UNIVERSITY OF CALIFORNIA,
IRVINE

Stuff's Cheap, Things Are Expensive: Recognizer Disparities for Object vs. Homogeneous Texture Patches

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Cognitive Sciences

by

Andrew John Burton

2020

DEDICATION

To my grandparents first of all, for everything.
And to my father,
for listening to all of my unconventional ideas
and advising me throughout life,
to my brother, sister, and mother,
most of my friends,
and all of the nice people I've met in restaurants.

"Tiger got to hunt,
Bird got to fly;
Man got to sit and wonder, "Why, why, why?"

Tiger got to sleep,
Bird got to land;
Man got to tell himself he understand."

and

"The only way I can feel the least bit important is to think of all the mud that didn't even get to sit up and look around."

–"Bokonon"
Cat's Cradle, 1963
(Kurt Vonnegut Jr.'s replacement thesis)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank Don, who has steadfastly been my advisor (and far more importantly, my friend) for many years at this point. He's one of the most expansive, unrestrained, well-read thinkers in science, and fearlessly goes after the big ideas. I meet and watch a whole lot of people around the world who know that, but I've gotten to perceive it firsthand. In general, he's made a real impression on how I prioritize things – and stuff.

I feel honored to have been able to make a contribution, however slight, to the lineage of computational vision researchers whose work I describe in this thesis.

My committee members, Mike D'Zmura and Jeff Krichmar, have also been very supportive of me during my time at UCI, and I thank them for far more than examining this dissertation. Jeff Barrett and Charlie Chubb served on my advancement committee, and you will find that they are also formidable thinkers about complexity. Charless Fowlkes, one of the greats in segmentation, served on my second-year exam committee, and taught my graduate class in computer vision. I had the good fortune to take vision itself with George Sperling, and a pivotal undergraduate perception class and Honors program with Ginny Richards.

Emre Neftci, head of the Neuromorphic Machine Intelligence Lab, also served on my second-year exam committee, and provided a bit of a second home for me while I was attempting to topologically accelerate (neuro)evolution. Georgios Detorakis, from the NMI Lab, has been an especially supportive friend and collaborator.

Alex Bower, from my PhD cohort, has been another very good friend and important collaborator. He's also pretty funny.

A focus on relevant official ties quite unfairly leaves out a large number of very close friends, classmates, faculty of the Department of Cognitive Sciences, undergraduate students, and even a few administrators I'm glad to have known. My former labmates at the Design Science Lab of VF Corporation and friends from the Campuswide Honors Program are certainly included among this number. I appreciate the social, emotional, and operational support of each of them.

Somewhat unconventionally, I also thank Ted Adelson at MIT. While we unfortunately didn't get to work together (perhaps the ill-timed worldwide phenomenon of a certain infamous dress cosmically got too much in the way), his exceedingly influential body of work and that of his students is rightfully quite well represented in this thesis. He also had the correct opinion about genetic algorithms, it turns out.

Very little of this project would have been practical without the open source community, particularly the contributors to the SciPy ecosystem. This work was overwhelmingly supported by teaching fellowships in general and experimental psychology.

VITA

Andrew John Burton

EDUCATION

Doctor of Philosophy in Cognitive Sciences (Cognitive Neuroscience) **2020**
University of California, Irvine *Irvine, CA*

Master of Science in Cognitive Neuroscience **2017**
University of California, Irvine *Irvine, CA*

Bachelor of Science with Honors in Cognitive Sciences **2014**
University of California, Irvine *Irvine, CA*

TEACHING EXPERIENCE

Teaching Assistant **2015–2020**
University of California, Irvine *Irvine, CA*

ABSTRACT OF THE DISSERTATION

Stuff's Cheap, Things Are Expensive: Recognizer Disparities for Object vs. Homogeneous Texture Patches

By

Andrew John Burton

Doctor of Philosophy in Cognitive Sciences

University of California, Irvine, 2020

Professor Donald Hoffman, Chair

Decades prior to the advent of deep learning, filter-based recognition and synthesis techniques functioned competently on patches of homogeneous texture ("Stuff"), as opposed to object-centric patches ("Things"). Since this competence gap was so obvious, requiring the revolutionary invention of new models to handle Things and scenes at all, the gap has largely eluded quantification. Using a subset of images from the Caltech256 and USPTex databases, this dissertation develops methodology to quantify a probable "Things vs. Stuff" processing dichotomy, examining the separability of these two metacategories in performance, similarity, and quality disparities they produce in both primitive, random-noise filterbank recognizers as well as a parameter-intensive ensemble of WGAN-GP discriminators. In the primitive recognizers, Stuff is shown to be categorically easier to recognize than Things, and the filter kernel size is shown to matter much less than the choice of filterbank histogram statistic. A "subordinal" statistic, the *signchain*, is introduced, and it is shown to be comparably effective to retaining the bins themselves. As a separate emphasis, the discriminators of the "GANsemble" are shown jointly to retain the ability to perform ordinary classification, even though they are trained only to spot fakes and are trained only on one class, provided each network's activation in the ensemble undergoes a form of subtractive normalization. To visualize discriminability lost with GAN loss, an identical-architecture ensemble with cross-entropy loss, the NONGANsemble, is created as a comparison. The pairwise firing affinity on real and fake images and pairwise model space distance (i.e. MSE, p-norm, Jensen-Shannon, signchain, SSIM of the weights) matrices are inspected for both, showing the Stuff detectors to be more promiscuous, and the Thing vs. Stuff distinction to visibly emerge "early or late" in terms of network layers depending on the choice of loss function. NONGANs are shown to generalize better to unseen classes (permitting an effective omnibus classifier for objectness), but at a cost of not being able to control synthesis and being potentially less compressible. Performance is at ceiling in the deep recognizers for Things and Stuff, but the competent synthesis of Stuff happens systematically before Things, even when using transfer learning to retrain Thing GANs. The popular Inception Score used for GAN quality assessment is shown to be unusably biased against Stuff because Inceptionv3 was trained on Things, and Fréchet Inception Distance is recommended in its place. Late in VGG16, Stuff classes occupy fewer filter channels but occupy them more fully. In the final chapter, Things and Stuff classes and networks are studied in terms of the separability of their MDS embeddings, and an algorithm and taxonomy was created that facilitates the conjecture that Stuff behaves more like a single Thing than all Things like a single kind of Stuff. Using interleaved rounds of MDS and Procrustes superimposition, the "embedding of embeddings" or *metaembedding* is introduced, visually reinforcing the main results found in earlier chapters (such as the fact that signchain distance resembles the Jensen-Shannon distance). Finally, the rankings produced by primitive and deep recognizers are combined in a higher-level embedding and also via the Borda Count method to produce a composite recognition difficulty ranking that supports Things being often harder to recognize than Stuff. The synthesis quality as estimated by FID is moderately correlated to ease of recognition, suggesting a "computational disfluency" account of image complexity as composite processing difficulty under fundamental operations (retrieval, segmentation, restoration, and destruction) is possible. This suggests Things and Stuff classes should not be naïvely combined in artificial vision systems, and should potentially be suspected to not be fully combined in natural vision systems.

Introduction

In a counterfactual visual world where there is no notion of time, and thus no need to analyze what's in front of us for potential uses or afforded actions, or at all then the ability of tracking coherently moving parts of the visual field, would things such as "**objects**" themselves need to exist? That is, in visual processing circuits, could everything instead ***safely*** be treated as fully egalitarian texture patches without pause, feeding all input blithely to communal neural networks?

In this dissertation, I argue that the answer to the second, better-defined question is **No**, because homogeneous material patches of texture are not only obviously different than individuated object-centric texture patches – they also can be expected to behave differently in subtle ways that, through a variety of methods and throughout a number of fundamental visual processes, you will be able to see.

We start by examining the history of convolutionally-founded visual recognizers and demonstrating the wide-ranging importance of reaching fair thresholds and extracting nuance from graded Same/Different judgments under uncertainty. Then we see that a homogeneous vs. heterogeneous texture divide along these lines exists in the very simplest and uninformed recognizers (linear classification on random noise filter features) as well as one of the most complex classes of recognizers known, an ensemble of deep neural networks actually designed for the much harder task of synthesis. Homogeneous textures are shown to be both easier to recognize and synthesize, although the recognizers for homogeneous textures are seen to be in some senses more promiscuous and brittle. This difference is pronounced enough to regularly survive projection of this data into low-dimensional representations. Satisfying a secondary objective, the potential for synthesis-controlling recognizers to emulate ordinary recognizers is substantiated and qualified. Along the way, new algorithms and visualization methods are proposed that touch on this homogeneity distinction but are for the most part generally applicable to the study of any gross behavioral dichotomy in visual recognizers.

To begin: An apex predator we contrive is combing its way through the tall grasses of an African savannah in search of an unaware or disabled prey animal. The predator should have the clear advantage: straight-ahead eyes, a faster top speed, a superior brain, and, perhaps more potentially, surprise. But if this hunter suddenly had to *consciously* enact the operations of the historically very fruitful token-based approach to perceptual organization conspicuously championed and developed by Marr (e.g. [114] [113] [116] [115]) – to extract all of the lines in the scenes and join them into contours, and fill those contours into surfaces, and parse those surfaces into objects, and segregate those objects into parts, and aggregate those objects into gestalts, would it not starve because of its inability to select, let alone track, a viable target in time? Even if these operations were carried out *unconsciously* instead by incredibly efficient, parallel circuitry exercising neurons, if not to the point of excitotoxicity at least to the edge of their practical neural speed limits induced by refractory periods, the number of potential tokens and candidate interpretations in a clearly viewable but complicated and densely populated panorama is sometimes so vast, or conversely so small under dimly detailed conditions, that the problem seems insurmountable – these operations must bear *some* cost. Costs can always be adjusted in a simulation-based argument to make a point. However, when you develop the abstruse edifice of all of visual processing on the foundation of edge detection, it seems that you will eventually be running up against an independently menacing feast-or-famine scenario that will eventually topple it: when looking for interesting regions in a sea of material, a surfeit of irrelevant edges could stop you, and so could a dearth of relevant edges.

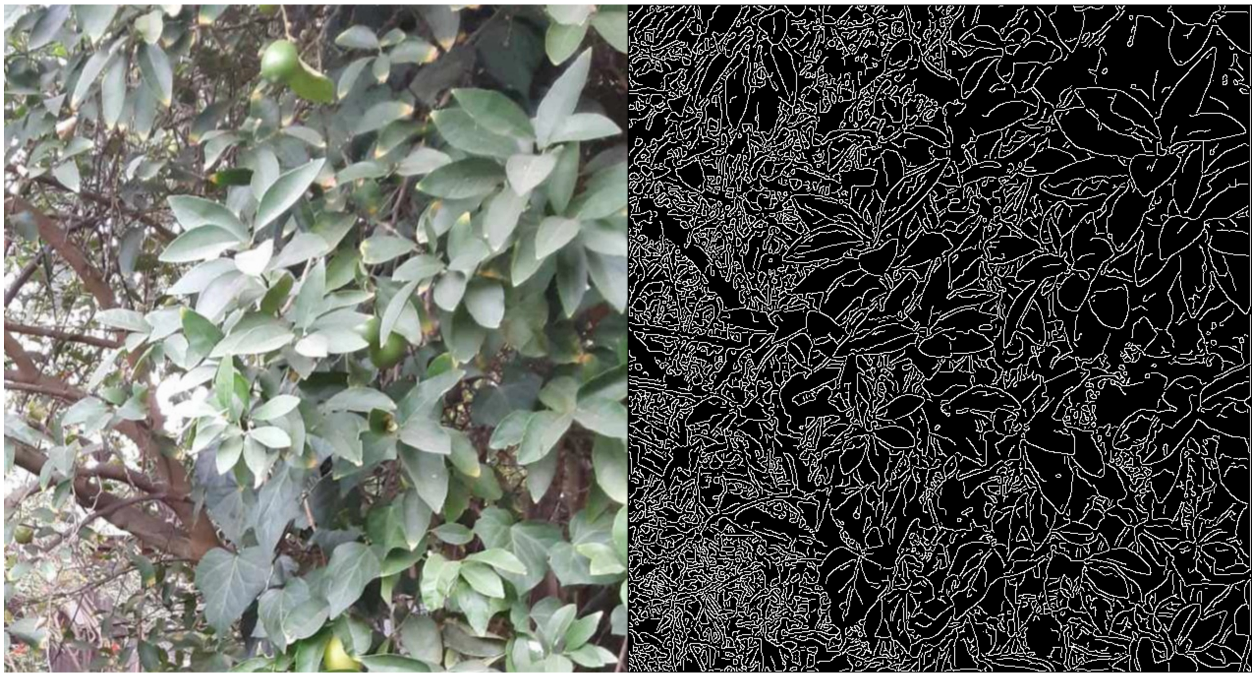


Figure 1: The *feast-or-famine* problem of edge-token-based vision. On the left, it is difficult but possible to see the two unripe lemons (center-top, center-bottom) using texture information. In the edge map induced by a Canny [25] edge detector, it is impossible. Edgels often either give too little information about important contours or too much information about irrelevant ones to depend upon to develop closed regions, surfaces, and objects by joining them, even if the method used for linking and denoising is an effective one. Often, we are interested in edgels from Things (objects) and not edgels contributed from Stuff (material).

Today, a vision researcher who adhered closely to the Marr philosophy might also starve, at least for funding. Modernly, the alternative view (emblematic of the outgrowths of the work of researchers like Malik and Perona in 1990 [111]) that the cells studied by Hubel and Wiesel [81] and explored at the computational level (and lower in Marr's hierarchy, see the quadripartite iteration in [116] that separates algorithms and mechanisms) by Marr likely carried out direct texture perception and not edge detection for more symbolic processing has won out on grounds of explicit technological proof, with the proliferation of the highly successful and yet still somewhat simplistic circuits known as convolutional neural networks (CNNs) [53] which in the course of effecting direct discrimination of input to categories induce sets of primitive "edge" detectors but do *not* use them to develop intermediate representations called edges.

In the generative rather than discriminative domain, parametric texture synthesis as a field (a la Heeger and Bergen [74]) has undergone a period of transformation and then a movement toward obsolescence. Careful research into which features of an engineered model are important for reproducing textures from a small set of statistical parameters (e.g. [132]) has been replaced by work that has shown that texture synthesis models based on CNNs can be largely based on coherencies within a model initialized with random noise filters (i.e. [163]). The generative adversarial networks of Goodfellow [63] have been expanded and combined to produce compelling models [185] of text-to-image translation. The ability to synthesize photorealistic scenes from text descriptions is intuitively a capability significantly more advanced than the ability to synthesize organic patches of mostly homogeneous visual material.

In 1991, Adelson and Bergen [2] described this tension as one between "things and stuff". Initially, this distinction referred to a belief that a focus on texture measurements (histogram statistics of filter response maps, such as means, variances, and kurtosis) would be more fruitful and naturalistic than an investigation into the token construction process (e.g. the inference of edge, blob, and bar primitives and their interrelations). A decade later the things vs. stuff distinction (Adelson 2001 [1]) emphasized investigation specifically into which techniques worked for perceiving object-like visual input vs. material-like visual input. In this body of research, work on the perception of homogeneous texture patches is motivated by practical concerns. Understanding domain-specific material perception facilitates intelligent robotic systems (Adelson et al. 2009 [88] describes a tactile approach) that not only know where an object to be manipulated is, but also how much force can safely be applied to the object. Combining texture classification techniques with human insight into the general behavior of materials based on their specularity, regularity, and lacunarity allows important automated predictions as to the behavior and quality of candidate cosmetics, foods, and art objects (paintings, computer renderings and photography are the explicit examples enumerated).

A great deal of texture research has been focused on naturalistic textures, to the point where the field has its own specialized textbooks and a prodigious number of journal articles dating back at least to the 1970s. The label of "machine vision" is often more associated with automated inspection systems that assess agricultural or industrial products as they leave an assembly line, applying various ad-hoc anomaly-detection techniques (for example, the general textbook of Davies [35] includes an amusing section on the specific case of detecting rat feces in cereal on a production line). Texture research has tended to be application-centric, whereas research into scene processing and object recognition has not retained this strict grounding. For example, the Kylberg texture dataset [102] contains classes such as rice and linseeds, and the CuReT dataset [34] focuses on characterizing the BRDFs (bidirectional reflectance distribution functions) of sample materials under differing illumination.

We all strongly tend to feel that there are clear differences between things and stuff – for one, objects are often segregated into parts at deep concavities (Hoffman & Richards 1984 [78]) whereas patches of regular texture often comprise large regions of these parts – but we still do not confidently know how strictly necessary these theoretical and intuitive differences are for visual processing in the computer and in the brain, especially for static images with no need to observe temporal coherencies (after all, a CNN recognizes "objects" just fine with no specific concept of objects in general). Attention has recently been given in the literature (Geirhos et al. 2019 [57]) to how texture-based vs. shape-based these direct neural-network approaches are. The aim of the proposed research program is to revisit and investigate the "things vs. stuff" distinction within computational vision in terms of the question: "What is the level of computational resources needed to process homogeneous texture (material) patches as opposed to object-containing patches?".

To summarize as well as to preview the remainder of this dissertation, the "things vs. stuff" distinction was important in the theoretical conflict between the token-based or symbolic account (edges $\rightarrow$ contours $\rightarrow$ regions $\rightarrow$ surfaces $\rightarrow$ objects $\rightarrow$ groups etc.) of perceptual organization developed by Marr and the direct texture-based or statistical approach which the success of simply discriminative neural network models has vindicated. In fact, much prior work in human and computer vision has indirectly addressed the question: histogram statistics and simple classifiers paired together have long sufficed to produce decent classification results on material patches, but only with the recent popularity of large-scale convolutional neural networks (CNNs) has object recognition been considered close to a solved problem; parametric *image* synthesis models in general have long produced good resynthesis of naturalistic visual texture patches (driving an essentially defunct field of *texture synthesis*), but it has taken the emergence of many deep convolutional generative adversarial networks (GANs) to generate plausible object exemplars and scenes.

Deep convolutional advancements in image segmentation have begun to revisit the "things vs. stuff" distinction in the specific research area of "panoptic segmentation" (Kirillov et al. 2019 [99]), wherein models are used to decompose a scene into object-like and material-like parts to build upon "semantic segmentation", the more intensive approach of labeling each pixel in the scene (using a CNN) as belonging to a class, as well as instance segmentation, the classical kind of segmentation. The Inception score (Salimans et al. 2016, [140]) quality measure critical to the development of convincing GAN output (believable rather than muddied hallucinations of new examples from a training set) and the evasion of artifacts associated with underfitting and the "mode collapse" problem is based on taking synthesized output and running it through a well-established classifier not used to train the GAN, and then examining the entropy of the label distribution. This approach can be generalized to work somewhat with classes out-of-vocabulary of the discriminative model by observing that high-entropy, low confidence label distributions are likely to be associated with the visual qualities of blurred "stuff" when the model has been widely and repeatedly trained on "things".

With performance of convolutional networks on mere discrimination of object sets in the thousands approaching a ceiling, it is now important to consider how many resources are minimally necessary in terms of training time or model capacity to achieve a desired level of performance, and the notion of textural complexity or, at the least, "objectness" (Alexe et al. 2012 [5]), the quality of patch that makes it seem like a "thing", may be a helpful predictor.

In addition, if discriminators for Stuff require less capacity or optimization, then a quick pass to carve the region into Stuff and Things could allow a sort of visual triage, letting the more sophisticated recognizer systems be brought to bear on windows of a scene that are not Stuff like. Given the much slower speed of neurotransmitters and neurons as physically-instantiated information processing units as compared to MOSFETs and microprocessors, and the demands of scene processing, it would be in that case a powerful heuristic to divert Stuff and Thing input to separate associative areas in cortex for filtering as well as for storage (i.e. V1 and V2, but also IT). The functional imaging studies of the future might be focused on locating Stuff-specific areas that can join the fusiform face area (FFA) [91] and the parahippocampal place area (PPA) [47] of Kanwisher and collaborators.

In consideration of these efforts, the business of disentangling "things" and "stuff" in precise quantitative terms continues to be relevant, and further knowledge of the distinction may be tied to advanced ability to diagnose the performance of state-of-the-art and resource-constrained models. Operationalizing the "Things vs. Stuff" distinction is naturally linked to the idea of coming up with a "fair" model of *texture complexity*. One way to attack this directly and briskly is to consider the construct of texture complexity very coarsely and with a simple design (to test the average performance of merely the class "things" against the class "stuff"), and to attempt to develop it as a measure established by converging operations (to measure this performance in the context of several fundamental but highly distinguished operations).

Further distilled:

- Scenes have multiple Things in them internally composed of Stuff. Scenes also host swathes of undifferentiated Stuff.
- Things and Stuff are essentially very high-level categories (metacategories) that sit atop the visual (conceptual) object hierarchy.
- We seem to want this distinction to exist, and the distinction seems to be relevant to research areas that have yielded compelling capabilities.
- But does it?
  - in a way that we can operationalize through a disfluency-based notion of textural complexity?

Entropy and Kolmogorov complexity as unsatisfactory measures of texture complexity

Whenever the notion of complexity is broached, there are two varieties that spring to mind. They are both wholly unsatisfactory where naturalistic images are concerned.

The first kind is the complexity that is suggested by the difficulty of compressing the image. An image is almost trivial to compress if it has the same gray level or intensity value everywhere. An image is very difficult to compress if no regularities at all can be exploited. When considering the pixels as being drawn from a discrete probability distribution, the first case corresponds to a (discrete Dirac-type, or Kronecker delta) spike centered at the gray level, and the second case is most similar to drawing from the uniform distribution over all achievable gray level values. This continuum is measured by the Shannon entropy [144] (defined for convenience in 2.3.8).

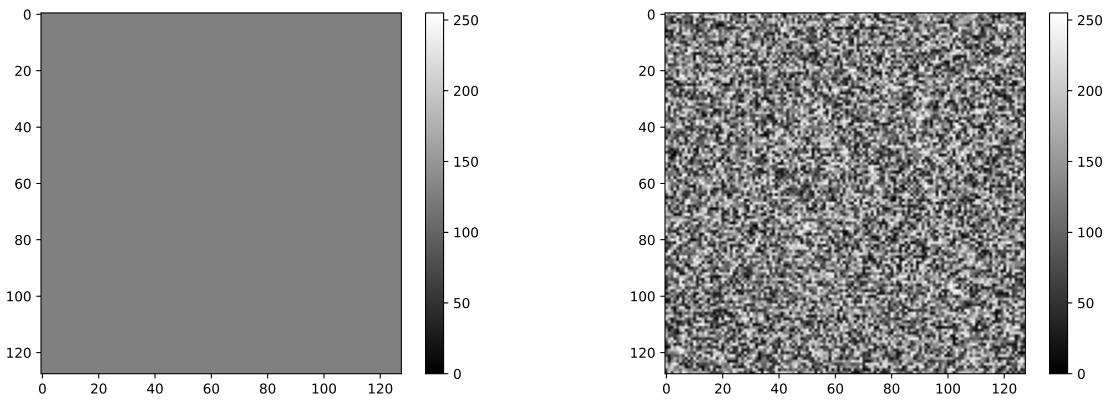


Figure 2: A very low entropy image (i.e. 0 for a truly flat color) and a very high entropy (uniform noise) image. Entropy is not a satisfactory measure of image complexity because large numbers of perceptually indistinguishable high-entropy images are easy to generate and lack interesting structure.

One doesn't need to actually calculate the Shannon entropy for any images to see that practically all interesting images the normative, nonspecialist subjective human observer would call "complex" inhabit the mid-entropy regime. Given that entropy is a quantity admitting a continuum, we can just look at the extreme cases to see this. The Shannon entropy as a form of image complexity certainly has one side of the spectrum correctly defined: a very low entropy image or a zero entropy image (such that there is no uncertainty about each successive pixel) is not complex in the least. Figure 2 shows an image with no pixel uncertainty and an image with maximum pixel uncertainty. The high-entropy image consisting of white noise should not be considered complex. Indeed when we see data of this character in nearly any scientific pursuit, we just wave it away and say of these images that they "look like we got a bunch of white noise". In specifying entropy in terms of the pixel distribution, there is no accounting for structure. Assuming that the pixel values actually followed the uniform distribution rather than being simply drawn from it, we could create a drawing program that allows someone to draw an interesting picture interactively by sorting the gray levels so that they become available to a drawer according to intensity and as long as all pixel values are conserved, the image would have the same empirically measured entropy.

Obviously, you can add "structure" to entropy by windowing the images and taking the probability distributions conditioned on window location to get some idea of how entropy varies over space. In this sense of hierarchical entropy calculation, perhaps patches whose subregions exhibit no variegation in entropy are reckoned "less complex". But unless your visual world consists of chessboards sometimes interrupted by the snow of white noise, this seems to miss something essential. Interesting images have self-similarity, semantic meaningfulness, composition, improbable angles, interesting occlusions, patterns that follow natural stochastic processes such as erosion or reflectance, and so on. Even in throwing out all of the images which constitute art, or depict scenes, or show natural or man-made objects and confining yourself to naturalistic texture, there is a rich complement of qualities that make an image improbable to generate by chance but perhaps conditionally highly probable based on the context. You can imagine mathematically reasoned, non-entropic proxies for complexity: perhaps seeing the number of times within an image the image displays fractal self-similarity. But it is the case that not all intruguing structure or evidence of generative difficulty or of image improbability falls under these tidy mathematical measurement schemes.

The other convenient mathematical construct which enjoys some universality in any conversation about complexity is the Kolmogorov complexity, or the algorithmic complexity. It is defined to be the length of the shortest description that can reproduce an object. Most commonly, it is seen as the shortest-length computer program that can compute an object into existence. It does not necessarily cohere with entropy, and it does not continuously develop. For example, a very large high-entropy image generated from the uniform distribution could consume billions of bytes. But a very small number of bytes suffices to regularly yield something nigh-on perceptually indistiguishable from the image: the implementation length of a program that implements a linear congruential pseudorandom number generator to reproduce other random images. Now, the definition of Kolmogorov complexity does not include "perceptually indistinguishable" in its definition, and maybe a variant of it ought to be developed to, but even under a condition of exactness there are problems. As Cover and Thomas [32] observe, "print out the first 1,239,875,981,825,931 bits of the square root of e" yields a number whose Kolmogorov complexity is (now supposed) much, much less than its neighbors on account of the existence of the command as embodying a "program" to compute the number. Indeed it is not particularly fair that the programming language can have almost whatever fundamental instructions or syntax or semantics you wish to reduce Kolmogorov complexity. The complexity has known and provable bounds for many kinds of objects (e.g. bitstrings in general), but it is not exactly computable or particularly meaningful for many objects.

Put another way, if the Kolmogorov complexity K for a string is defined [32] as:

$$K_C(s) = \min_{p:C(p)=s} l(p),$$

where C is a computer that can run any program, s is the string, p is the program, and l is a function that takes the length of the program, K is in some sense not very interesting unless we replace the *absolute minimum* (which is impossible to discover in general) with the average over some class of reasonable programs. In the visual analysis domain, this would be similar to seeing what the complexity of an image was by seeing what the complexity of the average competent recognizer itself was in terms of, say, parameter count of a neural network, but restricting ourselves only to a continuum of *subjectively reasonable* recognizers – because, at the end of the day after all effort was expended, the minimum-effort recognizer might only vacuously (e.g. accidentally) be able to make correct judgments about the image.

An average itself, to be actually useful for decision-making, needs to be a *practically realizable* and empirical average that does not consume undue time. To contrast, the example of the "halting probability" related to Chaitin's incompleteness theorem [27] and determining Chaitin's Ω number [32] perhaps demonstrates to the point of absurdity the lack of wisdom in relying on intractable (and in that case, uncomputable) accumulations of the behavior of an exhaustive space of programs. Chaitin's number's definition implies that if you authentically have in your possession a certain number of bits of precision of the number Ω then you must have known about the Halting Problem outcome of the programs (which temptingly include, for large enough binary representational precision, very interesting programs such as proofs of unsolved mathematical conjectures like Goldbach's conjecture [32] which are formulable as searches for conjecture violations) in the space, and also of length bounded by the number of bits of precision. However, the contributions to the accumulated probability-type summary of halting involve individual tests (i.e. the running of programs) susceptible to the Halting Problem. The number itself, then, while admitting a rigorous definition and able to be *say, empirically, found for some number of bits of precision* for individual machines (e.g. [23], which uses this kind of notion to deem the Riemann Hypothesis as more complex than Goldbach's conjecture under a specified machine language) and program encodings, is not generally computable. Non-practically computable (and certainly non-computable!) notions of complexity should arguably be put completely aside in our artificial vision analyses – they are obviously not what a natural vision system (which has to function reliably and in a tractable amount of time) would use for complexity estimation, if those systems do complexity estimation at all explicitly in anticipation of carrying out various stages of visual processing or encoding for storage.

Of course, in adopting a very different strategy incorporating some subjectivity and a notion of how "interesting" a texture patch is, we would have to be careful to not let complexity lose its more objective grounding in difficulty, resource-intensiveness, or surprise. There are admirable efforts [96] to estimate the memorability of images, as measured by recognition accuracy using human subjects and as later approximated by convolutional neural networks, but memorability is not complexity. We still want complexity to reflect difficulty, or the improbability of realizing a result.

To avoid this, "complexity" should be strongly grounded in the probability of computational processes producing an image, recognizing an image, repairing an image, retrieving an image, destroying an image, and so on. If human observers are involved, top-level biases and the impact of attention and any number of other experimental nuisance factors come into play in determining complexity as a result. The case of entropy shows that it is useless, though, to *absolutely* ground it in the bare probability of synthesis, assuming nothing about how images are actually generated, either by physical processes, computer programs, or biological or artificial neural networks.

Complexity should be estimated in light of the obstacle an image (in the case of instance-level complexity estimates) or classes of image (in the case of class-level complexity estimates) poses to information processing. It is perhaps helpful to view image complexity then as a characteristic of the image akin to friction, which prevents fluid and uneventful information processing – and to restrict our study of that friction to at-hand, practically-realizable systems.

This dissertation therefore focuses on operationalizing complexity as processing difficulty composited over several fundamental operations (which I hereafter term "computational disfluency"), focusing on the fundamental operation of recognition because it is potentially behind many of the other fundamental operations of visual processing and perceptual organization. The emphasis is placed on the distinction between Things and Stuff and the promise of showing Stuff to involve less disfluency-type complexity (interpose less difficulty to, for example, recognition) than Things. The proximal objective is to devise potential methodology for class-level estimates, not instance-level estimates, partially because of the belief that instance-level estimates should cluster around the class-level estimates, and partially because recognition normally exists at the class-level granularity, whereas repair, synthesis, segmentation, and effacement regularly need not necessarily.

Hypothetical distributed recognition and segmentation systems: feature thresholds & same-different judgments

Things and Stuff, however, might themselves be indirectly and partially defined by how they behave in response to processes that incorporate recognition. We briefly divert to highlight this possibility with two contrived examples.

Recognition as carried out by *recognizers* of any complexity often involves two fundamental objects of psychophysics: thresholds, and Same-Different judgments. For example, the binary case of classification (i.e. the target class vs. *not* the target class) occurs in logistic regression, and at the end of multiclass neural networks where a "one-hot" battery of terminal neurons are ON or OFF, and also construed to be heading towards mutually exclusive. An alternative scheme to the "multinoulli" case involves quantizing the activation level of a single terminal neuron and assigning bands of activation strength to individual categories, but this is generally chancier as the activation levels even of a functionally competent neural network approximating a straightforward linear classification might differ in response to such factors as nuisance input variation, training time, stochasticity of training, noise sustained even during inference, and, in neuromorphically-instantiated artificial neural networks, the uncertainty of the readout itself. In light of these sources of error which complicate assigning fair, contiguous thresholds to categories, a binary judgment allows for minimal confusion between categories.

Of course, recognition processes are not merely an end unto themselves. Recognition can also be used internally within some other process of importance, as in the example of controlling synthesis: a recognizer can be employed to detect changes in the level of conformance of the synthesizer to the important criteria of, with added training, producing patches that are different than they produced on the last step, and more likely rather than less likely of being confused with the images that are known to be members of the ground truth training set.

To contrive an example of where Same/Different judgments tightly control a process that does not constitute a normative form of "recognition", we can consider a type of segmentation that is neither semantic segmentation nor instance-level segmentation: the problem of establishing regions and contours by kinetic rather than osmotic spreading activation. There are two natural points of extreme contrast with spreading activation methods. A trivial contrast is the discriminative approach of deep learning: unless you are using a network specialized for semantic segmentation or incorporating, like the R-CNN of Girshick et al. [59], explicit region proposals, the notion of regions, similarly to the notion of objects vs. material, does not (seem to) need to exist in the first place. The second contrast is made with the historical, token based kind of segmentation. In the symbolic approach of Marr, the raw primal sketch [114] that is composed of a map of edges, solid bars, bar terminations, and blobs (putative regions of arbitrary geometry which exist with higher confidence because the edges already enclose them prior to adjudication) [115] is developed into the full primal sketch using processes that include (but are not limited to!) judgments of whether edge elements that look like they are going to be contiguous should be joined into a contour or not, and judgments of whether edge elements look extraneous or not. Sophisticated implementations of just this subset of judgments involved in the development of the primal sketch could become very involved, since varying scales of contours should be taken into account, as with the scale-space filtering methods developed by Witkin [178]. An early implementation, however, of these judgments is found in the description of the Binford-Horn Line-Finder [80], which also explicitly taxonomizes contour finding methods into those that are *edge-oriented* and those that are *region-oriented*. Of course, in the case of purely subjective contours whether static or moving [95], such as Kaniza triangles, or in the case of having to view the analyzed region incrementally through an aperture (anorthoscopic perception, [136]) edge-oriented methods would encounter major difficulties.

The region-oriented notion of endogenously computing boundaries by envisioning them as the limit of spreading activation is so ubiquitous in thinking about images (e.g. the intuitive flood-fill algorithm) but also neural activity that it does not seem specifically creditable to any one line of thought. Nevertheless, a thoughtful treatment of spreading activation's possible role in establishing contours was given by Ullman [162] in the course of his proposal of visual routines, or the notion that there may be many mid-to-high-level processes, not necessarily local and not necessarily passive that are involved in effective visual perception. A nonstandard use of spreading activation inward is certainly associated with the *grass fire transform* or *prairie fire transform* of Blum [14]. Blum's transform works (metaphorically, burns) inwards, revealing the "skeleton" of the region, which is for some very simple shapes a straight line medial axis.

We may consider these methods to be osmotic in that the spreading activation is radial up to the limit of the "resistance" provided by the contour. This is not the only alternative: for example, the spreading activation could be contrived to be ballistic. Casting rays from a centrally-located seed point is reasonable; after all, an intuitive way to describe regular polygon shape profiles in a 1-dimensional sense when the boundary is known is to consider the plot of the distance to the boundary. A circle, for example, maintains a constant-value centroidal shape profile commensurate with the constancy of the radius of a circle. This polar approach obviously quickly becomes unsatisfactory for irregular objects, as fugitive rays or lines of escape shot to the boundary can irregularly sample interesting geometry at different scales or be blocked from observing far boundaries by the interposition of near boundaries. The validity of a process founded on spreading in a single direction at a time to compute contours in all directions and escape some of these difficulties may not be immediately obvious, but it involves embodied cognition related ideas of using the environment to compute.

In an exotic example of something akin to spreading activation, Tero et al. [159] showed that the slime mold *Physarum polycephalum* could morphologically compute approximate minimum spanning trees: researchers placed oat-flake food sources in a configuration matching city locations in the Tokyo area onto a substrate, and an introduced slime mold first spread from its seed position to cover much of the space, but later contracted into a series of hubs and transport tubes that formed a network linking the food sources – a network highly similar to the Tokyo area rail system presumably designed with robustness as well as branching efficiency in mind.

In an example of biologically-inspired rather than biologically-instantiated computing, in the Ph.D. thesis work of Dorigo [40], an agent-based heuristic method for approximate shortest-paths graph search was proposed inspired by the pheromone-based navigation of ants and their own scouting of food sources. Ant colony optimization and related algorithms involve sending agents on trajectories through the graph. At each step, they deposit a "pheromone" value on what they have just traversed. Critically, the amount of pheromone deposited is decayed as time progresses so that later wandering which might prove to be aimless or indicative of graph cycles gets discounted. When the "ant" finds a target (analogous to a food source), it returns to the seed point, as a scout might to a real ant colony. The graph annotated with pheromone information for the original ant is likely to not represent an efficient path to the food source. However, when multiple ants are introduced, their pheromone annotations on the graph can be combined, and it can be seen that some arcs are more traversed than others. This tendency builds because of a further naturally-inspired innovation which is to make later ants attracted via pheromone to the choices made (the arcs traversed) by previous ants.

Leaving signals in the environment as a form of indirect coordination is known as *stigmergy* [46]. Of course, image analysis could make use of stigmergic processes: instead of an annotated graph as in ant colony optimization, we could maintain stigmergic maps. Through the introduction of stigmergic maps, it becomes possible to consider ballistic spreading activation from a seed point as a way of establishing contours by combining (through a union-type operation) individual trajectories "painted" onto stigmergic maps. The maps (slices of a stigmergic tensor) can be combined by some consensus operation to determine the region endogenously.

We can define a toy process of *kinetic image segmentation* (KIS) using a few simple objects and the guiding principle of imagining similarity of texture as related to a kind of permittivity within a medium, and a lack of permittivity at and beyond the interfaces. It will be an active sensing process for determining the bounds of the regions, similar to the active sensing philosophy of sonar, but for shape. This process is not intended to be immediately practical or even ever practical for segmentation problems. It is dependent on the scale of the textural input in ways that have not been compensated for, and at any rate, image segmentation has arguably already been "solved" by deep convolutional networks implementing semantic segmentation. However, it does illustrate an important point about stationarity through space as it relates to texture. The stationarity of a signal is usually defined in terms of how little the signal changes statistically over time. Since images are signals as well, to say that a class of moving images exhibits high stationarity is to say that the image statistics change little over the image sequence. If we consider time to instead be yoked to viewing small windows of an individual static image, a class of images (say even a broad high-level class, such as all Stuff) exhibits a different kind of high stationarity with viewing time if, when randomly viewing contiguous parts of the image as by smooth translation, the image statistics change little. For homogeneous image patches (continuous texture, or Stuff), this is to say that some statistics could be expected to be more often translationally invariant than for heterogeneous patches containing objects (Things).

We can begin to demonstrate KIS by taking perhaps the simplest kind of homogeneous image patch, a region of flat color. For extreme simplicity of demonstration, a black square inscribed in a white square field can be used as the target region to be developed. A seed point for the kinetically spreading activation is chosen near to the centroid of the square. The seed point will shoot agents of spreading activation that can be referred to as probes.

The seed point does not know where contours defining the boundaries of the region are likely to be found, so the probes are shot in all directions, as defined by choosing random angles spanning the unit circle. Probes advance a small amount of distance in a particular, randomly-initialized direction on each discrete time step. The probe coordinates need not be confined to integers, but the underlying scalar field of the image to be analyzed is discrete. Probes come equipped only with their angle of movement, their square sampling window, a window influence function, and a model of the texture at their seed point. The model of the texture at the originating seed point could be very complex. For example, it could be the confidence output by a deep convolutional neural network making a Same/Different judgment with the benefit of millions of parameters tuned by gradient descent. Given the simplicity of the black square stimulus, we will equip our probes with an equally rudimentary model of source-texture: the proportion of black pixels. If we start our probes in the center of a black square, then our excruciatingly simplified model of textural anomaly is initialized to expect a high proportion of black pixels (i.e. 1). Critically, it tolerates approximate matches, as all good recognizers should. For the time being, the threshold for being considered possibly the same texture is set arbitrarily to 0.5.

We can assume a window influence function that is constant across the square window admitting all of the pixels equally to the texture model, yielding the square itself. In practical terms, to handle corners better with ill-sized probe windows, the influence function should probably be defined to be a symmetric Gaussian. It is fairly critical that a small step size is chosen – if too large a step size is chosen, a tunnelling or temporal aliasing problem ubiquitous to physical simulations and collision detection can occur. Since the collision detection in KIS is approximate, the procedure is more equivalent to ray marching (collision is checked at incremental sample points) than ray tracing (where collisions are often analytically solved).

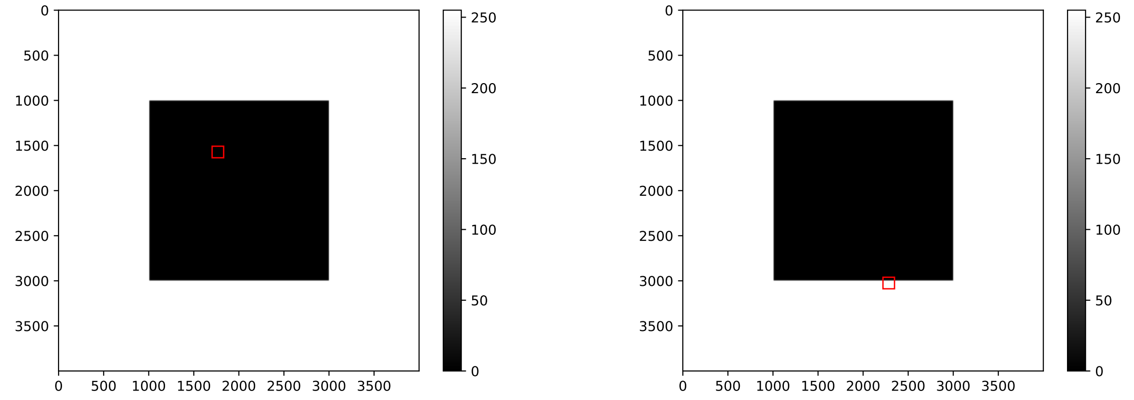


Figure 3: Kinetic image segmentation on a simple homogeneous square with a seed point near the centroid. A probe with a square perceptual window (128x128 pixels) and a constant influence function (as opposed to Gaussian falloff, which may be better for more complex region shapes), is shot from the seed point with a small step size ($s = 2$) in a random direction (i.e. $x = x + \dot{x}, y = y + \dot{y}$; $[\dot{x}, \dot{y}] = [s \cdot cos(\theta), s \cdot \sin(\theta)]$) on the original image (4000x4000). In the first case, the probe is in the interior of the region, and its simple model of texture (the proportion of black pixels) initialized based on the window at the seed point (the middle of the square) declares that the currently observed image window is consistent with its model (Same), avoiding any deflection. In the second case, the probe is straddling the boundary, nearing escape. When the texture model is inconsistent (Different), a different direction angle is chosen for the next timestep. This causes the probe to become mired if it does not return to the model-consistent region. The probe may escape the region significantly, in which case it will become permanently slowed.

When, after the probe has moved to its new position in a new timestep, the assumption of Same texture is violated (here, for example, when 49% of the pixels are black), the probe will deflect. If we sought to model the probe's collision as a billiard ball or pinball we would need the geometric description of the obstacle so that the angle of reflection could be calculated from the angle of incidence based on the contact point and accounting for any obstacle-related geometry. In the case of the crisp black-white boundary, this might be a well-defined boundary we could look for, but in the great majority of cases, the interface between domestic and foreign texture could be poorly defined.

Therefore, to make fewer assumptions, the probe is set to deflect in a random direction. Note that this could yield the same direction that caused the deflection, or a similar direction, fugitive from the seed point and headed out of the region. In this case, we will depend on the fact that further timesteps in that direction are relatively more likely to trigger future deflection events. On a more regular basis, the probe will be metaphorically stuck in the doldrums until it chooses a direction consistent with Same texture. Sometimes, however, the probe could move by fits and starts out into the foreign texture, where it will be trapped around the periphery of the object, not subject to the guarantee of a reasonably fast escaping random walk because the model is constantly issuing Different judgments leading to random movements.

On each timestep, probes paint themselves onto their stigmergic map, much like a stamp. Accordingly, until the less common event happens that the probe escapes the source region consistent with its texture model, the region of the image that is consistent with the model will tend to be painted over and over with paths determined by stochastic deflection. When the escape event does happen, the probe will not tend to move far out of the model-consistent region, so the error of the process is partially self-limiting. The word *partially* is chosen because with enough expended timesteps for the probe, the probability of observing a rare event such as an escape is more likely. Painting the negative space outside the region in an ambling path also becomes more likely, owing to the tendency of random walks to escape any area. No probe, therefore, should live forever. The demise of a probe due to old age is a constraint on the probe that leads to a termination condition, *probe senescence.*

Figure 4 shows the result of 100 probes painting the same stigmergic map, where each probe has a lifetime of 2,000 steps. The odd texture induced by the intersecting paths might be called the *resonance texture* of the paths, and the contrast in the texture which is reminiscent of liquid metal is actually due to integer overflow (an artifact of using an array of unsigned 8-bit integers to hold the map). 2000 steps of this small size using the probes described is enough to somewhat tidily define the square's boundaries kinetically.

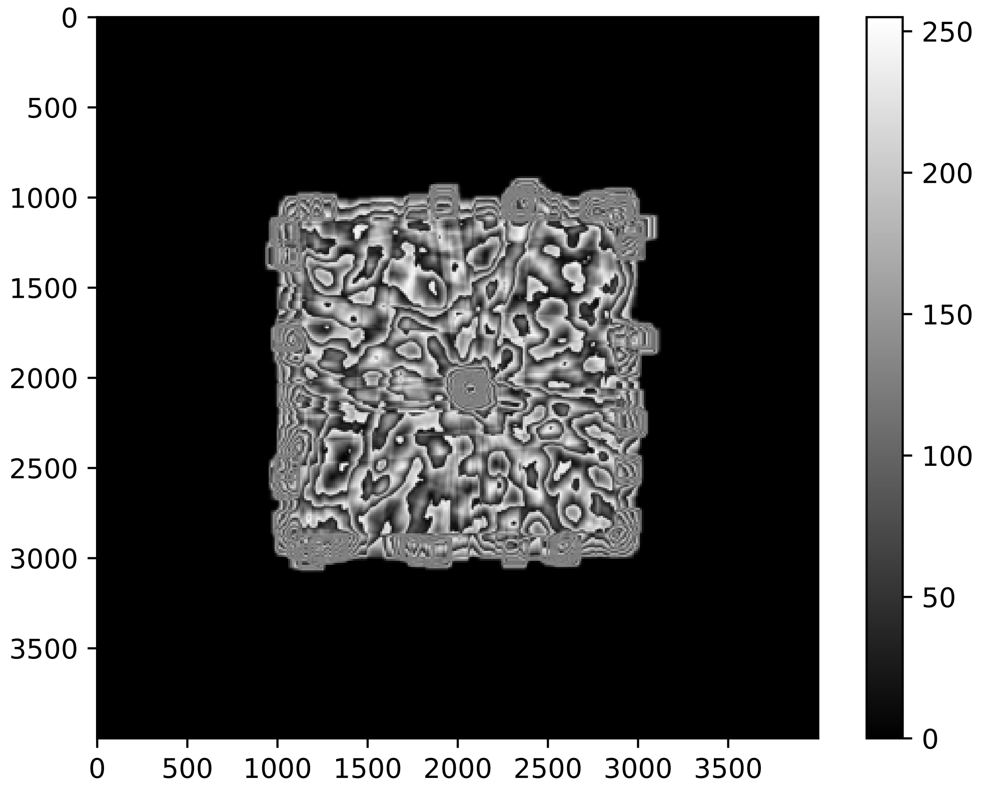


Figure 4: Kinetic image segmentation, under the influence of probe senescence. The collective stigmergy of 100 probes is shown, each probe with a life of just 2000 steps. Black-swan events of the probe escaping the bounds of the homogeneous square are reduced simply due to shortened observation times. The "resonance texture" exhibits topomap-like ridges and rectangular banding artifacts, and moreover reveals the probes' seed point texturally. The unusual banding is due to using the conventional 8-bit unsigned integer to hold the gray level for each pixel of the stigmergic map – even though each probe timestep imparts an activation of 1, overflow is permitted. Here, the activations are simply sequentially summed onto the same map, but the separate probe maps could be captured separately in a stigmergic tensor, a consensus function such as simple averaging down the volume could produce the average map.

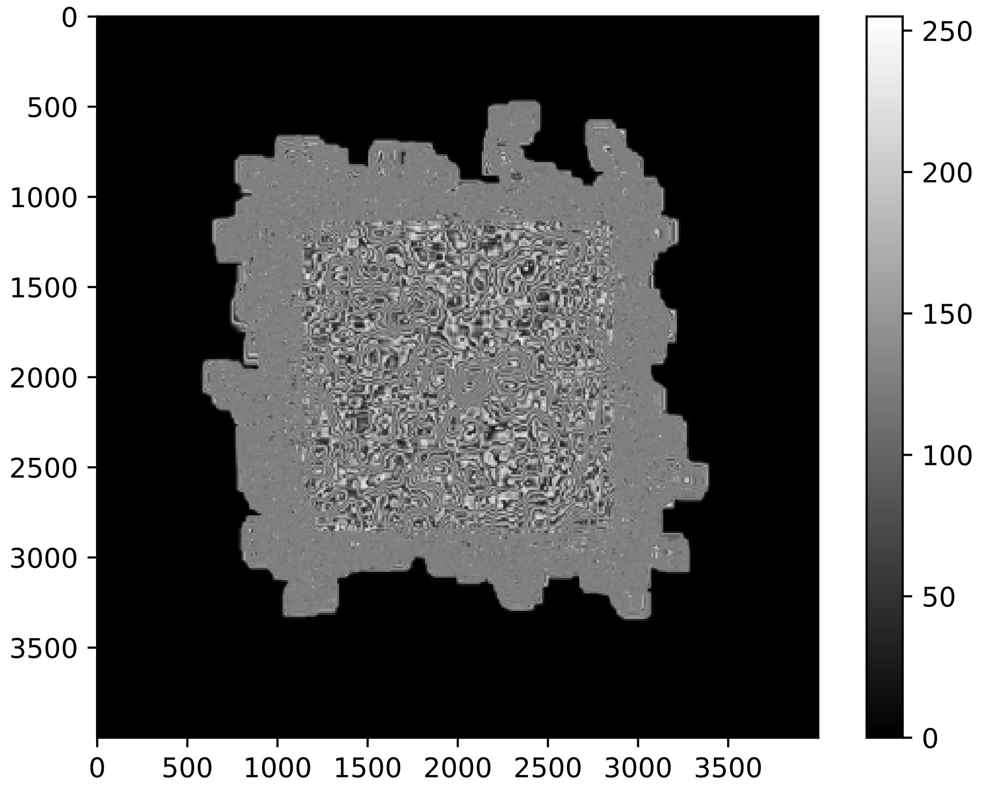


Figure 5: Kinetic image segmentation with 20000 step probes, beyond the help of a salutary effect of probe death by senescence. The inlier resonance texture is varied and complex. The outlier resonance texture is a flat gray, corresponding to regions painted with fugitive probes, that escape and then become stuck outside the originating region.

Figure 5 shows the result of a looser pressure of probe senescence, where probes are given 20,000 timesteps to live. When the time-to-live is increased too much, the rare events of probes worming their way out of the region with the seed point is increased, and the probes move on rambling trajectories seriously harming the definition of the region. It is interesting to note that the outlier paths induce a smooth texture (because there are fewer path intersections, and thus less overflow) than the inlier paths. Shapes can thus be analyzed by the texture they induce with Same/Different texture probes.

Figure 6 shows the KIS process allowing 20,000 steps per probe but with an active loitering or vagrancy detection process that terminates the probe before the end of its life if it is deflecting 95% of the time in a moving window of timesteps. The moving window can be shortened to make the 95% rule more draconian.

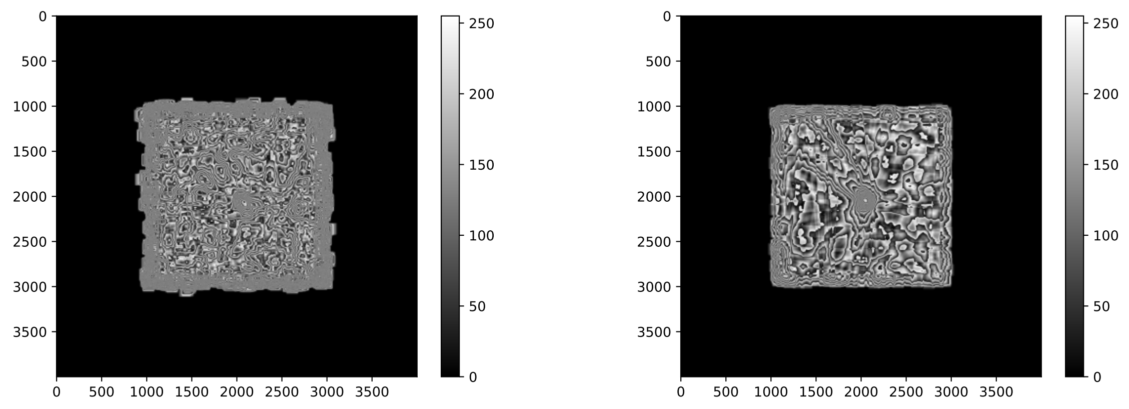


Figure 6: Kinetic image segmentation with 20000 step probes, but using an active detection routine that kills the probe early if it is “loitering”, or deflecting 95% of the time in a moving window of timesteps. The moving window is a) 1000 and b) 100 timesteps long, which controls the strictness of the loitering termination criterion. With stricter termination criteria, the stigmergic contour more closely matches the actual contour. The resonance texture on the boundary is qualitatively different from the coarse inner resonance texture.

In Figure 7a, we can see that the bizarre resonance texture disappears to be replaced by a more commonsense gray patch depicting the region. The change involved was increasing the representational precision from unsigned 8-bit integers to 64-bit floating point numbers.

The extra integer precision averts overflow for this number of timesteps. The seed point, since it always represents the starting position of the probe, is painted most often, and can be seen much more clearly than in the overflow-prone condition, where it can perhaps be diagnosed through texture analysis – here it is just the maximum of stigmergic activation. Figure 7b shows the path of only one probe before it is terminated for loitering. Appropriate corrections for the probe going out-of-bounds of the image are also something to think about – to kill the probe for leaving rather than to return it is a bias in some sense against exploring regions that are the background or the negative space of centrally located subjects.

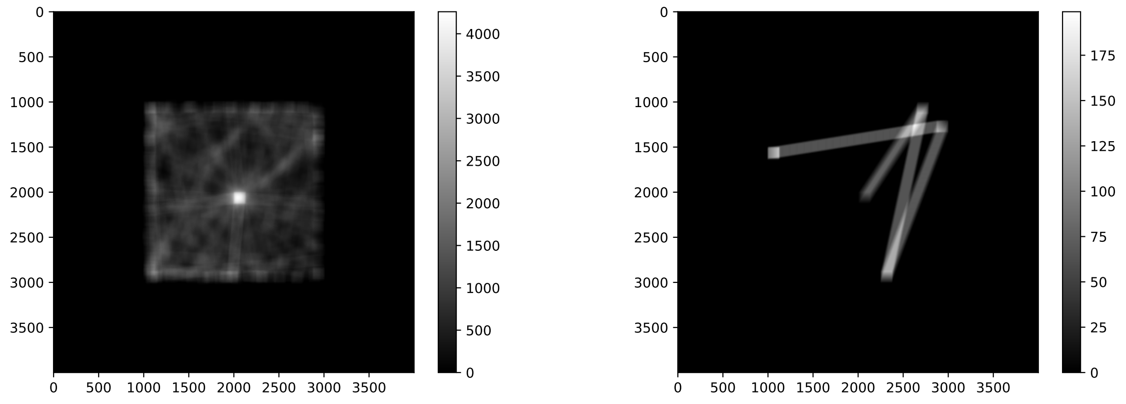


Figure 7: Kinetic image segmentation, a) identical to the preceding case, but using a 64-bit floating point stigmergic map (averting integer overflow). In this form, the seed point can be found from the consensus stigmergic map by thresholding, and the region can be estimated by blurring and thresholding. b) The result of only using one probe vs. 100.

It was earlier promised that this method would help to ameliorate the core insufficiency of centroidal shape profiles: concealing the influence of boundaries hidden behind other boundaries from the perspective of first-contact raycasts. To understand why, it is helpful to see the probe as akin to the proposal distribution in the Metropolis-Hastings algorithm [73]. In Metropolis-Hastings, the usually symmetric proposal or jumping distribution allows us to synthesize a new location by assuming some uncertainty around the current location and to compute an acceptance ratio used to determine if this will be the next location, and samples are drawn according to this acceptance. Markov Chain Monte Carlo sampling using this kind of algorithm allows us to sample some low-probability locations, although with a built-in local spatial dependence. Our probes also propose a new location, and decide whether the location is acceptable or not, and the Monte Carlo process of simulating multiple paths also allows us to penetrate (albeit more rarely) into harder to get to subregions of the region being analyzed by KIS. The difficulty of visiting a subregion is defined by its narrowness, how its shape interacts with probable points of pseudo-reflection elsewhere in the shape, and where the seed point dispatching probes is relative to connected subregions permitting entry into it.

The immediately obvious remedy to this difficulty is to provision more seed points, and to combine the consensus stigmergic maps originating from seed points with each other conditional on a hierarchical analysis that joins seed points if their maps have sufficient pairwise intersection and the texture models at each seed points also agree. The particulars of this graph-based analysis can be omitted to purchase some brevity, but it is likely hierarchical models like this have high utility in vision systems – for example, the action of Gestalt laws can be thought of as the graph theoretic problem of how a multigraph (edge strengths representing the force of individual laws like similarity and proximity acting between nodes) becomes a hypergraph (hyperedges representing scene elments an observer reports to be perceptually bonded).

Where to place the seed points? One rule for generating good candidates is suggested by considering Hoffman and Richards [78] in their statement of the minima rule:

"*Minima rule.* Divide a surface into parts at loci of negative minima of each principal curvature along its associated family of lines of curvature."

Figure 8 loosely reproduces the projected solid of revolution shape presented to illustrate the minima rule and subjects it to KIS at seed points roughly in the middle of each of the part boundary lines suggested by the rule. It can be seen that, in this instance, the part boundaries allot seed points that subjectively cover the object well, when the union of the seed points' stigmergic maps is considered. Of course, in the cases of completely simple flat color texture and completely known shapes, seed placement rules can be objectively evaluated by how efficiently the joined stigmergic maps of seed points generated by the rules tend to cover the shape without, as it were, "coloring outside the lines".

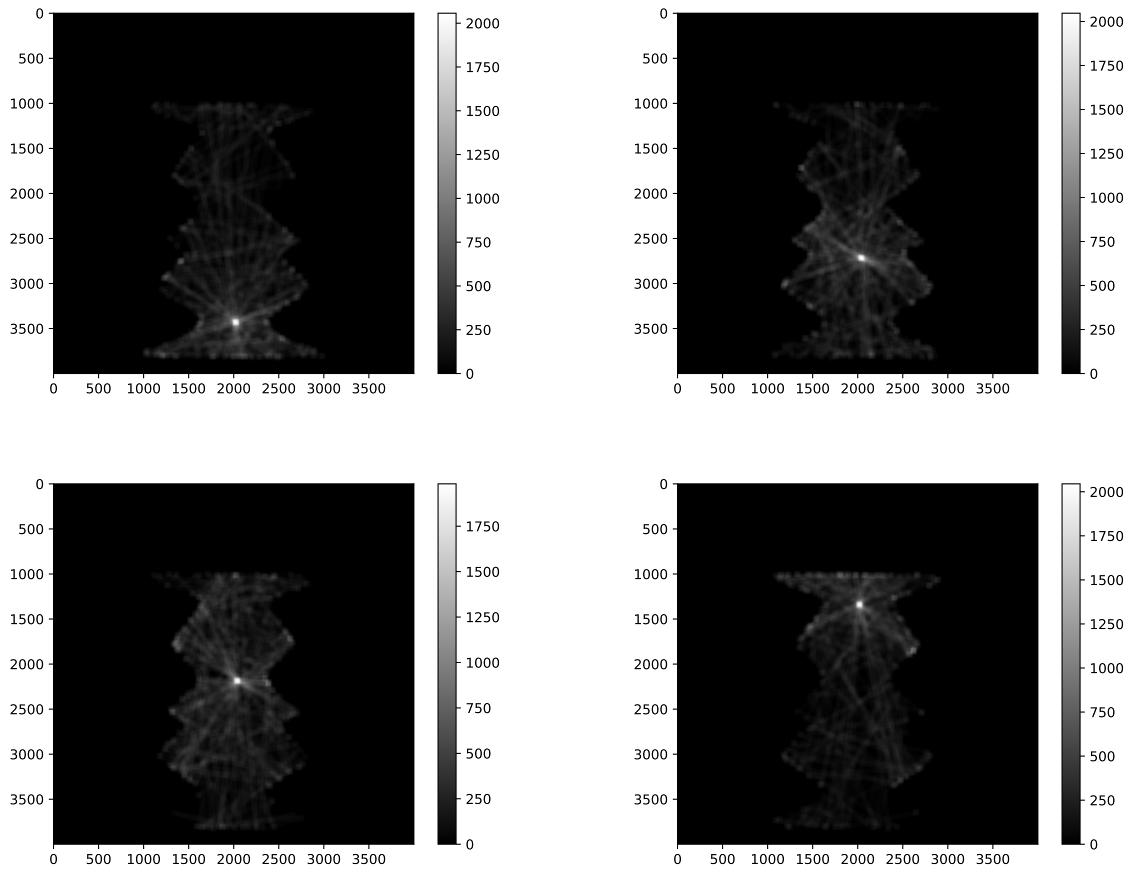


Figure 8: Kinetic image segmentation of a reproduced projection of a solid of revolution presented in Hoffman and Richards 1984, Fig. 16. The seed points (false color maxima) in each case are placed at approximately the midpoint of the parallel defined by each pair of negative minima of curvature, which are subjectively agreeable points for part segmentation. The negative minima can be seen to also define bottlenecks for a kinetically-spreading activation (as opposed to flood-fill). In a further graph-based analysis, seed points should be defined as nodes that are connected into graph components (regions) by defining edge strengths proportional to the degree their stigmergic map has significant overlap with other seed points, and the degree to which their texture model matches.

With various difficulties of KIS partially remedied, the main remaining bit of complexity concerns the texture model. It could be expected that tessellating a region with Stuff patches would be more amenable to KIS than a tiling with Thing patches. “Homogeneous texture” finds a functional definition, therefore, in how well it permits KIS. Of course, this conjecture is not possible to test until a competent model of Same/Different texture is arrived at. Various levels of model could be considered – the most primitive would directly report the degree of lack of pixel correspondence directly but would be very brittle to authentic variation, the somewhat-more-refined would calculate image statistics and try to match those statistics, and the more advanced would probably involve fine-grained discriminative neural networks making a direct judgment.

In addition to segmentation, recognizers can be critically important in supporting the process of retrieval. In the heyday of “shallow” neural networks, the Hopfield network [79] was perhaps the scheme most studied in the context of retrieval, but it has received renewed attention as networks which attempts to learn programs need auxiliary memory structures as components [134]. The Hopfield network implements an autoassociative content-addressable memory, which is to say that you can query the memory for a pattern by providing a sufficient part of the pattern. In the case of the Hopfield network and the Sparse Distributed Memory of Kanerva [90] the memories are fundamentally bitstrings. Though there is a certain universality to bitstrings, one can conceive of categorical alternatives (although alternatives without the advantage of built-in retrieval based on similarity).

Drawing inspiration from the simple letter-recognizing networks of the Connectionists [117], and also from experimental psychology in the tradition of discrete factors and levels, a data structure can be developed to serve as a content-addressable memory, based on constraint-satisfaction, for categorical values: the radial factor-level graph. Note that this hypothetical factor-level graph is wholly unrelated to the factor graph used in variational inference to associate observations or random variables with functions.

We can imagine all instances stored in the RFLG as being embodied as nodes in a basin. The data associated with the basin nodes are retrieved if they receive a prescribed number of units of activation, which they receive from level nodes. The level nodes and their parents, the factor nodes, belong to the list of factor-level trees (Fig. 9, Fig. 10) that comprise the outer structure of the RFLG. There exists one factor node for each factor being considered. We can take the example of a set of factors, backed by recognizers that perceive Low, Medium, and High levels of Rockiness, Muddiness, and Grassiness in an image patch. Assume that the image patch to go along with this example is a photographed section of a soccer field, the morning after a day it has been raining. Entertain that the soccer field patch is likely to be Low in Rockiness, Medium in Muddiness, and High in Grassiness.

To insert the patch into the RFLG (Fig. 11), we instantiate a basin node and attach the patch data to it. We walk the set of factor nodes and examine the children to locate the appropriate levels matching the patch and add a connection from the affected level nodes to the basin node. To retrieve matching patches, we first set the activation level of all basin nodes to 0. Then we take our search criteria (say the query is Grassiness=High AND Muddiness=Low AND Rockiness=Low OR High) and separately forward activation from the level nodes for each satisfied constraint. The node can be retrieved as an exact match if the number of factors in the query equals the number of activation units, and as a partial match if some lesser number of factor-level satisfactions is deemed permissible. The RFLG partially implements through the basin nodes the relations of relational algebra (note, though, that with only the structure currently in place it is unclear how to perform relational join operations).

In a vision system specifically, the query (the activity of the level nodes) is envisioned as driven by detectors of these various subjective textural qualities. Again, as with KIS this is meant to be taken as a toy example, as this data structure confers no clear general advantage if the activation polling is not very fast or somehow parallelized.

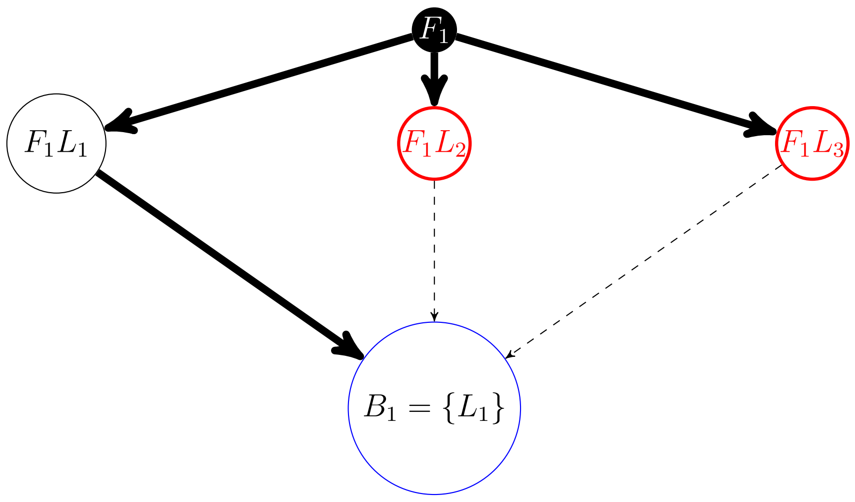


Figure 9: A "factor-level graph", containing just one instance and one factor, with three levels. On the outermost layer, a Factor node corresponds to a variable or feature. On the lowest layer, a Basin node stores an instance. In between are Level nodes (although for searching, this layer could actually be composed of search trees). For simplicity, disjoint levels can be assumed within each factor, such that each Basin node or instance is like a relation from relational algebra. The initial thick arrow denotes that when "activation" is given to the Factor node, the factor enters consideration. The thick arrow from the active level node to the retrieved basin node denotes that this instance was stored as exhibiting the first level of the factor. The dashed arrows represent nonexistent connections. Even when level nodes themselves are active, they cannot forward activation on connections that are weak or nonexistent, signfiying no stored basin nodes. When the factor is considered, the specific level or levels that are being queried are active, passing activation to any Basin nodes it is strongly connected to. Basin nodes with a sufficient level of "activation" are retrieved.

In a neural circuit, the level nodes would be driven by (not shown) detection mechanisms separately connected to level units, but in the simple data structure, the highlighted node corresponds to the Level (or possible value of the variable) we are querying for. Here, only the first Level is active, and so the basin node receives one level of activation. A reaping process in the basin selects every node in the basin that meets a threshold for activation, which is logically the number of factors considered in exact retrieval, and less than the total number in the case of approximate retrieval. If we consider only one factor, we should retrieve the node. There may be many basin nodes in the basin, only possibly connected to the level node shown.

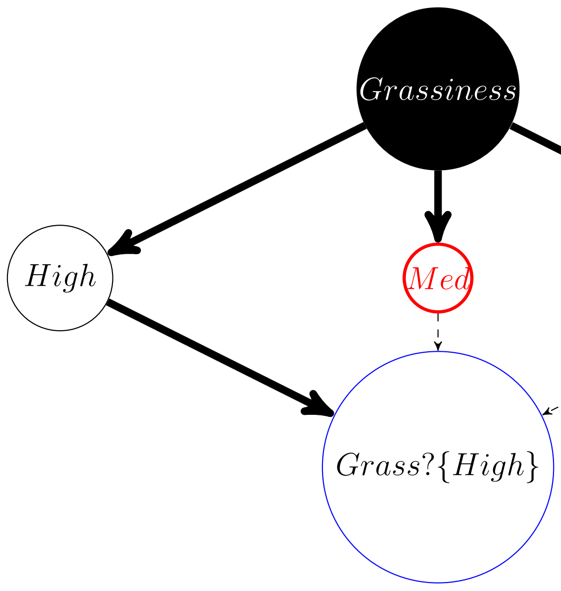


Figure 10: The factor-level graph made concrete. A single factor, Grassiness, is shown. To retrieve the image patches stored in the basin that are High in Grassiness, follow the pointer down to the High node and then push activation to every child connected to that node in the basin. A candidate node corresponding to a mystery image patch (Grass?) was apparently stored as being high in Grassiness.

If this represents an exact retrieval query with Grassiness={High} as the only factor, the mysterious patch stored in this basin node is returned as a search result.

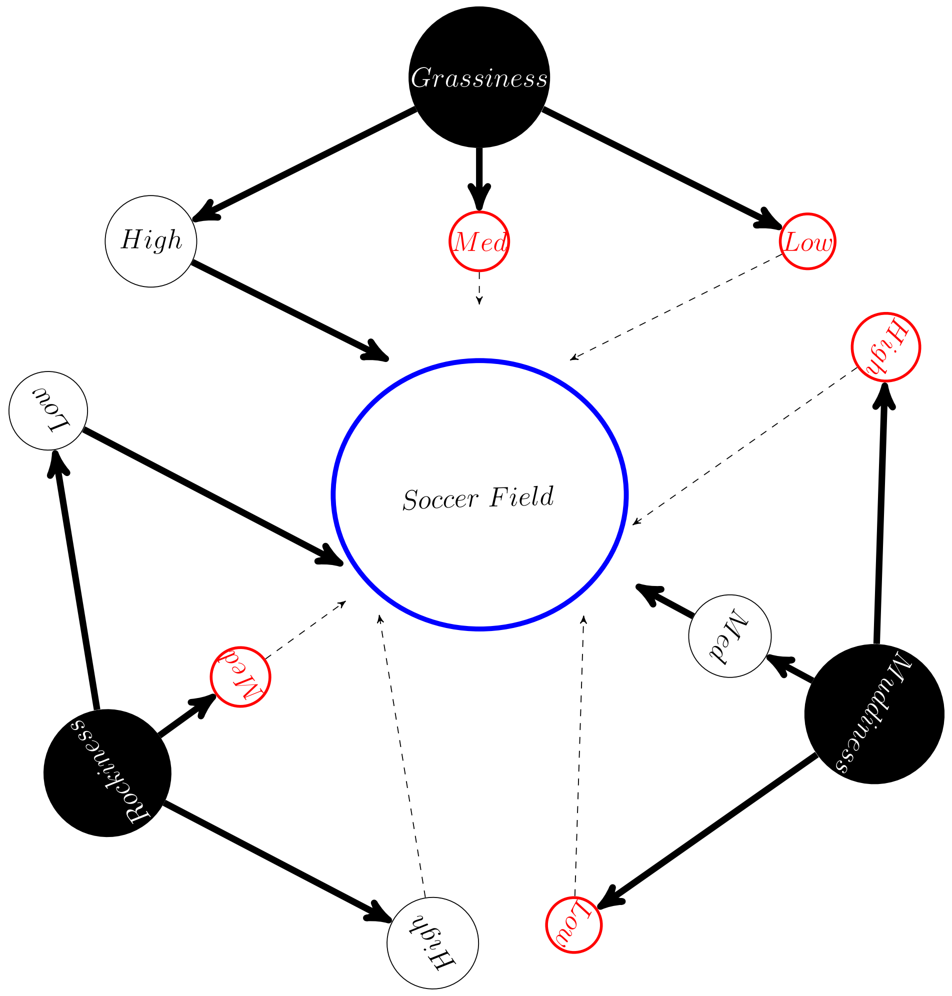


Figure 11: The radial factor–level graph is a collection of factor–level trees with a common basin (here only one basin node is shown). For example, the query "Grassiness={High} AND Muddiness={Low} AND Rockiness={Low OR High}" might return an texture patch of a soccer field the day after a rainstorm, that is High in Grassiness, Medium in Muddiness, and Low in Rockiness as measured by separate detectors with thresholded output. The activation level of the basin node is 3 units, corresponding to meeting three of the constraints, and in an exact retrieval scenario where these are the only factors, the patch should be retrieved.

The homogeneity of Stuff seems like it lends itself also to this kind of data structure being more appropriate. Separate instances of Things from a class seem less likely to be stored in the same basin nodes in the presence of noise in storage and retrieval processes than the homogeneous Stuff instances. Meeting the hard, categorical, and enduring thresholds that gate assignment to level nodes in this structure is an all-or-nothing proposition. It seems likely that it is worth more to adapt the input to be compatible with more flexible CAMs like the Hopfield network than to depend on recognizers' appraisals of High, Medium, or Low concordance with textural dimensions in a scheme like the one just described. It is unclear in the first place how one would come up with fair and meaningful dimensions for Stuff, let alone Things. Perhaps the fairest consideration of quality dimensions, assuming no other knowledge, is the "energy" or strength of association with a particular class from a dataset.

All of this speculative and perhaps unsatisfyingly incomplete algorithm design is to emphasize that providing quality recognition in the form of secure Same/Different judgments and stable thresholds could become critically important when recognition is applied to other primary processes of visual processing, and that Things and Stuff may behave with strong individual distinction in visual processes that are not simply recognition. To study the Things vs. Stuff divide in recognizers, then, is to see how well and from which origins Same/Different judgments and thresholds are developed between and among the two metacategories.

Given their potential importance in partially parallelizable non-recognition operations (e.g. retrieval, segmentation) of distributed visual processing, this dissertation will focus on the creation of recognizers and the analysis of the roots of their discriminability (specifically, models of Same/Different texture judgment or confidence judgments), with a particular focus on the important question of whether Stuff and Things can be safely combined within recognizers, or groups of recognizers acting as a composite recognizer. If recognition is, in fact, strongly linked to quality of synthesis (or in biological systems, "imagination" and hallucination), retrieval, and segmentation, the question becomes more important. This question is tantamount to whether a host of systemic differences exist between Things and Stuff (e.g. in the "power" or capacity of recognizer needed, in the confusability of textures, in the changes they induce in large recognizers like neural networks, in the amount of information they require or cause to be stored, in the character of separability that is observed between the feature vectors fed to drive them, or in the quality of low-dimensional representations for decision-making that can be learned from them) that would suggest that they ought to be processed by segregated systems – in digital computers and as well as brains.

0.1 Things vs. Stuff contributing datasets: Caltech256 and USPTex

The two datasets chosen to be representative of Things and Stuff are the Caltech256 [68] dataset and the USPTex dataset [10].

Caltech-256 images depict Things, are of varying extent and may involve odd viewing perspectives. They are a mix of background-containing and background-isolated images. For example, the classes `blimp, hamburger, waterfall, spaghetti, cannon, jesus christ, hot tub, and butterfly` often include backgrounds. The classes `chessboard, computer keyboard, flashlight, frying pan, french horn, joystick, knife, pci-card, vcr, yarmulke and revolver` often do not. Keeping the backgrounded and backgroundless images together is not naturalistic to pre-electronic-age human visual perception, but is naturalistic in the context of the datasets computer vision systems are often trained and deployed on, and the stimuli people currently perceive, when, for example, they shop electronically. Some classes are exceptionally non-naturalistic depictions (e.g. `eric cartman, homer simpson`) while some classes are quite specific (`palm pilot, ewer, ketch`). Caltech256 images are representative of many early image datasets that sourced their patches from researchers harnessing Internet image search.

USPTex texture classes are not given natural language labels but are assigned numbers. The USPTex dataset has only 12 samples for each texture class, whereas the Caltech-256 classes sometimes have in excess of 100 samples for the class (30,000+ images comprise the total dataset, but they are not evenly distributed amongst classes). This is unfortunate, but most well-known "texture"-focused datasets of a substantial size are not ideal to compare with image datasets because they are either photographing the same texture patch under different illuminants (designed to learn about the reflectance information modeled in bidirectional reflectance distribution functions, e.g. CuReT, ALOT [21], RawFooT DB [33]) or they are looking at entirely heterogenous images with the goal of identifying material used on an object (e.g. MIT VisTex [120], Flickr Material Database [145]). USPTex (in the vein of datasets with fewer classes such as the Kylberg agricultural dataset) otherwise begins from a position of strength in that it is capturing distinct patches of the same material.

Interestingly, there is a general lack of an appropriate, well-known database for the general study of homogeneous but actually uniquely sampled textures: possible future sources of good varied datasets could eventually include detailed video game worlds, and particularly satellite imagery by land type and land use, or entirely synthetic textures evolved by interactive methods from patches of stock images or hallucinated by GANs. ILSVRC 2017 (comprising 200 classes and derived from the ImageNet taxonomy) is a good alternative dataset for object images, although because ImageNet 2011 is a taxonomic database, there are very highly variable numbers of exemplars per category as well as very specific categories in the dataset.

Images from the two datasets are coerced to a standard resolution (128x128) which is less than the maximum of the two resolutions. It is possible to standardize the images as is very commonly done in preprocessing data for deep learning, but given the overall objective of this research program, it is a debatable question whether transforming the image gray levels into z-scores is a fairer assessment of the complexity of the images than using the images as provided (which may inhabit quite different gamuts), so this was not done.

Images were coerced to grayscale by applying just the luminance transform ([86], Table 4) of the ITU's YIQ NTSC color standard, producing the Y of YIQ:

$$E'_Y = 0.299E'_R + 0.587E'_G + 0.114E'_B,$$

where the E's are RGB gray-level intensities.

16 classes were chosen from each of Caltech256 and USPTex to serve as the Things vs. Stuff comparison dataset. They are depicted in Figure 12. Of note is the fact that these classes were not chosen at random but were picked for the potential of revealing some similarities among textures through various recognition, synthesis, and analysis processes. For example, among the Things the class `washing-machine` and the class `video projector` both often have ocular elements in the form of the projector's lens and the washer's laundry viewing window, the class `ak-47` has visual elements in common with the class `revolver` but also the class `swiss-army-knife`, and among Stuff there are multiple brick wall textures: `c191, c049, c047`.

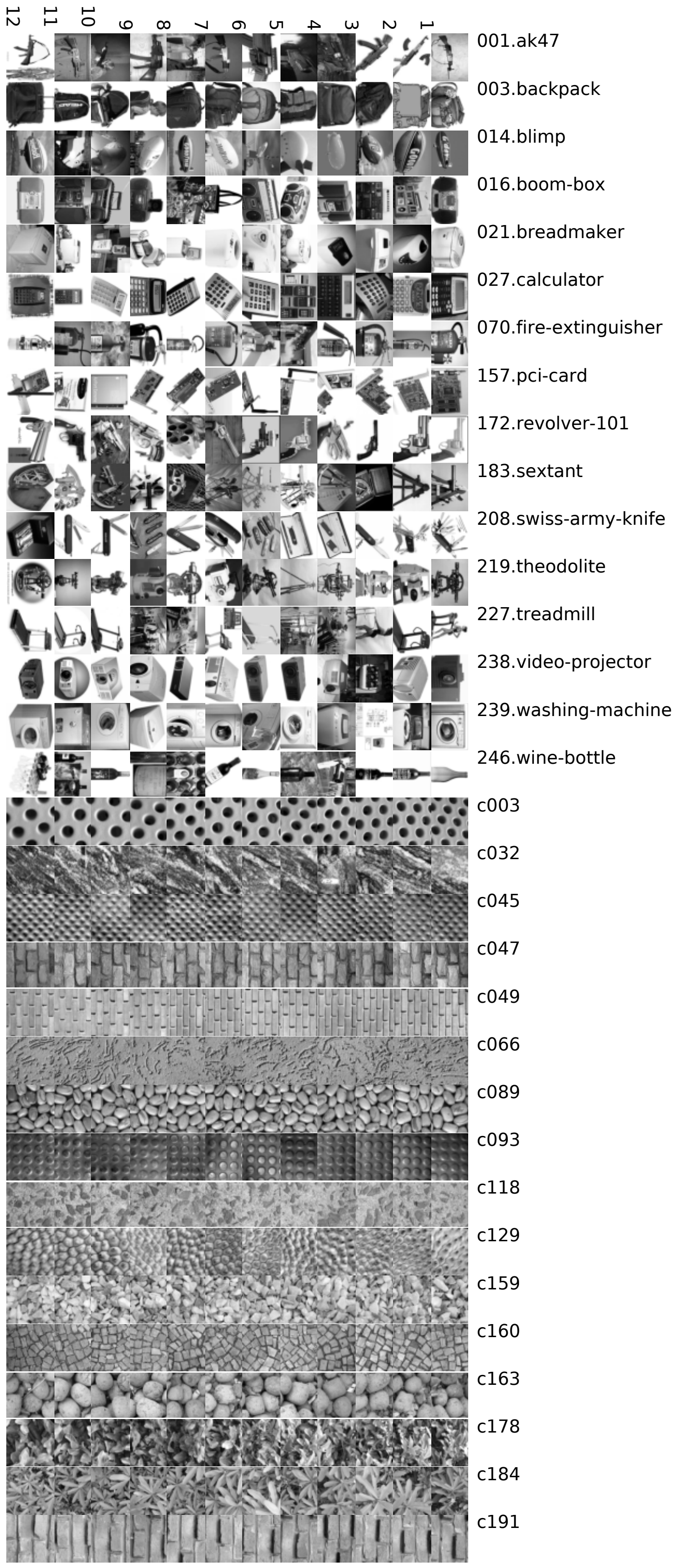


Figure 12: Sample images from single channel Caltech256 (things) and USPTex (stuff) classes used in the primary investigation

0.1.1 Superficial differences: simple image statistics

Things and Stuff as exemplified by the classes in our dataset differ more on the per-class deviation of their image statistics than the per-class mean of their image statistics. This is to say that Things are, instance-wise, more *individual* than Stuff. Of course, that very fact is built into our admittedly flawed *working* or adopted definition of Stuff as authentically homogeneous. Not all examples of a material from a certain class will be as homogeneous as these in this particular dataset. The differences between Things and Stuff developed in the remainder of this document may be somewhat exaggerated as an epiphenomenon of the lack of wide ranging material datasets where the subjects are all the same fine-grained kind of material (c.f. the Flickr Material Database, where material examples are arguably too individuated for this type of analysis, yet belong to a unitary class, like glass or metal). Regardless, it is important to establish the margin by which the Things and Stuff classes are already different from each other before being placed into the recognizers we will study in Chapters 1 and 2 and imagined in a common space relative to each other in Chapter 3.

The per-class deviations of Things are strongly significantly different (higher) than the per-class deviations of stuff, across all four of the simple functional-moment based statistics: means, variance, skewness, and kurtosis. The per-class means of means and of variances are significantly different, where Things enjoy a slighter advantage over Stuff. That is, the image mean gray-level value of Things as a metacategory is a bit higher on a per-class average basis, partially due to the backgroundedness distinction but not overwhelmingly due to it. The variance is significantly higher, again because of our notion of homogeneity. Whereas the greater deviation of moments of Things images is driven by the heterogeneity between instances, the per-class image variance being higher for Things is down to the heterogeneity within an image, or, as was discussed above with KIS, the lack of stationarity of pixel-based image statistics over space as you translate about the image.

While a future study could perhaps control the issue of varying per-class background proportion better by inserting standardized synthetic backgrounds behind already-background-subtracted instances, and could control the issue of largely unbalanced instance count by procuring or creating a new dataset with more instances, and both of these factors could contribute a difference that is not down to the essential differences of Things or Stuff, at some level the issue of heterogeneity vs. homogeneity is inextricably tied to the definition of material vs. object, unless you are comparing systems that process pictures of objects (Thing patches) but nevertheless reason about the material texturing those objects (i.e. the distinctly different aim associated with the compilation of the Flickr Material Database). In the succeeding sections, the heterogeneity of recognizer responses, recognizer constituency, and recognizer separability that will be on display has to be assumed to be affected simultaneously by input heterogeneity (depicted in this section), nuisance factors such as backgroundedness and sample intensity (i.e. relative number of image presentations), and any innate, more deeply meaningful differences between Thing and Stuff processing that might later be inferred.

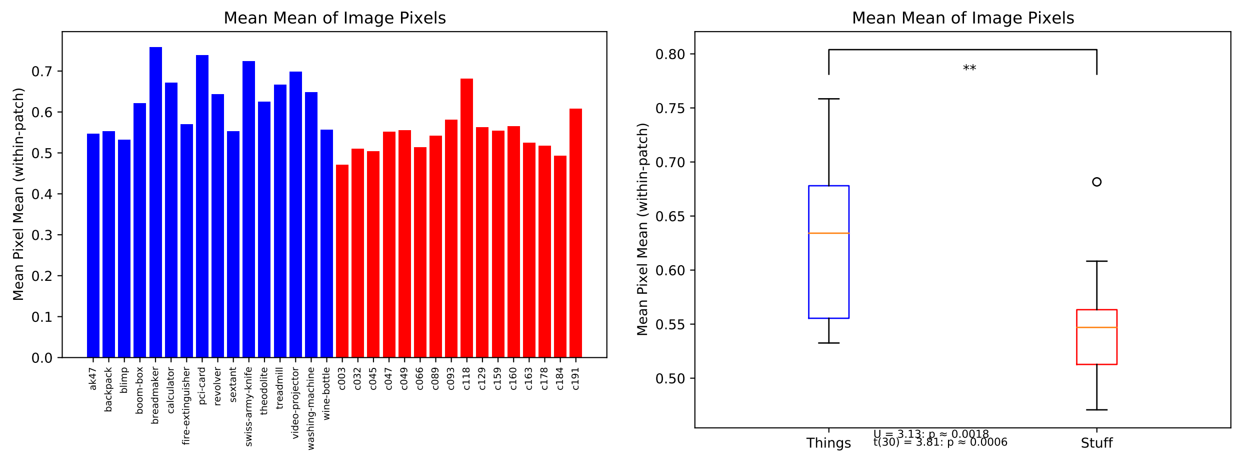


Figure 13: Per-class average mean gray level.

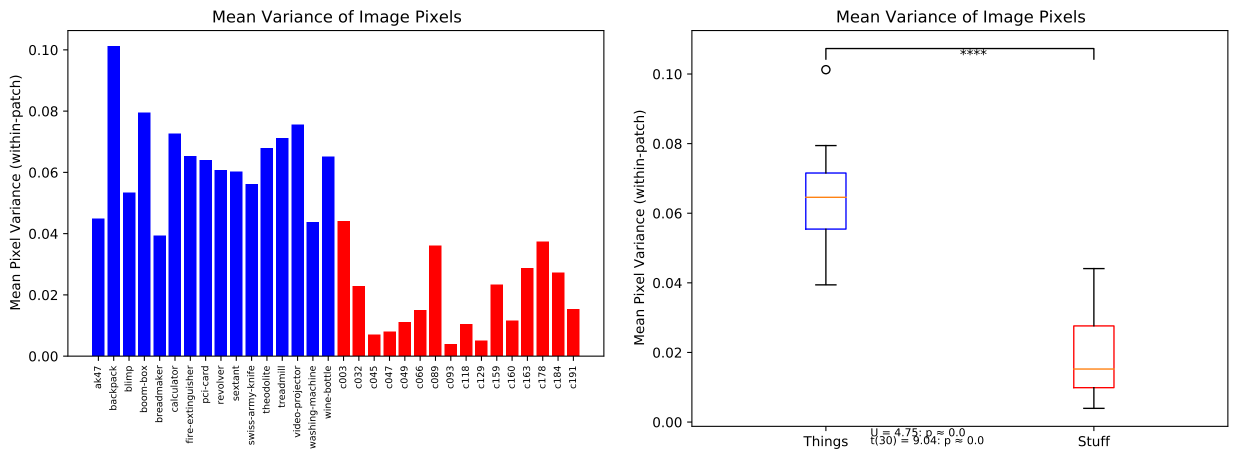


Figure 14: Per-class average gray level variance.

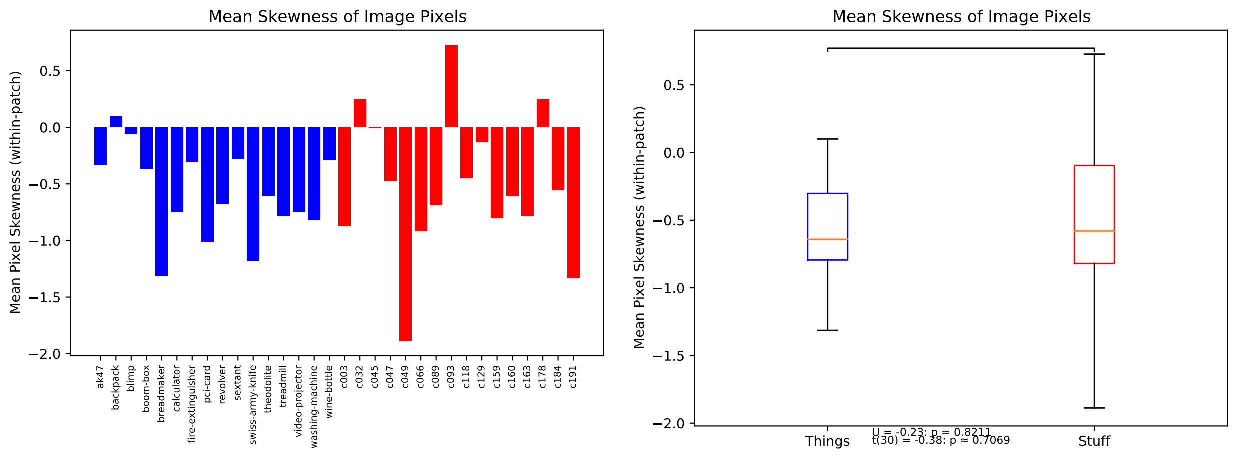


Figure 15: Per-class average gray level skewness.

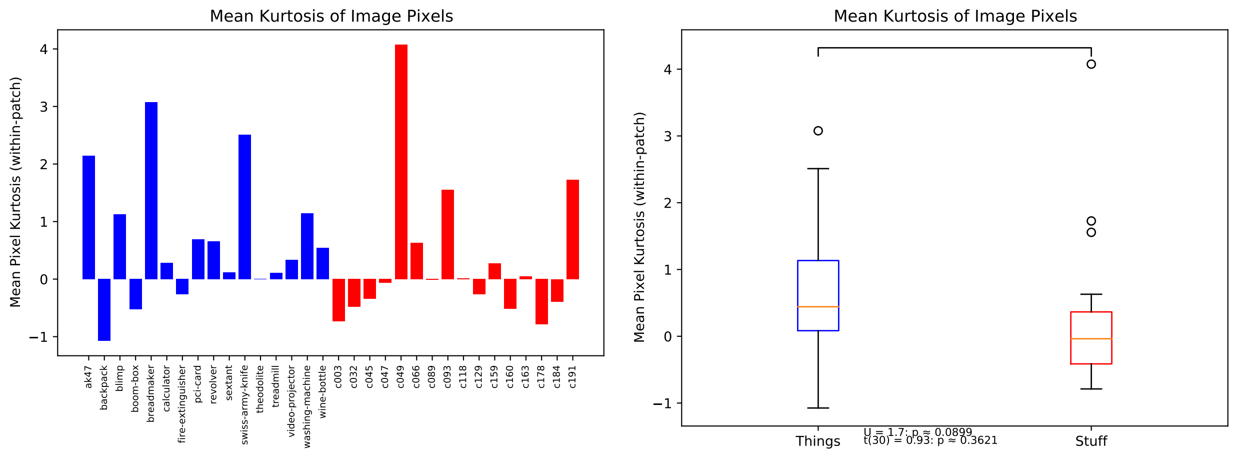


Figure 16: Per-class average mean gray level kurtosis.

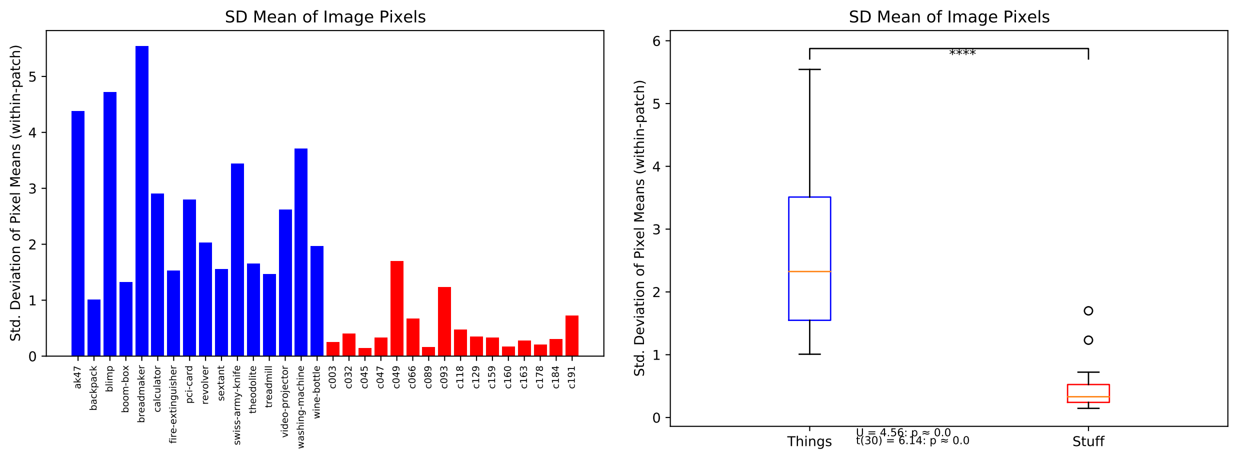


Figure 17: Per-class standard deviation of mean gray level.

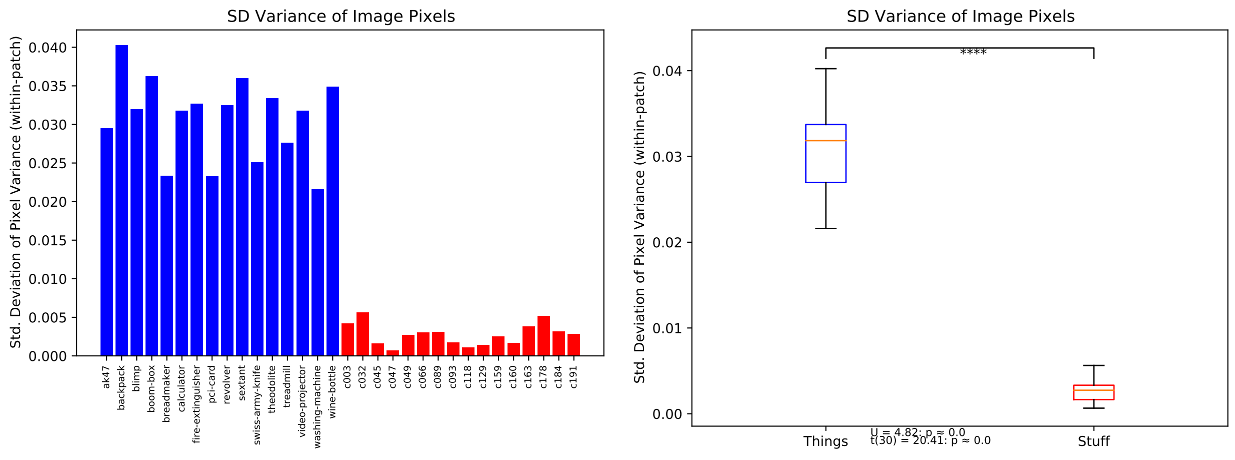


Figure 18: Per-class standard deviation of gray level variance.

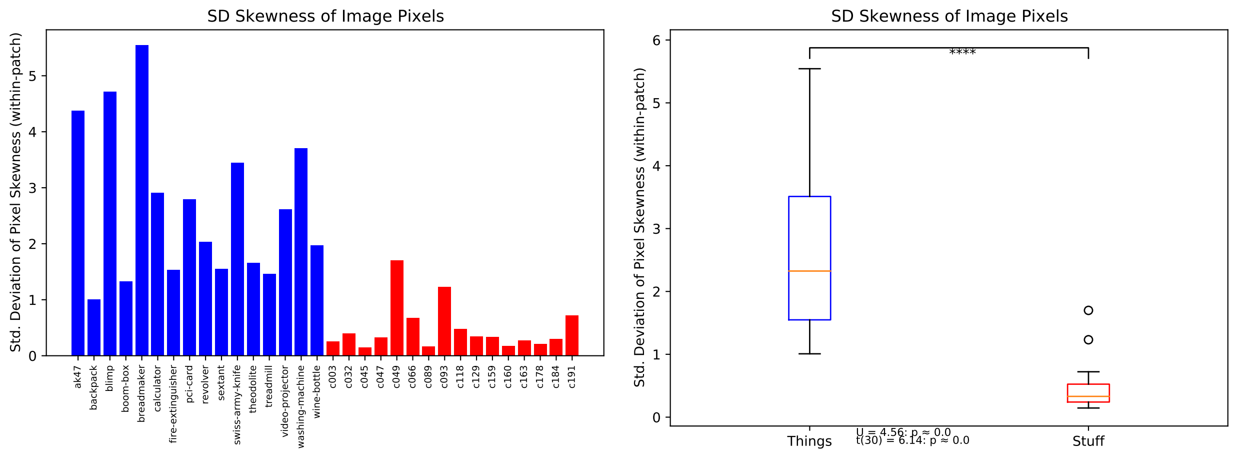


Figure 19: Per-class standard deviation of gray level skewness.

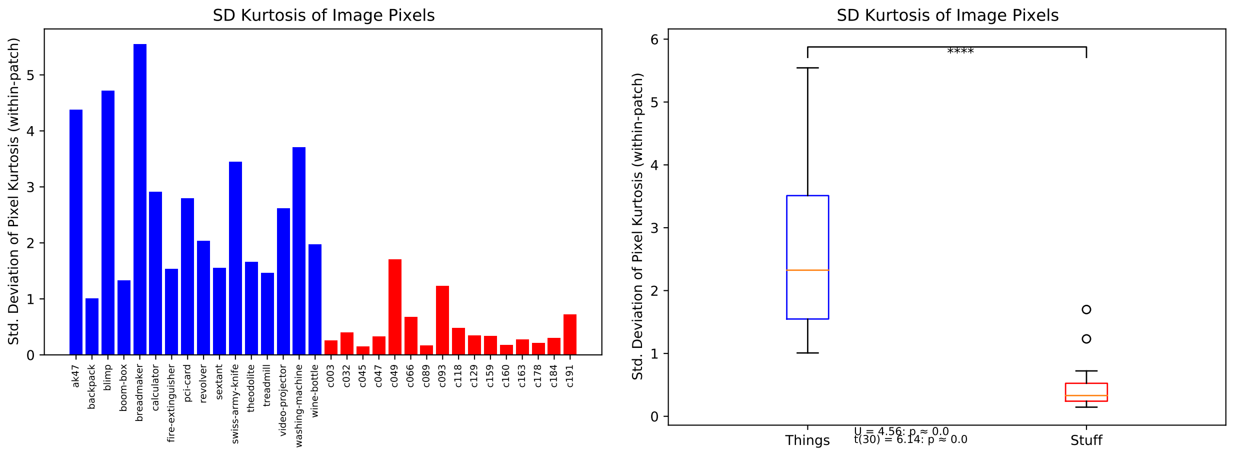


Figure 20: Per-class standard deviation of mean gray level kurtosis.

0.1.2 Hypothesized ease of stuff synthesis and recognition, favored statistical test for comparing Thing vs. Stuff class characteristics

The overriding primary hypothesis of this dissertation is that Stuff patches will pose less of a challenge for recognition and synthesis than Thing patches. This may equate also to the promiscuity of Stuff recognizers trained on only their own class, since during training they will encounter less varied data than the Thing classes, which have more varied instances. It is expected that Stuff-specific synthesizers will demonstrate higher competence at an earlier stage of training than Thing-specific synthesizers (precociousness), on a qualitatively-assessed basis. Cumulatively, this reflects the title of the investigation "stuff's cheap, things are expensive". It is expected however that semi-objective quality estimation methods (e.g. the Inception Score) involving an external oracle network will prove systematically biased against assigning elevated quality scores to Stuff, since most networks in the convolutional neural network literature have been trained on exclusively Thing classes.

The secondary hypothesis is that Stuff and Things will prove regularly separable in the features used in primitive recognizers, and in the models and judgments of deep convolutional recognizers, visually demonstrating the Things vs. Stuff dichotomy.

Throughout this report, statistical significance results are shown, in some places primarily as a secondary consideration to subjective visual analysis (this unusual methodological orientation is explained in 2.2.3). The classes and the per-class averages do not represent the sort of individual and independent responses of human subjects, so the p-values themselves are not even as valid as they usually are in psychological research. Always where a statistical significance is calculated it is between a proper vector or a flattened matrix of numbers pertaining to Thing classes and Stuff classes. The test statistic is calculated based on assignment to the two groups based on Thing or Stuff status. The hypothesis test is

p-value	Annotation
$p < 0.05$	*
$p < 0.01$	**
$p < 0.001$	***
$p < 0.0001$	****
$p >= 0.05$	

Table 1: Statistical significance thresholds for star annotation.

always the Wilcoxon rank-sum test [177], also known as the Mann-Whitney U test as Mann and Whitney more fully described it and the U distribution [112]. The implementation used (`scipy.stats.ranksums`, [170]) does not implement the tie or continuity correction that sometimes appears. The rank-sum test is used in place of, for example, Welch's t-test, because it is a nonparametric method – we cannot assume that all of the Things vs. Stuff measurements are normally distributed. In addition to rank-based methods seeming somehow appropriate thematically in light of the *sub*-ordinal statistic proposed in Chapter 1, rank-based methods are somewhat more robust to outliers relative to traditional hypothesis tests. The development of Bayes factor type renovations to the Wilcoxon tests has been a shockingly recent development [165] and so these methods are avoided because they are not yet in common currency.

The plan of attack for diagnosing a pervasive Things vs. Stuff dichotomy in recognizers is tripartite: two extremes of recognizer sophistication will be presented in Chapters 1 and 2 (i.e. some of the very simplest recognizers, working very close to the features will be presented first, and then some of the very most intensive recognizers currently practical, which also touch on synthesis, will be presented second), and then the behavior (as opposed to the raw performance) of the extremes will be compared using the manifold-learning based analysis methods presented in Chapter 3.

Chapter 1

Things vs. Stuff in Primitive Filterbank Recognizers

Statistical texture recognition using banks of linear filters and the gray-level statistics of their filter response distribution of has been studied intently for at least the last forty years, with object recognition that is both broad and fine-grained only becoming practical in the last ten. Typically, filterbank based image recognition has involved choosing a set of filters (traditionally, steered [51] edge detectors and combinations of Schwartz-like, or symmetrically falling off, functions), convolving the patch being classified with each filter, and then taking a moment-based histogram statistic (e.g. the mean) for each distribution to use as a per-filter feature for classification.

The abstraction of having filter "channels" analogous to color channels is long-established, dating back at least to the work of Campbell & Robson in 1968 [24] on Fourier composition of simple grating textures and the measure of contrast-sensitivity functions therefrom. Richards [135] explicitly described the hope that a small number of "texture primaries" could be discovered to account for most of human texture discrimination, founded on the observation that there were many effective texture “metamers” (perceptually indiscernable but source-distinct textures) among the early digitized and noise textures studied on some of the early computer displays, and so possibly dimensionality reduction and discrimination techniques from colorimetry could apply.

Because the level of orderliness of biological visual systems is incompletely known, it pays to consider algorithms, data structures, compression strategies, and signal transformations that at first would appear to be implausible and worse-performing compared to computer models with well-behaved and studied, idealized components. How might the brain pass around very approximate information about signal distributions in a convenient form? What level of statistics is really needed to perform easy texture recognition tasks – do marginal (intrafilter) and very low order (means, and comparing only very few filters) statistics suffice? How far can you torture probability information and still derive useful results? From a computational neuroscience of vision perspective, these are questions still needing to be completely answered.

In resource-critical contexts (such as differential and interplanetary remote sensing) and on very easy tasks (homogeneous texture recognition) we might like the complexity of a classifier and the volume of retained input data to be very low compared to the prima facie complexity of the imagery. Particularly in geospatially-correlated similar image search, where it could be desirable to have a very simple and parallelizable process perform a first pass over a mammoth dataset to flag up patches for more careful processing, it could be seen as acceptable to sacrifice even a considerable degree of accuracy. A space probe distant enough to have very slow accurate transmission rates back to Earth is in a position to capture much more imagery in acceptable resolution than it can send over its lifetime – especially when fastest transmission endangers that lifetime and shortens the window of useful scientific work. A real-estate startup or participant in a building detection and labeling competition might want to avoid the cost of storing large and complete sets of data. A geospatial intelligence agency with enough capability to quickly image patches of the planet might want to quickly focus on suspicious developments or look for wreckage faster than they can conventionally pull down imagery and classify it with state-of-the-art accuracy – even if, once the data is appropriately stored, inference through large and well-trained networks would be fast. In other areas of pattern recognition where the image data is likely to be very overcomplete for the limited purpose of a classification, such as for detecting almost-identical copies of video or audio data, it pays to be able to move across a large amount of signals data and gain some rough knowledge based on very few samples ruling out a match: it may take almost no information to diagnose a merely abused signal from an entirely alien signal.

Early texture analysis approaches on practical datasets like those created from subsets of imagery from the LANDSAT survey used ad-hoc statistical approaches based on classification using the distributions of arbitrary measurements of images, such as the Fourier power spectrum ring integrals, the pixels themselves, means and variances of the raw input, local neighborhood and path sums, sustained gray-level run lengths, whole image entropy measures, and so on ([176] provides a prototypical example of early kitchen-sink style discriminating features on LANDSAT in the mid-1970s).

Models of texture discrimination by 1990 [111] shared an emphasis on maintaining a bank of multiscale edge detector based linear filter kernel patches (typically rotated Gabor wavelets or Differences of Gaussians (DoGs)) which are convolved with the input image to produce filter responses images (forerunners of the "feature maps" or texture channels discussed with CNNs). First-order statistics or moments (e.g. single-filter means, as a proxy for "activation" of a neural unit sensitive to a filter pattern standing in for its "receptive field") of the distributions of these response images were used as reduced-dimensionality feature vectors for classifiers. The quality of texture discrimination then depended on how the intrinsic difficulty of the classification task interacts with the choice of the filters. A filterbank which is sufficiently large and well-chosen for the task should perform well; this thinking motivated much research into collecting well-tuned filterbanks, mostly as rotated, contrast-adjusted, and mixed combinations of simple primitive shapes serving as the filters. Modernly, following the deep learning [62] “revolution”, filters are learned for the dataset in a multiscale, layered approach by stochastic gradient descent directly instead of being engineered from mathematical or practical intuition. The fact that early layers tend to specialize similarly for naturalistic target images and that those early layers are likely edge detectors can be viewed as a partial vindication of the old fashion.

In 2003 Varma & Zisserman [168] posed the question “Are Filter Banks Necessary?”, arguing that progress into texture classification in the 1990s was based not on picking good filterbanks (c.f. the classifier of Leung & Malik 2001 [104]) but using more filters, considering their joint statistics, and including the distribution itself as a feature (allowing the comparison of distributions by divergence statistics). One successful family of classifiers they proposed [169] combined the conventional machine learning techniques of vector quantization, k-means clustering, and chi-squared divergence between histogram “models” which are constructed by greedily picking the most informative dimensionality-reduced indices to perform statistical “texton” matching using a volume created by stacked filter maps.

Concurrent with skeptical research into how little domain-specific and carefully-motivated statistics were necessary for recognition work with texture, research into parametric texture synthesis was attempting to find the minimal information needed to reconstruct textures from statistics. Heeger and Bergen 1995’s [74] perturbative parametric texture synthesis procedure used Laplacian image pyramids (Burt & Adelson 1983 [22]) to optimize a noise image into a multiresolution histogram match with the statistics of the texture to be synthesized. The algorithm depended on a coarse-to-fine approach where “subband” histograms created by convolving low and high resolution versions of each patch with a small set of steered filters were matched first at the top of the pyramid between noise and reference images (with decorrelated gray-levels for each color channel being separately matched), and then the pyramid was collapsed. Portilla & Simoncelli 2000 [132] presented a variation on this approach where a texture was represented durably as a point in a 710-dimensional space. The individual features were derived from a 4-orientation, 4-level complex-valued steerable pyramid representation, and were meant to be dataset-agnostic, measurable for all naturalistic textures. These parameters included: marginal statistics (skewness and kurtosis of lowpass reconstructions, variance, min, max of raw gray levels, and the variance of the highpass band); autocorrelation statistics of lowpass images, coefficient-magnitude statistics (autocorrelation of magnitudes of subbands, cross-correlation of subband magnitudes with same-scale oriented copies, and cross-correlation of subband magnitudes with coarser orientations); and "cross-scale phase statistics" (cross-correlated real parts of coefficients with complex coefficients between all orientations at coarser scale).

Portilla & Simoncelli's ablation-study type approach involved the authors demonstrating that removing certain of these statistics would cause dependable degradations in the quality of resynthesized images (patches which were analyzed to yield a feature vector and then synthesized from pyramid matching these statistics). With marginal statistics removed, the resynthesized textures were globally and locally lacking in contrast; with autocorrelation constraint statistics removed, straight lines become warped; without magnitude correlation constraints, local regions twist and randomly mix, and without the cross-scale phase constraints, orderly elements in the texture become mixed and blurred. The specific procedure for matching these statistics itself had to be specially determined as simultaneously optimizing them would not yield usable textures. Around 15 years later, the parametric synthesis model of Gatys et al. 2015 [56] improved upon this without the benefit of this feature engineering approach by using an already-trained 19-layer convolutional neural network (VGGNet) and the Gram matrix (in this case, the sum of inner products of all feature map pairings within a layer). The CNN outputs are used to compute Gram matrices at each level for the texture to be resynthesized and a parallel network computes a layerwise counterpart for the white noise image being optimized. A cost related to the comparison of Gram matrices is propagated back to correct the image being synthesized. The output textures are of an impressive quality that handles orderly textures (such as brickwork) without the muddying and bending artifacts seen in the Portilla and Simoncelli model, but with as many as 852,000 parameters (vs. the 3000 or so used for the color extension of the earlier model). Ustyuzhaninov et al. [163] revised this model quickly by showing that a single network using random filters and one layer can generate a Gram matrix and the network can be trained using backprop to produce comparable results. If the feature maps encompass a variety of kernel support extents (filter sizes), performance is about as good as Portilla & Simoncelli and Gatys et al. using responses from 1024 filters (but these are randomly initialized).

As the first part of this investigation into a dichotomy of Things vs. Stuff in recognizers, the most primitive histogram statistics operating on the most uninformed filters (random noise) will be used to construct low-quality convolutional recognizers. In Chapter 2, relatively high-quality deep convolutional recognizers will be developed as a contrast to these, and inspected very deeply.

To preview this chapter:

- The process of extracting histogram features for filterbank-based texture recognition is reviewed.
- A sub-ordinal statistic with some interesting properties, the signchain, is introduced.
- An even more information-poor feature, the signchain-of-signchains, emerging from signchaining the concatenations of bijections of signchained histograms, is briskly detailed.
- A case for using Linear Discriminant Analysis as the classifier algorithm of record is made.

- Similarly, a case for using the Matthews Correlation Coefficient as the performance index instead of accuracy, precision and recall, and d-prime is put forward.
- The classification performance of LDA classifiers using mean, median, signchain, signchain-of-signchain, and full bin heights of histograms is assessed to see if there is a Things vs. Stuff difference in competency of recognition.

1.1 Filter-based histogram statistics for recognition

Histogram-based filterbank recognition is implemented by convolving each patch $p \in P$ (where P is the set of patches in the dataset) to be classified with a battery of filters $f \in F$ (where F is the list of filters in the battery) producing filter activation maps $a \in A$ (where A is the list of activation maps, ordered parallel to F). The statistical features of the distribution (which may be means, variances, or other arbitrary quantities) of interest are concatenated from each contributing filter to form a combined feature vector for that choice of statistical feature or features.

All histograms computed within this investigation could be considered "extemporaneous" histograms in that they are potentially taken over different probability spaces each time – the spaces are not guaranteed to be the same, even as they may tend to share values in common. Hardware recognition circuitry based on computing distributions and doing fast, low-information recognition operations with logical elements would likely be more suited to a histogram over a constant probability space, where the total range of realizable values (e.g. the minimum and maximum mean gray level activation) is known ahead of time; this could be considered the "preplanned" histogram. Here, only "extemporaneous" histograms are used, which may limit the utility of histogram feature recognition to be confidently and immediately implemented in fixed-range contexts and also present an issue when, for example, the entropy for empirical distributions of neural network quantities is calculated for different networks.

Histogram computation can vary so that it is adaptive [50] to the features of the data; in particular, unequal width histograms can be useful for approximately fitting idealized distributions or being convolved with 1-dimensional filters themselves to produce kernel density estimates. Here, only non-adaptive, equal bin width histograms are used. For example, the 32-bin histogram is computed simply by splitting the interval between the minimum presented and maximum presented values into 32 bins.

1.2 Signchains as sub-ordinal statistics of histograms

The signchain S of a list L of n elements is a bitstring of $k = n - 1$ elements. In the straightforward case, let E be a totally-ordered set, $E*$ the set of all finite strings composed of elements of E, and L an element of E* of length n. Then the signchain S of L is the sequence that results from evaluating the `signchain-bit` function on the tuples (L_i, L_{i+1}) for $i = 1...(n-1)$. In other words, the ith bit of the signchain is defined by the relationship between the corresponding element of the list and its immediately following entry.

$$\text{signchain-bit}(L, i) = \begin{cases} 1 & \text{if } L_{i+1} >= L_i \\ 0 & \text{if } L_{i+1} < L_i \end{cases}$$

In reality, only adjacent elements in the sequence must be comparable, which means that only a (non-strict) partial order on the elements in E and arranged in the list L is truly required.

An unbroken series of 0s in the signchain signals that for that many consecutive change-intervals the original sequence is strictly decreasing.

For instance, for the 6-dimensional vector $< 1, 2, 5, 4, 3, 3 >$, the signchain of its components is [11001].

When applied to the sequence of probabilities in a probability mass function, the signchain captures what could be called *sub*-ordinal information about the probability distribution. That is, not even reconstruction of the rank ordering of non-consecutive original values is possible (in general, except for the notable cases of a signchain consisting of all 1s or all 0s) because comparisons were only made between the current and next value.

Multiple signchains can be considered to help define a deterministic finite state machine that is an acceptor for signchain-decimated probability distributions. Many possible distributions can be consistent with a particular signchain and thus a path through the machine. Multiple signchains can be arranged in a trie (prefix tree) structure so that distributions that are signchain identical are mapped to storage associated with the same leaf node. Signchains of symmetrical distributions like the Gaussian and signchains that are strictly monotonic can be efficiently run-length encoded.

The signchain itself was inspired by the Markov language models highlighted by Shannon in [144] as “the series of approximations to English”. A second order word approximation was defined as constraining the generated stream of words by bigram frequencies, the first order by word frequencies, and the zero order by nothing, leading to equiprobable selections of words, which leads one to ask what a “fractional”-order Markov process would look like in terms of a constraint on generation that is informed by only *relative* symbol probability information.

Hierarchical summarization of binary input data by has been performed by others in the past, for example, in the hierarchical bitvectors [60] which are a simpler precursor of the van Emde Boas Tree [166] which implements an associative array using bitvector keys but also additional range data. A system of hierarchical bitvectors allows fast determination of set membership in a contiguous set by using the leaf nodes of a tree to represent collectively a set membership vector which is 1 where the value is in the set. Interior nodes of the tree summarize their childrens' set membership by taking on 1 only where at least one child has a value of 1. The binary derivative part of the signchain operation, it turns out, is also quite similar to the early stages (prior to monosequence finding) of the more involved sequential-spectrum operation of Stepien [153], proposed as an alternative to the frequency spectrum in some cases for analyzing signals with missing periodic sections.

The signchain-like idea of irrevocably throwing away information about encountered items and setting up equivalence classes of streams of symbols so that multiple realizations can map to the same irreversible summary is behind the foundations of probability in the first place. In the description (see [32]) of the method of types, a type P_s is the proportion of occurences of a symbol for some defined alphabet $\mathcal{X}$ and a defined production s. For a sequence $s = 1231233$, the type P_s is:

$$P_x(1) = \frac{2}{7} \qquad P_x(2) = \frac{2}{7} \qquad P_x(3) = \frac{3}{7}$$

And the equivalence class called the *type class* is composed of the productions of the same length drawn from the same alphabet whose symbol occurrences match the appearance frequency constraints of the type. Just as probability mass functions are information-losing acceptors for lists of data, signchains are information-losing acceptors for probability mass functions and other sequences, and induce their own equivalence classes based on which sequences conform to the constraints.

In this way it is possible for many activation maps of filterbanks to be signchain-equivalent under a particular adaptive histogram.

For a signchain representing a histogram, the most significant bit (MSB) of the bitstring is arbitrarily the first change interval from the lowest-valued to the second-lowest-valued bin, and the least significant bit (LSB) represents the final change interval between the ultimate and penultimate bin. For a signchain representing a filterbank histogram feature, the patch $p \in P$ is convolved with the filter $f \in F$ to produce the activation map a. The extemporaneous equal-bin-width n-bin histogram is taken and the $(n-1)$-bit signchain accepting the histogram is computed, comprising the recognition feature associated with the filter.

A signchain's decimal representation is the "natural" bijection of the bitstring to the non-negative integers.

For a bitstring $b_K, b_{K-1}, ..., b_1, b_0$, the decimal representation d is:

$$d = \sum_{k=0}^{K} k_i \cdot 2^i$$

The decimal representation of the signchain should be expected to be less useful to a linear classifier than a bitvector. One disadvantage is that this bijection much more strongly changes the value at the MSB end of the signchain. The lowest value bin is not inherently necessarily much more valuable than the highest value bin in a histogram.

At a loss of substantial visual interpretability, a fair permutation of the bits suited to classification could perhaps be discovered. Arrangements of binary codes with special adjacency value properties occasionally have useful applications, as with the binary reflected Gray code, which ensures that adjacent decimal values are associated with binary encodings that are adjacent in the sense that only a single bit changes. The BRGC is used in a handful of settings mostly involving minimum effort switching but especially electromechanical position encoding elements that needed to ensure that each adjacent position value update changed only one bit at a time.

The integer signchain distance Υ between two sequences $\vec{a}$ and $\vec{b}$ of equivalent length is defined to be the Hamming distance (number of components which do not agree) between their signchain representations, or, equivalently, the two bitstrings XORd with the population count (the number of 1 bits, or the binary *Hamming weight*) taken therefrom.

$$\Upsilon(a, b) = popcount(signchain(a) \oplus signchain(b))$$

The population count can be accumulated directly, which would be most natural in an integrated circuit setting where two registers carrying the signchains to be compared could be XORd and the result popcounted by an efficient parallel device. In the odd case you had the XOR already computed in decimal form, the equation for the popcount of n-bit decimal representations (along with more efficient algorithms for specific machines) is given by [175] as:

$$popcount(x) = x - \left\lfloor \frac{x}{2} \right\rfloor - \left\lfloor \frac{x}{4} \right\rfloor - \ldots - \left\lfloor \frac{x}{2^n} \right\rfloor$$

The signchain distance proportion $\hat{\Upsilon}(\vec{a}, \vec{b})$ is the "fractional" Hamming distance, or $\frac{\Upsilon(\vec{a},\vec{b})}{K}$ where K is the number of bits in each signchain. When *signchain distance* is mentioned, the proportion $\hat{\Upsilon}$ is connoted.

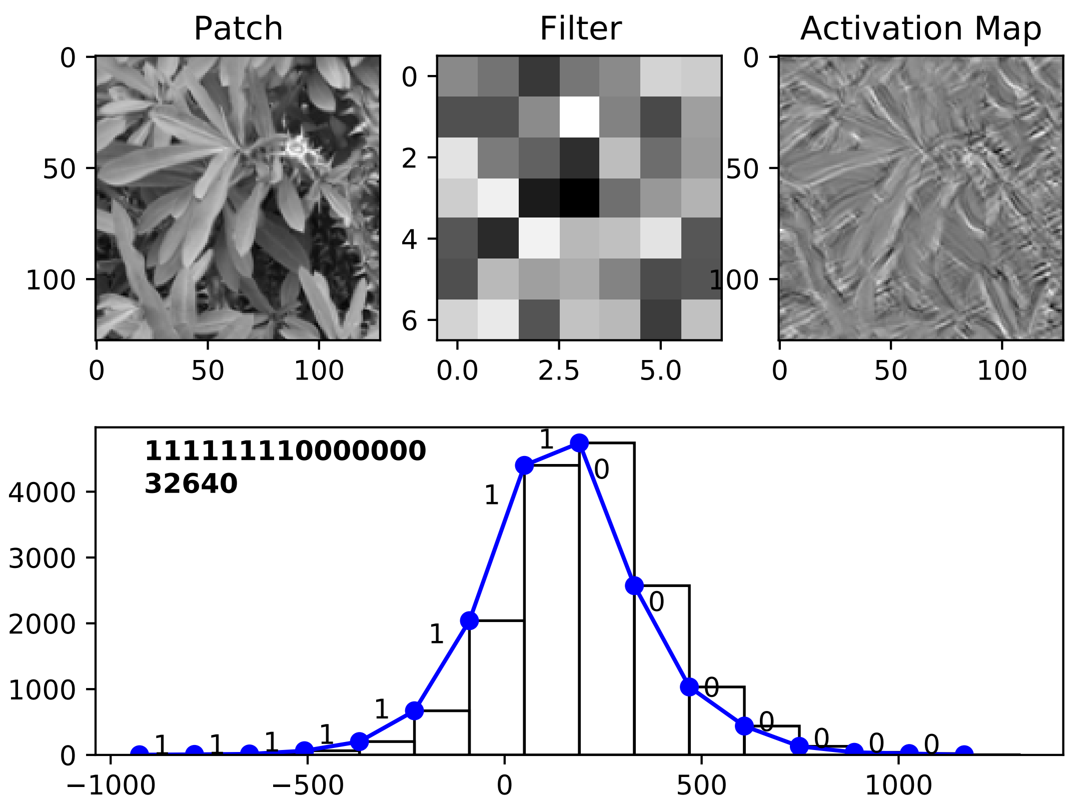


Figure 1.1: Example calculation of 15-bit signchain feature from convolution of patch with a single filter. The patch is convolved with the filter kernel, producing an activation map the size of the patch. The adaptive histogram with 16 bins is calculated. A signchain is then determined by examining the 15 change intervals and producing a sequence of bits that is 1 where the interval is increasing (but not strictly, so no change also results in a 1) and 0 where it is decreasing. The decimal bijection of the signchain bitstring used as a member of the list used to create a *signchain-of-signchains* from multiple filter signchains is also shown.

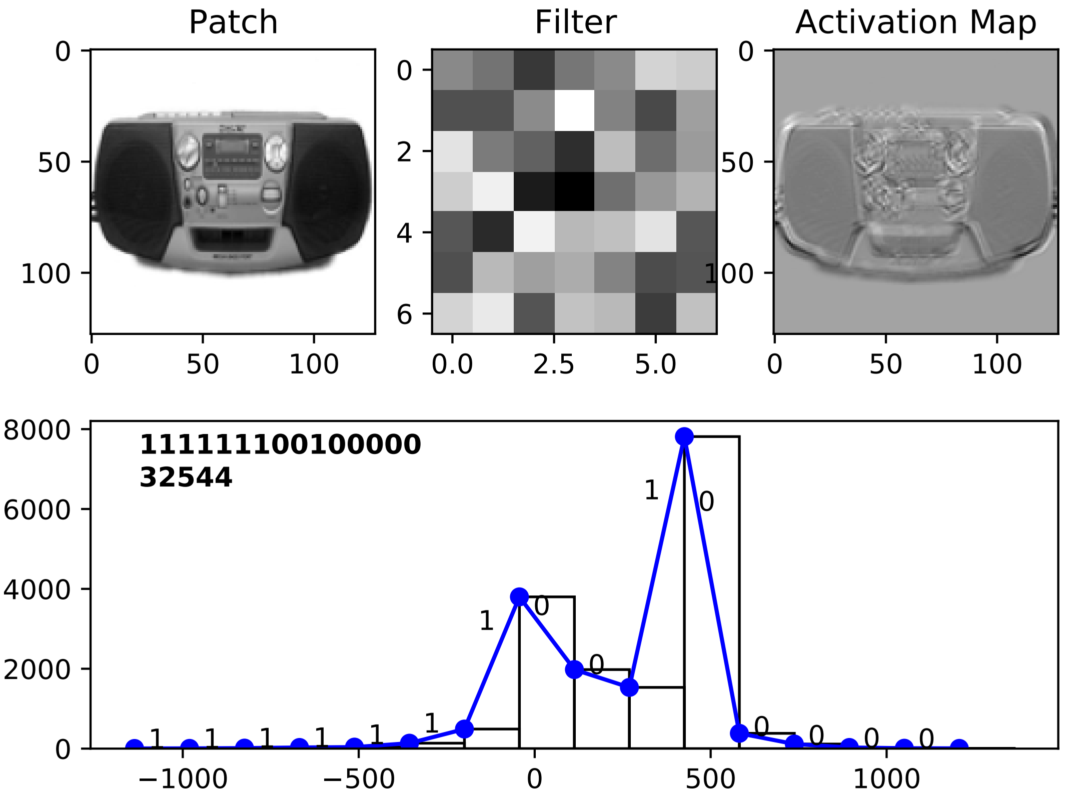


Figure 1.2: Calculation of signchain feature from convolution of patch with a single filter, using a thing patch

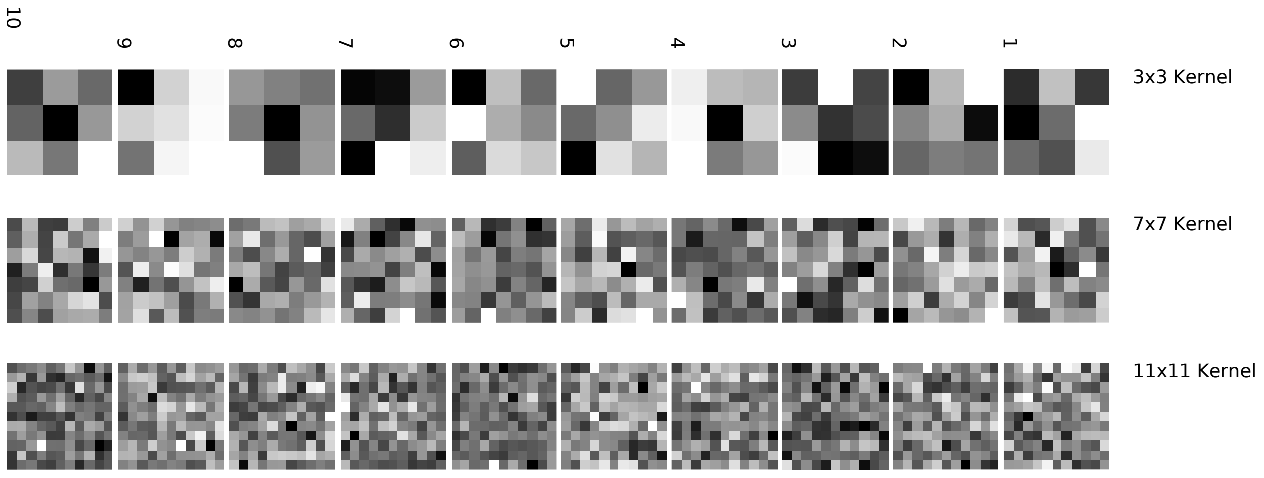


Figure 1.3: First 10 filters of each kernel size (3x3, 7x7, 11x11) used to compute histogram features

1.3 Signchains-of-signchains as even lower-information descriptors

If the signchain is somewhat like an atavistic version of the derivative on the discrete probability space that throws away a particularly extreme amount of information, the question arises as to whether there is a natural way of degrading the probability information again to produce an even-lower-information statistic. One answer lies in applying the signchain process itself to *signchain-decimate* the feature vector of signchains. To signchain-decimate a feature vector, we merely subject the list of elements in the feature vector to the signchaining operation.

In the case of a signchain, which is a bitstring, being the feature extracted for each filter, it is not particularly interesting to take this signchain of bits. With each application of the derivative on the sequence you are consuming one bit (i.e. done once, this is essentially the concatenation of binary second derivative evaluations). As an alternative, the successive conventional decimal bijections of a signchain bitstring to a decimal signchain number may conceal some order that the signchaining operation, combined with a linear classifier, could reveal. Thus this, the signchain of the list of signchain numbers contributed by each filter, is defined to be the canonical *signchain of signchains* for filterbanks.

As compared to the binary nth derivative formulation computed on the bitvector representation of the signchains, this method throws away significantly more information (over the whole filterbank), specifically arriving at one-bit per filter comparison. Neural systems benefit from summaries that can be expressed as small bitvectors because these can be most efficiently and robustly transmitted and processed by circuits, and the burden of information storage or the provision of any necessary error-correcting redundancy is lessened. Not all bits are created equal. This fact continues to be an important theme in computational neuroscience; for example, research by Stringer et al. [154] examined the eigenspectrum of V1 neuron response in mice viewing stimuli from the popular ImageNet dataset and discovered the presence of "fractal" $1/n$ power-law scaling of principal components. This suggests the importance of a few key features in early vision, and a predictable falloff in the importance of less important features, as verified by looking at the complexity of the shapes of random projections using various neural tuning curve schemes (spanning the spectrum from orthogonal "efficient coding" to highly overlapping channels) organized by eigenspectrum decay. The authors claimed that "if the variance spectrum was to decay more slowly then the population code could not be smooth, allowing small changes in input to dominate population activity". In any case, visual features that are somewhat correlated (permitting stimulus generalization) but which destroy enough low-level nuisance information (overly "fine" features, whose destruction tends to prevent widely different mappings of subjectively Same stimuli) are ideal. You can see the uselessness of overly fine features by just passing the raw image pixels to a classifier; if this was ever a highly valid choice, there would not be much of a field of texture analysis prior to deep learning. Even very impoverished information about random noise filtering of the stimulus introduces more salutary coding or hashing (for categorization) and improves upon the feature vector created by the stimulus itself.

The signchain of signchain numbers is an ambitious summarization theoretically capable of large ultra-lossy "compression" rates, similar to the extreme case of classification labels (e.g. with representational precision of a bit or a small number of bits) themselves: if you retain only 49 signchain-of-signchain bits from even a small image (128x128 pixels, 3 channels, using conventional machine word sizes) you can achieve "savings" of three or four orders of magnitude. Of course ultra-short summaries are useful only if they are at all competent.

1.4 Recognition performance of LDA classifiers using signchains and signchains-of-signchains, mean, and full histogram features

For the primitive recognizer features in this chapter, and the deep convolutional recognizer features in Chapter 2, per-class classification performance results are reported using an overall multiclass classifier that employs Linear Discriminant Analysis. The specific implementation is that of [130], using no regularization, prior class proportions from the labeled data, and a small threshold 0.0001 for testing the significance of singular values for assessing rank deficiency – the LDA process used involves a "Fisher score"-maximizing (maximally mean-separating accounting for combined standard deviation) vs. PCA-style (maximum variation direction capturing) projection implemented with a singular value decomposition that is too involved to relate here (but see the source code for [130]'s `sklearn.discriminant_analysis` in v.0.23.2 and [122], 8.6.3). The idea of the Fisher LDA score objective is to ground mean separation in terms of standard deviation separation, which is an idea that is reflected at least also in Student's t-test, Cohen's d, the d-prime discriminability measure, and the F statistic calculated in ANOVA.

Linear discriminant analysis is a form of discriminant analysis, a generative technique (see [122] for a detailed explanation) that involves calculating a decision surface based on estimating the probability of class membership for data points via their distance, or distance under a specific projection, to the centroid (aka class-conditional mean) of a not-necessarily symmetric multivariate normal distribution inferred for each class. The distance is corrected for the multidimensional variance (i.e. the covariance matrix Σ_c) of each class's Gaussian; specifically, it is the Mahalanobis distance, which is a component of the Bhattacharyya distance between multivariate Gaussians:

$$d_{\text{mahalanobis}} = (x - \mu_c)^T \Sigma_c^{-1} (x - \mu_c)$$

Inference traditionally follows Bayes' Rule (it involves evaluating a posterior based on the prior probabilities of the class and the likelihood under each multivariate Gaussian model, with a change in the class label arguments, so LDA needs no multiclass strategy like one vs. rest), so where all $\Sigma_{c \in C}$ are diagonal, the gaussian-based discriminant analysis ("GDA") is the Naive Bayes classifier. If a specific, enduring decision boundary is desired, though, a threshold for equal interclass posterior probability with Gaussian density (a term of which is the Mahalanobis distance) has to be determined. In the case where the class-conditional covariance matrices are tied, meaning that they are all identical, the equation for the posterior can be shown [122] to be simplified so that one of the x-containing terms cancels out in the consideration of other classes under Bayes Rule. This leads to a linear decision boundary between two densities being compared, vs. a quadratic decision boundary (two x terms remained; and note that the parabolic shape is not the same as an isocurve or level set of distance ellipse of the Gaussian).

The class associated with the Gaussian-based model with highest posterior probability was chosen as the predicted class.

1.4.1 Using class-specific Matthews Correlation Coefficients instead of alternatives in the presence of highly underpowered and overpowered classifiers

When considering per-class performance in a multiclass scenario we are imagining each class as having its own classifier working on a very unbalanced classification problem. Even for the small number of patches we use (approaching 2000), the vast majority of the cases an imaginary "sub-classifier" (a contrived object that we pretend is associated with the performance of a classifier on a single class) encounters are not positive. Negative cases are much more prevalent than positives. The per-class accuracy calculated in this manner can therefore be misleadingly high. When each binary "classifier" is considered, 4 contingencies can occur: true positives (declaring the target class when it is the target class), true negatives (declaring it is not the target class when it is not the target class), false positives (declaring it is the target class when it is not), and false negatives (declaring it is not when it is).

Various disciplines attempt to improve on accuracy by using performance indices that take into account some or all of these four possible contingencies. For instance, in machine learning and particularly in computer vision, the convention seems to be to report the F1 score which is the harmonic mean of precision and recall.

If TP is the count of true positives, TN is the count of true negatives, FP is the count of false positives, and FN is the count of false negatives:

$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$, where precision is $\frac{TP}{FP+TP}$ and recall is $\frac{TP}{TP+FN}$.

More vividly, in the context of an imaginary duck hunt, recall would be associated with asking the question "*Did I shoot all the ducks?*" while precision would be associated with the quite different "*Was everything that I shot a duck?*".

There may appear to be a problem with the definition of precision and recall in two cases. First, when there are no true positives and no false negatives: this corresponds to there being no missed positives and no claimed positives, which adds up to no positive cases in general. Typically a researcher does not set themselves up to encounter this eventuality when testing classification in contrived or preplanned situations, but it is eminently possible in data you might naturalistically receive involving multiclass one vs. rest classification scenarios. According to our colloquial definition above, the recall could be argued to be 1, as you claimed detection of everything there was to claim (*no duck escaped being shot*).

For some this definition might be unsatisfactory. Second, when there are no true positives and no false positives – no declarations were made in favor of the presence of the target class. Precision is 1 under our definition (*no non-ducks were shot*). Blindly implementing the conventional formula will produce undefined values that will propagate to the calculation of the F1 measure. And of course, neither precision nor recall includes the number of true negatives, so if you are given the number vs. the proportion of contingency types and not also the total number of cases, distinct behavior for differing numbers of correct rejections can go unsurveilled.

In experimental psychology, the use of d-prime (d'), associated with signal detection theory, is the prevailing convention. Higher positive d-prime values are associated with the upper-left corner of Receiver Operating Characteristic (ROC) space (see Figure 1.11a) signifying progression towards perfect detection and strong negative d-prime values towards perfect perverse detection (perfect detection if the labels were to be inverted) in the bottom-right corner. The space defined by precision and recall imparts (see Figure 1.11b) slightly different information, with the quality of the classifier on the ascending diagonal and the extremes in the top-left and bottom-right corners corresponding to all claims made of the target being valid and all valid claims being made respectively.

d' imagines the "signal" (target presence) and "noise" (target absence) distributions as normally distributed such that the distance between them can be thought of as the distance between the idealized distributions' mean parameters accounting for their variance parameters, but it is conventional to provide the index specifically as:

$d' \approx \Phi^{-1}(\text{hit rate}) - \Phi^{-1}(\text{false alarm rate})$

where Φ^{-1} is the inverse CDF (or quantile or point-percent function) of the Normal distribution, also called the probit.

The hit or true positive rate (sensitivity) is $\frac{TP}{P}$ or $\frac{TP}{FN+TP}$, and the false alarm or false positive rate (1-specificity) is $\frac{FP}{N}$ or $\frac{FP}{TN+FP}$.

d' presents several disadvantages specifically in the situation in which you have very poor classifiers intermingled with very good classifiers. The most superficial disadvantage is that like a number of inverse CDF functions, the probit function is not calculable in closed form and relies on consulting tables to interpolate a value, using differential equations, or employing occasionally opaque numerical methods. Perfect detection yields a d' of infinity, which hinders convenient visualization where finite and non-finite values must be simulatneously plotted. Simply observing a d' of infinity can also be highly misleading without further context. If the false alarm rate is 0, the probit evaluates to negative infinity – the hit rate then needs not be even close to 1 to result in $d' = \infty$. Further, quite elevated d-prime values indicating high-performance classifiers can hover in the 3 to 4 range, which is not especially obvious to those uninitiated in signal detection theory. The area under the curve (AUC) of ROC space is directly related to d-prime; [151] is a classic compendium on niceties of calculating and interpreting d', AUC, and the bias (conservative or liberal) quantity of SDT.

Clearly, a performance index that still accounts for unbalanced classifiers (thus improving upon accuracy) but whose maximum value is finite and is uniformly interpretable as good performance is desirable.

The Matthews Correlation Coefficient is produced from a number of terms from the confusion matrix, which are additive combinations of true and false in the denominator and multiplicative combinations within true and false contingency types in the numerator:

$$MCC = \frac{(TP \cdot TN) - (FP \cdot FN)}{\sqrt{(TP + FP)(TN + FP)(TP + FN)(TN + FN)}}$$

or, when a term in the denominator being zero sets the denominator to zero,

$$MCC = \frac{(TP \cdot TN) - (FP \cdot FN)}{1}$$

to gracefully avoid a problematic division by zero.

Of course, when there are no positive cases available (no false negatives and no true positives), the MCC's value should not be thought to be reflective of the classifier's intrinsic quality. Nevertheless, the MCC seems preferable to d-prime because with the correction for the denominator it is always defined and finite and does not require evaluation of the inverse CDF. Chicco & Jurman 2020 [28] further make the case against the F1 score citing the true negative problem mentioned above and specifically the asymmetry of different F1 scores resulting from flipping the positive and negative class. Here in the one vs. rest multiclass scenario there is no ambiguity as to which is the negative class, but the MCC is chosen because it retains the interpretability of perverse performance observable in negative d-primes while not retaining the worst disadvantages of d', and also gaining the familiarity of the range $[-1, 1]$ of a correlation coefficient.

1.4.2 Performance data for primitive filterbank recognizers

A constant set of 50 random noise filters (separately generated for extents 3x3, 7x7, and 11x11, see Figure 1.3) was generated according to the standard normal distribution. For each patch in the dataset, the grayscaled, single-channel pixels of the patch were convolved with each filter to produce a stack of activation, or feature, maps. For each map, an extemporaneous histogram was calculated with 16 bins, and the signchain operation was applied such that the mean activation of the feature map, the median activation of the feature map, the histogram bin heights of the gray levels, and the corresponding signchain could be extracted for each filter to serve as partial feature vectors. Additionally, the signchains were bijected to their decimal equivalents and then the list of these (over filters) was signchained itself, producing the signchain of signchains. The other feature vectors were concatenated over filters to form full feature vectors to join the signchain-of-signchain features, which are calculated above the single-filter granularity.

These feature vectors, along with the appropriate class label, were passed into separate LDA classifiers for each type of feature, for each patch. In total, there were 3x5 (kernel size x feature type) LDA classifiers. 3-fold cross-validated labels were collected from each LDA classifier. The classifiers had a grand accuracy calculated using the labels and the predictions. Per-class performance was also assessed, imagining each as a separate binary classifier for the purpose of tabulating signal-detection measures. An example of this tabulation for the well-performing 11x11 Medians classifier is shown in Table 1.1.

11x11 MEDIANS LDA

Class	TP	TN	FP	FN	Precision	Recall	d'	MCC
001.ak47	15	1643	129	83	0.1042	0.1531	0.431865	0.0671
003.backpack	76	1653	66	75	0.5352	0.5033	1.777930	0.4782
014.blimp	8	1701	83	78	0.0879	0.0930	0.357162	0.0453
016.boom-box	12	1738	41	79	0.2264	0.1319	0.876934	0.1411
021.breadmaker	68	1455	273	74	0.1994	0.4789	0.949788	0.2201
027.calculator	11	1732	38	89	0.2245	0.1100	0.797786	0.1247
070.fire-extinguisher	8	1742	44	76	0.1538	0.0952	0.657058	0.0889
157.pci-card	17	1670	95	88	0.1518	0.1619	0.622192	0.1049
172.revolver-101	5	1734	37	94	0.1190	0.0505	0.395686	0.0448
183.sextant	13	1697	73	87	0.1512	0.1300	0.610050	0.0953
208.swiss-army-knife	25	1707	54	84	0.3165	0.2294	1.130154	0.2314
219.theodolite	4	1741	45	80	0.0816	0.0476	0.288231	0.0291
227.treadmill	38	1592	131	109	0.2249	0.2585	0.784325	0.1713
238.video-projector	20	1727	46	77	0.3030	0.2062	1.124322	0.2166
239.washing-machine	7	1728	58	77	0.1077	0.0833	0.462611	0.0575
246.wine-bottle	15	1700	69	86	0.1786	0.1485	0.719526	0.1195
c003	8	1855	3	4	0.7273	0.6667	3.375753	0.6944
c032	0	1853	5	12	0.0000	0.0000	-inf	-0.0042
c045	12	1858	0	0	1.0000	1.0000	inf	1.0000
c047	6	1847	11	6	0.3529	0.5000	2.516857	0.4157
c049	8	1854	4	4	0.6667	0.6667	3.285578	0.6645
c066	10	1850	8	2	0.5556	0.8333	3.594529	0.6780
c089	12	1858	0	0	1.0000	1.0000	inf	1.0000
c093	0	1853	5	12	0.0000	0.0000	-inf	-0.0042
c118	1	1851	7	11	0.1250	0.0833	1.289232	0.0973
c129	1	1844	14	11	0.0667	0.0833	1.047699	0.0679
c159	11	1850	8	1	0.5789	0.9167	4.010102	0.7264
c160	11	1842	16	1	0.4074	0.9167	3.764910	0.6078
c163	9	1847	11	3	0.4500	0.7500	3.191347	0.5776
c178	5	1841	17	7	0.2273	0.4167	2.149078	0.3018
c184	7	1848	10	5	0.4118	0.5833	2.760688	0.4862
c191	10	1842	16	2	0.3846	0.8333	3.349338	0.5624

Table 1.1: Example signal detection calculation table for the 11x11 medians LDA classifier. The Matthews Correlation Coefficient is always finite and does not suffer as directly from edge cases in the case of no true positives (precision and recall) or perfect performance (i.e. an infinite d-prime in the case of only true positives and negatives).

3x3 MEANS LDA

Class	TP	TN	FP	FN	Precision	Recall	d'	MCC
001.ak47	2	1759	13	96	0.1333	0.0204	0.394968	0.0327
003.backpack	98	935	784	53	0.1111	0.6490	0.492957	0.1053
014.blimp	0	1781	3	86	0.0000	0.0000	-inf	-0.0088
016.boom-box	0	1779	0	91	1.0000	0.0000	NaN	0.0000
021.breadmaker	95	1158	570	47	0.1429	0.6690	0.877489	0.1877
027.calculator	0	1770	0	100	1.0000	0.0000	NaN	0.0000
070.fire-extinguisher	0	1786	0	84	1.0000	0.0000	NaN	0.0000
157.pci-card	0	1765	0	105	1.0000	0.0000	NaN	0.0000
172.revolver-101	0	1771	0	99	1.0000	0.0000	NaN	0.0000
183.sextant	0	1770	0	100	1.0000	0.0000	NaN	0.0000
208.swiss-army-knife	0	1761	0	109	1.0000	0.0000	NaN	0.0000
219.theodolite	0	1786	0	84	1.0000	0.0000	NaN	0.0000
227.treadmill	14	1432	291	133	0.0459	0.0952	-0.350617	-0.0537
238.video-projector	0	1773	0	97	1.0000	0.0000	NaN	0.0000
239.washing-machine	0	1786	0	84	1.0000	0.0000	NaN	0.0000
246.wine-bottle	0	1769	0	101	1.0000	0.0000	NaN	0.0000
c003	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c032	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c045	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c047	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c049	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c066	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c089	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c093	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c118	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c129	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c159	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c160	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c163	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c178	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c184	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c191	0	1858	0	12	1.0000	0.0000	NaN	0.0000

Table 1.2: Example signal detection calculation table for the perversely performing 3x3 means LDA classifier.

3x3 SOS LDA

Class	TP	TN	FP	FN	Precision	Recall	d’	MCC
001.ak47	3	1704	68	95	0.0423	0.0306	-0.102005	-0.0091
003.backpack	38	1519	200	113	0.1597	0.2517	0.524161	0.1106
014.blimp	5	1712	72	81	0.0649	0.0581	0.175953	0.0187
016.boom-box	3	1740	39	88	0.0714	0.0330	0.176699	0.0160
021.breadmaker	38	1579	149	104	0.2032	0.2676	0.744291	0.1602
027.calculator	6	1707	63	94	0.0870	0.0600	0.249513	0.0291
070.fire-extinguisher	3	1737	49	81	0.0577	0.0357	0.117151	0.0104
157.pci-card	27	1643	122	78	0.1812	0.2571	0.830184	0.1598
172.revolver-101	4	1691	80	95	0.0476	0.0404	-0.052433	-0.0052
183.sextant	4	1711	59	96	0.0635	0.0400	0.083229	0.0083
208.swiss-army-knife	15	1651	110	94	0.1200	0.1376	0.443310	0.0705
219.theodolite	5	1746	40	79	0.1111	0.0595	0.447811	0.0502
227.treadmill	25	1555	168	122	0.1295	0.1701	0.342007	0.0642
238.video-projector	9	1692	81	88	0.1000	0.0928	0.364404	0.0488
239.washing-machine	3	1739	47	81	0.0600	0.0357	0.135188	0.0121
246.wine-bottle	3	1693	76	98	0.0380	0.0297	-0.167876	-0.0149
c003	10	1851	7	2	0.5882	0.8333	3.639648	0.6979
c032	1	1846	12	11	0.0769	0.0833	1.103052	0.0739
c045	9	1846	12	3	0.4286	0.7500	3.160536	0.5634
c047	2	1845	13	10	0.1333	0.1667	1.490008	0.1429
c049	0	1845	13	12	0.0000	0.0000	-inf	-0.0067
c066	9	1853	5	3	0.6429	0.7500	3.457716	0.6922
c089	3	1846	12	9	0.2000	0.2500	1.811557	0.2180
c093	1	1841	17	11	0.0556	0.0833	0.976513	0.0607
c118	3	1850	8	9	0.2727	0.2500	1.952618	0.2566
c129	5	1846	12	7	0.2941	0.4167	2.275618	0.3451
c159	2	1842	16	10	0.1111	0.1667	1.414495	0.1293
c160	5	1849	9	7	0.3571	0.4167	2.376348	0.3815
c163	2	1836	22	10	0.0833	0.1667	1.294838	0.1098
c178	0	1840	18	12	0.0000	0.0000	-inf	-0.0079
c184	2	1849	9	10	0.1818	0.1667	1.619355	0.1690
c191	1	1842	16	11	0.0588	0.0833	0.998922	0.0629

Table 1.3: Example signal detection calculation table for the 3x3 signchains-of-signchains LDA classifier.

In the SDT calculation, all data is considered from the perspective of just one class at a time. The number of true positives, true negatives, false positives, and false negatives is accumulated from considering each patch's case. Then the performance measures of Precision, Recall, d-prime, and the Matthews Correlation Coefficient are computed according to the description in Section 1.4.1. Note that the MCC is defined and finite for all classes in the example table, where as d-prime reaches infinite values and precision and recall sometimes take on potentially misleading extreme values. The MCC is at its maximum value for some classes under this classifier, such as c045 and c089 which means every case was either a true positive or a true negative: no mistakes were made. Given the multiclass nature of this problem, the number of true negatives and true positives is necessarily very unbalanced, which is why binary-judgment level accuracy was not reported.

The full signal detection table for two more classifiers, the 3x3 Means classifier and the 3x3 Signchains-of-Signchains classifier are reproduced as Table 1.2 and Table 1.3. They represent respectively, the worst performing classifier and an intermediate case. The means-based classifiers performed very poorly. This could be the result of a bad interaction between the means features and the LDA classifier, but given exploratory classification with a linear SVM and QDA, it is probably mostly due to the unsuitability of mean activation in the face of using random noise, rather than principled, filters. The LDA does seemingly institute a bias towards conservatism, since the mean LDA classifiers report an MCC of 0 often, and this MCC is due to often issuing no positive declarations at all.

In Figures 1.4 through 1.8, the per-class MCCs are shown for each filter kernel size, grouped by feature type. Within types the results are grossly similar to the eye, and so filter size is not nearly the factor feature selection is. The means only perform above zero for a few Thing classes. For one in particular, a negative MCC is observed. This is the trend towards perverse detection, which is worse-than-chance performance.

For all remaining features, the per-class MCCs are systematically, but not strictly, higher for the Stuff classes as a group. This comports with the hypothesis that Stuff will be easier to recognize, in general, than Things on account of its increased describability by simple texture statistics that are stationary as you move about the image.

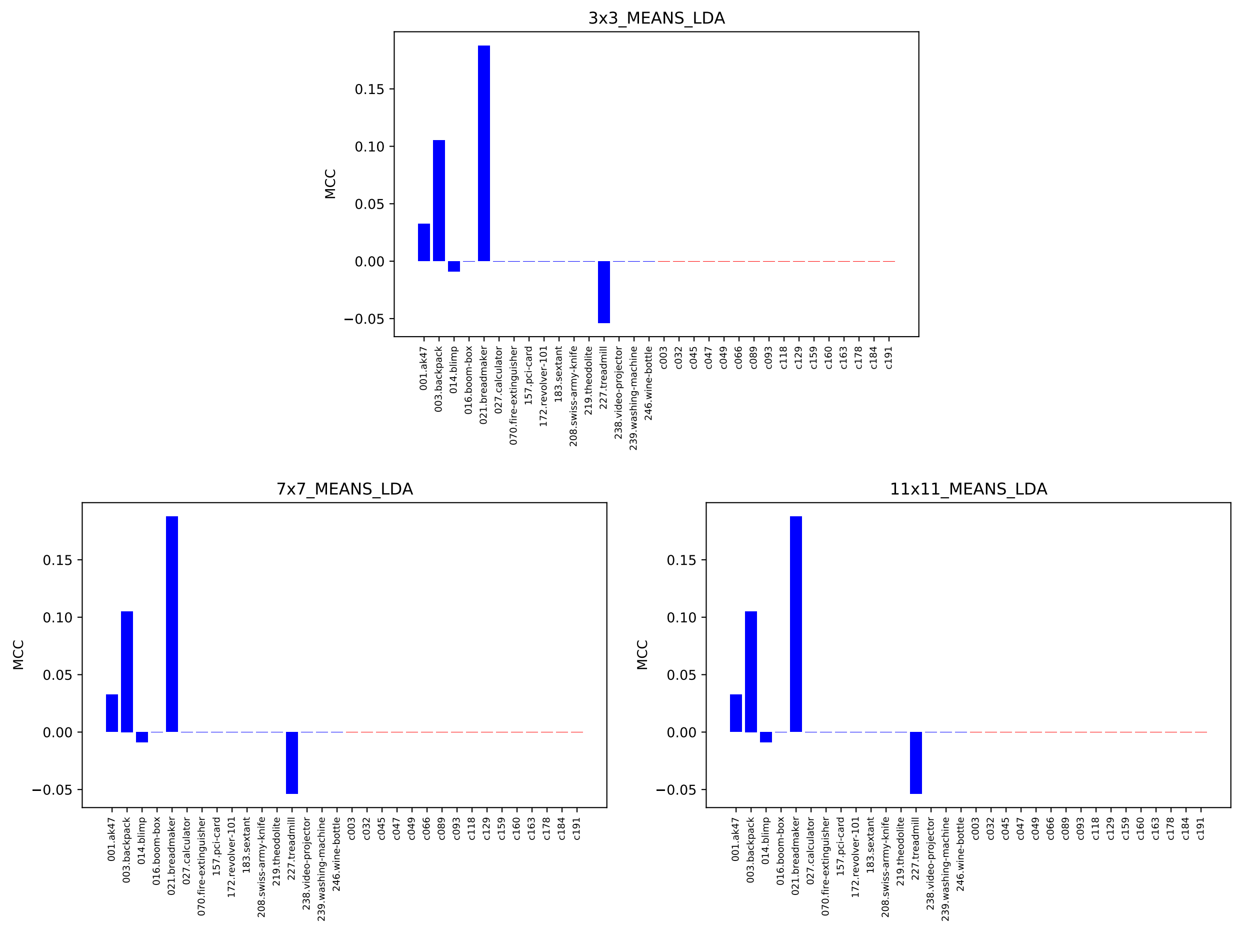


Figure 1.4: Per-class Matthews Correlation Coefficients of LDA classifier using Means, for a) 3x3, b) 7x7, c) 11x11 kernels

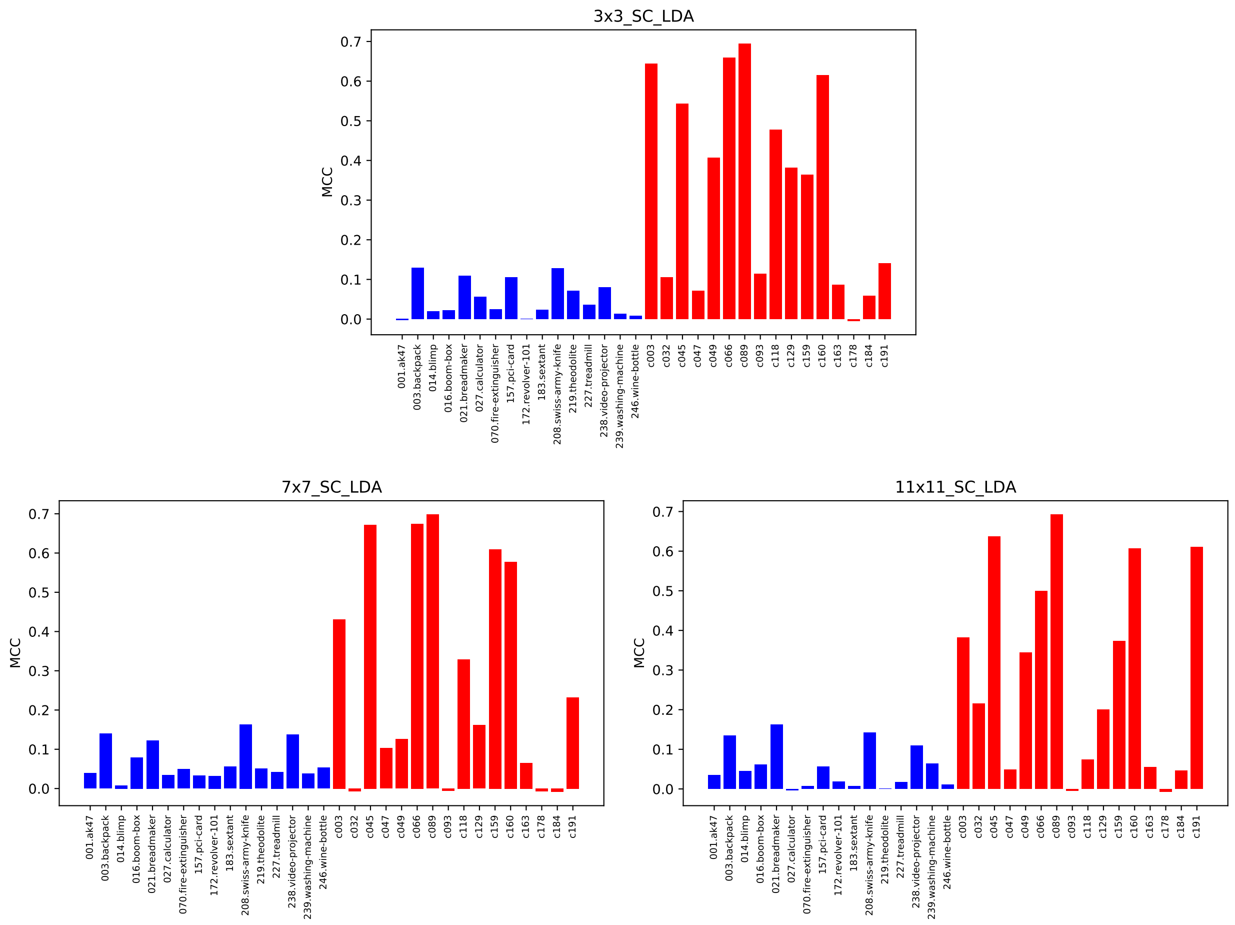


Figure 1.5: Per-class Matthews Correlation Coefficients of LDA classifier using Signchains, for a) 3x3, b) 7x7, c) 11x11 kernels

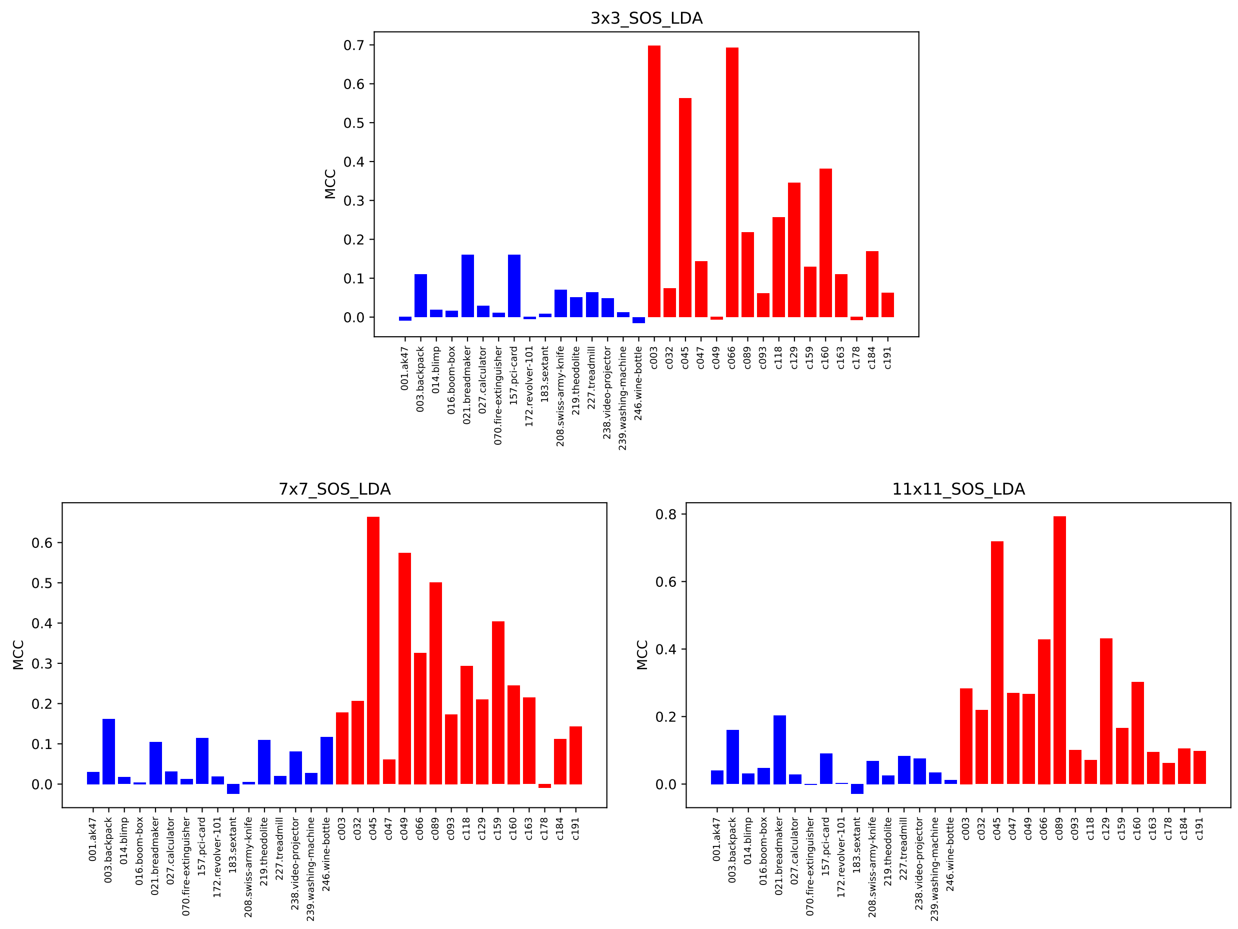


Figure 1.6: Per-class Matthews Correlation Coefficients of LDA classifier using signchains-of-signchains, for a) 3x3, b) 7x7, c) 11x11 kernels

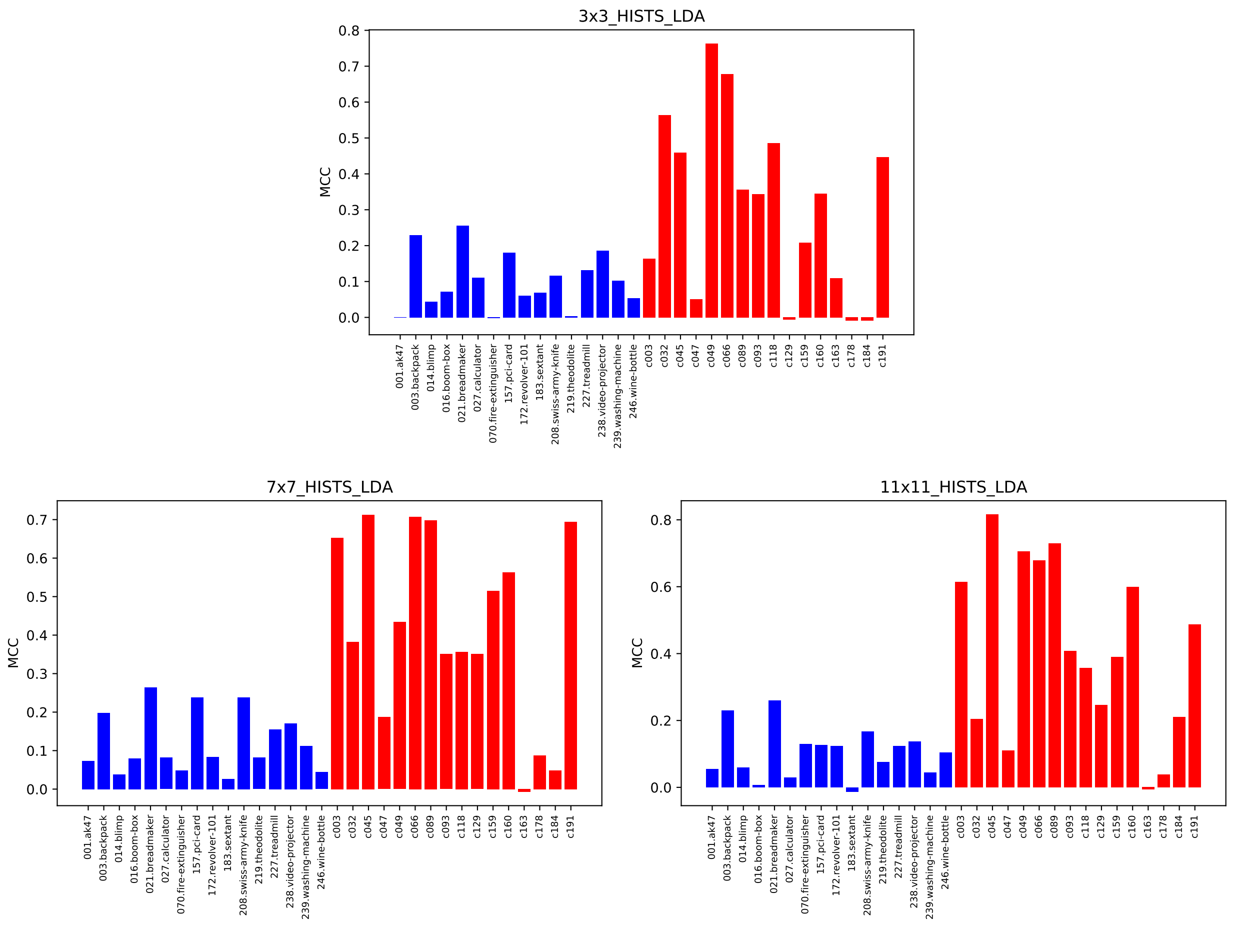


Figure 1.7: Per-class Matthews Correlation Coefficients of LDA classifier using full histogram bin heights, for a) 3x3, b) 7x7, c) 11x11 kernels

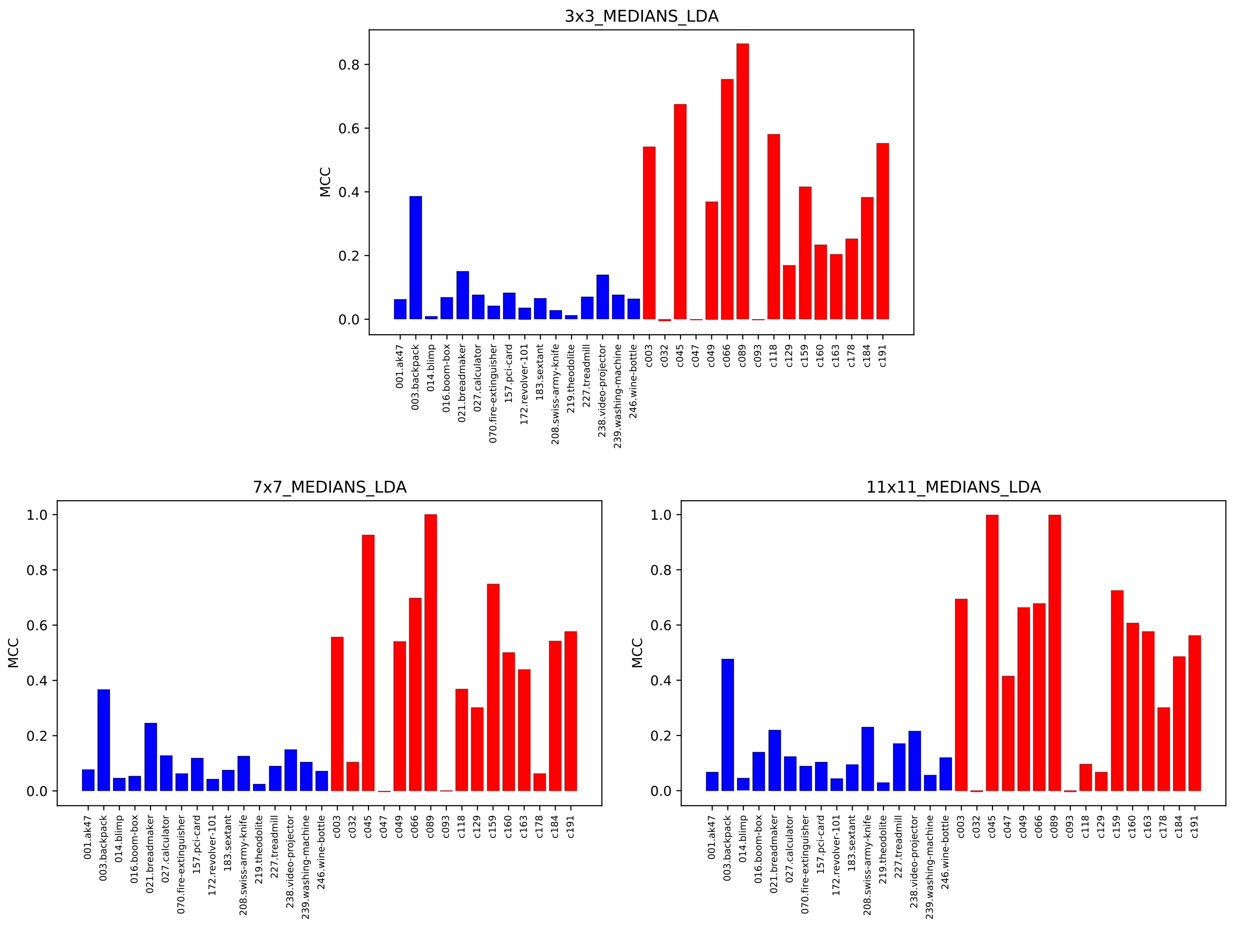


Figure 1.8: Per-class Matthews Correlation Coefficients of LDA classifier using medians, for a) 3x3, b) 7x7, c) 11x11 kernels

The medians classifiers occasionally reach the maximum MCC score of 1, but only for the larger filter sizes, 7x7 and 11x11. Signchains, signchains-of-signchains, and the full bin heights all reach MCCs of 0.7 or 0.8, again for larger kernel sizes.

The Reciever Operating Characteristic space is based on the true positive rate and the false positive rate, and since the false positive rate is necessarily low for competent classifiers such as the LDA on a sufficiently multiclass problem, ROC space is mostly uninformative to examine. Figure 1.9 shows the Receiver Operating Characteristic space plot for classes in the best-performing 11x11 Medians classifier. In this case, the classes (discrete points in ROC space) stay at a very low FPR and rise up the TPR axis, with a discontinuity at very mediocre performance (0.15-0.4) and a range of classes elsewhere in the interval $[0, 1]$. The Things classes for this classifier fan out somewhat between the Stuff classes and the line-of-chance performance. The upper-left corner in ROC space represents an approach towards perfect detection, with high or infinite d-primes and MCCs approaching 1 in this area.

In the more modestly-performing 3x3 signchain-of-signchains classifier's ROC space in Figure 1.12a, the Stuff classes are more concentrated in the poor performance area along the TPR axis that was unpopulated for the 11x11 medians classifier. Also, relative to it, the Things classes are rotated towards the line of chance, and some things classes fall very slightly below the line, towards mild perverse performance. In the very poor 3x3 Means classifier's ROC space in Fig. 1.11a, nearly all classes from both Thing and Stuff metacategories are concentrated at the origin, which represents no false positives and also no true positives (negative, or Different, judgments only). The few exceptions to this are a few Thing classes, some of which inhabit the middle of space above the line of chance (these points get closest to the upper-right hand liberal corner than any others), and one (which we know to be the treadmill from the MCC plot in Figure 1.4a) below the line of chance.

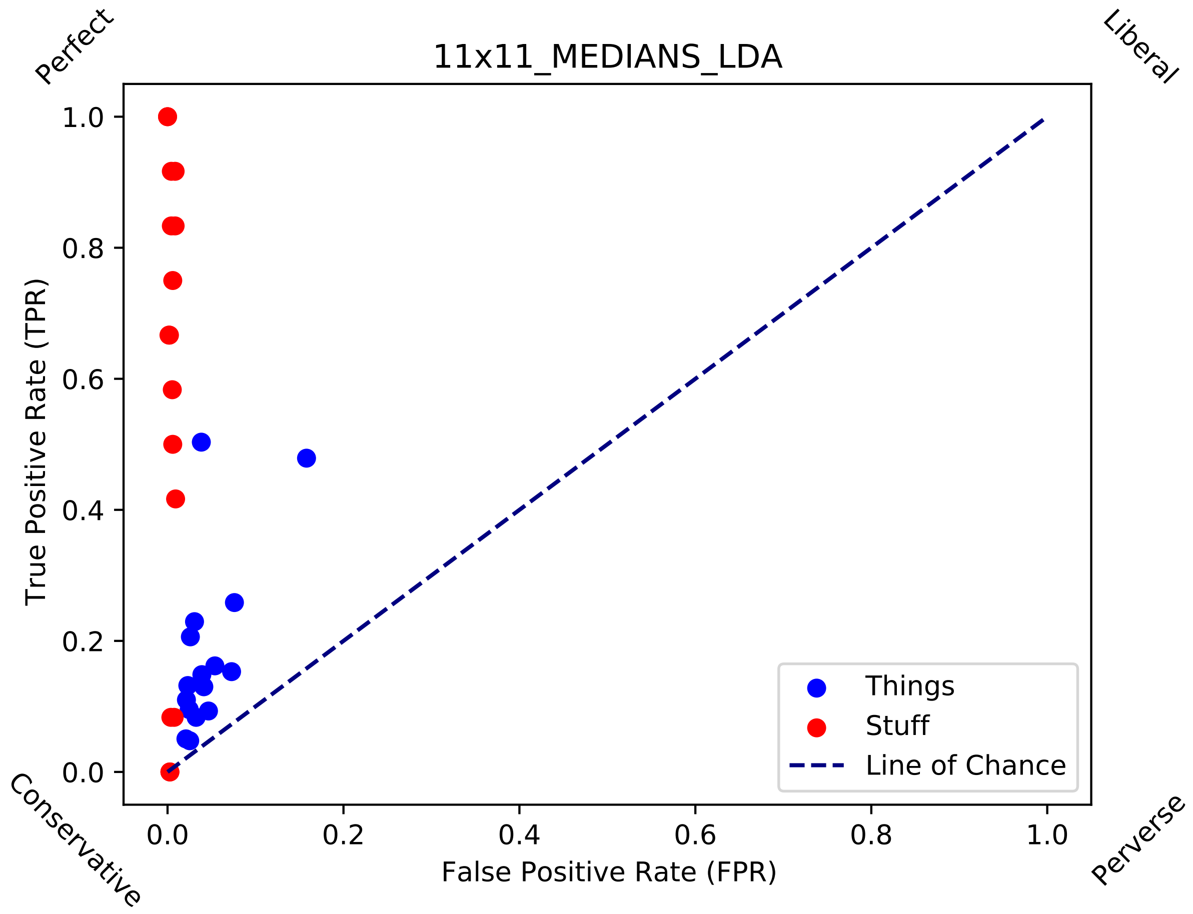


Figure 1.9: Receiver Operating Characteristic plot showing where classes (points in ROC space imagined as one-vs.-rest classifiers) fall in terms of hit rate and false alarm rate for the well-performing 11x11 medians of filterbanks classifier. Things classes stay mostly near the line of chance bisecting the space. Stuff classes range in quality of classification but mostly avoid issuing false positives. None of the classifiers are very liberal, nor are any in the corner approaching perfect perverse performance (perfect classification if labels were flipped).

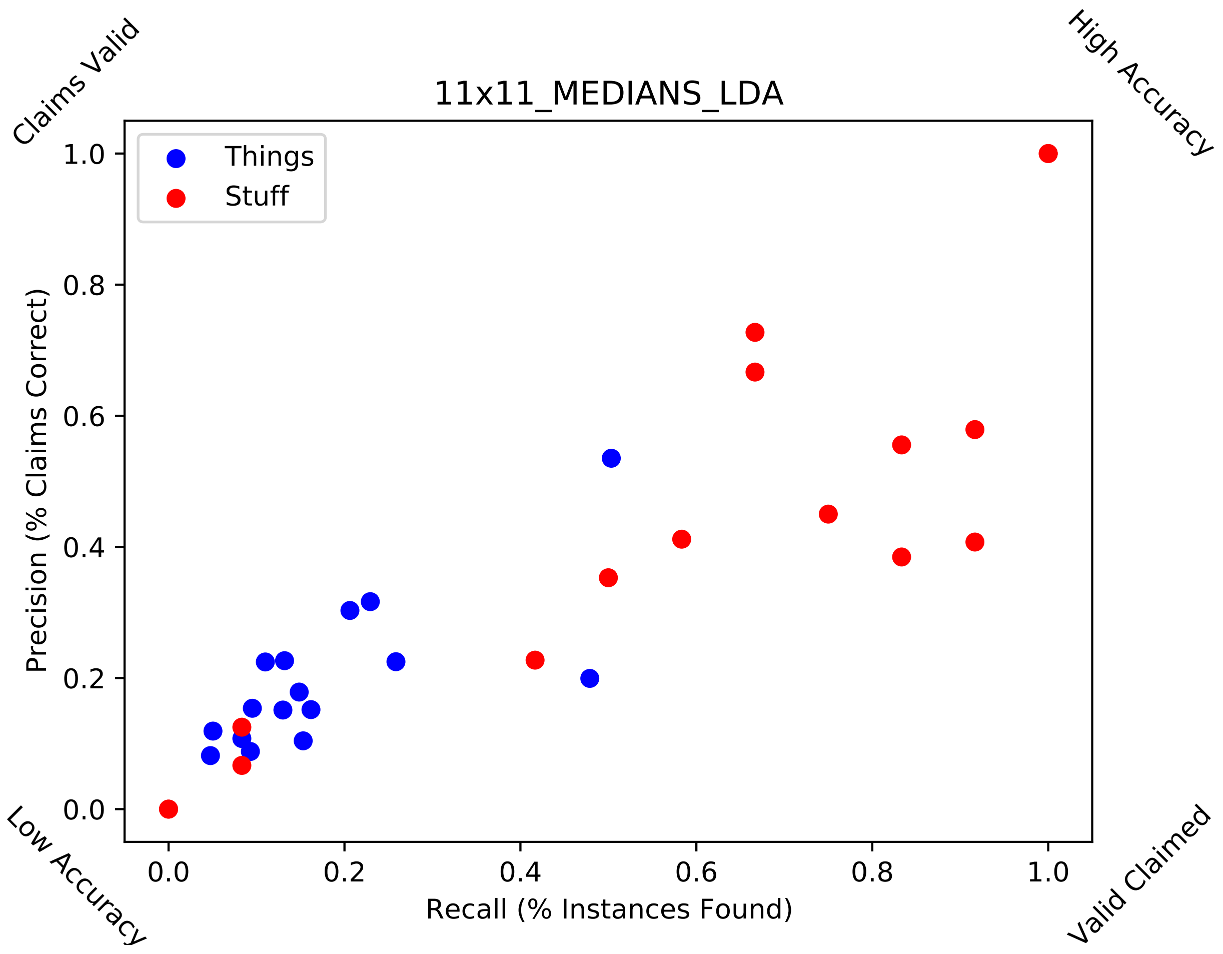


Figure 1.10: Precision/Recall plot showing where classes fall in terms of precision and recall for the well-performing 11x11 medians of filterbanks classifier. Things classes have low precision and low recall, while Stuff classes have intermediate precision and recall. Stuff classes make claims of Same that are more valid on average and have made a greater percentage of the possible valid claims.

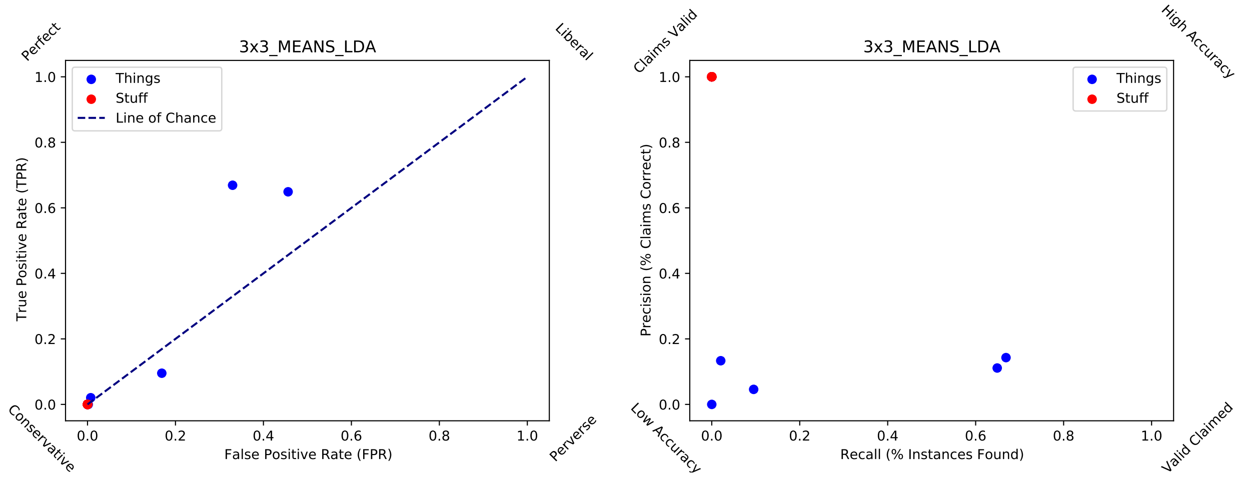


Figure 1.11: ROC and PR plots for the very poorly performing 3x3 filterbank means classifier. The LDA classifier elected to make very few claims of Same at all, so nearly all classes are overplotted at the origin in both spaces. One class did notably worse than chance. The failure of means classifiers across kernel sizes, presumably due to convolutional outliers given the noise construction of the filters, suggests that means are not at all appropriate for *random noise* filter classification.

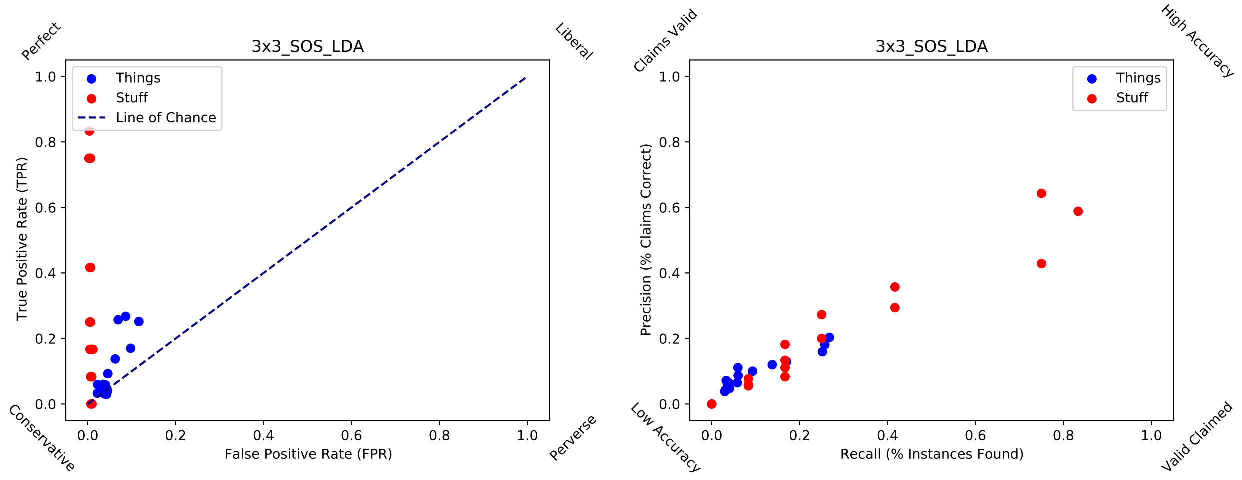


Figure 1.12: The 3x3 filter kernel signchains-of-signchains classifier was the worst performing non-means based classifier, but it provides a more valid comparison with the best-performing classifier. Compared to it, fewer classes approach perfect performance and there is less separation between stuff and things.

To see this data another way, Figure 1.10 shows the Precision-Recall plot for the well-performing 11x11 Medians classifier. Stuff goes farther along the diagonal towards high accuracy (high recall and high precision) while Things classes mostly stay in the regime of quite low recall and precision. However, they are seen to be joined by some Stuff classes. For the mediocre signchains-of-signchains classifier, the Precision-Recall plot (Fig. 1.12b) shows that only a few classes of Stuff are in the reasonably high (0.75 to 0.9) range of recall (percentage of found instances) and intermediate Precision (0.4-0.6). The rest exist closer than for the best classifier to the Things classes, in the bottom left corner. For the poor means classifier, the Precision-Recall plot (Fig. 1.11b) is close to useless. A few Things vary mostly on recall with some reaching just shy of 0.65 recall but less than 0.2 precision, arguing that the claims they make of Same are false ones. Stuff is visible at the upper-left corner of perfect precision but zero recall, meaning simply that none of the Same instances were found and all of the claims were speciously correct (as none were made).

In Table 1.4, the grand accuracy (merely tabulating the correctness proportion of predictions) is shown for each classifier. The accuracy is the mean accuracy of those obtained from the 3-fold cross validation. The accuracy is also shown among just the Things and just the Stuff. Note that the accuracy for just the Stuff is systematically among classifiers (except for mean classifiers where, perhaps because of the conservative nature of the LDA) it is effectively zero (accuracy results were rounded to 2 places). The MCCs from the signal detection tables for each classifier were averaged and are also shown in this table, subsetted similarly for Things and for Stuff. Again, these accuracies are systematically higher for Stuff as compared to Things, in many cases. MCCs were nonzero for Things (rounded to 3 places here) sometimes but only for the very poor means classifiers. These differences can be compared to see which are the classifiers with the greatest change in mean Thing MCC vs. main Stuff MCC – the difference as viewed through the Wilcoxon rank-sum or Mann-Whitney U test is also listed. Negative values are indicative of Stuff being easier than things. The most negative U statistic is that of the 11x11 signchains-of-signchains classifier,

Classifier	ACC	ACC(things)	ACC(stuff)	Avg. MCC	MCC(things)	MCC(stuff)	U(MCC)	p-val
3x3_MEANS_LDA	11.18	12.46	0.00	0.008	0.016	0.000	0.302	0.763
3x3_MEDIANS_LDA	18.18	14.72	48.44	0.230	0.085	0.374	-2.789	0.005
3x3_HISTS_LDA	17.91	15.85	35.94	0.205	0.101	0.309	-2.073	0.038
3x3_SC_LDA	13.10	11.03	31.25	0.193	0.052	0.335	-3.392	0.001
3x3_SOS_LDA	13.16	11.38	28.65	0.144	0.046	0.243	-3.015	0.003
11x11_MEANS_LDA	11.18	12.46	0.00	0.008	0.016	0.000	0.302	0.763
11x11_MEDIANS_LDA	24.22	20.38	57.81	0.316	0.140	0.492	-2.789	0.005
11x11_HISTS_LDA	18.50	15.49	44.79	0.257	0.103	0.411	-3.279	0.001
11x11_SC_LDA	12.67	10.49	31.77	0.176	0.054	0.298	-2.676	0.008
11x11_SOS_LDA	14.33	12.34	31.77	0.165	0.054	0.276	-4.070	0.000
7x7_MEANS_LDA	11.18	12.46	0.00	0.008	0.016	0.000	0.302	0.763
7x7_MEDIANS_LDA	21.39	17.40	56.25	0.286	0.111	0.460	-2.940	0.003
7x7_HISTS_LDA	20.37	17.52	45.31	0.271	0.121	0.421	-3.354	0.001
7x7_SC_LDA	14.12	12.10	31.77	0.179	0.067	0.291	-1.922	0.055
7x7_SOS_LDA	13.58	11.44	32.29	0.160	0.052	0.269	-3.882	0.000

Table 1.4: Summary of filterbank features LDA classifier recognition performance. Total accuracy is shown, followed by accuracy only among things, accuracy only among stuff ground truth cases, and then Matthews Correlation Coefficients averaged over the classes. Subset averages for things classes and stuff classes follow, then the Mann-Whitney U/ Wilcoxon rank-sum test statistic (negative favoring better performance on stuff, positive favoring better performance on things), and the associated p-value.

even though it is not the classifier with the highest accuracy and average per-class MCC (the 11x11 medians). This table also lists the associated p-values for the Wilcoxon test statistic. The U statistics are negative and the p-values below the conventional significance threshold ($\alpha < 0.05$) for well over half of the classifiers assessed. A negative U but insignificant p-value ($\alpha = 0.055$) was found for the 7x7 signchains classifier, and of course, for the degenerate means classifiers with small positive U (things easier than stuff), the p-values were high and extremely insignificant ($\alpha \approx 0.763$).

Based on these results, there is a significant recognition difference between Thing and Stuff patches in the very most primitive filterbank convolutional recognizers, such that Things appear more complex than Stuff in terms of recognition difficulty, as would be expected from knowledge of the history of filterbank-based recognition and synthesis, where in particular a significant amount of time (decades) had to elapse before Things could be competently synthesized.

Chapter 2

Things vs. Stuff in Deep Convolutional Recognizers

In this Chapter, we will see that a greater competence for working with Stuff is not cleanly limited to uninformed, primitive recognizers; it also appears in synthesis, and in extremely large, intensively trained models. A class of synthesis-controlling recognizer networks will be shown to retain somewhat weaker general classification ability in groups when a simple transformation of their output is adopted, and the divergent development of these networks when they are trained with an ordinary objective function for recognizers will be studied to measure the model performance and model constituency differences wrought by specializing networks for adversarial synthesis control. The statistical homogeneity of Stuff input will be seen reflected in these networks as well as several other deep networks, with implications for mixing Things and Stuff in a unitary visual processing system being naturally suggested.

Is image synthesis more expensive in terms of model maturity (training time, or number of epochs) for Things as opposed to Stuff? There are several varieties of image synthesis. For homogeneous textures, there are nonparametric and parametric methods: parametric models (e.g. [132]) are able to synthesize entirely unseen textures from a reasonably small set (e.g. 3000) of statistical measurements by perturbing a noise image to match the statistics, while nonparametric methods often merely sample and stitch together image blocks for an extended or removed region depending on a nearest-neighbor type query of patch similarity [45]. As mentioned in Chapter 1, relatively recent developments [163] have been able to use the Gram matrices of convolutional neural networks with merely randomly initialized ("random noise") filters to constrain targeted statistics and even to directly implement the synthesis of regular texture. Irregularities with textures involving regular patterns often arise among certain classes of texture in parametric models – these *synthesis failures* are what researchers historically looked for in the feature engineering process of stepwise inclusion and exclusion of low-level image statistics to be matched.

For objects and scenes, only generative adversarial networks (GANs [63] [61]), improved feverishly over the last five years, have proved highly and generally effective. GANs involve learning an effective latent space for producing convincing hallucinated images that look like the training set, and are generally not studied for their synthesis failures because, in part, they like the other purely discriminative CNNs reject the feature engineering approach and directly learn filters through deep learning.

There are many variants of GANs, but they all involve two networks, the discriminator and the generator, which are jointly and adversarially trained. A discriminator network attempts to accurately produce a confidence or judgment that an image is not from the training set. A generator, which can be viewed as a decoder, the latter, production-oriented half of an autoencoder network, attempts to produce plausible fakes from input noise in its code layer (often denoted z). Traditionally, the discriminator's confidence is only used through training to cultivate the GAN's generator, not for classification. This is presumably for two reasons: 1) classification has already been practically "solved" by CNNs which are designed and perform well at the problem, and 2) the confidences are only on the basis of "real" (from the training set) and “fake” (not actually from the training set). Given the fact that a discriminator must manage this bipolar outcome which is distinct from that of a class membership judgment, it remains to be seen how effective a GAN’s discriminator could be in classification. Arguably, GAN research is very rarely focused on anything other than multiclass data because the emphasis is deservedly on broadly-competent, high quality synthesis, explorations through the code space, and reverse correlating relationships between generated images to relationships in low-dimensional projections of that space. Correspondingly, there has not been emphasis on determining the residual power of GAN-origin-discriminators, which is an important consideration because single-class GANs by construction only see their target class. One conjecture that is possible but is not yet sustained by evidence is that such a network could have to internalize that class more deeply because synthesis is a harder problem than mere recognition.

The GAN’s unique ability is to convincingly learn to fill a latent code space (z) such that nearly all patterns produced by the downstream generator network ($G(z)$) plausibly originate from an input training set. An ordinary compressive or denoising autoencoder (see [62] for a discussion of these and variational autoencoders) forms an information bottleneck that is believed to help the network learn an efficient representation of the training set, compressing inputs into the lower-dimensional code space. However, even for a well-trained autoencoder, there is very little guarantee that each point specified by a specific pattern of code layer activations corresponds to an image that looks like the training set. Adversarial training of the generator in a GAN bakes into the objective function a reward pressure that fosters output image plausibility across (with enough training) all possible configurations of random input noise. An autoencoder is merely trained on the inputs in the training set, but the GAN’s generator inputs derive from a (usually static and Gaussian) noise distribution that the network must adapt to while still fooling the discriminator – the difficulty of this task constrains outputs to realistic productions under “proper” training.

Despite the general effectiveness of the "GAN loss" based on minimax or adversarial training of the generator and discriminator networks, GANs have undergone many renovations to improve the quality of generated images. Without these changes, the synthetic samples still look quite muddied (e.g. beds blending into windows and walls and pillows in the LSUN [182] Bedrooms dataset). "High-resolution" (greater than 128 pixels square) GAN samples were first popularly trialed using a Laplacian Pyramid GAN approach [38], which also introduced the conditional GAN or CGAN (input noise is augmented with a special label input that tells the network which class to produce hallucinated samples from). The DCGAN (deep convolutional GAN) merely takes the GAN architecture modeled on an MLP-based autoencoder and applies the convolutional-volume based innovations of the CNN [133] – nearly all image-producing GANs of interest are fundamentally DCGANs for this quality purpose. The DCGAN additionally demonstrated that linear interpolations between the code layer representations of random rendered bedrooms produced relatively smooth transitions between the resulting hallucinated bedrooms produced by the interpolations. More surprisingly, visual analogies were demonstrated, showing that deep convolutional GANs can learn a low-dimensional embedding that tracks an interesting manifold where linear arithmetic operations implement analogies: the coordinates in code space for an average smiling woman could be subtracted by neutral woman's coordinates and added to neutral man's coordinates to see an unseen smiling man produced by the resulting coordinates. Specific manifolds within that space of interest could be approximated linearly or nearly linearly by finding two images on the sides of a continuum (left-facing and right-facing face, all other attributes held constant) and mapping the expanse between them.

Quality (diversity and convergence) concerns drive much research into GAN architectures and training tricks. One somewhat inelegant solution type reminiscent of mixtures of experts or gated models used in ordinary deep learning research to combat catastrophic forgetting (negative transfer inflicted on a task by learning another, even closely related task) is to create an ensemble of generator networks in the GAN (e.g. AdaGAN [161]) and make their output an additive mixture with new generators added to the mixture when synthesis problems are detected.

The most exciting investigations relating to GANs involve amazing applications of multi-class synthesis where details are not probed if quality and performance are not helped; they are not stuffy methodological investigations probing the exact details of why, for example, “things” are statistically distinct in some ways as compared to “stuff”. Pix2pix’s “image-to-image translation” [84] and CycleGAN [188] involve learning highly convincing “neural style” transformations which can turn, for example, overhead schematic map views into fake satellite views and horses in a video sequence into zebras. The cycle of CycleGAN refers to the “cycle consistency loss”, which measures how much detail is lost in a reconstruction of a roundtrip from the original style to the target style and back again – if the transformation is very well learned (or trivial) the restored image should exhibit only modest degradation. CycleGAN enables unpaired translation, meaning that it does not need to learn a unidirectional transformation on matched pairs where the transformation has been demonstrated by human intervention. The StackGAN [185] involves two stages of GAN and incorporates word embeddings. On the Caltech-UCSD Birds 200 dataset, the StackGAN is capable of *text-to-image translation*, hallucinating novel bird photos of good resolution from short text descriptions that do not just resemble images from the training set – a problem that GANs can sustain related to mere memorization of training data.

However, if deep learning methods are to be used with decreasing caution in critical areas or enterprises are to plan effectively for the expensive training time of networks, the expected synthesis quality of individual image classes and the time to cultivate that quality must be reliably predictable. As a blunt instrument, exploring the Things vs. Stuff dichotomy would seem to provide a good methodological proving ground, and can perhaps even give us developmental insight into how convolutional neural network recognizers are organized through through the clarity afforded by a dichotomy.

One hypothesis is that, as in the case of recognition, there is a performance gap that disfavors object vs. material synthesizers – training time to produce good quality fakes should be longer for objects, and using the discriminator from the GAN for classification (if this is even feasible in the first place) should produce poorer results for longer (i.e., with less training time). Presumably, the same pattern might even be seen when the GAN has to synthesize image couples (two images concatenated together), especially for images of dissimilar or broad (all dogs vs. a specific breed of dog) classes, in which case this technique can investigate the relatedness and breadth of image categories, inducing a novel kind of similarity graph.

While there are numerical quality measures for synthesis that are partially objective, they do exhibit the potential to be biased, especially in a way related to the Things vs. Stuff dichotomy. Absent these, a study of synthesis quality of Things and Stuff class-specific networks given a common amount of training time with which to strive towards competence will have to be highly qualitative. Consequently, there is motive to see whether discrimination can be used as a proxy for synthesis quality. First, of course, synthesis has to be competent for Things and Stuff. The discrimination process during early training steps prior to the point of generator fakes that seem reminiscent of training data to humans might be unstable and uninformative of later quality. But at the point of bare competence, does the computational fluency of Stuff recognition seen for primitive filterbank recognizers in Chapter 1 reemerge for deep convolutional recognizers?

There are three broad classes of outcome: 1) neither metacategory is competently discriminated by GAN-origin-discriminators, 2) both Thing and Stuff classification performance is at ceiling, which is uninformative, 3) there is intermediate performance that hits neither floor nor ceiling. Then, only in the eventuality of the third class of outcome can we detect a possible Things vs. Stuff performance difference. A priori one might expect that even without much hyperparameter selection and architecture finding an ordinary CNN discriminator would be at ceiling performance since even at relatively shallow depths (3-5 convolutional layers) they were transformatively effective in computer vision, but that might not be the case for a GAN-origin-discriminator. And if GAN-origin-discriminators are weak classifiers, certainly they can be improved from their individual performance somehow by being bound up in a conventional (i.e. non-synthesis-oriented) ensemble?

To preview this chapter:

- The DCGAN architecture used for the Thing vs. Stuff investigations is specified.
- The use of the WGAN-GP rule with no rectification is justified for reasons of overcoming a lack of diversity.
- Synthesis results for Things and Stuff are shown at early and late development, and qualitative results are discussed.
- Brief reeducation of GANs towards a single Thing and Stuff target is accomplished.
- Synthesis failure classes are presented for stuff and catch duplicate Thing networks are introduced for contrast.
- The ensemble of GAN discriminators is detailed, and the joint activation affinity matrix is motivated over graph-based analysis.
- Subtractive normalization of activation affinity is recommended based on the GANsemble's reaction to real and fake images.
- Problems with the naïve argmax classifier operating on the spectrum of class activations given synthesis failures are anticipated.
- The characteristic pattern quadrant appearance of the pairwise model parameter dissimilarity matrix is introduced and connected to signal detection theory.

- MSE, the L1 and L2 norms, Jensen-Shannon distance, Signchain Distance, and SSIM (of the weights coerced into an image) are brought in to compare network weight similarity at each major layer.

- The "NONGAN" is introduced with ordinary cross-entropy loss yet identical architecture to the WGANs.

- Affinity matrix and parameter dissimilarity matrix analysis is repeated for the NONGAN, demonstrating early vs. late vision differences between GAN and NONGAN based on content.

- NONGAN and GAN complement dissimilarity vector analysis is used to assess when networks diverge most across Things vs. Stuff divide based on objective function.

- A NONGAN direct-discriminator of objectness (Things vs. Stuff) is strongly cross-validated using a held-out, second dataset.

- Out-of-vocabulary misclassifications using the held-out dataset are used to assess the retention of potential for strong similarity.

- Differential activation tracing of two non-target classes and a target class during training of a GAN and NONGAN is used to reveal developmental divergence of activation.

- The possibility of an implicit bias towards Things in the popular Inception Score is evaluated, and the popular alternative Fréchet Inception Distance is probed for bias.

- The effect of the Things vs. Stuff difference on late-stage filter-related neuron activations and channel occupancy is examined.

- Performance of the naïve argmax and LDA classifiers based on GAN ensemble and NONGAN ensemble class activation emissions is assessed using the Matthews Correlation Coefficient, as in Chapter 1.

2.1 Synthesis of Thing vs. Stuff patches with Generative Adversarial Networks

Before proceeding to the indirect measurement of quality through recognition, it pays to consider the actual subjective quality, and it is necessary to specify the synthesis model itself.

2.1.1 WGAN-GP-compatible DCGAN-type Model for Synthesis and Recognition of Things and Stuff

The Deep Convolutional Generative Adversarial Network architecture arrived at for the Things and Stuff investigation is described in this section, and was implemented with the high-level neural network library Keras v2.3.0-tf from Chollet et al. [29] included in TensorFlow v2.2.0, using two Nvidia RTX 2080Tis in parallel. The specific WGAN-GP learning rule implementation was the reference version provided in the Keras documentation by A.K Nain [124]. A high-level reference implmentation of neural network operations wherever possible was deemed very preferable so that these operations could be efficiently and accurately employed; no innovations to learning algorithms or special corrections were planned.

Hand tuning of parameters occurred with the video projector class, choosing the parameters including training time that subjectively seemed necessary to get decent quality samples produced within the space of a targeted maximum of 3.5 hours of real world training time.

The tensor schematic layout of the discriminator model used is shown in Figure 2.1.

The model begins with a layer of input neurons, accepting a single-channel (grayscale) image patch with gray levels on the interval $[0, 1]$.

The first stage of the discriminator passes this input forward through 5 convolutional blocks, specifically consisting of Conv2D-LayerNorm-LeakyReLU layer groupings.

Conv2D layers are 2D convolutional layers implemented with matrix multiplication. Strided convolution was employed constantly throughout the network, using a symmetrical stride of 2 in either direction, which has the effect of reducing the image extent in each convolutional block by half. A telescoping-up sequence of filter channel counts (32,64,128,256,512) was used, settled on through hand-tuning. This had the effect of taking the single-channel 128x128 input to 64x64x32, 32x32x64, 16x16x128, 8x8x256, 4x4x512 tensors progressively through the network: this prototypical DCGAN-like progression was observed to be necessary to prevent the network immediately failing to converge (producing saturated images with input equal to 1 everywhere) or consuming too much of GPU video memory (ensembles of DCGANs may easily grow to the size that constituent networks need to be spread to multiple cards or swapped on and off disk). A perceptron-like bias vector added to the layer inputs was opted for. The *filter* kernel extent was held constant, with 5x5 filters, and convolutions were configured to be padded so that output shape resembled input shape before accounting for stride.

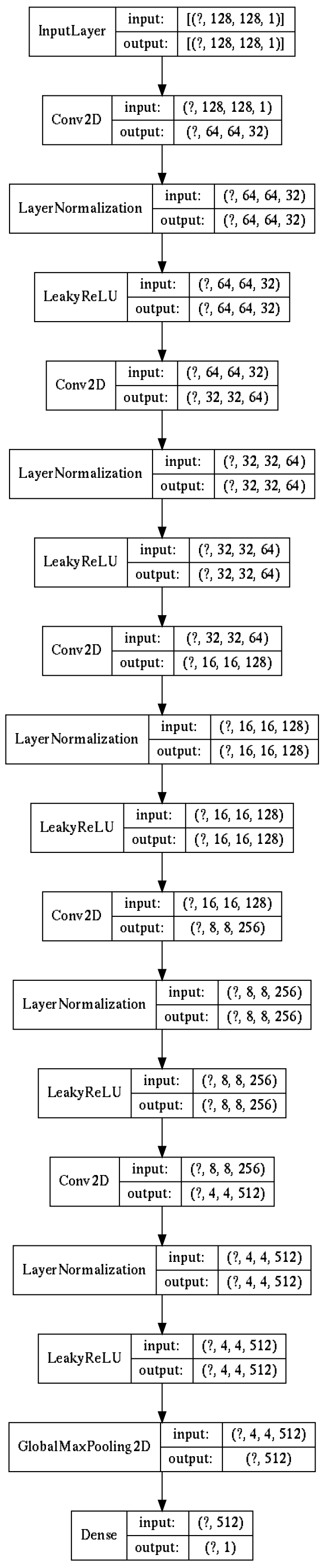
Each convolutional block subjected the Conv2D tensor to layer normalization [9] (this too was observed to be indispensable for convergence of the WGAN-GP, but not for the mode collapse experiencing ordinary DCGAN which had originally been trialed). LayerNorm adjusts the input to neurons based on normalization using the observed mean and variance of all summed inputs to neurons within the layer and computes an internal neuronwise bias and gain parameter; the stock Keras implementation was used with the default parameter $\epsilon = 0.001$.

The end of each convolutional block was a rectification with the LeakyReLU [109] function, with $\alpha = 0.2$ arbitrarily for every layer.

$$\text{LeakyReLU}(x) = \begin{cases} \alpha x & \text{if } x < 0 \\ x & \text{otherwise} \end{cases}$$

Following the five convolutional blocks, a max pooling operation is applied, which takes the maximum activation value from the network. The max pooling used was global, meaning that for each channel the global rather than a local maximum was taken.

The 512-dimensional resulting activation vector was fed forward through a final Dense (fully-connected) layer to produce a single scalar output. The discriminator in total has 4,356,289 trainable parameters.



InputLayer | input: [(?, 128, 128, 1)] | output: [(?, 128, 128, 1)]
Conv2D | input: (?, 128, 128, 1) | output: (?, 64, 64, 32)
LayerNormalization | input: (?, 64, 64, 32) | output: (?, 64, 64, 32)
LeakyReLU | input: (?, 64, 64, 32) | output: (?, 64, 64, 32)
Conv2D | input: (?, 64, 64, 32) | output: (?, 32, 32, 64)
LayerNormalization | input: (?, 32, 32, 64) | output: (?, 32, 32, 64)
LeakyReLU | input: (?, 32, 32, 64) | output: (?, 32, 32, 64)
Conv2D | input: (?, 32, 32, 64) | output: (?, 16, 16, 128)
LayerNormalization | input: (?, 16, 16, 128) | output: (?, 16, 16, 128)
LeakyReLU | input: (?, 16, 16, 128) | output: (?, 16, 16, 128)
Conv2D | input: (?, 16, 16, 128) | output: (?, 8, 8, 256)
LayerNormalization | input: (?, 8, 8, 256) | output: (?, 8, 8, 256)
LeakyReLU | input: (?, 8, 8, 256) | output: (?, 8, 8, 256)
Conv2D | input: (?, 8, 8, 256) | output: (?, 4, 4, 512)
LayerNormalization | input: (?, 4, 4, 512) | output: (?, 4, 4, 512)
LeakyReLU | input: (?, 4, 4, 512) | output: (?, 4, 4, 512)
GlobalMaxPooling2D | input: (?, 4, 4, 512) | output: (?, 512)
Dense | input: (?, 512) | output: (?, 1)

Figure 2.1: The discriminator part of the DCGAN used throughout this research takes 128x128 patches and subjects them to 5 Conv2D-LeakyReLU blocks with inner *layer* normalization, a single max-pooling layer, and then a terminal densely-connected layer that produces a single activation value

The tensor schematic layout of the generator model used is shown in Figure 2.2.

Whereas the discriminator ends with a Dense fully-connected layer prior to the output, the generator begins with a densely connected layer following the input, which is a 256-dimensional noise vector. For all investigations described, the noise vector was generated using a pseudorandom number generator targeting the standard normal distribution (with $\mu = 0, \sigma = 1$).

The noise vector is densely connected to a shallow layer of 16,384 neurons and reshaped into a tensor output with image extent 4x4 and channel count 1,024.

According to the DCGAN design, the telescoping-down sequence of filter channel counts halves each time (1024, 512, 256, 128, 64), and the extent doubles corresponding to a symmetrical "stride" of 2 in a Conv2DTranspose layer. The Conv2DTranspose (or "deconvolutional", or "fractionally-strided", see [43],[184]) layer is not a simple upsampling as by nearest-neighbor but is a dilation using a learnable kernel that capitalizes on the interaction between stride and the transpose, nevertheless enlarging the output. As in the discriminator, the filter kernel extent was held constant, using 5x5 filters, and convolutions were padded so that output shape resembled input shape before accounting for stride. Contrary to the discriminator, there was no bias layer used in the convolutional layers of the generator.

The convolutional blocks are BatchNormalization-LeakyReLU-Conv2DTranspose blocks. Batch normalization [83], a forerunner of the layer normalization that was used in the discriminator, performs a similar function but the averaging is taken over the (mini)batch per neuron rather than over the layer within a single image presentation. The LeakyReLU was configured to use the same $\alpha = 0.2$ setting used in the discriminator. Regularization via Dropout [150] (using $p = 0.1$, 0.2, and 0.5) was informally found to be not helpful and in some design attempts even inimical, so it was not used following the rectifications.

The final convolutional layer differed in that its units were specified to be rectified with the hyperbolic tangent function:

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

Note that for the discriminator, no nonlinearity was applied. This has implications for classification. The output does not immediately resemble a confidence [0,1] that can be likened to a probability.

It is conventional in normal discriminative networks that do multiclass (i.e. "multinoulli") labeling to have the rectified output of each terminal neuron transformed by the softmax function so that the transformed node activations represent probabilities:

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{c \in C} e^{z_c}}$$

([62], e.q. 6.29), where here i is the class of interest, and C is the complete list of classes.

This choice was originally made because it did not seem to forestall synthesis, and perhaps there was a chance useful information would not be thrown away or numerically obscured (especially from the point of view of planning to use a linear classifier) by forcing a nonlinearity that could be used by the ensemble.

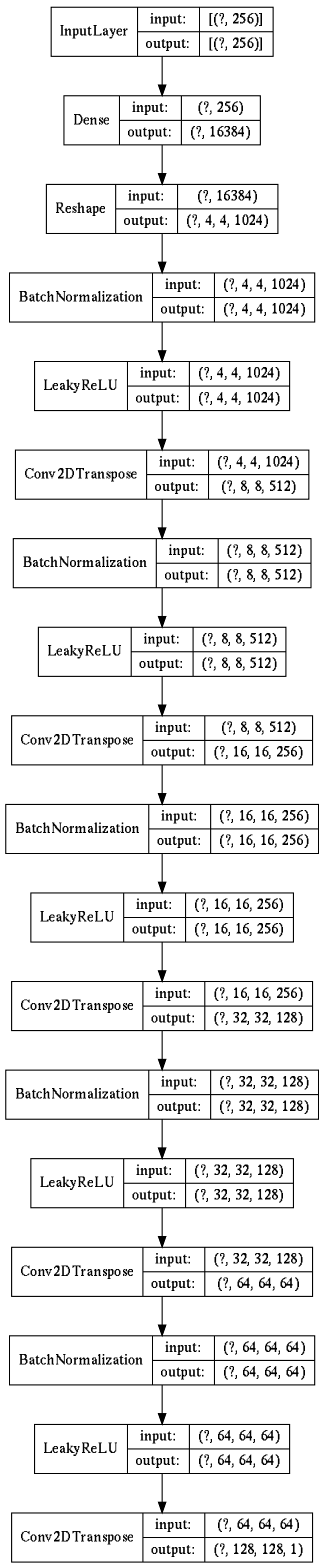


Figure 2.2: The generator part of the DCGAN used throughout this research takes 256-dimensional noise vectors and subjects them to a densely-connected layer followed by 5 Conv2DTranspose-LeakyReLU blocks with inner batch normalization to produce a single-channel 128x128 synthesized image

The conventional loss function that *could* be minimized at the end of the (binary) discriminator is the binary cross-entropy loss. The cross-entropy in general (where p and q are distributions) is

$$\mathbb{H}(p,q) = -\sum_{c \in C} p_c \log q_c$$

In the general cross-entropy loss, the prediction $\hat{y}$ and the target value y are being compared for classes $c \in C$ in the same way p and q are the different interpretations being compared. The binary cross-entropy loss is a special case that is based on connecting the Bernoulli distribution's PMF $p^k(1-p)^{1-k}$ log transformed ($k\text{log}p + (1-k)\text{log}(1-p)$) with this idea (since there are 2 alternative considerations for a true and false prediction) and the log-likelihood's relationship to cross entropy:

$$BCE(\hat{y}, y) = -y\text{log}\hat{y} - (1-y)\text{log}(1-\hat{y})$$

(see [89] for a partial derivation).

The GAN learns something a lot more complex than (even iterated) binary classification, but its loss is conceptually simple at a high level. Following closely the notation of the NIPS 2016 GAN tutorial [61], if x represents true data (real samples from the training set), D is the discriminator, G is the generator, and z is the noise that G consumes, the long-range objective of the discriminator D (if it is emitting probabilities, but see above) is to output something close to 1 when it is factually fed x and 0 when it is *actually* fed $G(z)$. The goal of the generator G is to emit output given z that makes $D(G(z))$ approach 1 as often as possible.

Because of the positive logic involved, the cost of the discriminator can be formulated by combining half the expectation of $D(x)$ with half the expectation of $1 - D(G(z))$.

$$-\frac{1}{2}\mathbb{E}_{x\sim p_{data}}\log D(x) - \frac{1}{2}\mathbb{E}_z\log(1 - D(G(z)))$$

The network output incorporating the nonlinearity is necessary to comply fully with these theoretically-defined losses. Training proceeds by mixing in fake images produced by G with real images. The generator's cost function for reasons of gradient computation (see [61] 3.2.3 for why GANs although adversarial do not actually stick with a simple game-theoretic minimax objective) focuses just on tricking the discriminator:

$$-\frac{1}{2}\mathbb{E}_z\log D(G(z))$$

In practical terms, this boils down to injecting some random noise into the generator as defined above and collecting the resulting images for a training batch. The discriminator's loss is determined by the discriminator's binary cross-entropy loss previously mentioned on the real and fake samples combined with the *correct* labels. The labels, being not strictly binary, are corrupted with some noise for easier training in the context of all of the involved nonlinearities and to leave some uncertainty. The generator's loss is determined by the discriminator's binary cross-entropy loss sustained on the real and fake samples but combined with *misleading* labels (i.e. all labels declare "this is real!"). The gradients are separately taken with respect to these losses and gradient updates are made. Conventionally, the discriminator is updated first, and the generator second.

Throughout, the networks were optimized using the popular Adam [98] method, with a parameter of 0.0002.

During training, you may observe that the quality of the fakes is not strictly related to the losses. But the losses (especially given our lack of constraint on the output of the discriminator) can inflate, and the quality of synthesis convergence will suffer. This may suggest balancing the discriminator and the generator when they get out of balance w.r.t. their losses. You can invent any number of balancing schemes (e.g. for every doubling, give the disfavored network a power of two of the doubling count in terms of updates), but, at least through the lens of casual experimentation, unprincipled ones seem destined to fail.

2.1.2 Observed mode collapse and non-convergence given low sample count prior to introduction of WGAN-GP

Initially, in the attempt to find networks with the power to synthesize things and stuff with a DCGAN architecture as above using GAN loss, convergence was slow and unreliable with increasing loss occasionally developing quite quickly regardless of terminal application of the nonlinearity. Many deep convolutional network architectures easily capable of generating MNIST [180] digits or hazy patches from the LSUN dataset popular for use with DCGANs [182] were not capable of producing even barely acceptable quality yet still "creative" (or not merely memorized) 128x128 patches of all of our data, which was presumably a combination of several factors. First, 128x128 patches were once considered "high-resolution" for GANs prior to the advent of coarse-to-fine improvement architectures like ProGAN [92]. Second, our dataset was massively sample-impoverished: sample counts of 12 or even 100 are much lower than the thousands or tens of thousands or more used in most casual demonstrations of GANs. Third, the Things data particularly is more structured than Stuff data which synthesizes fine with networks of lesser capacity and power – it could be even argued that some muddled synthesis results on RGB images that seem convincingly creative and even have the benefit of thousands of real input images actually do not vary substantially from training set samples except by errors made and decoherencies introduced chromatically.

Besides trying to prevent GAN training from diverging and making it reliable and fast, the other main objective of research into improving notorious weaknesses of unstable GANs is explicitly a prevention of the creativity-loss scenario just mentioned, but formally known as mode collapse [61]. Mode collapse is so-called because it can be observed even on data that is a simple mixture of Gaussians. Instead of synthesizing the full mixture, the GAN learns to produce distorted samples from each individual Gaussian (a mode of the full mixture). Mode collapse was certainly experienced in this project, in trying to develop GANs with the power to synthesize the Things patches. Since the Stuff patches are already defined to be quite homogeneous it is difficult to perceptually diagnose this condition in samples, although it was informally observed to be less of an issue alongside the synthesis quality being good early for many Stuff classes (with the exception of c003, the "holey" texture, which has very large sized repeating features that could be easily smeared in GAN loss syntheses). It appears that there is factually a competence hierarchy for synthesis that is, like the case of recognition with primitive filterbank recognizers, strongly categorical: some combinations of architectures, standardization tricks, and learning rules are capable of producing monochrome digits, others low-resolution homogeneous patches, and others high-resolution object patches.

A philosophical question connected with this is whether GANs are actually just faulty memorizers ([123] preliminarily investigates criteria for departing memorization), and that any profound creativity, for example in creating completely novel and convincing high-definition human faces (as with the state-of-the-art StyleGAN [93] that uses a progressively-grown ProGAN discriminator and private noise channels skipped to layers of the generator), is an example of failed memorization that just results in a useful discrepancy if the synthesis is deep enough and the features sufficiently high-level.

In classification rather than synthesis problems, something that ameliorates incompetence given low sample data count is nearly any form of data augmentation (see [147] for a readable survey), or creating new "synthetic" samples. Traditional data augmentation involves trivial transformations of the sample patches, such as flips, rotations, small translations, and contrast and whole image scale adjustments; an ideal classifier should, after all, have performance that is invariant to these considerations. For example, deep convolutional networks enjoy some translation invariance from directly pooling operations within one layer over the strided action of the included filter kernels, and target patterns can be recognized from the nonlinear combination of focal activations across layers. Many higher-level, nonlocal statistical homogeneities of the class samples might not change, or might change trivially with these transformations. Using data augmentation might make classifier networks more robust to these changes which is clearly desirable, but they might also facilitate smoother approximation of the network by gradient descent in the first place, which is of foremost and initial importance. Aside from these practical and observable considerations, elementary data augmentation by image transformations is then somewhat akin to the general statistical technique of "bootstrapped" [44] resampling-based estimates, which may come off as inherently unpalatable because you are reasoning on "made-up" data. Data augmentation for training *synthesizers* has not been well-studied; in fact, looking for research into GANs and data augmentation almost invariably results in finding many papers (e.g. [13]) which advocate *using GANs for* advanced data augmentation. Indeed, where the GAN in question has converged to produce factually diverse samples that are acceptable to a human observer as coming from the class, there is little reason to be too suspicious of these synthetic samples, and so this is a preferable source of augmented data for training vision systems.

Conceivably, even fake visual worlds could be used to train new synthetic vision systems for the real world. Object-rich video games like *Fallout 4* contain hundreds of textured 3D model assets that could be positioned into scenes automatically and "photographed" from many directions using placement markers, room templates, a random number generator, and an internal scripting language, and this approach could easily produce millions of scene samples. These samples could be used for general vision systems or for specific vision systems such as those with tactical applications relating to, for example, automatically computing an estimate of the concealment potential a scene affords from the point of a combatant trying to conceal themselves from snipers, or networks also incorporating EEG or galvanic-skin response measurements to assess the joint effect of environment and type of scare in a survival horror game. The idea of representing say schema of what belongs in a bedroom or bathroom with Hopfield network [79] style content-addressable memories is reborn somewhat in the original DCGAN [133] work which produced patches based on the LSUN-Bedrooms dataset, but GAN synthesis of what we might call sparse output problems (e.g. vectors of indexed objects and positions rather than image data) remains an important gap in the literature. To illustrate further, using GANs for automated level design where the level design is specified by dense signals or where a natural dense representation can be computed has already been studied by researchers [58] who used a GAN to generate images that could be used to create *DOOM* levels (the authors show how different image channels can be used to build floor maps, wall maps, height maps, and Things maps that place enemies, activators, and powerups within the level and that these can be automatically transformed into the somewhat unconventionally mostly geometry-based level data used in the DOOM WAD file format – see [141], Ch. 4, for an in-depth discussion). Similar research [171] has been done using evolutionary strategies refinement of GAN output generating levels for *Super Mario Bros.* – a game that was necessarily tile-based due to the design of the Nintendo Entertainment System's video hardware. It is unknown at this point if the quality of synthetic levels for object-intensive games where the architecture and landscape often are implemented as placed objects would be better using a non-convolutional GAN on the object entity data (positions and identity vectors) or if some 2D information packing scheme could be developed to optimally make use of research results derived from computing with fully naturalistic images. In any case, it seems clear that deep convolutional recognizers and synthesizers like the GAN will find ample future potential in research into closed-loop design (automated generation and subjective evaluation) of video game environments, and that this research will potentially bleed back into mainstream computational vision. These datasets may not have widely emerged for reasons of resource-intensiveness (i.e. training the synthesizers might require organizing a contest to produce 1000 houses in a level editor) or for reticience to rely on the fair use doctrine in involving commercial games so deeply in academic research. Nevertheless, the case for public use of modern complex virtual environments for cognitive science research is compelling; properly instrumented games could allow us to inspect player patterns of exploration, dwelling, looking, structure-building, or inventory-level acquisitiveness for markers or clusters of mental illness, creativity, high-level perceptual acuity, or leadership.

To return from this playful digression to the motivation for the choice of the WGAN-GP learning rule, the kind of mode collapse that was initially observed on our low-sample count (as few as $n = 12$ within some classes) dataset is presented in Figure 2.3.

Under the GAN learning rule, the architecture we detailed above suffered mode collapse of considerable severity. Mode collapse of the highest severity would be naturally defined by producing exactly the same, memorized samples. Here, it is clear that the GAN was memorizing input samples almost completely: this is best shown in the distinct hourglass-shaped front panel model boom box and the docking station style boom-box (3rd and 4th images, first row, reading left-to-right), but even the lower quality samples demonstrating more peripheral image noise have only trivial variability.

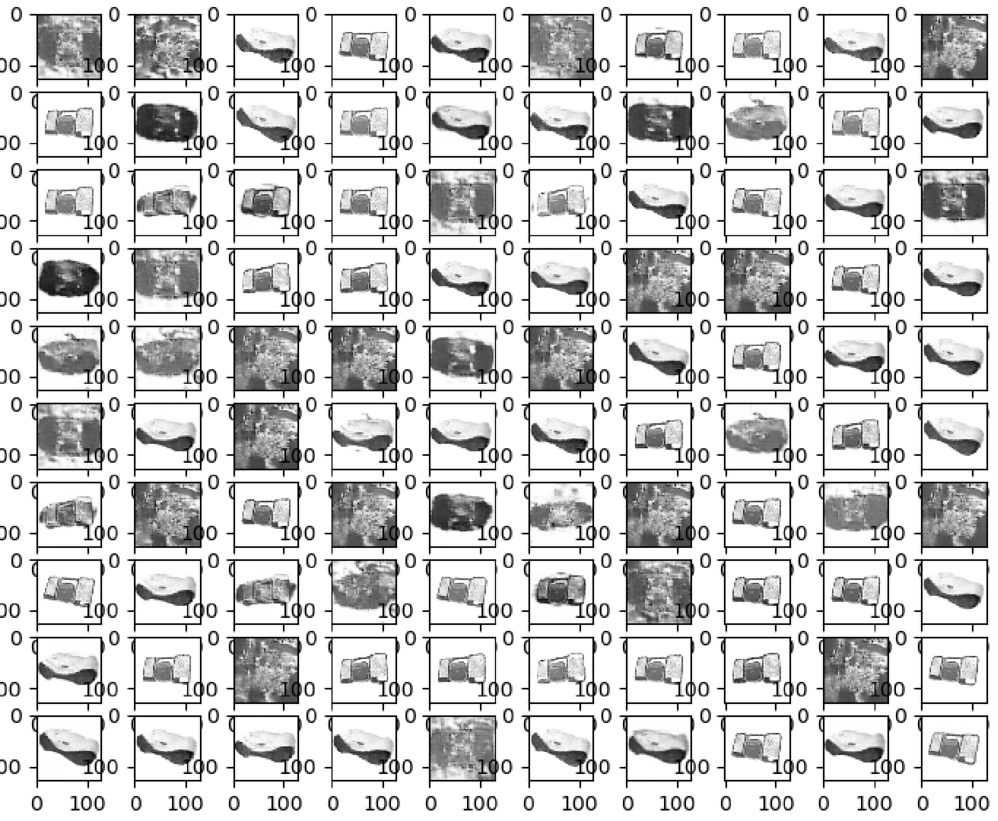


Figure 2.3: Illustration of mode collapse on boom box images, where a GAN produces only a few samples of the class, usually memorized samples from the dataset (as shown here), a problem perhaps exacerbated by extremely low sample count relative to most GAN datasets (tens of thousands of samples). WGAN-GP was needed to escape this problem entirely.

An example of attempting to solve this frustrating problem using traditional data augmentation (here, vertical and horizontal flips and 45 degree range rotations of the input images were permitted) to train the GAN synthesizers is presented in Figure 2.4. Amusingly, the instability of training manifests in averaging the rotation of the image input *directly into* synthesis! The resulting images look like they approach radial symmetry, taking the oblique views of the AK47 rifles and rotating them through, most commonly for this rotation-randomization range of 45 degrees, 2 or 3 orientations, making the magazines look like wings in poor-quality images of birds. Increasing the randomization range was found to exacerbate this problem,

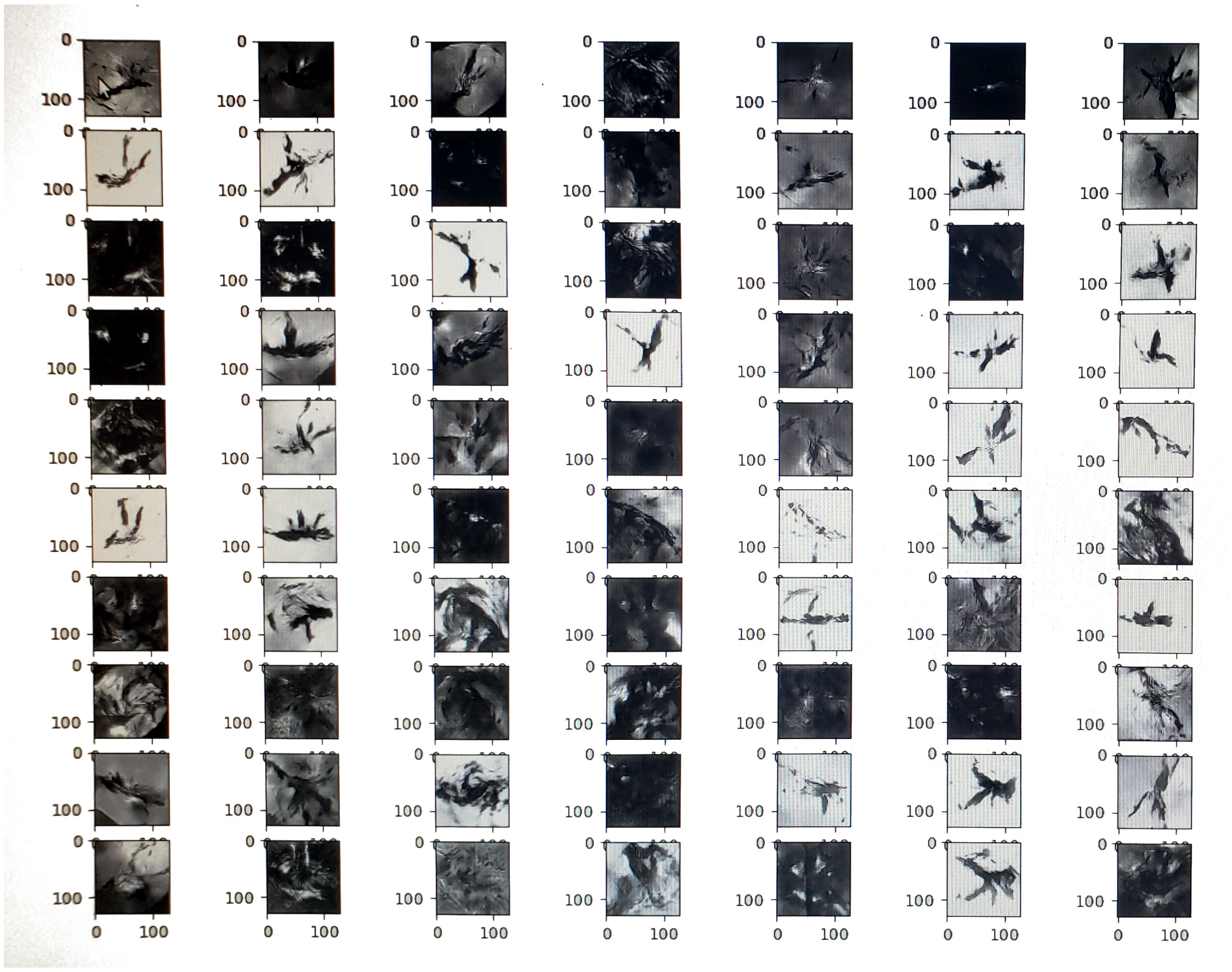


Figure 2.4: Ordinary data augmentation via transformed (rotated, scaled, contrast adjusted, translated) input data produced variety in samples, but not accurate samples. For example, these samples could be better mistaken for poor pictures of birds than AK-47s. Qualitatively, less rotation in input had less of an effect (not shown) on mode collapse. All input experienced random flips and rotation of 45°.

producing more radially-symmetrical fakes; small rotation range (e.g. 5 degrees) triggered mode collapse comparable to that previously observed, although with some rotation in output images.

The use of traditional data augmentation useful for classification should thus be suspected in easing performance failures in synthesis problems, because the GANs capture the transformations but at the cost of losing the natural transformation-related invariance in the produced results. In the real world, rotating a picture of an AK-47 does not change the number of magazines on the AK47 in the picture, but in the world of unevenly trained GANs, exactly these things are possible.

An avenue that remains open despite this result in addressing mode collapse is in making training stable. When training is stable (as measured externally only by loss), it may happen also that the network learns at a more similar rate throughout the layers and filters of the network, preventing chimeral mixtures of representations from desynchronized gradient updates. Under "stability", the discriminator and the generator are less likely to depart their Nash equilibrium like [61] stability, as when the discriminator loss (assuming unrectified output for the discriminator as detailed above!) stays stable and the generator loss (based on fooling the discriminator) flies off uncontrollably, so progress towards convergence and thus higher quality samples are likely to materialize.

Assuming that the GAN process is bringing the fake distribution and the real distribution into alignment, and thus to make the distance function used to assess the distance between distributions smoother is to make an overall loss function using this distance smoother, was the main insight underlying the authentically landmark Wasserstein GAN of Arjovsky et al. [8]. They explicitly proposed avoiding the rectification on GANs (which this investigation did for the mostly unrelated reason of trying to assure an ensemble of class-specific GANs had the most unadulterated information from the member networks) and substituting in the Wasserstein-1 distance for the notion of the disconnect between the distributions. The 1st Wasserstein distance is also called the Earth Mover's Distance because it measures the distance between distributions by shifting probability mass from value to value in the space and considering that the "transport cost" of that move is based on how many adjacent values the mass has to move. Particularly, the EMD is the *optimal* transportation of mass plan, which means that it is not completely straightforward to calculate (falling into the category of planning-related quantities that need linear programming or other more intense methods to solve and which are not susceptible to general closed form expressions).

Wasserstein-1 has the following definition [8]:

$$W(\mathbb{P}_a, \mathbb{P}_b) = \inf_{\gamma \in \Pi(P_a, P_b)} \mathbb{E}_{(x,y)\sim\gamma}[|x - y|],$$

where the greatest lower bound of an expected probability value disparity is taken over the possible joint distributions. A joint distribution γ is considered to be evaluated on the cost of moving the mass from one position in the probability space to another. Arjovsky and colleagues considered the total variation distance (which is just the supremum of the norm of the difference of value, or the distance, at each probability value), the KL divergence both ways, the Jensen-Shannon divergence, and the Wasserstein distance and showed that the latter places less topological constraint and is "weaker" than the other divergences, converging in a certain case where they did not. They then proved that using EMD as the basis of a loss function would lead the loss function to be continuous everywhere and differentiable almost everywhere under some assumptions, and that the gradient of the EMD is equal under these assumptions to the negative of the expectation of the gradient of a function like that of the discriminator judging the output of the generator given the noise. The most notable assumption involves Lipschitz continuity (which relates to the bound on the maximum value of the derivative attainable with a function). From a sequence of insights, they develop a poor-man's method of enforcing Lipschitz constraints in the case of the GAN: clipping the weights after gradient updates to a specific range.

The WGAN algorithm eventually developed renames a slightly-modified discriminator to the critic. As happens when you would attempt to balance the discriminator and the generator through some likely-doomed *ad hoc* method of your own invention, you define a number of surplus update steps the discriminator gets for each step the generator gets, n_{critic}. While training, you compute the gradient updates for the discriminator. The gradient is determined from considering the difference between the activation of the discriminator on real samples and fake samples as in traditional backprop. However, the gradient descent procedure is, critically, modified. Instead of using the ordinary gradient descent with no momentum (or memory of the last gradient, with the family's start being credited to Polyak [131]) or the popular Adam optimizer, they selected RMSProp (introduced in lecture notes by Hinton et al. with the descriptive title "rmsprop: Divide the gradient by a running average of its recent magnitude" [77], and named because some unpublished work found that it worked better when adding a square root to the mean square operation). The gradient updates the weights using RMSProp but then at the last moment, the weights are clipped between $-c$ and c, a clipping parameter. The choice of this hyperparameter is free and therefore subject to being chosen poorly. Analogously (the negative of) the gradient based on the average of the critic activations on the generated fakes batch is determined and updated with RMSProp and no clipping.

The qualitative results of the experiments in [8] were impressive. WGAN produced better quality images than a stock DCGAN, did this without needing BatchNorm, and most interestingly suffered no mode collapse generating Bedrooms, which was a source of excitement in GAN research.

For whatever reason, when the WGAN algorithm was tried with our reasonably standard architecture and using the default parameters in an attempt to derive the no mode collapse benefit, it failed to converge. Rather than moving to complete a detailed parameter sweep, an improvement was chosen, the WGAN-GP (Wasserstein GAN with Gradient Penalty) algorithm of Gulrajani et al. [71]. The WGAN-GP method revises out the weight clipping parameter and replaces it with a gradient penalty, which seems more appropriate since it is not constraining the weights maximum achievable value after the update directly.

The WGAN-GP also specifies a number of surplus passes for the modified discriminator, and the total number of passes is n_{critic}. It lacks the clipping parameter c but has a gradient penalty parameter λ and uses the Adam optimizer instead of the RMSProp optimizer. For each critic round, the loss is accumulated over the batch subject to the sum of the discriminator's activation on the fake image, the negative of its activation on the real image, and a gradient penalty term

$$L \leftarrow D(fake) - D(real) + \lambda(||(\nabla_{interpolated} D(interpolated)||_2 - 1)^2$$

The interpolated term may seem bizarre, and arguably it truly is. The authors relate enforcement of the Lipschitz continuity without resort to gradient clipping by bounding the constituent samples being compared between each other, and this is done with alpha blending. A random ϵ is chosen $\epsilon \sim U[0, 1]$, and the epsilon is used with the familiar alpha blending linear interpolation equation to produce the blended sample.

$$interpolated = \epsilon real + (1 - \epsilon)fake$$

Adam is then used to get the weight update for the critic corresponding to the averaged losses from the batch. For the generator pass, fakes are created and then a similar average is run through Adam using the more familiar GAN formulation of the discriminator (which has now been updated several times) working on the patches the generator has built from noise.

For the purpose of reproducibility and reliability in this involved process, the implementation used in this investigation was exactly that of [124] A.K. Nain in the [29] official Keras documentation. The number of discriminator steps n_{critic} was 5, and the weighting of the gradient penalty λ was 10, using the architecture described above.

Gratifyingly, the WGAN-GP process worked acceptably despite our sample impoverished data. Convergence happened for *almost* every class, and mode collapse was clearly not a problem, although sample diversity could still be described as low, especially among Stuff classes where this is definitionally a given. The training time expended for each GAN, each working to synthesize new samples for a specific class, was measured in completely nonstandard "units of training": triples of 32 image batches, and these numbered 15,000 per network trained. Epochs (whole passes through the data) were not used because of the incomparable class sample sizes.

2.1.3 Qualitative Differences in Things vs. Stuff Fakes

With the training regimen of the networks specified, the qualitative results of the synthesis that results can be considered. The 15,000 training steps were chosen to bring the Things to the edge of competent synthesis, since it was expected that these would be most difficult.

Sample results from within the first 1,000 steps are shown in Figure 2.5 for Things and Figure 2.6 for Stuff. It is interesting to notice that at the earliest shown points (e.g. 50-300 elapsed units), the proximal "goal" of the optimization is dichotomous for Things and Stuff. That is, the "proto-stuff" first focuses on filling out the image plane in a grid like pattern. Radial (ellipsoidal) zones in the texture that start as more rectangular grids indicative of the influence of local window filtering operations differentiate over time into the proper detail. Interestingly, even by the end of training and certainly by the first 10,000 steps (Fig. 2.8), some Stuff classes have not converged. Interestingly, they were classes whose target form is reminiscent of the "proto-stuff" pattern – an organic texture with Voronoi or leaflike cell-shaped structure and a truncated dome texture. It could possibly be that these synthesis failure classes emerge because the target is too similar to early-developing Stuff synthesis and the optimization cannot escape or re-penetrate these areas of the parameter space. Proto-things, meanwhile, appear to focus on establishing average object bounds in the earliest steps of training – and then adding detail or less common forms later. Despite the fact that the "backgroundness" proportion of each class in the Things dataset varies significantly, especially in the earliest steps there is an apparent bias for most classes against backgrounded samples. To confirm this tendency with high confidence though, a neural network backgroundedness classifier could be trained so that thousands of samples could be automatically graded on having a background or not.

Observed synthesis failure classes and intentional "catch" duplicate networks

Since the synthesis failure classes (c093 and c129) are so dissimilar in quality from their peers, it can easily be conjectured that they are arrested at a different state of neural development than these peers: this would be bound to be evident in their behavior, but also likely the weight patterns of the neural networks contributing to that behavior. Studying the synthesis failure classes would then allow qualitative synthesis failure to be predicted from behavioral and parametric anomaly and would provide explanation of those anomalies upon visual inspection.

Because this phenomenon is potentially useful to gather information about and apparently as far as is demonstrated here, is a feature particular to Stuff, the preplanned Stuff classes resulting in synthesis failure were retained for investigation.

In addition, it was decided that duplicate networks showing the divergence of training and reflecting on multiple realizability of synthesis would be included as catch networks. The theodolite class was used to train two networks that diverged only in the stochasticity of training, and the boom-box class was also given a duplicate network, but this was trained for 25,000 units. The hypotheses related to the foregoing "catch networks" would be that 1) the simply duplicated network, if not parametrically similar, is behaviorally similar, and that 2) the network with extra training produces markedly higher quality output and that a divergence from the other networks on parametric lines could be statistically observed in a joint analysis of parameters, perhaps with a tendency the opposite of the incompetent synthesis failure networks.

These qualitative results build on the notion of Chapter 1 that there is a competence hierarchy of broad classes of texture that makes Things harder to recognize than Stuff, and affirms strongly that there is a similar synthesis quality competence hierarchy.

Since synthesis quality is controversial to quantify, the bulk of the remaining investigation focuses on recognition of fakes and real images involving the collection of WGANs in the line of assessing the practicability of GAN-based recognition as a rough, highly indirect proxy for synthesis quality.

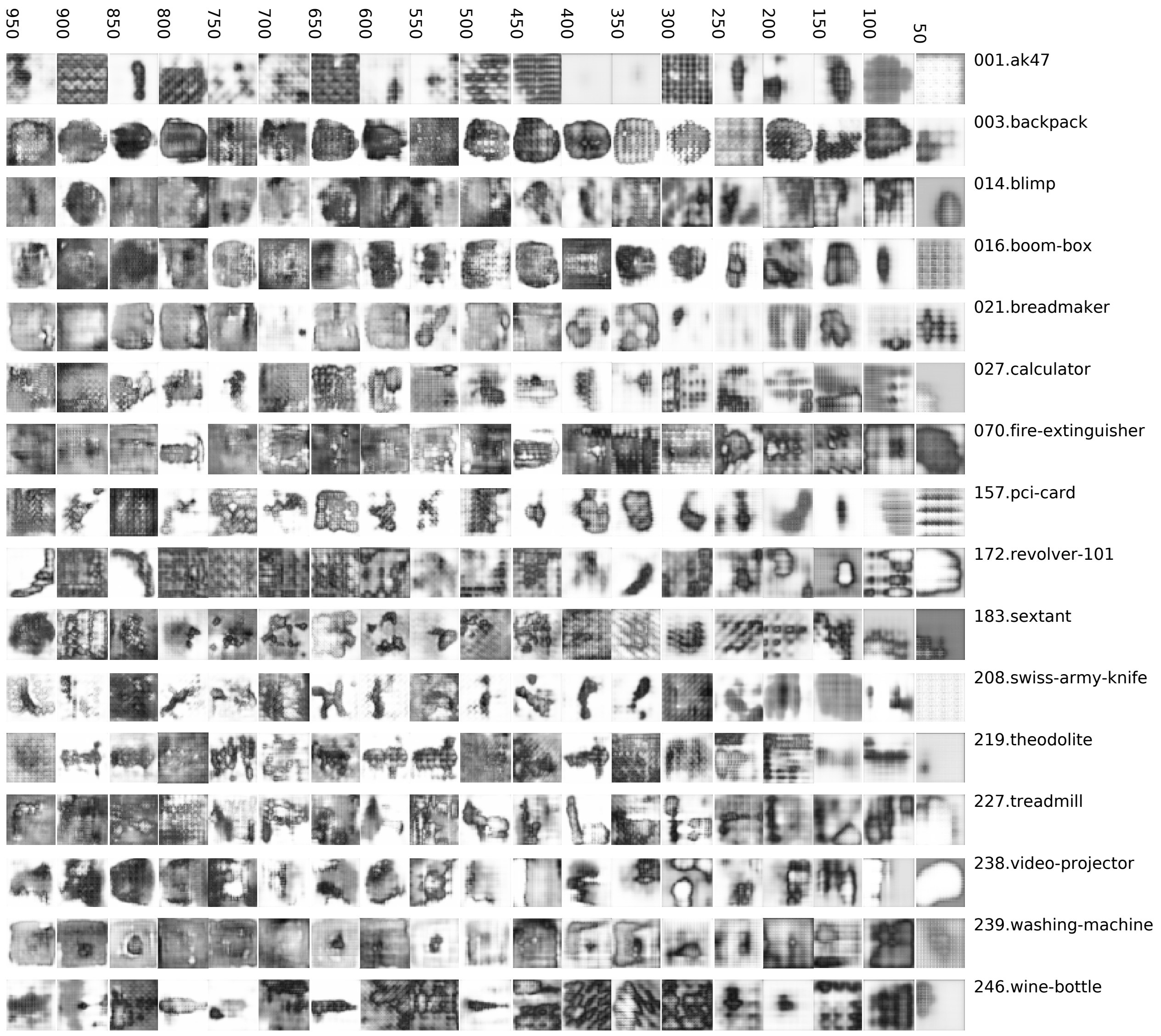


Figure 2.5: Thing Synthesis samples from the WGAN-GPs in the GANsemble, before 1,000 units of training (3x batches of 32 image presentations) elapsed. In the earliest stages of training, the patterns of Thing and Stuff synthesis are qualitatively dissimilar in how they depart from the nothingness of saturated output. "Proto-things" try to establish object boundaries first. Object boundaries are not very uniformly established within Things, suggesting that they take longer to produce minimum-acceptable synthesis results.

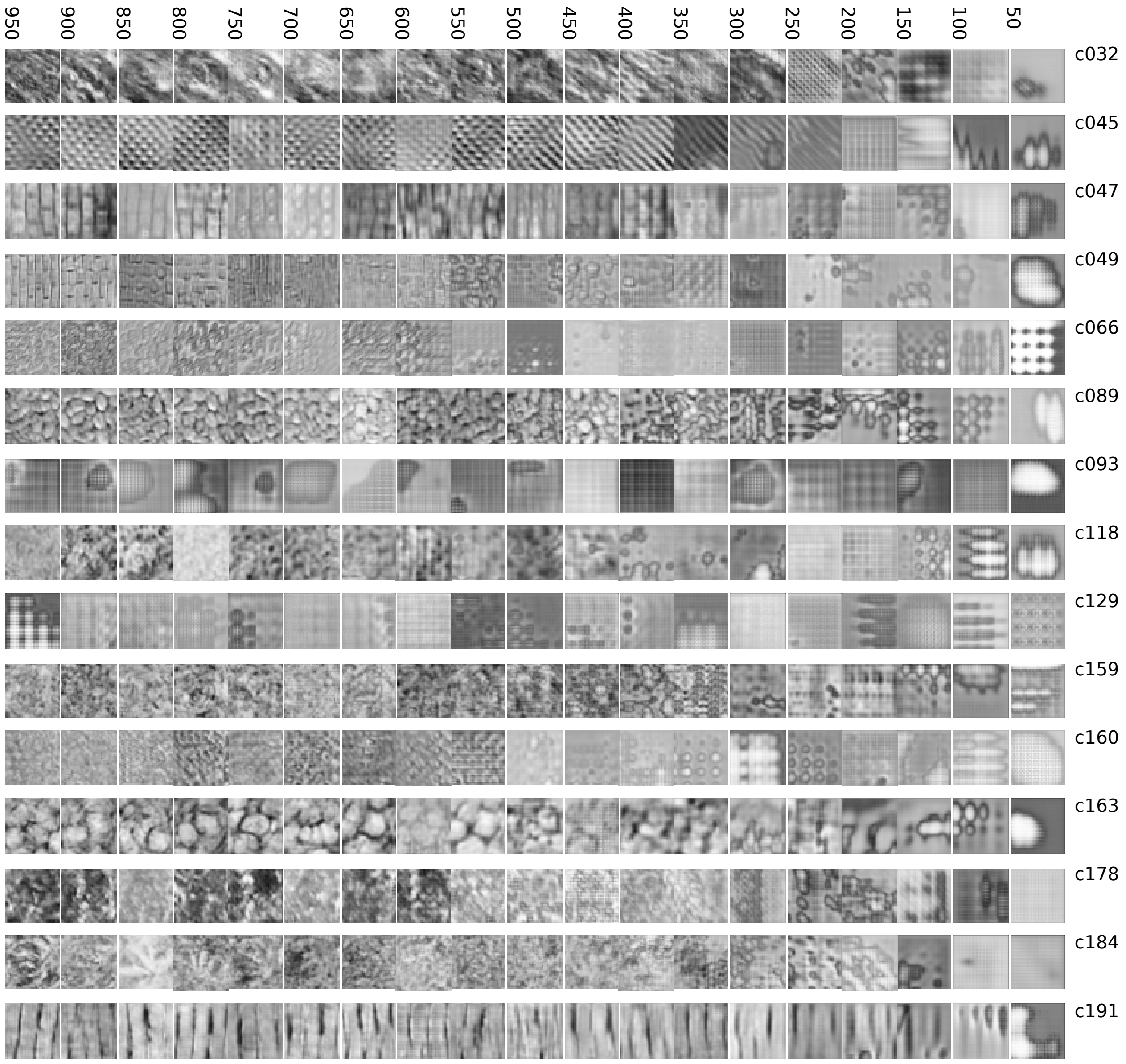


Figure 2.6: Stuff Synthesis samples from the WGAN-GPs in the GANsemble, before 1,000 units of training (3x batches of 32 image presentations) elapsed. “Proto-stuff” prioritizes tessellating the plane before adding detail, succeeding around training unit 600. Exceptions exist for synthesis failure classes which only exist in Stuff: c093 and c129, both circular grid patterns similar to proto-stuff, do not converge at all during this time.

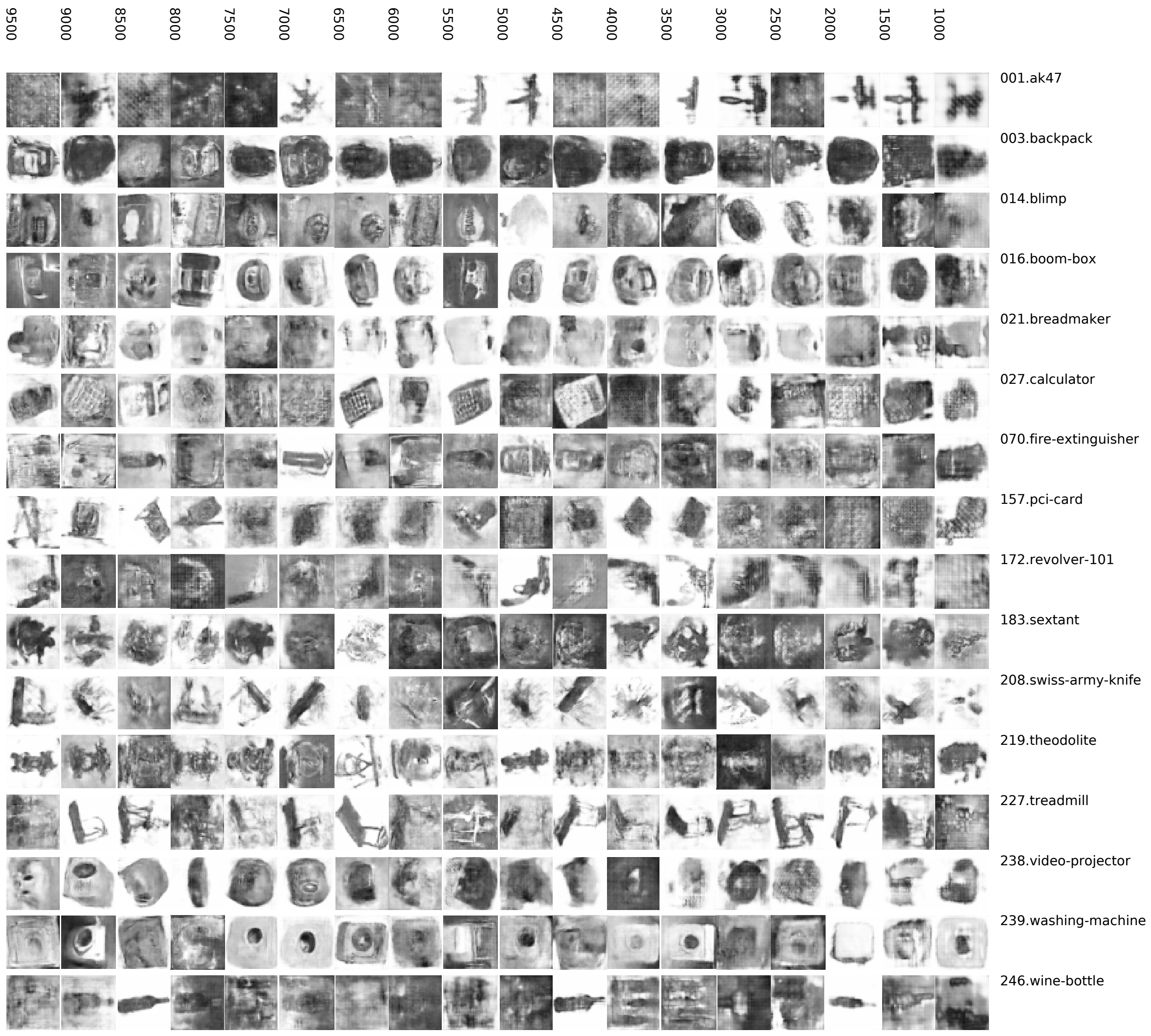


Figure 2.7: Thing Synthesis samples from the WGAN-GPs in the GANsemble, before 10,000 units of training (3x batches of 32 image presentations) elapsed. Things are still poor quality, with some classes like video projector, washing machine, and boom-box doing well.

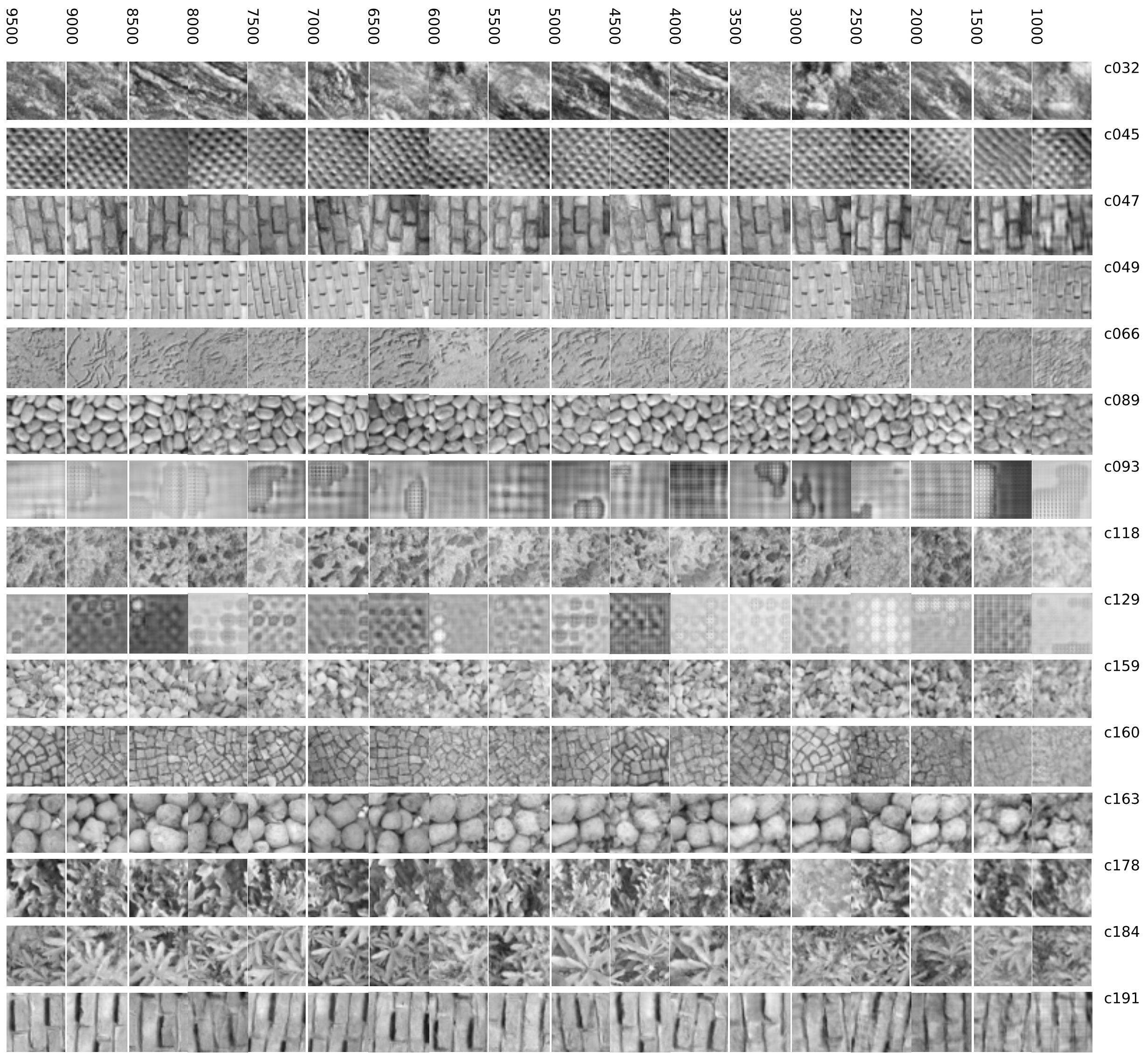


Figure 2.8: Stuff Synthesis samples from the WGAN-GPs in the GANsemble, before 10,000 units of training (3x batches of 32 image presentations) elapsed. Excepting synthesis failure classes c093 and c129, Stuff is near a quality ceiling.

2.1.4 GAN "morphs": retargeting pretrained GANs

One rough, primarily qualitative measure of the similarity of a bank of class-specific GANs is to subject them to a period of reeducation where they are trained towards a single synthesis target class. Stored GANs originally trained towards their own class-specific objectives and on their own data are loaded into memory, and retasked to the new objective: the single target class is fed to the list of all of them.

Consider the case of this battery of new GANs receiving a very small amount of extra training: say 150 units of training as opposed to 15,000. If an ordinary discriminator without the GAN objective were loaded to be paired with any of the trained generators, presumably quite an extended period of training would be required, comparable to training from scratch. The performance could be hypothesized to be terrible. In the other extreme, if the discriminator is one that is substantially similar or the same to the target, it should only act as purely surplus training time, further refining the result. In intermediate measure, the other networks that have been specialized for image synthesis but particularly for classes such as brick walls and boom boxes should require a middling level of adjustment in the form of allowed image presentations and weight updates to reach the level of quality produced by the originally trained class-specific GAN.

It could perhaps be the case that the speed of this kind of transfer learning [17] could be reliably proportional to the similarity of the network-originating class, but this would require an agreed-upon quantitative measure of image quality and also deep class similarity, and opinions on these measures are unsettled. If this strong hypothesis is true though, a weaker version must surely prevail as well: that it is easier to train Things networks to produce Things and Stuff networks to produce Stuff than to cross over this important metacategorical divide in the synthesis reeducation domain. The pattern that is observed is that the metacategory itself is more determinative of quality: stuff is easier to synthesize for both reeducated networks than things.

In point of fact, if Figures 2.9 and 2.10 are correct, both these hypotheses would appear to be false on visual inspection. However, it is the case as seen in Figure 2.11 that taking a discriminator of equal architecture with a cross-entropy loss and substituting it in for a discriminator with the GAN-oriented loss does not produce acceptable synthesis results with only 150 units of training and the network as a whole will require substantial revision.

In addition to morphing class-specific GANs into each other with transfer learning, with better automated quality and similarity measurements, the prospect of optimally pulling GAN layers apart for recycling into new Frankenstein or chimera networks could be studied using methods for gradientless discrete optimization methods (such as evolutionary computation) and perhaps a small allowance of fine adjustment polish training – but it would be consistent with these results that it would be hard to improve on a coherent network with linear or, especially, layer-substitution mixtures of networks since it takes time to cure inconsistencies at the level of discriminator substitutions, let alone swaps at the granularity of layers or linear combinations of layer weights. Possibly beyond the things vs. stuff distinction, it will be tough to predict optimal mixtures; however, this does rule out an effort to store a library of class-specific networks and to try to predict which will be the best candidates for transfer learning given a particular target.

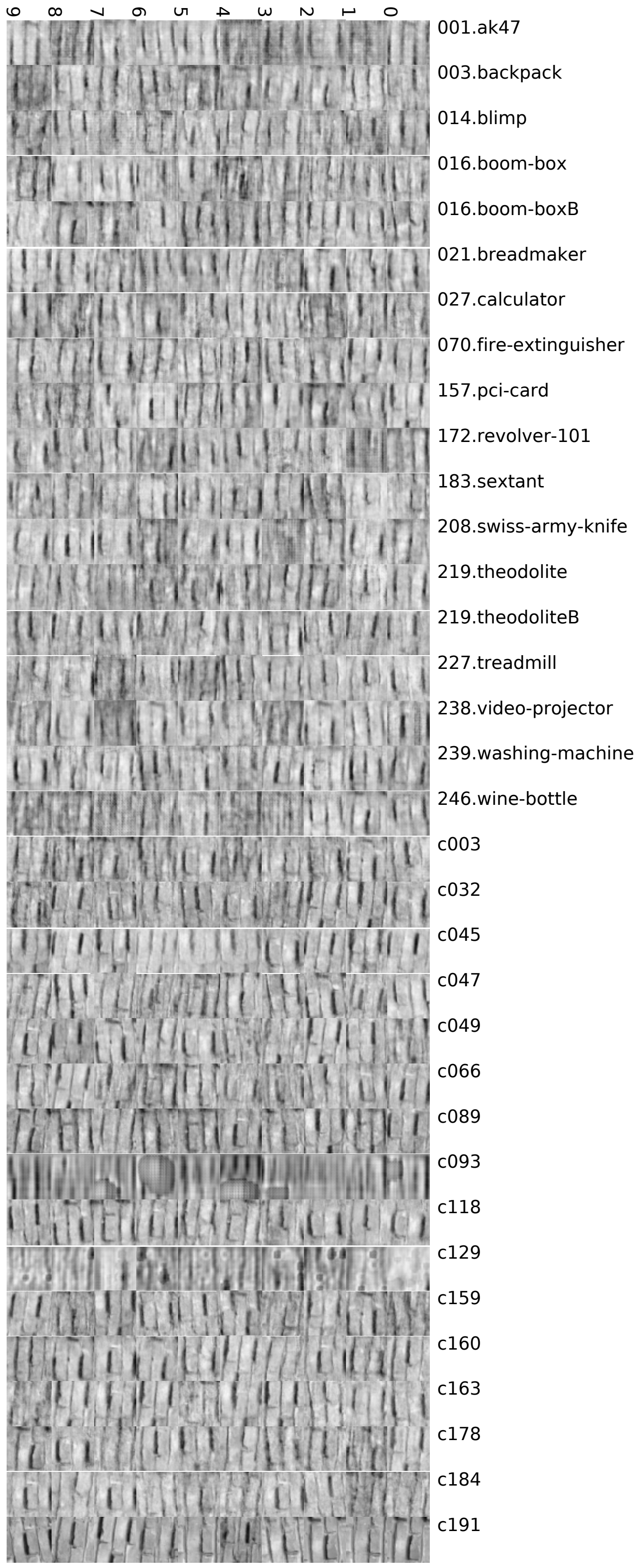


Figure 2.9: Each class-specific GAN's 10-sample rendition of a brick wall. All networks are trained for 150 extra training units but targeting c191's discriminator. c191's advantage (including over c047 and c049) in simply refining itself is very barely perceptible, excepting failure classes c093 and c129

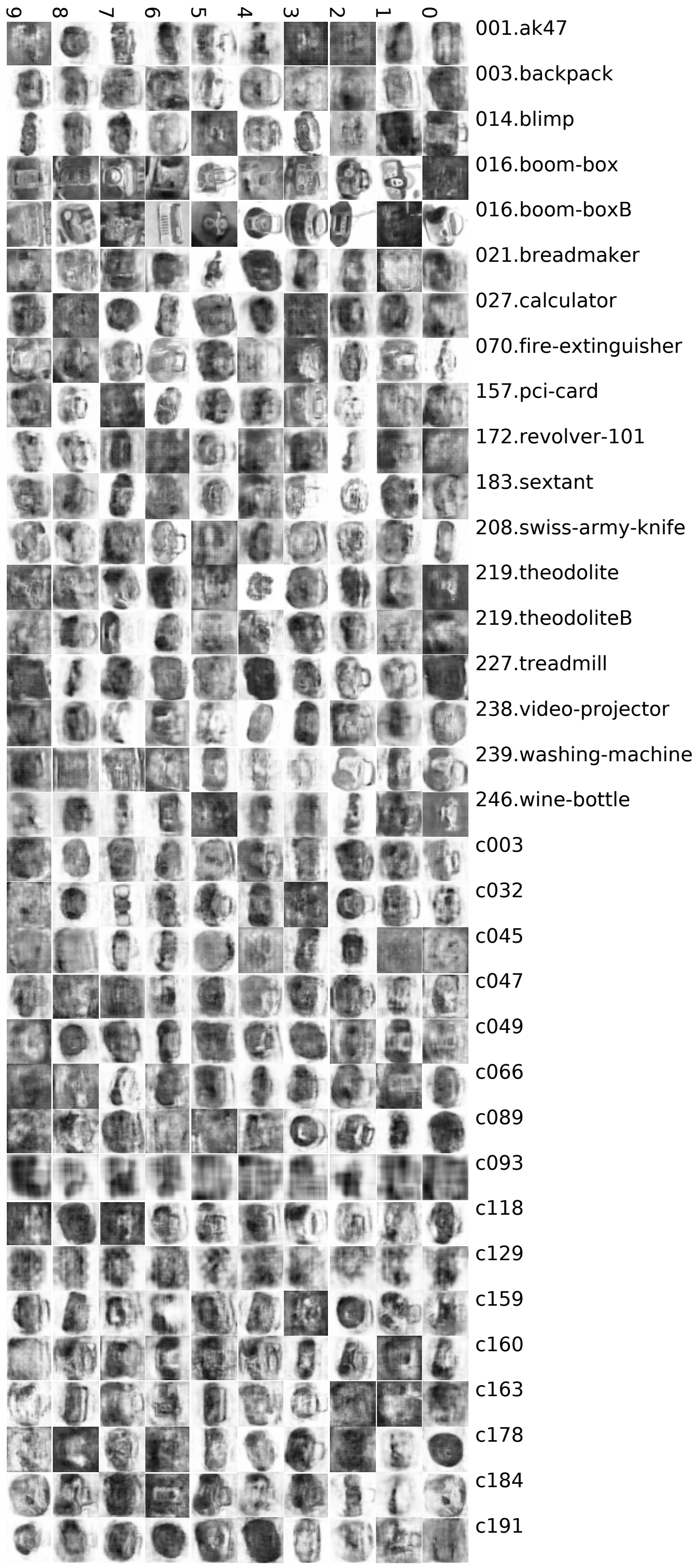


Figure 2.10: Each class-specific GAN's 10-sample rendition of boom boxes. All networks are trained for 150 extra training units but targeting the less trained boom box discriminator. Stuff members produce qualitatively more texture-like patches still and thing class synthesis demonstrates better defined boundaries. The boom-boxB network with 25,000 + 150 training units produces especially clear samples.

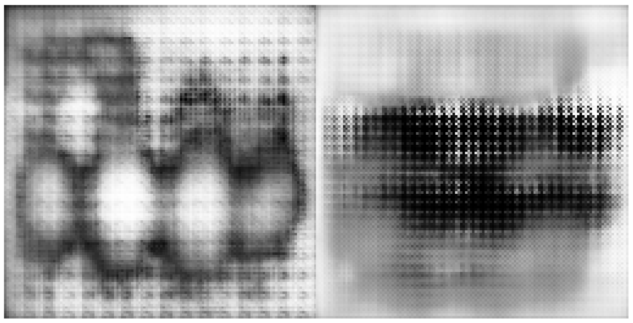


Figure 2.11: Discriminators not trained with the GAN-style loss fail in their attempt with the old generators to synthesize more than primordial proto-thing samples with the provision of 150 training units. The donor networks were the a) pci-card and b) fire extinguisher networks.

2.2 GANsembles: committees of one vs. rest GAN discriminators and their interclass label and weight affinities

As previously alluded, GAN-origin-discriminators are trained to aid synthesis, not do efficient classification. For a GAN targeting a single class for synthesis, they are exposed to the target class of interest and not any other class. We should expect them to suffer when compared to a network for classification. Therefore, they can be assumed individually to be weak classifiers. A way of improving the results of weak classifiers taken in isolation is to consider the joint behavior of the weak classifiers operating on the same input – this is the approach of the ensemble in machine learning, which has many varieties depending on how the information from the contributing classifiers, sometimes called the committee, is combined. Anticipating the frequency of reference to this new object, we presently term the ensemble of GAN-origin-discriminators to be the GANsemble. Previous work [173] [108] has created ensembles of GANs but these are primarily ensembles of the whole GAN and may even be confined to testing different initializations of generators to potentially expand the creative output in terms of fake patches of the collection with the idea that this could improve synthesis results.

At this point, the present investigation is no longer primarily concerned with synthesis results, but classification.

2.2.1 GAN-origin discriminator interclass labeling affinities on fake vs. real images

Before settling on a method to combine the information gained from each GANsemble member network, the joint output distribution of the networks should be considered. To estimate this distribution, we have two natural choices: we can estimate the output distribution on the real images used to guide synthesis, or we can use the fake images which result from synthesis. Any dichotomous Things vs. Stuff results (when thought of in terms of signal) for the fake images are assumed a priori to be "weaker" versions of the results obtained for the real images, since the synthesis has only come to the point of being barely competent in perceptual quality.

A fully satisfactory characterization of the joint distribution of output values is unlikely, so we simplify this to consider just an average terminal activation of networks. For a patch p, each network D_k in the GANsemble emits an activation value a_k that contributes to the GANsemble's activation vector A, which we can term the *label emission spectrum* of the GANsemble after the example of Fourier analysis of signals and spectral analysis of chemicals, since we are looking for the "power" of activation at each (admittedly complex and unorderable) sensor level.

For real images, each member network D_k in the GANsemble receives all of the patches p in $P_{c_i \in C}$, where C is the totality of classes in our dataset, and i is an index for D_k that indicates the target class c_i, since with duplicate networks a crosswalk vector may be needed to correspond networks with non-uniquely occurring classes. It emits a "label" that because of the lack of rectification for the WGAN is actually just a bare activation, but one that should be thought of as consistent for that patch given the network as it was trained since we will not update the gradients and weight values in a backward pass for merely measuring activation.

We can construct the Real firing or "label" *pairwise affinity matrix* $RR(k_1, k_2)$ by dividing the vector of activations collected by firing D_{k_2} on the D_{k_1} network's corresponding target class C_i. The patches come from an unchanging authoritative source – the dataset.

The Fake pairwise affinity matrix $FF(k_1, k_2)$ is the result of firing D_{k_2} (the *fake-receiving network*) on a batch of generated image fakes from G_{k_1} (the *fake-producing network*). The image fakes produced vary because they are the result of random input noise vectors (e.g. $G(z)$), so we are computing a Monte Carlo average with the idea of approximating the asymptotic behavior of infinite generated images. The count of image samples $n_{\text{MC}} = 100$ used for Monte Carlo averaging is small to keep our computations relatively quick since the number of the entries in the matrix in the first place grows with the square of the number of partipating networks (i.e. 34^2).

The fake images affinity matrix computed is displayed in Figure 2.12. For lack of more precise language, it is vertically streaky. This corresponds to the fake receiving networks emitting activation at proprietary (and not directly comparable) ranges. For example, the wine bottle class fires highest. Most other classes fire higher than an absolute 0 indicating some acceptance of the fake patch. A few classes, notably the synthesis failure classes c093 and c129, fire with inverted sign and out-of-band magnitude compared to the rest of the field. Of the receivers that fire high, the wine bottle class fires highest.

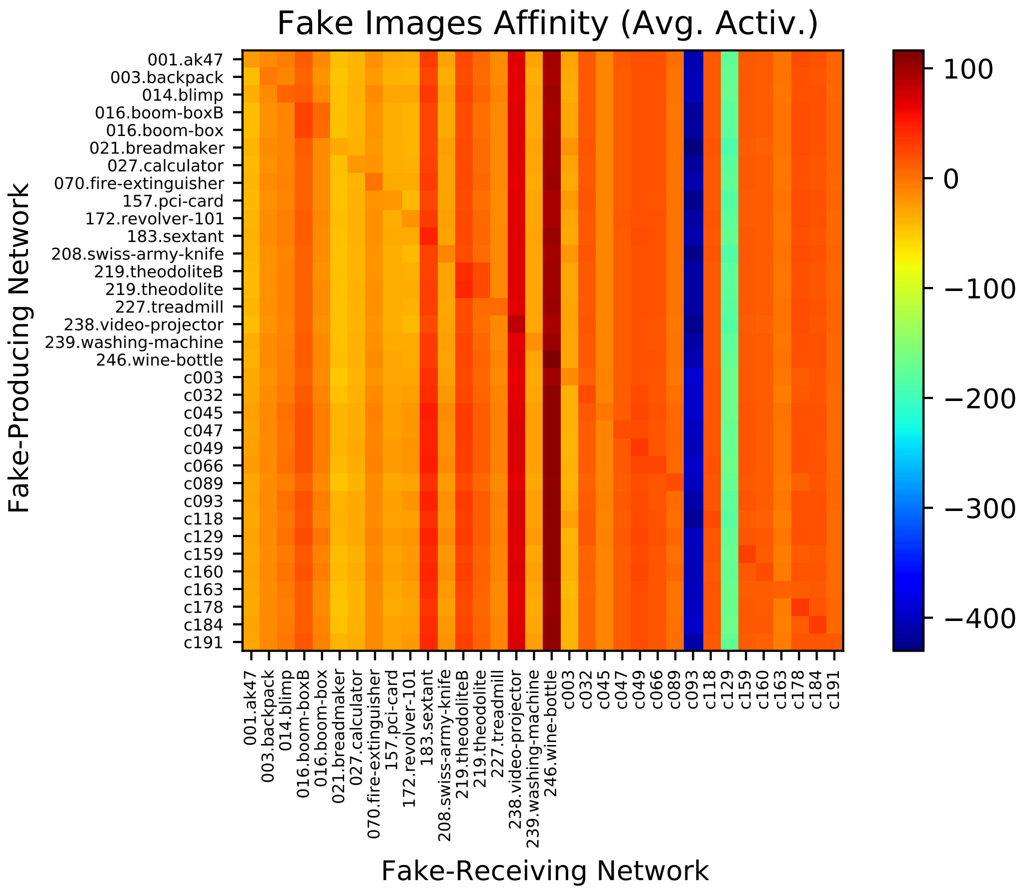


Figure 2.12: Per-class activation Monte Carlo average of each discriminator in the GANsemble to every other network plus itself (diagnonal) calculated over the 100 fakes produced by each network. Some networks fire higher than others systematically, preventing even the discriminability of things vs. stuff. Synthesis failure classes c093 and c129 fire systematically lower. Similarity of identical catch classes boom box and theodolite is barely discernible.

The pairwise affinity matrices should live up to their name in that we use them to estimate the dissimilarity of classes. That is, we should arrive at a representation where the diagonal is close to a relative reference zero point (which may or may not coincide with the real absolute zero point) and so the activations departing from the reference point are strongly related to class dissimilarity. Recall that the WGANs, being unregulated by a nonlinearity, tend to return high values when they perceive a real image, and low values when they perceive a fake image. Intermediate values near zero are what indicate non-confidence in this unrectified case. This will soon become important.

Now, the fake affinity matrix representation is in some sense dual to an directed graph representation where nodes are classes and the edge weight or potential in one direction corresponds to the nearness of two networks where one is the receiver and one is the producer. An opposite potential reverses the role. Once we arrive at a transformation of the affinity matrix into a more proper affinity matrix that more closely represents dissimilarity we might consider these potentials to be part of a dissimilarity graph.

Graph representations of similarity are appealing because we can apply the interesting toolbox of graph theory methods to image class similarity, permitting some level of studying the categorization induced by deep neural networks, rather than with a human-in-the-loop looking at the graph and suggesting important classes and important taxonomic divisions. For example, in graph theory, the rich-club coefficient [187] of a graph concerns how connected the pivotally well-connected node (hubs) of the graph are, the minimum spanning tree describes efficient paths through the graph, the isoperimetric or Cheeger-related constant [121] describes how bottlenecked a graph is, and the Fiedler [48] vector values (the second-lowest-eigenvalued eigenvector of the graph Laplacian matrix that is the difference of the degree matrix and adjacency matrix representation of the graph) collectively provide a recipe for locating ideal cuts and counting connected components. Some number of these techniques are restricted to undirected graphs, however.

For our inspection, however, the directed affinity graph corresponding to the fake affinity matrix can be drawn, although since it is fully connected to start, it is almost an almost completely uninformative depiction (the classes are arranged in a ring with all of the possible connections creating some aesthetically interesting intersections but little else). If we want to gain some information from the visualization we can use a force-directed graph drawing algorithm, such as Fruchterman-Reingold [52] off the shelf (using [72]). A discussion of the Fruchterman-Reingold algorithm is beyond the scope of this project, but suffice it to say that the idea of force-directed graph drawing in general is roughly based on the idea of using the weight potentials in the graph to define a kind of Hooke's Law like spring attraction quantity that can be combined with an electricity like repulsion between nodes so that the system can be physically simulated until an acceptable equilibrium is reached. This heuristically more usually avoids intersecting edges but also has the benefit of often placing bonded nodes near each other when they are similar (if similarity or dissimilarity information is represented on the graph's arc weights).

Figure 2.13 is the force-directed digraph corresponding to the fake images affinity matrix. The positioning of networks or nodes on this graph is not expected to be replicated in a subsequent run of the simulation. Even relative proximity information is likely to be somewhat highly disrupted by the complexity of this process and the multiple realizability of equilibrium.

This problem is exacerbated in that the arcs included involve weak potentials which may have a very-non-directly-related effect on placement. If you were to try to take the entire all-to-all connected graph (the complete graph) with its 34 nodes and its $2\frac{34(34-1)}{2} = 2 * 561 = 1122$ edges and request from the graphviz tool [101] the default `dot` drawing style which is aesthetically pleasing for trees, you might not even get an output. If you wanted to decimate the graph so that only the most informative pairwise distances were represented you could set a threshold. The threshold could be symmetric (favoring both extremities $x \in (-\infty, -100] \cup [100, \infty)$) or asymmetric (favoring just one). Figures 2.14 and 2.15 show the effect of graph-thresholding with the force-directed drawing method preserved; the graphs are very distinct where the "highly-positive pass" one is very uninformative, capturing only the streak of maximum activation corresponding to the wine-bottle class from the matrix.

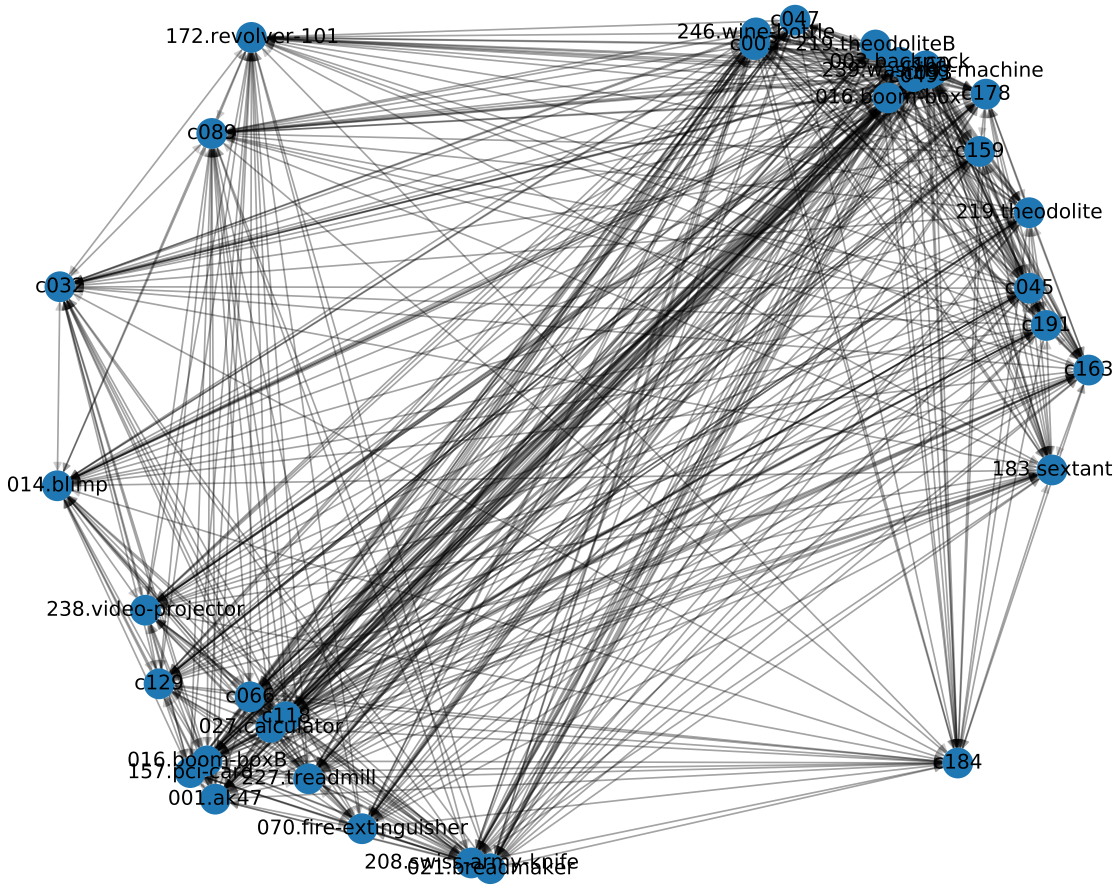


Figure 2.13: Fruchterman-Reingold graph drawing of the graph corresponding to the GAN Fake images affinity matrix. Arrowheads denote that this is a digraph, where the directions along the arc are related to fake-producing and fake-receiving role.

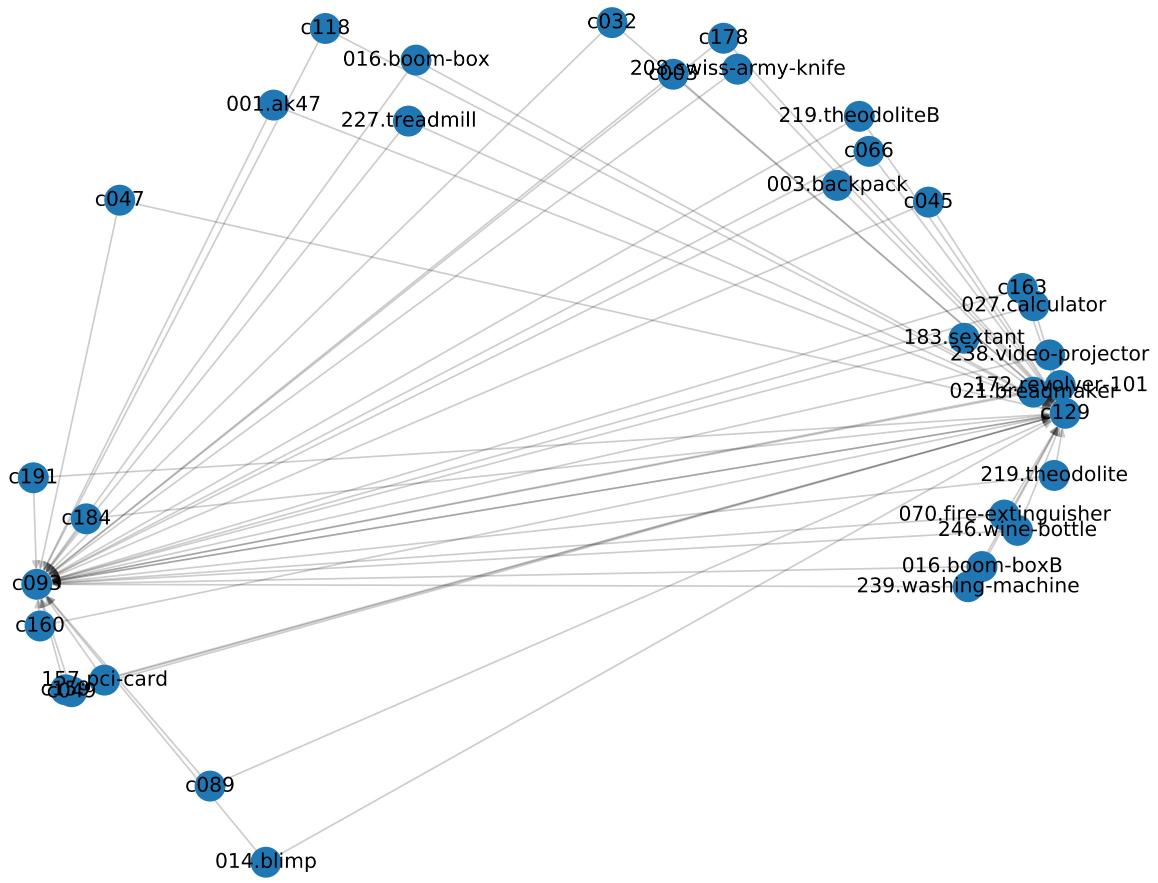


Figure 2.14: Fruchterman-Reingold graph drawing of the "highly-negative pass" graph corresponding to the GAN Fake images affinity matrix. Only entries from the matrix whose values are less than or equal to -100 are included as arcs.

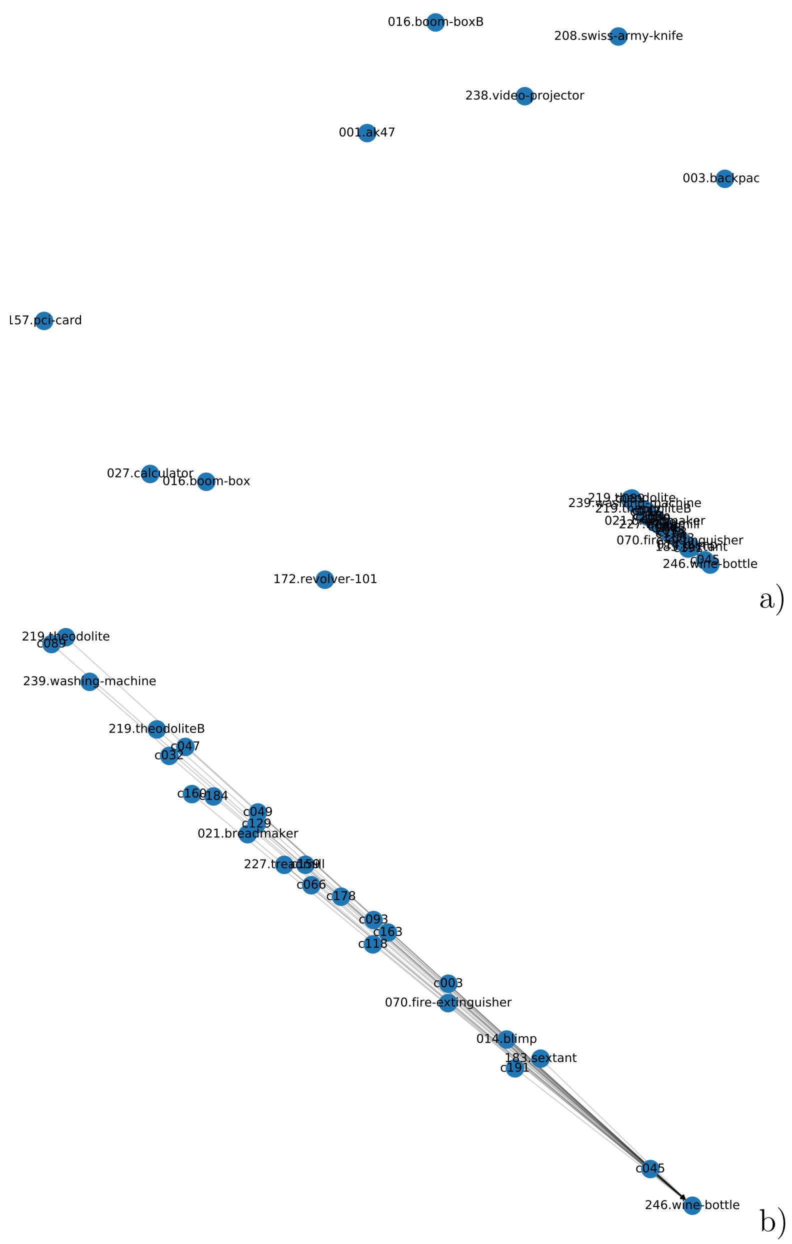


Figure 2.15: Fruchterman-Reingold graph drawing of the “highly-positive pass” graph corresponding to the GAN Fake images affinity matrix. Only entries from the matrix whose values are greater than or equal to +100 are included as arcs. a) The total graph is shown. It has disconnected components. b) The area of the graph which is connected is enlarged to show that the very positive activations are all directed onto the wine-bottle class.

What you learn from similarity graphs, even with only 34 networks and 32 image source classes here, is very limited compared to the space and effort they consume, and the informativeness that does remain is very sensitive to your hyperparameterlike free parameter setting of thresholding. Additionally, the force-direction does not stably capture similarity the way other manifold learning methods (such as MDS, which will be visited in detail in Chapter 3) might, although the foray into graph visualization does give one a clue that manifold learning (or dimensionality reduction) methods which rely on the (unweighted) graph adjacency matrix (for example, Laplacian eigenmaps [11]) may do poorly on this particular kind of dense data.

Even if the affinity matrix is preferred, it is apparent that it is still grossly unsatisfactory. The streaky pattern does not show even a difference between Things and Stuff, our high-level metacategories! Could this be the end of our project, that GAN discriminators are almost completely uninformative, even in ensemble?

It turns out that it is not, as the reader who has either the heft or the view of the remaining pages could easily intuit: a simple transformation yields some promise of the Things vs. Stuff dichotomy, and even a pattern of graded negativity in responding that could potentially be used to aid classification (as well as class similarity judgments).

The reference firing level seeming to float for various of the deep convolutional recognizers in our GANsemble seems to suggest the curative use of normalization. Normalization computations are useful in early visual circuits accounting for contrast, and it is suggested by Heeger and others that normalization is “a canonical cortical computation” [26]. Now, the neurons in our GANs bear such a pitifully passing resemblance to biological neurons that it could be jokingly suggested that no real neuroscientist should be involved in studying the behavior of devices that are composited from them, but when it comes to normalization, there are two familiar kinds from mathematics: normalization by subtracting a baseline value and normalization by dividing by a baseline value. It turns out, of course that the choices have different implications, and these implications were studied in terms of the long range effects on dynamics in the case of Hebbian learning in work that imagined the necessary firing-rate control constraints [119] on idealized circuits preventing runaway activation growth as influenced by regularizing-penalty-like subtractive normalization vs. multiplicative normalization. Specific meaning is given to these terms in that work and what is tried below is not completely identical.

Given the work done with our nonstandard definition of affinity (which should ordinarily be a byword for similarity, not "ur-similarity") in the affinity matrices presented, we can come up with a sensible definition for a baseline. It is the Monte Carlo average (as recorded in the Fake matrix, on the diagonal) of a network assessing its own generator's fakes, the auto-association.

Divisive normalization and subtractive normalization are trialed in Figures 2.16 and 2.17. The divisive transformation does not improve on the lack of quality seen in Figure 2.12. The subtractive transformation evidences a strikingly different pattern, and one that we come to see the Things vs. Stuff dichotomy appear within. It can be presumed that subtractive normalization is justified in the following sense: if extra training or extra proficiency with the same amount of training tends to *add* activation, the sensible undoing operation is to take away, or *subtract* activation, not to divide. It may also be that divisive normalization and subtractive normalization rules work differently depending on the within-network vs. between-network variance of activation, and a scalar field of F-statistic-like quantities could be sampled in a simulation and annotated with the superiority in terms of separability able to be produced by either rule to see under which variance conditions each rule is useful. For the time being, though, we leave the superiority of subtractive normalization in this particular case as a simple empirical fact.

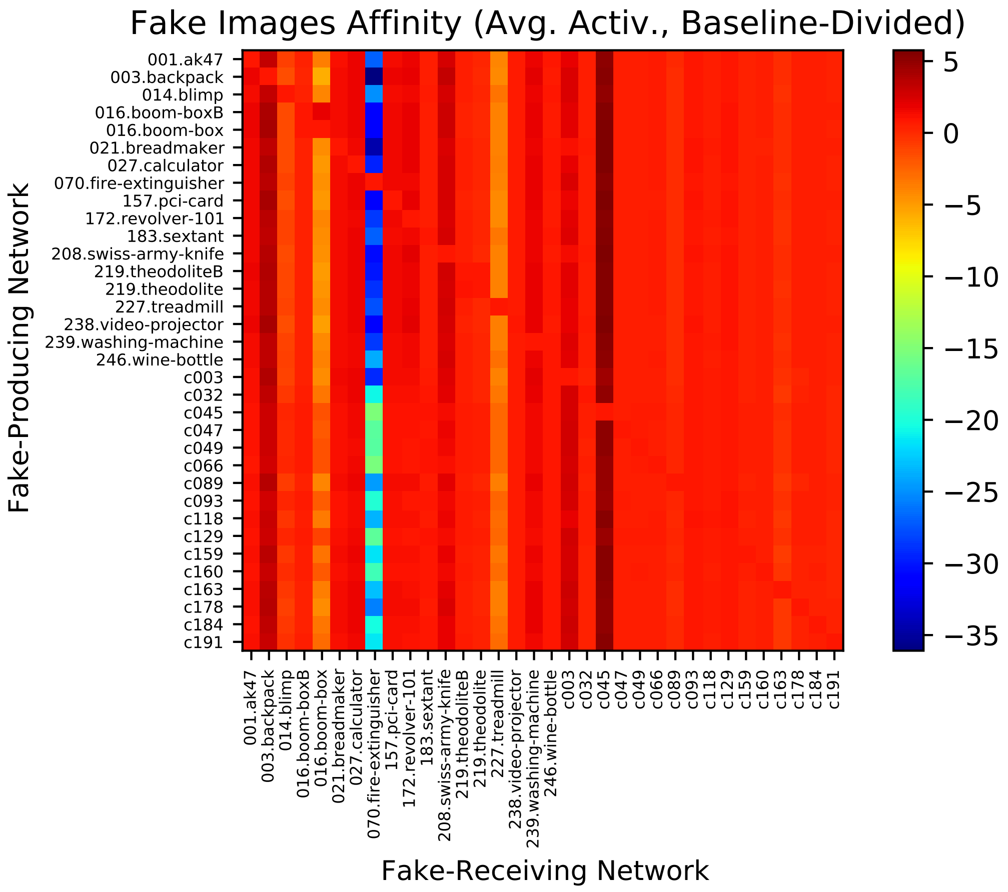


Figure 2.16: GANsemble average activation matrix for fake images, but with each activation divided by the receiving network's MC average baseline self-activation. The fire-extinguisher class fires strongly negatively to every other class, and the treadmill and only the less trained boom-box discriminator fires higher to most other classes. Again, things and stuff are not at all discernible.

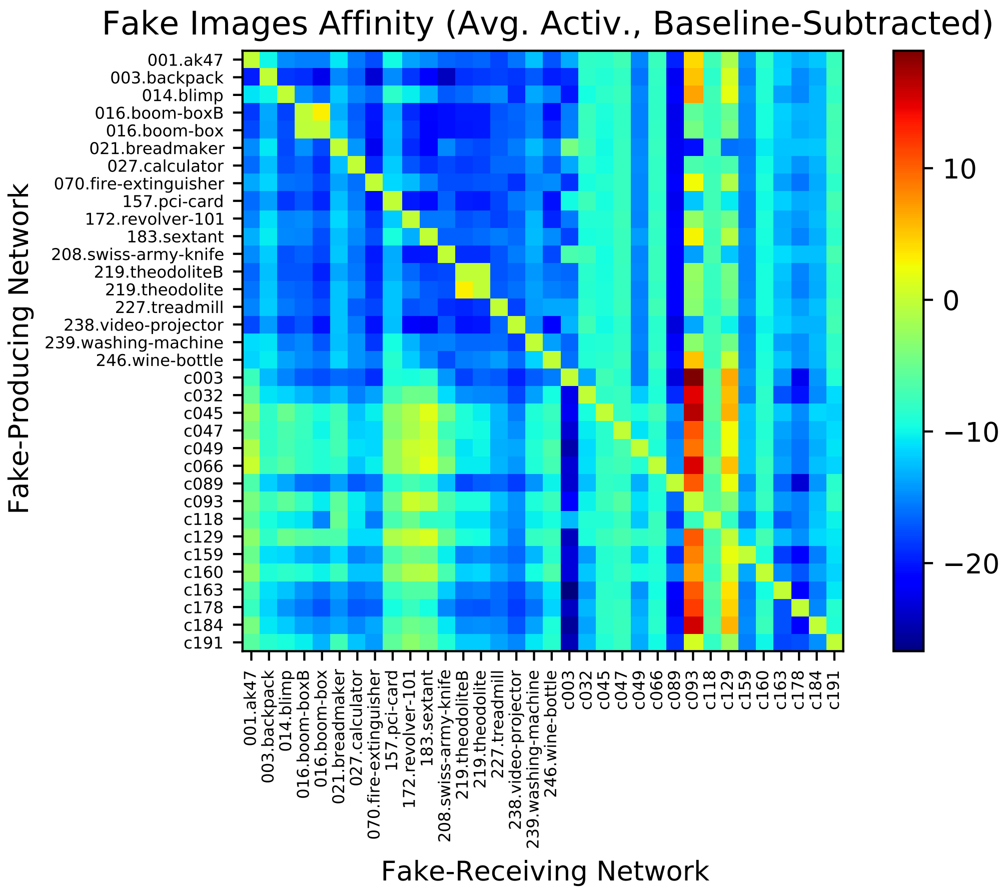


Figure 2.17: GANsemble activation affinity matrix for fake images, but with each activation ***subtracted*** by the receiving network's MC average baseline self-activation. Things to things comparisons (upper-left quadrant) show graded negativity facilitating similarity and classification whereas stuff-to-stuff comparisons (lower-right quadrant) show less useful patterns. Things-to-stuff and Stuff-to-things activations are also less informative. The promiscuous firing of the synthesis failure classes and the expected discriminator affinity of the catch duplication classes (boom-box and theodolite) are both clearly observed.

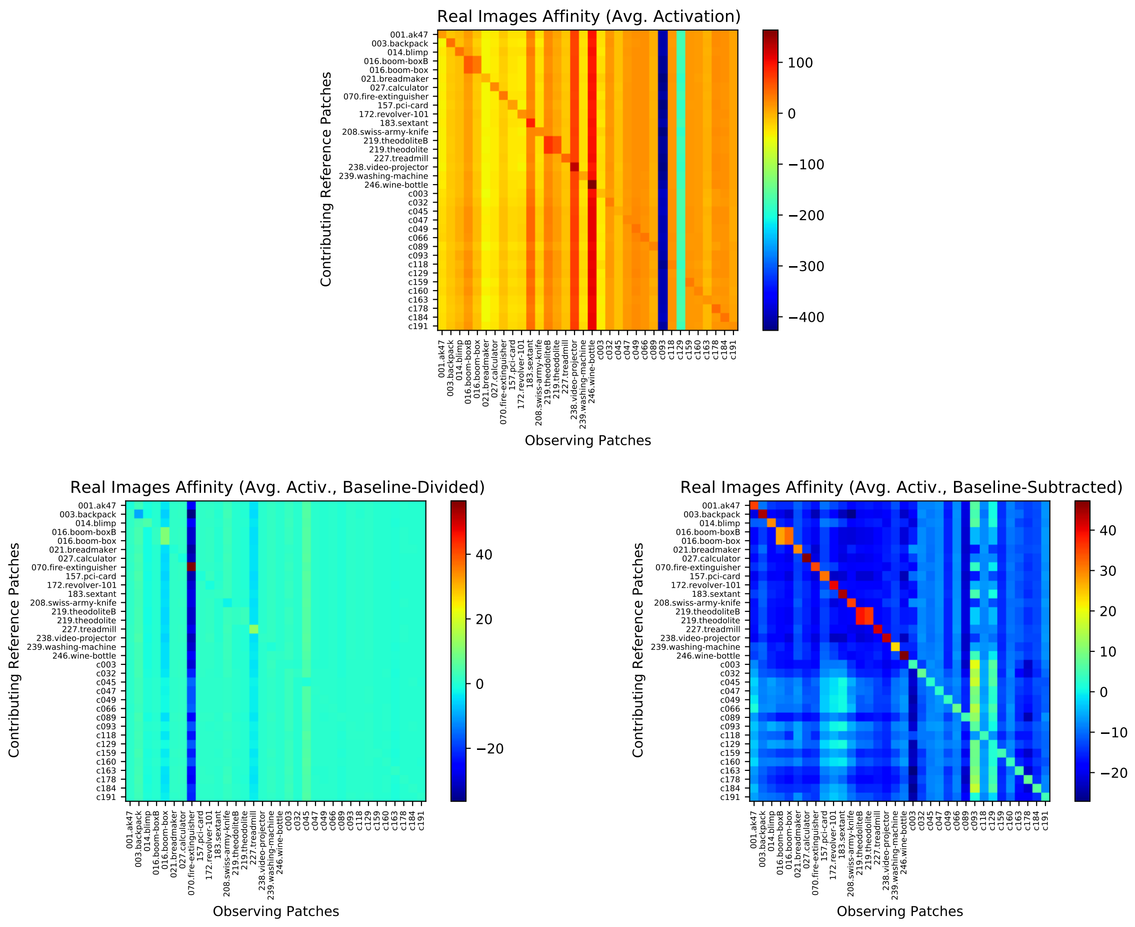


Figure 2.18: GANsemble firing affinity matrices ***on real images*** for a) raw-activation, b) baseline-division, c) baseline-subtraction. The patterns are grossly similar as for the fake images, except that the quadrants have traded off somewhat in vareigation and the self-activation diagonal is zero as expected for inter-stuff comparisons but high for inter-thing autoactivation, illustrating that GAN discriminators after baseline subtraction fire towards a high ***and*** a lower baseline point to the target class depending on whether it is seen as a real or a fake, with other (e.g. sub-baseline) activations portending a non-target patch.

2.2.2 Naïve argmax classification, and synthesis failures as problematic defectors on the committee

Now that the ensemble is established and a transformation discovered to its internetwork firing affinity matrix that yields graded dissimilarity, the stage is set for the ensemble to perform classification.

For each classifier d_k in the battery of networks, presenting a patch p leads to a vector of emissions composed of each patch-network combination activation $(d_k(p))$. We might call this vector the label (or activation, since the WGANs are unrectified) emission spectrum of the GANsemble. The term spectrum is chosen to provide loose analogy to chemical and light analysis techniques appealing to an estimate of "energy" or "power" which exists on an ordered determining axis. No single fair and deeply meaningful way of ordering textures is established, but it is suggested in the possibility of a Things vs. Stuff dichotomy that textures will eventually prove locatable even within the metacategories on a multifactorial composite notion of complexity or computational difficulty of processing.

Hence, we consider perhaps the simplest method for combining the judgment of impaneled ensemble members as expressed in the activation spectrum, the naïve argmax classifier:

$$\hat{y}_p = \operatorname*{argmax}_{d \in D} d(p)$$

Metaphorically, this corresponds to the committee of networks choosing the "the loudest voice". At the risk of personifying relatively uncomplicated neural networks, it is naïve in assuming that committee members will press their individual claims honestly. Of course, one modification suggested already by the data was the subtraction of the Monte Carlo estimate of the expectation of the activation of a GANsemble member to fakes from its own associated generator when given random noise (i.e. baseline subtraction).

$$\hat{y}_p = \operatorname*{argmax}_{d \in D} \left(d(p) - \mathbb{E}(d(g_d(z))) \right)$$

If the activation of the discriminator is stable over time, or for nearly everywhere past the initial steps of training, then this is a possibly workable assumption. We will see in subsequent sections that being almost-fully-naïve does happen to be practically workable (although only for the one-half of the Things vs. Stuff dichotomy where graded negativity of the affinity matrix exists in some measure for fakes and reals).

However, if the level of activation is developmentally-dependent, and some networks are at an arrested state of development, then one possibility is that the developmentally-arrested networks may emit systematically lower activation on average regardless of input compared to peers. This would cause the naïve argmax classifier never to declare for these networks' classes. A much worse result for the classifier manifests if the opposite possibility is realized: with a systematically high activation that is not conditional on input, the anomalous development classes emit a high value for all presented classes, winning all or nearly all contests. In this context, the classifier becomes useless.

Under this possibility, which is actually the case, synthesis failure classes in a committee therefore present an existential threat to GANsembles using just the naive argmax classifier and therefore the prevailing action of one network. This suggests two important research thrusts: settling on a decent classifier that is not as naïve or easily manipulated as the argmax classifier employing all joint activation in the emission spectrum, and automatically detecting synthesis failure with neither human observers nor specialized auxiliary neural networks trained to find synthesis anomalies. This kind of quality control network could realistically be brought to bear in non-GAN parametric texture synthesis, such as the Portilla and Simoncelli models [132], which have extensions to color textures where the color channels can become focally decoupled in mixing the feature vectors of texture statistics to produce novel textures, producing local color anomalies that would have to be statistically repaired and patched, perhaps using quite simple statistical tests of variance on windows and nonparametric [45] or stitching-based "synthesis" for texture anomaly inpainting. While auxiliary networks for synthesis quality control present an interesting direction for study, it remains possible as conjectured above that the telltale signs of synthesis nonconvergence are discoverable from variance in the model weights.

2.2.3 GAN-origin discriminator interclass model parameter distances

The pairwise firing affinity matrices just computed are useful for considering the joint interaction behavior of the GANsemble outputs, but while this kind of analysis is top of mind, it is convenient to consider the similarity of the models themselves, not just the classes as viewed by the collective of models. Deep neural networks facilitate massively multiple realizability of image judgment operations and it is interesting to discover if any weak constraint on this realizability leads to detectable similarity between models trained to deal with classes that are similar. Particularly if GANs are glorified memorizers of their input, we should expect to find some similarity.

Measuring the effectiveness of class similarity induction by a committee of networks is not completely straightforward. It would be possible, although resource-intensive, to assemble a panel of human subjects and have them create a set of similarity judgments over classes, for example by way of analogy (e.g. 98 percent of subjects agreed that theodolites were more similar to sextants than backpacks were to sextants) and to take the high-concordance rules and use them as inputs into a massive constraint-satisfaction problem to score GANsembles on their agreement with the greatest number of prevailing empirically-discovered analogical constraints. If there were a rigorous idea of the qualitative "principal components of texture" or something you could imagine as the fairest set of adjustment "knobs of texture", then other neural networks could be trained to discover these qualities, regress unseen textures reliably on them and compute an agreeable composite of these quality dimension wise features. Of course, there may not be natural texture dimensions of a parsimonious sort if, analogous to color sensing, there are texture channel (texture channels were discussed by [135]) activations and those channels are determined, by say, the action of a large set of completely random noise filters that are different for every perceiver.

In some highly constrained and contrived spaces of texture, there are natural "knobs". For example, you can consider the highly constrained set of textures that come about from computing the 2-dimensional inverse discrete Fourier transform of a forward-transformed blank image whose phase information is then corrupted by being painted over with energy according to a Lévy flight [146] or other random walk initialized by a few latent parameters. With Markov Chain Monte Carlo methods, you could even discover the likeliest parameter settings that produced a group of textures. In this and many other special class of texture investigations, such as work concerning isodipole textures [70] or simple point clouds [156], the stimuli are so completely non-naturalistic that these researchers are almost studying a completely different thing from researchers in neural networks who even use the concept of the degree of naturalism for practical purposes (e.g. constraining neural networks by the imposition of natural image priors, [126]) under the heading of research into "texture". In another situation, names are given to the parameters generating the image; this is the case with the libnoise [12] implementation of the popular 2D self-similar Perlin noise which allows you to specify dimensions like *lacunarity*, *octave count*, and *persistence* that all are meaningful with respect to either the calculation or the visual experience (i.e. high lacunarity noise has wide subjective "holes") of the produced patches.

This is not at all practically applicable to the space of naturalistic textures we contend with throughout life. Laborious work involving collecting multiple-stages of judgments can

[152] associate human preferences reliably with these subjective and “naturalistic” labels laymen or designers might converse about, like the humorous case of “premiumness”, but the determination of the labels themselves is not systematic. In other words, we could be reduced to scoring our GANsembles on similarity more objectively by running them through a layer of indirection to subjective judgments of arbitary features, like pointyness, ornateness, bulbousness, ocularity (perhaps this would be the confidence or multiplicity of containing an atrium-like inner hole or annulus in a patch), but it is not immediately clear what the value of that effort would be, except to see which architectures for synthesis or methods for embedding would have trouble with producing or closely placing patches containing elevated or reduced magnitudes on these subjective quality dimensions. The special case of considering class identity as wholly categorical and not decomposable into qualities leads to a brutish criterion of measuring similarity by at least assuring that the duplicates of the same class are located in similarity space amongst themselves across multiple recognizing or synthesizing network realizations, or analysis methods of the outputs of those realizations.

Measuring the similarity of models rather than classes is comparatively much easier, but it is important to remember that it is not completely determined; it is just the case that the pure and applied mathematical community has settled on a few different notions of “distance” (beware: here we use distance and metric colloquially rather than according to their technical definitions!) that are unique enough from each other to be considered distinct and that each have a practical purpose in various fields of inquiry. This investigation considers only a handful of them that are considered most important.

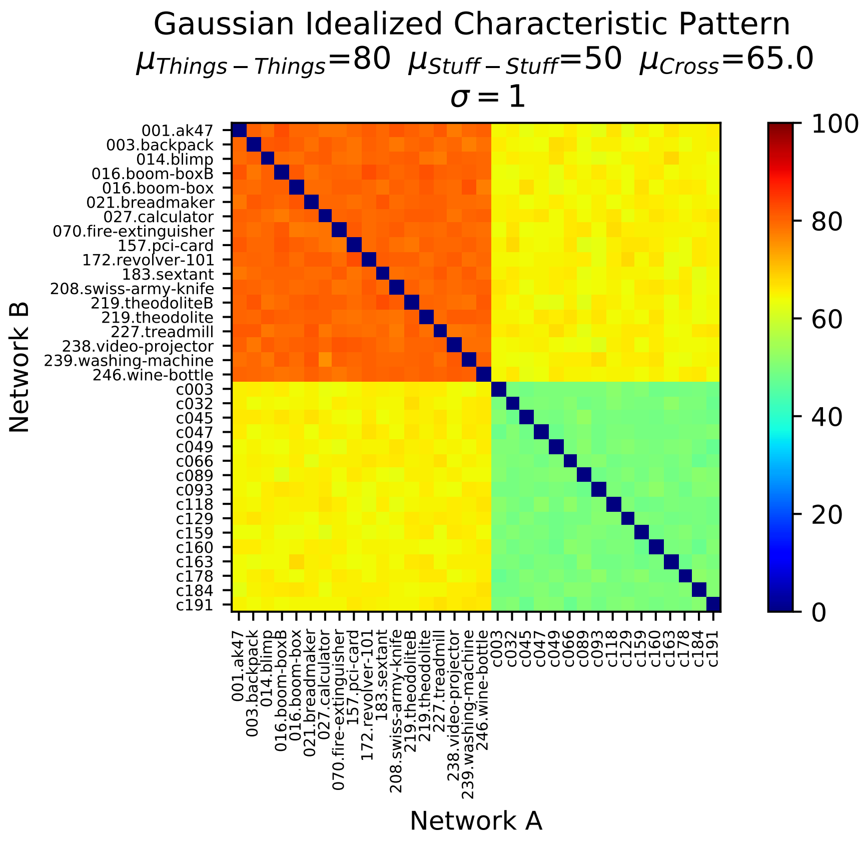
Consider the pairwise *model parameter* affinity matrix $\Xi(N_1, N_2, f)$ to measure the distance between N_1 and N_2 according to some single “distance” function f. In a deep convolutional neural network, the network has trainable (updated by gradient descent) and not trainable weight and preprocessing parameters, and many of the trainable ones concern units whose parameters are grouped in discrete layers. For example, there is the notion of a first, second, third, fourth, and fifth convolutional layer in our GANsemble discriminator networks when counting away from the input towards the terminal densely-connected layer and the output neuron whose activation stands for that of the whole network as far as the ensemble is concerned. In the discriminator, our layers have biases, and we can consider the bias weights alone as associated with a conceptual layer if the definition of layer is relaxed to not strictly mean a discrete stage of transformation of all the input.

Therefore we can stack the pairwise affinity matrices of different layers to produce the pairwise model parameter affinity tensor $\Xi(N_1, N_2, L_i, f_j)$ where i varies the index of logical weight groupings temporarily considered "layers" and j varies the choice of distance functions considered. Holding j constant, we have a stack of matrices corresponding to the actual affinity matrix (cf. the pseudo-"affinity" matrices of firing) between the two networks based on the distance between their parameter vectors. We only consider the Dense and Conv2D(Transpose) layer weights (filter and bias) in this investigation, even though parameters of BatchNorm and LayerNorm, which include learned elements, can also be considered. So these are proper weight vectors. While in a library such as Keras these weights may be stored as tensor objects in light of meaningful spatial dependence (e.g. slices of the convolutional block activations are channel activations) that is perhaps also meaningful for distributing computation in memory and affecting locality, we flatten them to vectors to treat them as homogeneous quantities and to permit most distances to be calculated on them.

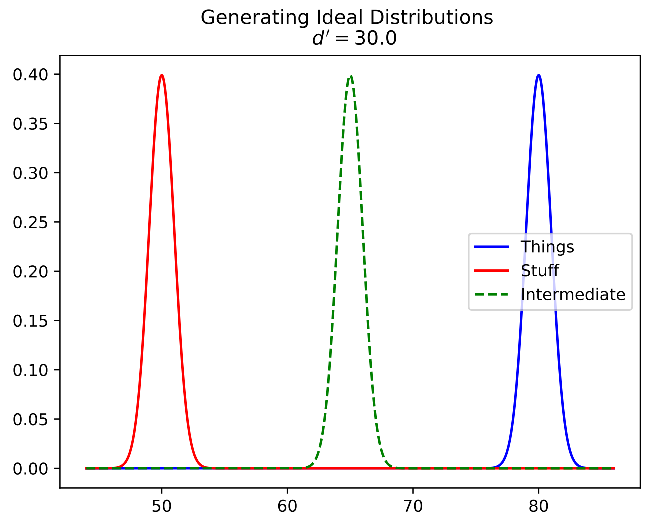
The pairwise matrix $\xi(N_1, N_2)$ for a layer and a specified distance function can have a number of extreme appearances in view of a possible Things vs. Stuff dichotomy. The most indicative of the dichotomy might be called the *characteristic pattern*. In all valid patterns, the distance between a network and itself should be zero, corresponding to the diagonal. When an equal number of things classes and then stuff classes are grouped together (i.e. all 16 things, then all 16 stuff) on the axes of ξ, four quadrants develop. In the case of imbalance between the things and the stuff class, ξ is still square but the quadrants are not proper quadrants. The upper-left quadrant (conventionally, II) represents Thing-to-Thing intermodel distances. The lower-right (IV) represents Stuff-Stuff distances. The remaining quadrants (I,III) represent Thing-Stuff, or "cross"-metacategory comparisons. While this matrix exhibits symmetry when only symmetrical distance functions are chosen, not all distance functions are symmetrical (e.g. the directed Hausdorff for point sets, and the Kullback–Leibler for distributions), so we can mock the "idealized" characteristic pattern by drawing from a distribution separately per quadrant (for the symmetric case, one triangle is not allowed to vary).

In the expected pattern, the least dissimilarity (i.e. 0) is always expected for a network being compared against itself, and this expectation should only be violated when using approximate or stochastic distance functions. The next least dissimilarity is expected among Stuff classes. Since Stuff classes are homogeneous within the class, are mostly statistically stationary as you move around the patch, and can be recognized and synthesized by techniques of lesser "power" (e.g. primitive filterbank recognition, or Heeger & Bergen [74] or Portilla & Simoncelli [132] for Stuff synthesis), we might *a priori* expect less distance in the models (models for Stuff are closer to each other than models for things, which may be aligned or may not be aligned with the models being easier to morph into each other by retargeting gradient descent). Quadrant IV should then exhibit an elevated, but only slightly elevated average value. In the "idealized" case, the variance is low so that the dichotomy is clear (the kurtosis might be negative if we used a uniform distribution) within the quadrant – but the actual interesting pattern would develop if model distances were closest to actual similarity, and the presence of graded positivity of the dissimilarity in this quadrant would be necessary but not sufficient for the model distance capturing similarity. We could correlate the firing affinity matrices cellwise with model matrices, but in reality we possess no known very competent similarity oracle for the purposes of this initial investigation, and won't assume directly that the firing affinity matrices constitute such oracles. Quadrant II should be like

Quadrant IV but at a notably higher dissimilarity level, connoting the greater model distance within Things (amongst Thing-Thing network parameter comparisons). The Thing-Stuff comparisons quadrants (I, III) should be of intermediate dissimilarity. Figure 2.19 shows the mocked-up pattern when a high, low, and intermediate value are chosen to control generating equal-variance Gaussian distributions with narrowly tuned $\sigma = 1$ for the quadrants.

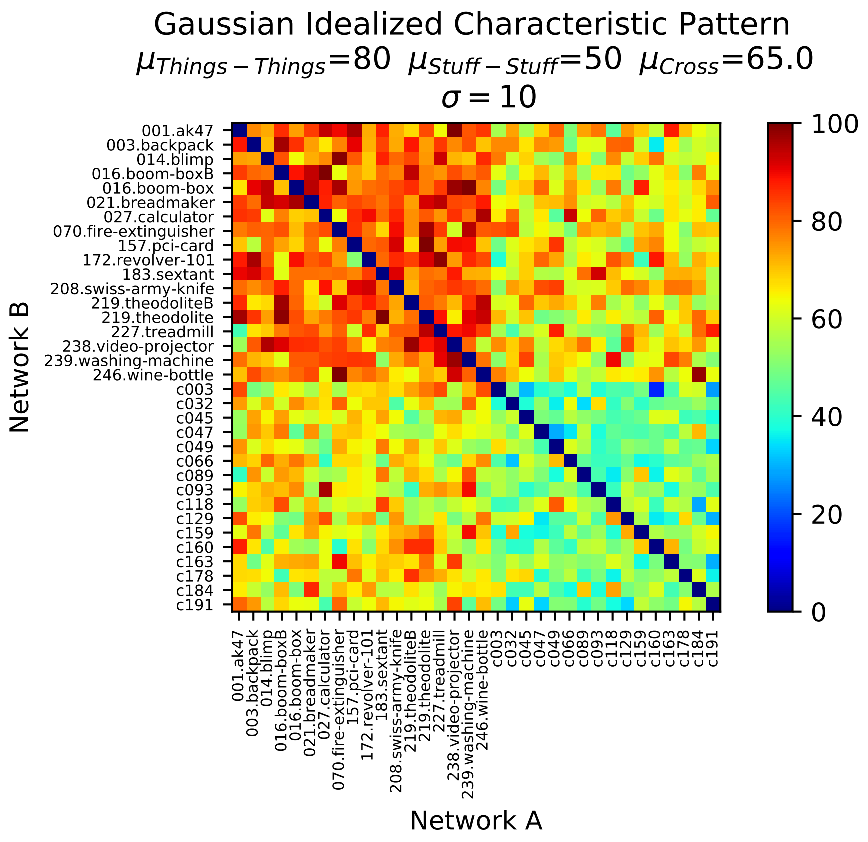


a)

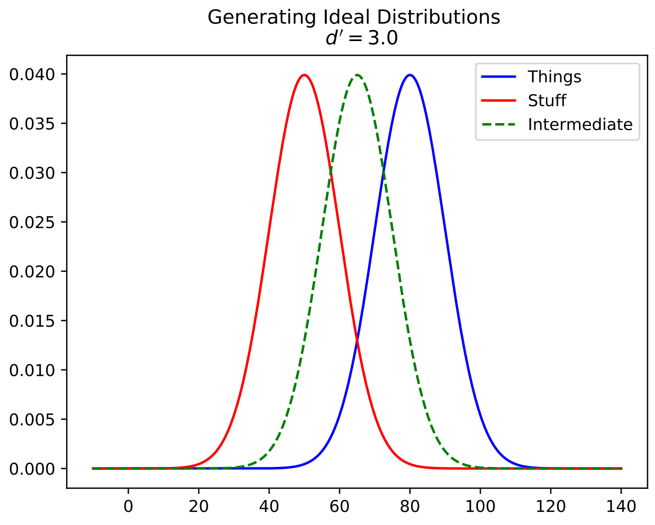


b)

Figure 2.19: a) The idealized *characteristic pattern* of the Things vs. Stuff dichotomy in model distance "quadrants" in which there is higher dissimilarity in Thing-Thing model comparisons (Q2), lower dissimilarity in Stuff-Stuff comparisons (Q4), intermediate dissimilarity in cross-comparisons (Q1 and Q3), and definitionally no dissimilarity in self-comparisons (matrix diagonal). Simulated values within "quadrants" are drawn according to b) idealized normal distributions with sigma 1 and resultingly high "idealized d-prime" separation ($d' = 30$).



a)



b)

Figure 2.20: a) Weaker version of the idealized characteristic pattern of the Things vs. Stuff dichotomy. At $\sigma = 10$, the pattern is much weaker to see because the b) generating distributions for the quadrants produce much more mixed values with the same mean separation but increased bandwidth. Alternatives to the characteristic pattern are 1) the inverse characteristic pattern (higher dissimilarity among Stuff networks), 2) severe cross-comparisons pattern (higher dissimilarity in quadrants 1 and 3 corresponding to Thing-Stuff comparisons), and 3) and roughly-constant pattern (which may be low or high relative to true outliers like synthesis failures).

When the bandwidth of the Gaussians is widened to $\sigma = 10$, the characteristic pattern becomes noisy and weakened, appearing as it does in Figure 2.20. The generating distributions exhibit lower separation as the mean activation values have not moved to compensate, and so their degree of overlap increases. The reader with signal-detection theory experience will note that this notion of separation (between two distributions, the signal and the noise distribution) coheres with the idea of d-prime. As listed in Chapter 1, d' is conventionally defined specifically in terms of the probit function on the false alarm and hit rates, but an idealized d-prime is really just a distance between two means accounting for spread, where there are various possible ways of accounting for spread; for example the difference family of effect size measurements including Cohen's d and Hedge's g are mean differences that express slightly different opinions on the proper way to pool effect size.

If we arbitrarily take the "idealized" rather than the empirical d-prime to be equal to the less-commonly encountered root-mean-square variant of Cohen's d ([31],eq. 2.3.2)

$$d'_{\text{ideal}} = d_{\sigma_1 \neq \sigma_2} = \frac{\mu_1 - \mu_2}{s_{p_{\sigma_1 \neq \sigma_2}}} = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2 + \sigma_2^2}{2}}}$$

It can be seen that with the increasing dissimilarity quadrant generating distribution sigma, this "idealized" d-prime or Cohen's d decreases, showing less of a chance of taking a random value from ξ_{mock} and associating it and those near it in value uniquely with a quadrant.

The second extreme appearance to be considered is the "inverse characteristic pattern" which merely exchanges the lower and higher activation value connoting more model similarity and less model similarity. Observing the inverse characteristic pattern on a dissimilarity matrix would connote high Stuff-Stuff dissimilarities and low Thing-Thing dissimilarities. Of course, when the dissimilarity is exchanged for the similarity there is a change in polarity, so consider the characteristic pattern for *similarities* to resemble the inverse pattern for *dissimilarities* and vice versa.

Another extreme appearance, then, would be observed low values of dissimilarity in the metacategory-consistent quadrants (Thing-Thing or II, and Stuff-Stuff or IV) and high values in the ordinarily intermediate metacategory-inconsistent or Thing-Stuff or cross-comparison quadrants (I,III). When the cross-metacategory comparison quadrants are high, consider this the *cross-comparisons most severe* case.

Finally, the last extreme appearance considered is that of the dissimilarity being roughly constant across all quadrants. The degree of constancy becomes referential to outlier comparisons which are substantial. Synthesis failure classes, where they induce large dissimilarities with the internals of comptetently trained networks can set the degree of this high dissimilarity fluctuation. If synthesis failure class activation values dominate the variation, they would tend to obscure the pattern somewhat in an automated heatmap process, so if the characteristic pattern appears, this is an additional testament to the strength of the apparent dichotomy.

A last possibility is no pattern at all, or seeming visual noise, consistent with taking the mock characteristic pattern and letting the σ increase.

With the characteristic pattern and the other conspicuous cases defined, it is now possible to consider each $\xi \in \Xi$ to determine at which stage of visual neural differentiation or inverse visual (hallucination) differentiation the Things vs. Stuff dichotomy develops. In the spirit of varying distance as established by the transition from the GAN to the WGAN, the question of whether it develops early or late (e.g. in Conv2D layer 1 in the discriminator or Conv2D layer 5's bias) may critically depend on the nature of the distance chosen to examine it.

Notably, the boom-box class had two network variants, one with the benefit of 25,000 training units, so this extra training could be detected too if the activation were the result of the amount of training and not a mere competence-incompetence distinction. The duplicate theodolite networks (identical training time of 15,000 units) give a rough idea of the degree of multiple realizability with the same training, as it may be that networks targeting synthesis of the exact same image set, with the exact same training time, and the exact same architecture, loss function and optimizer could exhibit completely different weight structure.

Each distance function detailed below contributes to a cell in the ξ matrices the result of comparing all the weights associated with a logical grouping of weights across two networks.

Corresponding-weight disparity: MSE

The most immediately obvious distance to calculate is the familiar mean-squared error used, arguably poorly, to assess the average non-correspondence at the pixel level in images.

Recall that it, for the case of I being one image flattened to one dimension, and $\hat{I}$ being a comparison image with the same N pixel locations, is defined:

$$\mathrm{MSE}(I, \hat{I}) = \frac{1}{N} \sum_{n=1}^{N} (I[n] - \hat{I}[n])^2$$

The mean squared error takes the average of the squared pixelwise or valuewise disparity. When we use it on the weights of a neural network in some layer, we flatten the weights of that layer to a single dimension and compute the MSE on that vector.

With millions of weight locations to compute, there are bound to be outliers and the MSE is sensitive to them. However, it is easy to compute, ubiquitous in image quality measurement comparisons, and has a satisfying low-level basis in the accumulating the average (squared) pixel disparity. It is perhaps more clever to instead use a variation of this idea that is less sensitive to outliers, for example the median absolute deviation, which is usually advocated (e.g. [105]) over standard deviation for characterizing magnitude of deviation in the form of the median of the absolute-valued deviations from the median, but could be instead set to track the median of the absolute valued difference of corresponding weights, but this is not standard practice in either image quality assessment or the loss function of neural networks. Suffice it to say that the MSE may not be very informative, and that other average direct correspondence measures could be slightly less uninformative, but average non-correspondence being low for blur on homogeneous images may say something also about its inability to discern neural network weights, which look may look reasonably homogeneous when plotted as an image.

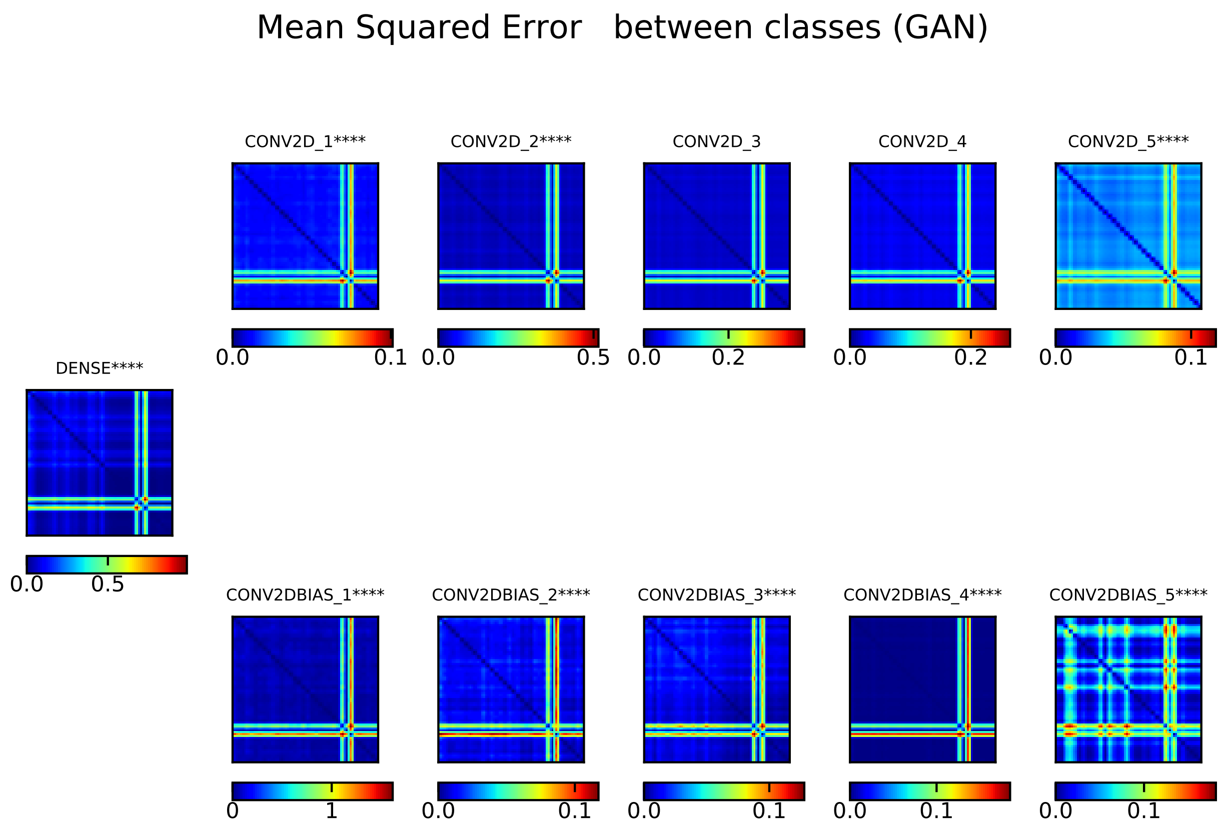


Figure 2.21: Discriminator weight mean-squared error (MSE) distance matrices calculated for each convolutional layer (and for the learnable layer biases separately) and the terminal Dense layer. Stars next to the layer name conventionally indicate the significance of comparing the things-things to the stuff-stuff comparison quadrant using a Wilcoxon rank-sum test, *which is highly misleading given the lack of independence and the susceptibility of hypothesis tests to outliers*. The synthesis failure classes (bright lower cross converging in lower right) dominate the comparisons as their network weights are strongly different to the others on a point-to-point basis. The strongest Things vs. Stuff distinction can be perceived in the terminal Dense layer, the 3rd convolutional layer's bias, and the 5th convolutional layer, with things networks being more dissimilar from each other in these areas. By the final convolutional layer, some strong dissimilarities have emerged in the bias. Developmental differentiation under MSE for content/class is thus relatively late for GAN discriminators. *Hotter colors indicate more dissimilarity between a pair of networks.*

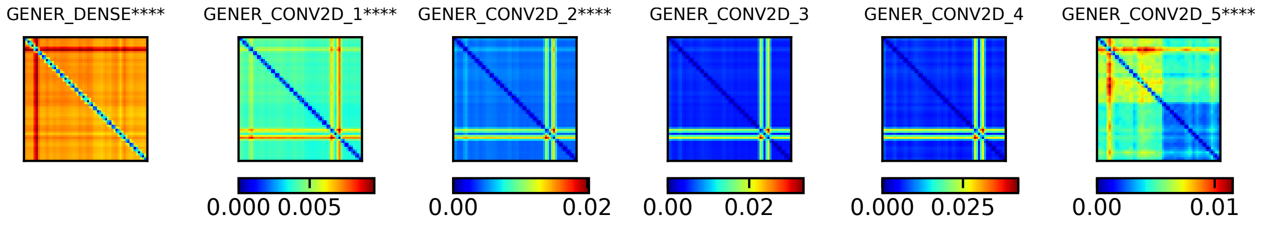


Figure 2.22: Generator weight mean-squared error (MSE) distance matrices calculated for each convolutional layer (there are no learnable biases) and the initial Dense layer. The extra training of the second boom-box network is clearly visible in the upper left, and the synthesis failures also remain visible. A pronounced difference in Thing-Thing vs. Stuff-Stuff comparisons is only seen in the beginning Dense layer which translates the incoming noise of encoding space and then this strongly reemerges by the final convolutional layer which produces the output patch. The characteristic pattern consists of close to zero individuation between stuff, maximum individuation between things, and intermediate individuation where Stuff networks are compared to Thing networks. Interior layers in GAN generators are thus relatively similar on a point-to-point basis despite large changes in content. *Hotter colors indicate more dissimilarity between a pair of networks.*

Geometric-location-of-model disparities: L1 and L2 Norm

Alternatively, networks can be imagined as points in a high-dimensional space. Our discriminators, for example, can be considered in terms of their learnable parameters as existing as points in a 4,356,289-dimensional design space.

Then the natural distance is the Euclidean distance, the Minkowski distance associated with (the difference is taken in each term contributing to) the L^2 norm in the family of L^p norms:

$$d(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + ... + (a_n - b_n)^2}$$

$$||v||_2 = \sqrt{(v_1)^2 + (v_2)^2 + ... + (v_n)^2} \quad ||v||_p = \sqrt[p]{(v_1)^p + (v_2)^p + ... + (v_n)^p}$$

Exploring other p-norms, the infinite norm $||v||_\infty = \max(v_1, v_2, ..., v_n)$ doesn't contain much information. But the L1 norm, whose corresponding Minkowski distance is also known as the Manhattan (or, for easily-intuited-to-be-related reasons, the taxicab) distance, is interesting because it is at least as big as the Euclidean distance and because the distance is simply accumulated on each coordinate axis (dimension).

$$d_1(a, b) = |a_1 - b_1| + |a_2 - b_2| + ... + |a_n - b_n|$$

Of course, the L1 and L2 norm can be expected to be quite similar in how they portray the pairwise relationships between networks because they are distinguished only in the squaring of the terms and then the square root of the sum. However, the squaring is likelier to be susceptible to the influence of outliers, and since some outliers are eventually going to surface in very-high dimensional data, it is of some interest to see if these distances differ qualitatively.

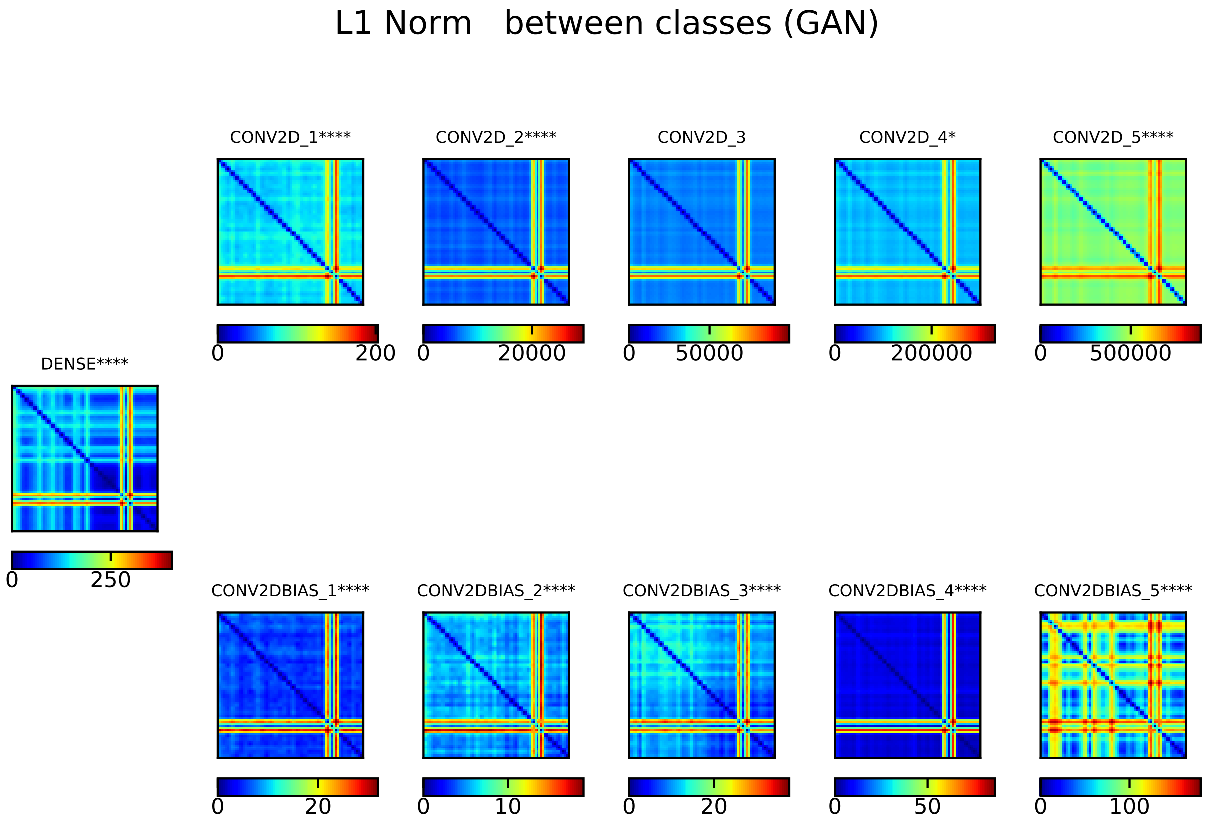


Figure 2.23: Discriminator weight Manhattan (L1 Norm) distance matrices calculated for each convolutional layer (and for the learnable layer biases separately) and the terminal Dense layer. Unlike the MSE, the L1 norm notion of distance does not reveal the strong characteristic pattern of Thing-Thing vs Stuff-Stuff separation in the convolutional layers proper, but does consistently in the bias (where dissimilarity grows until the 4th convolutional layer's bias, and then recurs very strongly in the final bias). The terminal Dense layer also exhibits a pattern of low L1 norm separation between Stuff networks. *Hotter colors indicate more dissimilarity between a pair of networks.*

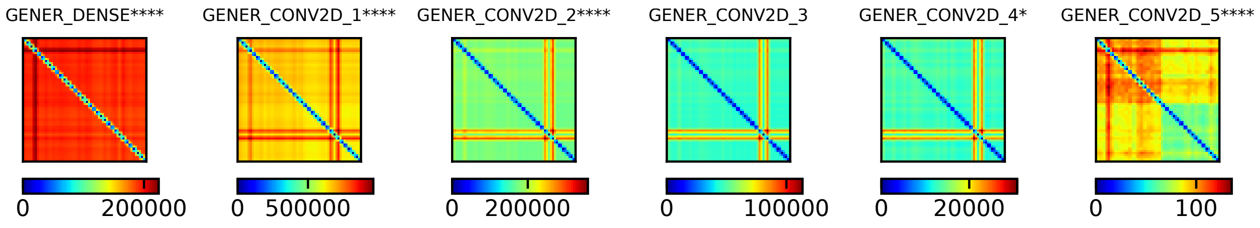


Figure 2.24: Generator weight Manhattan (L1 Norm) distance matrices calculated for each convolutional layer (there are no learnable biases) and the initial Dense layer. The characteristic pattern is only observed in the last convolutional layer of the generator, suggesting that content specialization is late as observed through the L1 norm. *Hotter colors indicate more dissimilarity between a pair of networks.*

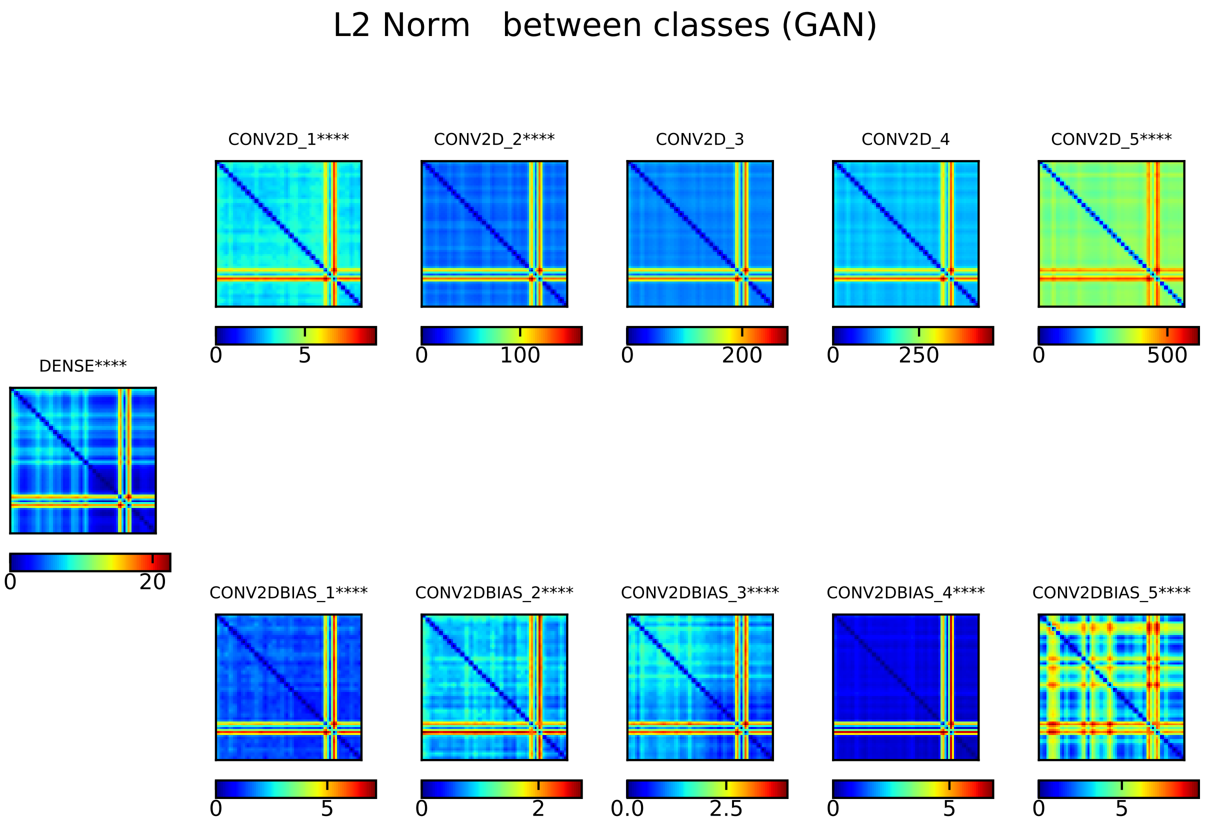


Figure 2.25: Discriminator weight Euclidean (L2 Norm) distance matrices calculated for each convolutional layer (and for the learnable layer biases separately) and the terminal Dense layer. The layer disparities are extremely similar to those calculated for the L1 norm, which is reasonable given the similar formulation of the L1 and L2 norms and the extremely high dimensionality. Blue pixels indicate pairs of networks that are relatively more close to each other in a 4.3M-dimensional space when the networks are thought of as points. *Hotter colors indicate more dissimilarity between a pair of networks.*

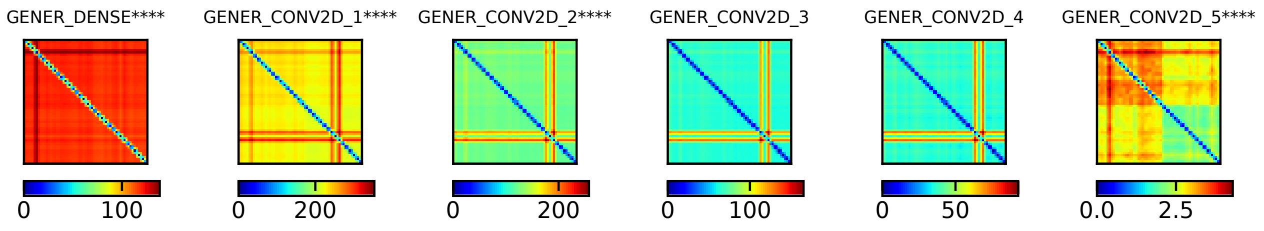


Figure 2.26: Generator weight Euclidean (L2 Norm) distance matrices calculated for each convolutional layer (there are no learnable biases) and the initial Dense layer. Again, the characteristic pattern is observed for the generator only in the final convolutional layer. The pattern is again extremely similar to that of the L1 norm, suggesting that having two of these Minkowski norms is redundant when comparing GANs. Blue pixels indicate pairs of networks that are relatively more close to each other in a 21.6M-dimensional space when the networks are thought of points. *Hotter colors indicate more dissimilarity between a pair of networks.*

Weight distribution disparities: Jensen-Shannon & Signchain

The direct notions of distance can be supplemented with a different notion of disagreement of network parameters. If a similar pattern of connectivity in a layer emerged but with the pattern shifted or otherwise not aligned, ordinary distances would fail to credit the similarity. One way of seeing if the values are comparable is to examine the distance between the empirical weight distributions.

One such distance is the Jensen-Shannon Divergence, which is chosen over the Kullback-Leibler divergence because the former is symmetric, or insensitive to the order of the arguments.

The Jensen-Shannon Divergence is:

$$D_{JS}(P||Q) = \frac{D_{KL}(P||\frac{(P+Q)}{2})}{2} + \frac{D_{KL}(Q||\frac{(P+Q)}{2})}{2}$$

where the KL divergence (or relative entropy) between two distributions P and Q over the same space X is:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

In this case, we actually use the derived Jensen-Shannon distance d_{JS}, which is the square root of the divergence $d_{JS} = \sqrt{D_{JS}}$.

As stipulated in Chapter 1, the empirical distributions are extemporaneous (meaning that the range of observations and bins is not preplanned), so the Jensen-Shannon distance is not necessarily measuring the same weights equally. If weight ranges were vastly different between comparison networks, this would not be appropriate – even as a preplanned histogram for all comparisons might also be misleading if, choosing the largest range in one example, it put all bins in one place through the in-common histogram function. However, we already know through the weight clipping of WGAN (vs. the gradient penalty of WGAN-GP) that neural network weights within a circumscribed range still function to compute as long as the range is suitably commodious relative to precision (e.g. [8] prescribed a clipping parameter of 0.01, leaving weights bounded in [-0.01,0.01]), and if neural networks being trained mostly fill out representation space (this appears an unstudied question), then this is an acceptable approximation.

As a companion to the Jensen-Shannon distance, we can consider the signchain distance proportion, which also can measure how well discrete distributions cohere, although on a sub-ordinal basis. As such, if it measures something like the JS distance does, we should see a similar pattern in the dissimilarity matrix, but one that appears to be a weaker signal than the JS distance.

The signchain distance proportion $\hat{\Upsilon}(\vec{a}, \vec{b})$ was defined in 1.2, but is simply the proportion of bits in the signchain which do not agree.

In total, the weights of a layer are flattened for the two networks, the extemporaneous histogram of 32 bins is taken separately so that resulting vectors have their JS distance calculated, and also the extemporaneous histogram of 33 bins yielding 32-bit bitstrings is taken and $\hat{\Upsilon}$ is calculated for the bitstrings.

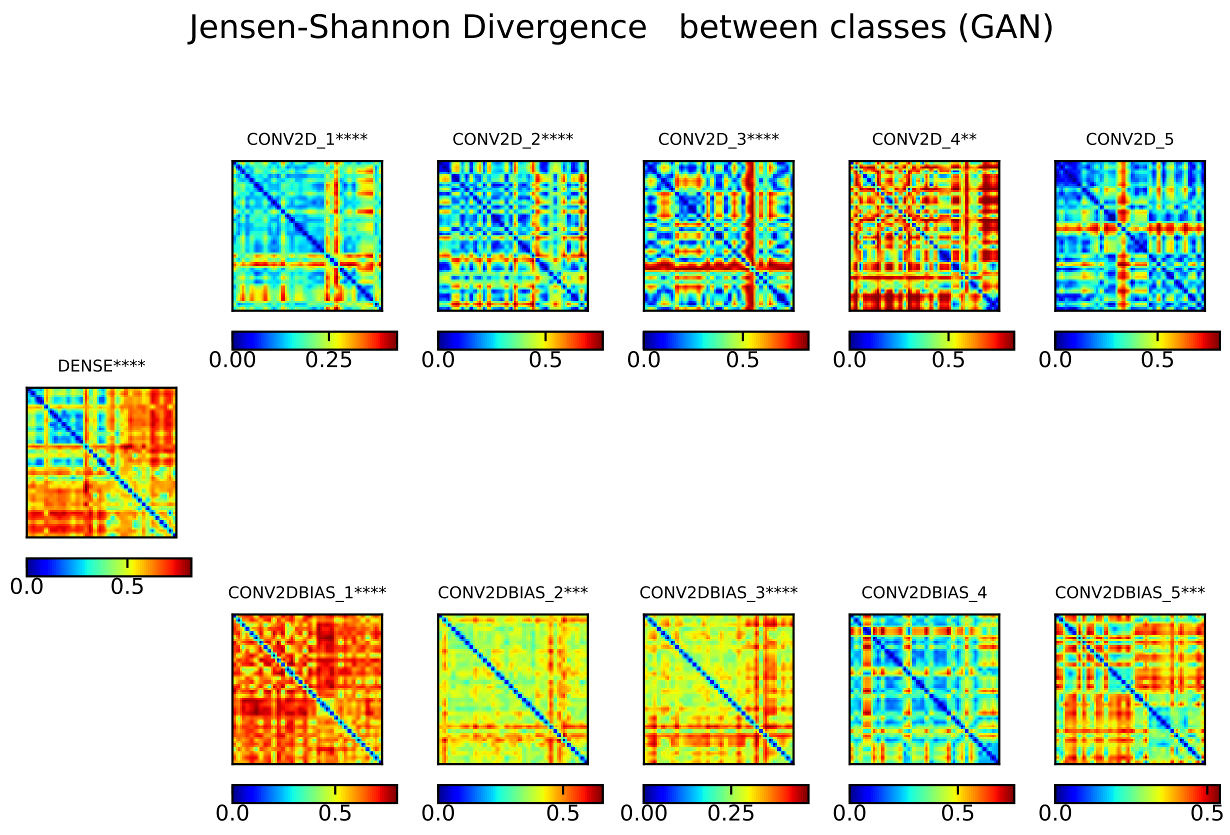


Figure 2.27: Discriminator weight distribution Jensen-Shannon distance matrices calculated for each convolutional layer (and for the learnable layer biases separately) and the terminal Dense layer. Distributions are estimated from a 32-bin adaptive histogram of the weights. While quadrants can be visually individuated (especially in the Dense layer and final convolutional bias weights) with some difficulty, the characteristic pattern is not visible. The things vs. stuff cross-comparisons show that network weight distributions are maximally individuated when comparing Thing networks and Stuff networks. Things and Stuff GAN discriminators thus seem to differ detectably in their weight distributions in the final Dense layer, whereas their spatial locations and MSE match differ more among Things networks. *Hotter colors indicate more dissimilarity between a pair of networks.*

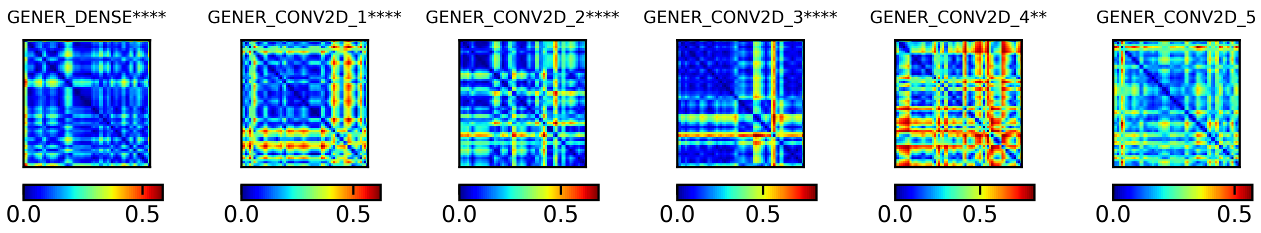


Figure 2.28: Generator weight distribution Jensen-Shannon distance matrices calculated for each convolutional layer (there are no learnable biases) and the initial Dense layer. No characteristic pattern is observed, nor is any fully consistent pattern. There is somewhat reliably less divergence among Thing classes. *Hotter colors indicate more dissimilarity between a pair of networks.*

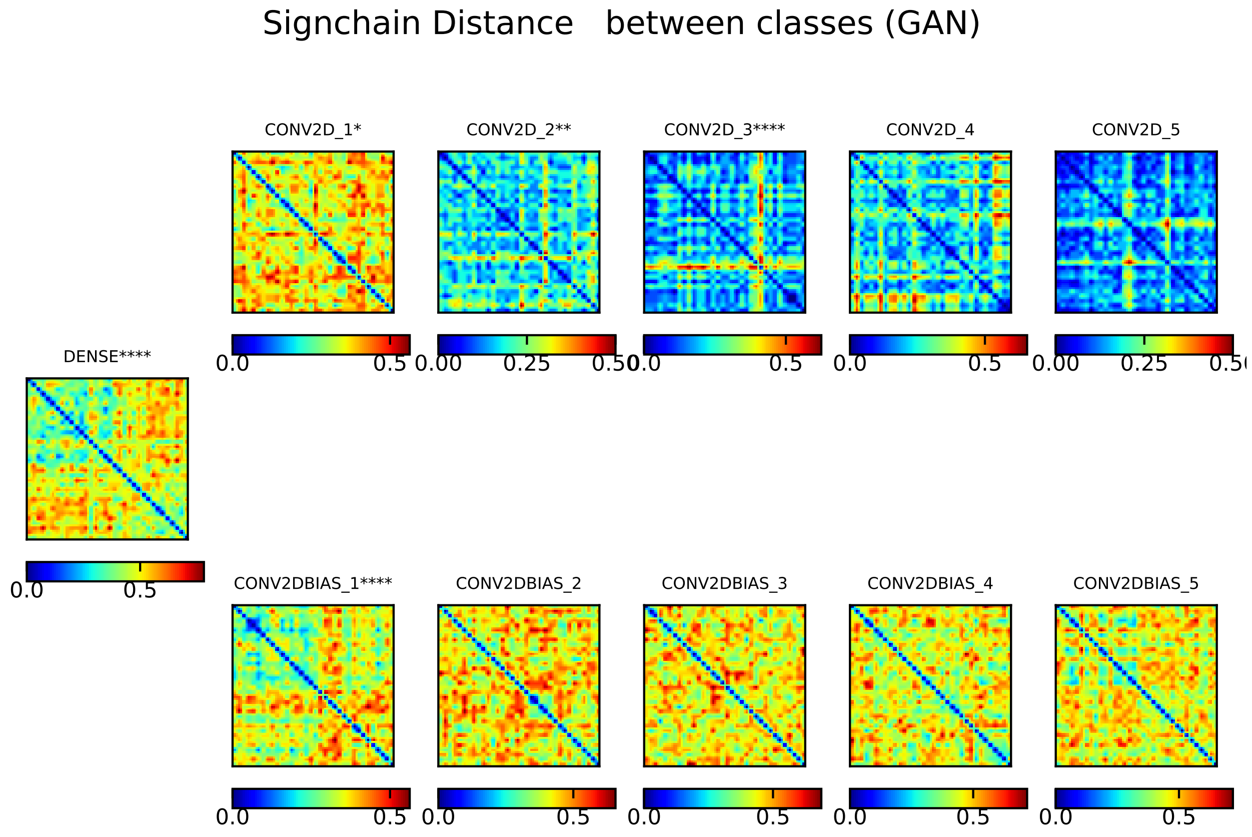


Figure 2.29: Discriminator weight distribution Signchain distance matrices calculated for each convolutional layer (and for the learnable layer biases separately) and the terminal Dense layer. Distributions are estimated from a 33-bin adaptive histogram of the weights, giving a 32-bit signchain. The Dense and first convolutional bias layers are reminiscent of a weaker form of the pattern established for the Jensen-Shannon distance in the Dense layer. This suggests that the signchain distance only weakly performs a similar function to the Jensen-Shannon distance. The remaining patterns effectively resemble visual noise. *Hotter colors indicate more dissimilarity between a pair of networks.*

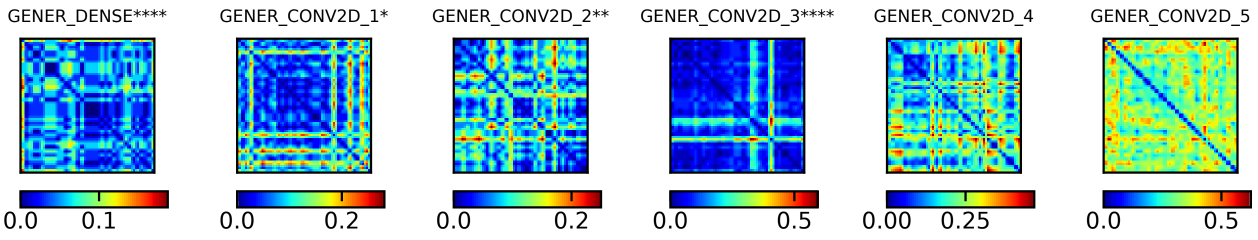


Figure 2.30: Generator weight distribution Signchain distance matrices calculated for each convolutional layer (there are no learnable biases) and the initial Dense layer. Every layer is grossly visually reminiscent of a weaker version of the GAN generator results for the Jensen-Shannon distance, with this relation being stronger than that for the discriminator. *Hotter colors indicate more dissimilarity between a pair of networks.*

Weight neighborhood "appearance" disparities: SSIM of "Weight Portraits"

In addition to the traditional notion of distance, the distributional divergences, and the point-to-point lack of correspondence, there is also the notion of visual nearness, or similarity. Of course, machine estimates of visual similarity are the province of neural networks. To avoid the problem of having to specify some auxiliary neural network and attempt the learning of discrete levels of similarity, a researcher might turn to one of the image similarities often said to be "perceptually-motivated". Perhaps the most famous putative perceptual quality measurement is the Structural Similarity Index metric, or SSIM, of Wang et al 2004 [174].

The SSIM is defined as a function (for simplicity, the combination function's exponents are set to 1) of three components, a luminance term, a contrast term, and a "structure" term:

$$S(x,y) = f(l(x,y), c(x,y), s(x,y)) = l(x,y)^{\alpha} \cdot c(x,y)^{\beta} \cdot s(x,y)^{\gamma}$$

The terms are computed where "luminance" is based on average pixel intensity and a denominator smoothing constant C_1 based on the pixel range of the image and a small coefficient,

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2\mu_y^2 + C_1}$$

constrast is computed with an exchange for the standard deviation,

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2\sigma_y^2 + C_2}$$

and "structure" is computed using the joint variance (read: covariance) over the product of variances (in total, the correlation coefficient):

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x \sigma_y + C_3}$$

Importantly, the means and standard deviations are *local*, computed in the neighborhood around a pixel for the pixel. The neighborhood is weighted with a symmetric Gaussian to falloff evenly so that closer pixels to the current pixel are weighted more and to reduce blocking artifacts. The MSSIM (or mean SSIM) is merely the mean of the scalar field created corresponding to image pixels for the SSIM. When the "SSIM" is reported, it is typically actually the *MSSIM*.

The implementation used in this investigation [164] ignores pixels around the edge in the SSIM field as the filter implementation may vary for how it deals with pixels whose windows go off the edge, and uses a 7x7 kernel and window, a $\sigma = 1.5$ for the Gaussian windowing, $K_1 = 0.01$, and $K_2 = 0.03$ for the constants K used to set the constants: $C_1 = (K_1 L)^2, C_2 = (K_2 L)^2$ based on the range L which is 255 for (the 8-bit) images, and the maximum 32-bit floating point number $\sim 3.4\text{x}10^{38}$ otherwise (including here, for weights).

The (M)SSIM is often shown in direct comparison to the MSE, by revealing some images with large perceptually apparent degradation from a reference but equal MSE. The SSIM of these images varies. However, there is little reason to suspect that this cannot be just as easily done for the equal-SSIM "hypersphere". How to accomplish that was studied by Dosselmann and Yang in 2008 [41], who characterized this kind of demonstration as a "deception" and showed that the luminance and contrast terms taken away yielding just the structure term was heavily related to MSE. Amusingly, they derive equations for transferring back and forth from each term (with the exception of contrast) to MSE! As SSIM has proliferated and has become used in computer vision and computer graphics (and also the entertainment industry, with the original authors winning an Engineering Emmy in 2015), researchers [127] have become very concerned with discovering its edge cases, such as where there are patches of low-intensity or regular intermediate-range variation that is missed due to the windowing.

While the SSIM is not deeply reminiscent of the human visual system in any way, and this is perhaps obvious to some, it does degrade very severely with the addition of noise, and noise is what our weights might look like. To get the weights in a format that is suitable for the 2D windows of SSIM, the weights are flattened as usual but then reshaped with suitable trailing zero-padding into the nearest square matrix, which we may call the *weight portrait*. The MSSIM of the two weight portraits are what is compared across networks.

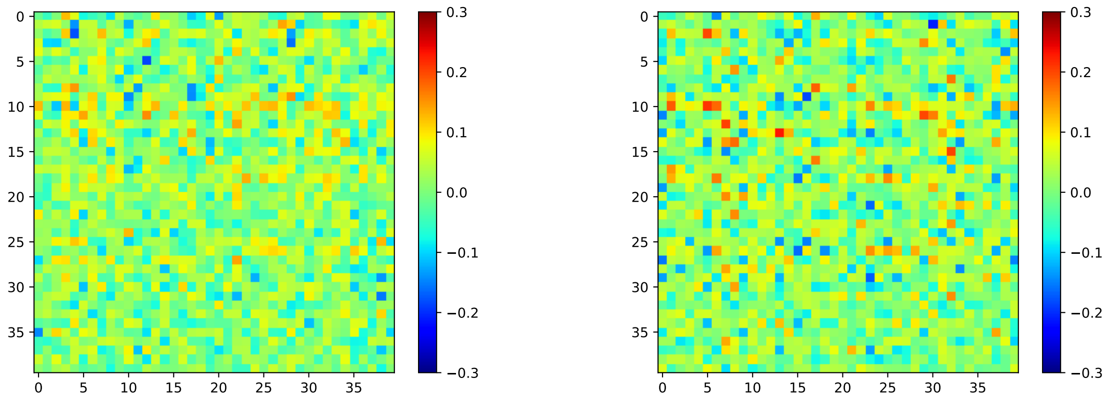


Figure 2.31: Weight portraits of the final convolutional layer of the generator for the AK47 network and the boom-box network. Note that there is no padding because $\sqrt{1600} = 40$.

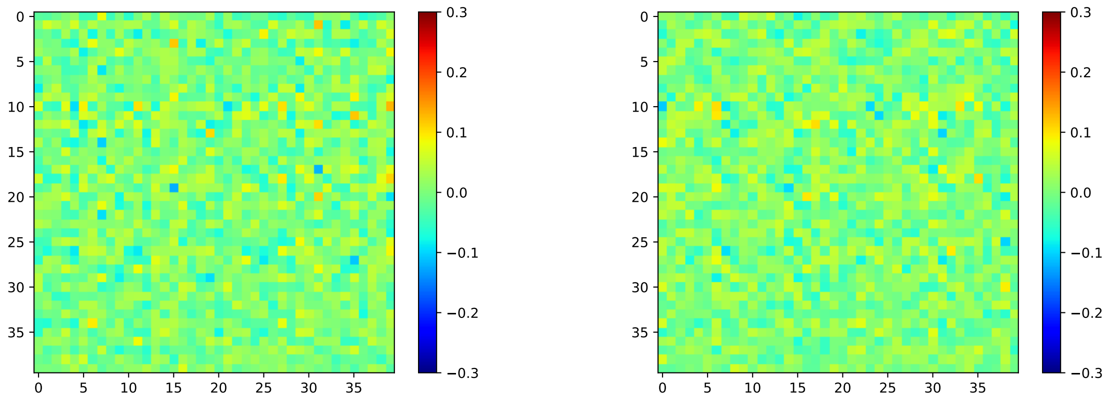


Figure 2.32: Weight portraits of the final convolutional layer of the generator for the c047 network and the c191 network, plotted with common color axes. The stuff networks are much more visually similar and homogeneous than the thing networks.

It can be seen when plotting with identical false color assignment the last convolutional layer of the generator for a pair of Things (Fig. 2.31) and a pair of Stuff (Fig. 2.32) that the weight portraits for Things and Stuff look qualitatively different. The weight portraits of the Stuff are themselves homogeneous, whereas the weight portraits of the Things are not, looking like the Stuff portraits overlaid with irregular (and thus possibly meaningful) noise.

If the pattern of these pairs holds up, it predicts a SSIM similarity matrix for this layer will show the characteristic pattern (low variation, or high similarity, for Stuff).

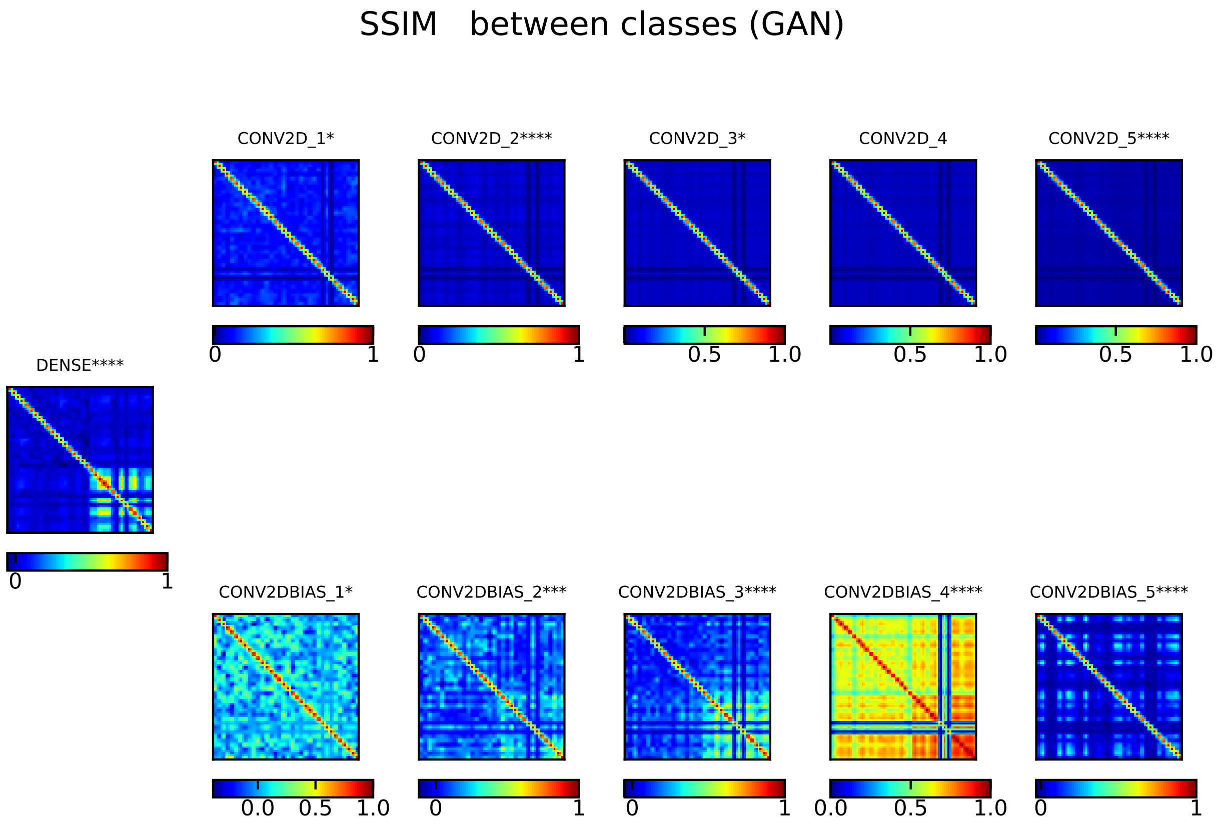


Figure 2.33: Discriminator weight portrait SSIM matrices calculated for each convolutional layer (and for the learnable layer biases separately) and the terminal Dense layer. Barely any similarity can be observed between anything in the convolutional filter weights proper, but Stuff-to-Stuff comparisons show that many Stuff networks appear similar to each other in intermediate convolutional layer biases. The weights are flattened into a list and then reshaped into the closest fitting square with zero-padding added so that the SSIM, which processes statistics on windows of 2D images can be defined. *Hotter colors indicate **less** dissimilarity between a pair of networks.*

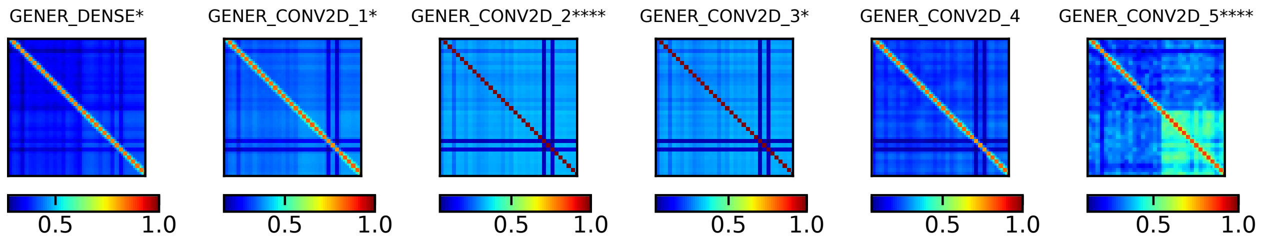


Figure 2.34: Generator weight portrait SSIM matrices calculated for each convolutional layer (there are no learnable biases) and the initial Dense layer. SSIM of Stuff-Stuff network comparisons is high *except* in the intermediate layers, but the characteristic pattern (including intermediate distance for Thing-Stuff comparisons) is very firmly established by the final convolutional layer just prior to the synthesized image. If SSIM were truly perceptual, stuff heatmaps of the weights themselves would by this layer look relatively similar compared to thing heatmaps. *Hotter colors indicate **less** dissimilarity between a pair of networks.*

Notably, the synthesis failure networks (which were deficient performers) and the duplicate boom box network with more training (which was for obvious reasons an above-average performer) are both visible within many of the distance matrices. The synthesis failures are seen in the cross that intersects in the middle of the Stuff-Stuff quadrant, and the overtrained network is visible more rarely (for example, it is easily seen in the pure-distance MSE and p-norm matrices) in the upper left hand corner, in the beginning of the Thing-Thing quadrant.

This suggests that with a field of enough networks with convergence and common training time, that this method of visualization could be used forensically to discover synthesis failures and, perhaps more practically in some abstract contest, cheating by overtraining.

To confer an idea of the importance of studying the characteristic pattern, the Wilcoxon or Mann-Whitney U statistic's induced level of significance separating the Thing-Thing quadrant (QII) affinity values from the Stuff-Stuff quadrant (QIV) values is hinted at using the conventional star-notation for significance. The precise Us and p-values are not reported, somewhat for brevity, and somewhat because these are *not* valid or effective tests. To see that they are not effective, notice that the Wilcoxon results are unusably more promiscuous than visual confirmation of a characteristic pattern. This is because hypothesis tests are normally formulated in terms of means or things like means and they are then subject to the influence of outliers, which is extremely significant here due to the presence of the synthesis failure classes. In the presence of anomalies, this is fatal. Further, the test is not legitimate because even though it is nonparametric, the activation values are anything but statistically independent due to such factors as the dependence induced by row and column relations corresponding to primary and secondary network in the pairwise comparison.

Strong Characteristic Pattern (low Stuff-Stuff network variation, high Things-Things variation)	5th Convolutional Layer, Generator (MSE, L1Norm, L2Norm, SSIM)
Weak Characteristic Pattern (low Stuff-Stuff network variation, high Things-Things variation)	2nd Convolutional Bias Layer, Disriminator (SSIM) 3rd Convolutional Bias Layer, Discriminator (L1Norm, L2Norm, SSIM) 4th Convolutional Bias Layer, Disriminator (SSIM) Dense Layer, Discriminator (L1, L2, SSIM)
Strong Inverse Characteristic Pattern (high Stuff-Stuff network variation, low Things-Things variation)	(none)
Weak Inverse Characteristic Pattern (high Stuff-Stuff network variation, low Things-Things variation)	(none)
Cross-Comparisons Most Severe Pattern (highest Stuff-Things network variation)	5th Convolutional Layer Bias, Discriminator (JSDivergence) Dense Layer, Discriminator (JS Divergence, Signchain Distance)

Table 2.1: Subjective presence of characteristic Things-Things vs. Stuff-Stuff quadrant separation in interclass GAN model comparisons

2.3 Comparative Behavior of Architecturally Identical GAN-origin vs. non-GAN/direct Discriminators

In the previous sections, the labeling affinity and the model parameter affinity of the classes in the GANsemble were described. DCGANs, and especially committees of WGAN-GP models, are clearly not the most efficient networks to train if all you desire is recognition.

Since GAN-origin-discriminators only experience their own class and their loss is tuned towards avoiding being tricked by their generator, it is interesting to consider what precisely you give up discriminability-wise in assuming the GAN loss and forgoing experiences out-of-class.

This area is not at all well-studied: of course there are investigations into ensembles of

GANs, given the immense popularity of the family of architectures, but the focus is on synthesis quality, not comparative studies of discriminability, and in at least one case involves ensembles of GANs where the ensemble is mostly comprised of the action of generators with different initializations being picked randomly to produce outputs [173] [108].

2.3.1 The "NONGAN", and its loss function

A priori, you can expect that the pressure of GAN synthesis isn't going to produce a good classifier since it is not constrained to bear directly on the universe of alternative input at all. Admittedly, to compare state-of-the-art, specifically designed recognition networks to any GAN on its own is foolish.

It is simply unfair to compare any GAN-origin-discriminator against a good purpose-built discriminator of varying architecture. The fairest comparison is to be had with its NONGAN-origin-complement, which is the architecturally exact equivalent that differs only in the training rule.

Gone is the WGAN-GP procedure with its interpolative gradient penalty, the weight clipping of WGAN, and the extra discriminator passes afforded by the w_{critic} hyperparameter. The loss for a NONGAN we define presently to be just the binary cross-entropy loss previously mentioned in the theoretical description of the GAN.

However, owing to our previous intuition, and to assure maximum validity of comparison, no nonlinearity is revised into the architecture. This is seemingly inappropriate given the likelihood of saturating one side of the loss asymmetrically, but it does preserve the wish of no change, however minor, to the architecture.

For each network in the GANsemble, a NONGAN complement has been trained, forming a NONGANsemble.

One believes that the detailed comparison of the GAN and NONGAN, afforded by the Things vs. Stuff distinction, is its own important contribution not just because it considers the loss sustained in assuming the GAN loss function, but also the ability of a Same/Different classifier to recognize and generalize, and beyond this, the possibility that GANs and NONGANs (or at least a collection of each embodied in a GANsemble and NONGANsemble) can be shown to measure different things. It could then be, in the future, eventually discerned if the judgments of GANs and NONGANs and special divergences between their behavior and their models could inform a hybrid model or new diagnostic criteria for the training or evaluation of recognizers and synthesizers.

2.3.2 NONGAN interclass labeling affinities on fake vs. real images

The NONGAN discriminator does not have the burden of determining real from fake images and bootstrapping synthesis with a generator; we should expect more, then, of the NONGANsemble in terms of facilitating recognition and similarity. This expectation is fulfilled, as we can see when we examine the pairwise activation affinity matrices for real images and fake images (Figures 2.35 and 2.36).

Immediately, it can be seen that baseline subtraction is not needed to get rid of the streaky pattern indicative of individual firing regimes in the unsubtracted activation of the GANsemble. For both reals and fakes, this pattern holds in the raw activation and the baseline-subtracted activation (critically, the baseline is still assessed on fakes from the GANsemble). The divisive normalization undoes this progress in both cases, revealing the individual firing regimes. Otherwise, for both reals and fakes, this individual regime pattern is evinced most in the baseline-subtracted affinity, but only for NONGANs that were trained on Stuff (see the streaks predominantly down the right side of the baseline-subtracted affinity matrices).

This suggests that direct discrimination of Stuff, like synthesis of Stuff, induces a lesser pressure on improvement of recognition quality than Thing networks require in the course of their training, or that there is at least less information they get to absorb on account of the homogeneity of their input.

Additionally, in the raw activation with no baseline correction condition, the fake and real affinity matrices have less graded and less severe negativity in the Stuff-Stuff quadrant, which is as it was with the GANsemble, but presenting as less severe of a problem. Again the streaked pattern is perceived somewhat, in the upper-right quadrant comparing Stuff and Things, yet again when the Stuff networks must make determinations; baseline-subtraction of the NONGAN matrices had also made the Stuff-Stuff comparisons more promiscuous. In the case of the NONGAN, there is from the outset (when division is avoided) an asymmetry of polarity favoring negative activation, and in fact, only the target class gets positive activation. Just as with the GANsemble duplicate Thing classes of boom box and theodolite can both be seen distinctly as squares partially off the diagonal.

The preferred strategy, then, for the NONGAN is to not undertake baseline-subtraction since it is naturally a notion needed for dealing with the bipolarity of the WGAN-GP-induced activation and the compensation factor (i.e. the baseline) is dependent on the GAN anyway. The graded dissimilarity being more pronounced everywhere here is strongly indicative of the better quality of direct discriminators for classification and similarity, although the joint information of a classifier more complex than the argmax is likely to overwhelm this difference, at least with the small number of classes (32) we have studied. Producing similarity judgments, however, is a subtler distinction to make, and since the potential informativeness (especially for Thing-trained perceivers) is so rich, we should expect to see for the GANsemble relative to the NONGANsemble a serious degradation in coarse measurements of similarity (such as top misclassifications) and maybe some significantly perceptible loss in a method that uses joint information from affinity effectively (such as producing a low-dimensional embedding of the matrix into an informative plane).

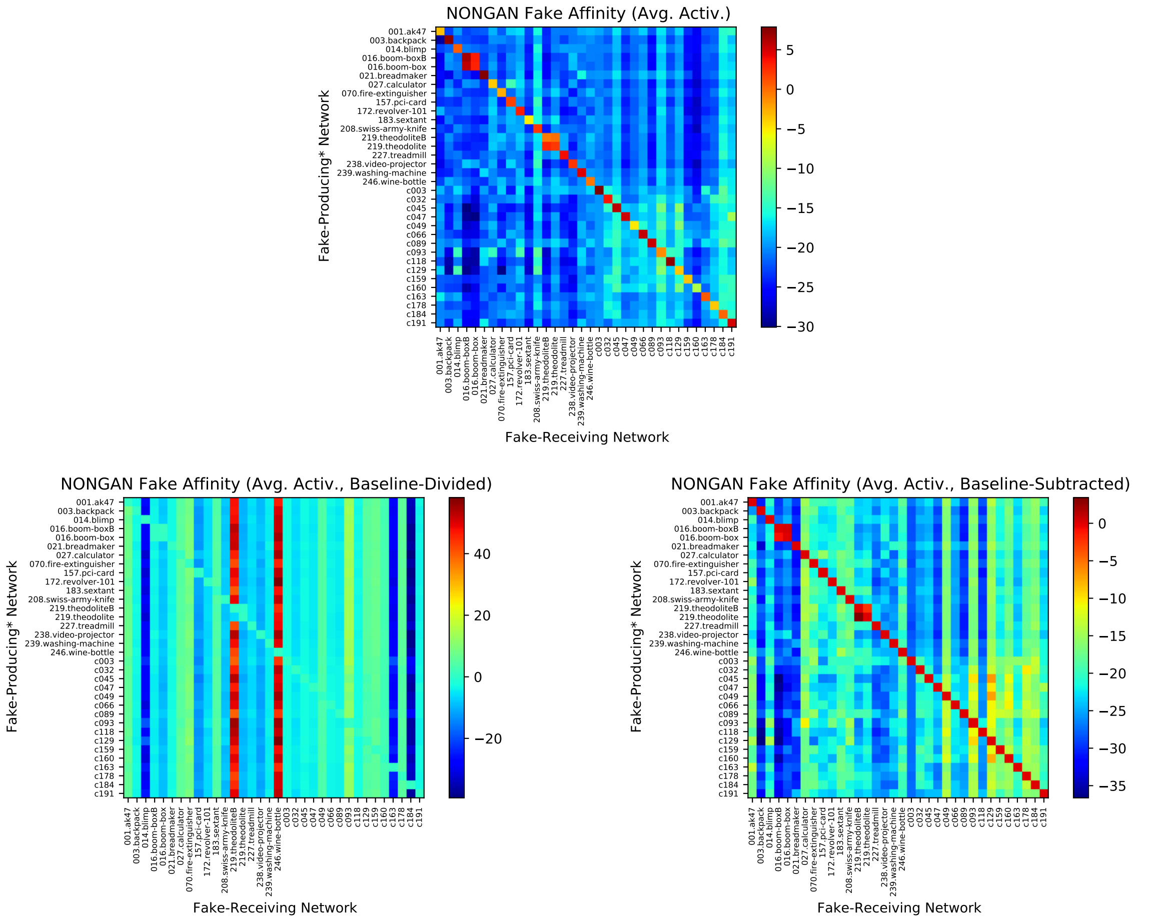


Figure 2.35: NONGANsemble firing affinity matrices on fake images for a) raw-activation, b) baseline-division, c) baseline-subtraction.

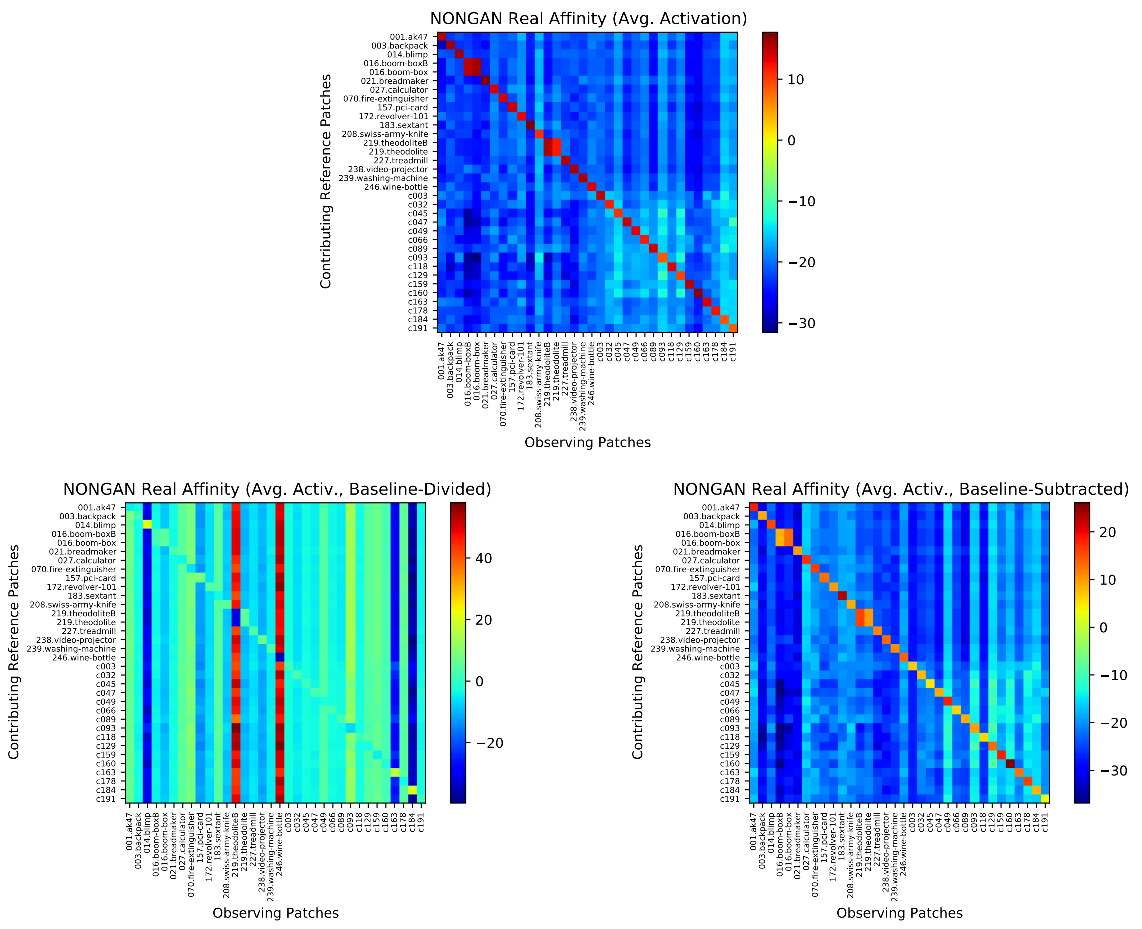


Figure 2.36: NONGANsemble firing affinity matrices ***on real images*** for a) raw-activation, b) baseline-division, c) baseline-subtraction.

2.3.3 Interclass model parameter distances redux, for NONGANs

The visualization-based demonstration along the lines of characteristic pattern quadrant analysis demonstrated for the GANsemble in Section 2.2.3 is repeated in this section, but for the parameter "distance" matrix created by deeply comparing the discriminators of the NONGANsemble.

There is little to note here beyond what was developed already, except that whereas the GANsemble did not experience the inverse characteristic pattern, the NONGANsemble did, along the pure distance type measurements as well as the SSIM in the bias of the first convolutional layer. The characteristic pattern was observed for these same measurement types in the Dense layer. Only the Jensen-Shannon distance of the first convolutional layer's bias evidenced maximum network distance in the Thing-Stuff cross quadrants. This is in stark contrast to the GANsemble, which experienced the majority of its stereotyped patterns in the ending stages of the network, with weak characteristic pattern developing throught the network for SSIM.

The conclusion, then, probably is that the NONGAN develops the Thing vs. Stuff dichotomy very early in the course of its discrimination: it has demonstrated this facility by the end of the first convolutional layer, and the separation recurs somewhat more weakly in the terminal dense layer, but with alternate, and now canonical, polarity: low Stuff-Stuff variation, high Things-Things variation. Why Stuff-Stuff network comparisons are emphasized early is a bit of a mystery in terms of possible explanation, but it could be simply that, for the NONGAN, it is possible immediately from the edge detection level of abstraction filters in the first convolutional layer to disentangle Stuff, and that these earlier layers are highly specialized for that purpose: the notion of "early" or "late" here is not that of training time but of the early vs. late debate concerning circuitry of the human visual system.

Object (or "Thing") perception has long been linked with activity in inferotemporal cortex as a later stage of processing past the early visual cortex, and the single-cell recording work in 2010 of Rust and DiCarlo [138] showed in rhesus macaques that neural activity in V4 and IT could be combined with a support vector machine to predict which of several objects, presented under differing simple transformations, was being presented at the time of recording. Although extracting this information from macaques and subjecting it to an SVM on an external computer doesn't particularly *reverse-engineer the algorithms used by the brain* (i.e. it is not a Marr level 2 investigation), it does show that more of the available information is present (in a linearly separable sense) by the point of IT, and demonstrated a "V4-to-IT gain in tolerance" to simple transformations – but, appropriately, not necessarily identity-changing transformations, as Rust and DiCarlo amusingly used the failure case of Portilla and Simoncelli synthesis, the tendency to turn inhomogeneous texture such as faces and figures into flowy, veiny, streaky silk or rock-like homogeneous texture, to produce "scrambled" stimuli.

Considerably less research focuses on localizing likely neural representations of homogeneous patches, as the complexity continuum's extremes of simple gratings and objects have attracted more functional neuroanatomy interest. Notably, though, a study by Jacobs et al. in 2014 [87] used fMRI in two ways to try to get at the loci of Stuff representations. In the first paradigm, viewers looked at material patches like wood, fabric, and stone, and a linear naïve Bayes classifier was used on the full brain as well as mostly early region (V1, V2, V4, and also the PPA) region-of-interest voxel data. V1 voxels were the most informative of early areas for the ROI analysis, and the full brain analysis yielded informativeness arising from the supramarginal gyrus, which the authors claimed has shown more activation to roughness judgments than aesthetic judgments in their work. In the second, an adaptation task, similar analysis highlighted the right parahippocampal gyrus, and again but less strongly the supramarginal gyrus. Because research concerning Things and Stuff, beyond early visual cortex, currently focuses on different areas, it is currently difficult to make a determined stance on whether Thing processing is conclusively later than Stuff processing. Early areas, such as V2, of course show specific responding (in macaques) to complex stimuli like hyperbolic and polar gratings [75] that might resemble some classes of homogeneous texture, and these areas naturally exhibit activation for visual stimuli in general.

If the NONGAN is comparatively better at discrimination of final results because it does not have to contend with discerning current fakes as the GAN does, then there may be more opportunity allocated to tuning early vision to do better on otherwise similar homogeneous texture. The GAN, by constrast, would seem to develop this capability for representing the dichotomy only once, and in late stages of this highly simplified model of the visual system, whereby it is able to pull in all of the previous nonlinear projection capability of the earlier layers of the network.

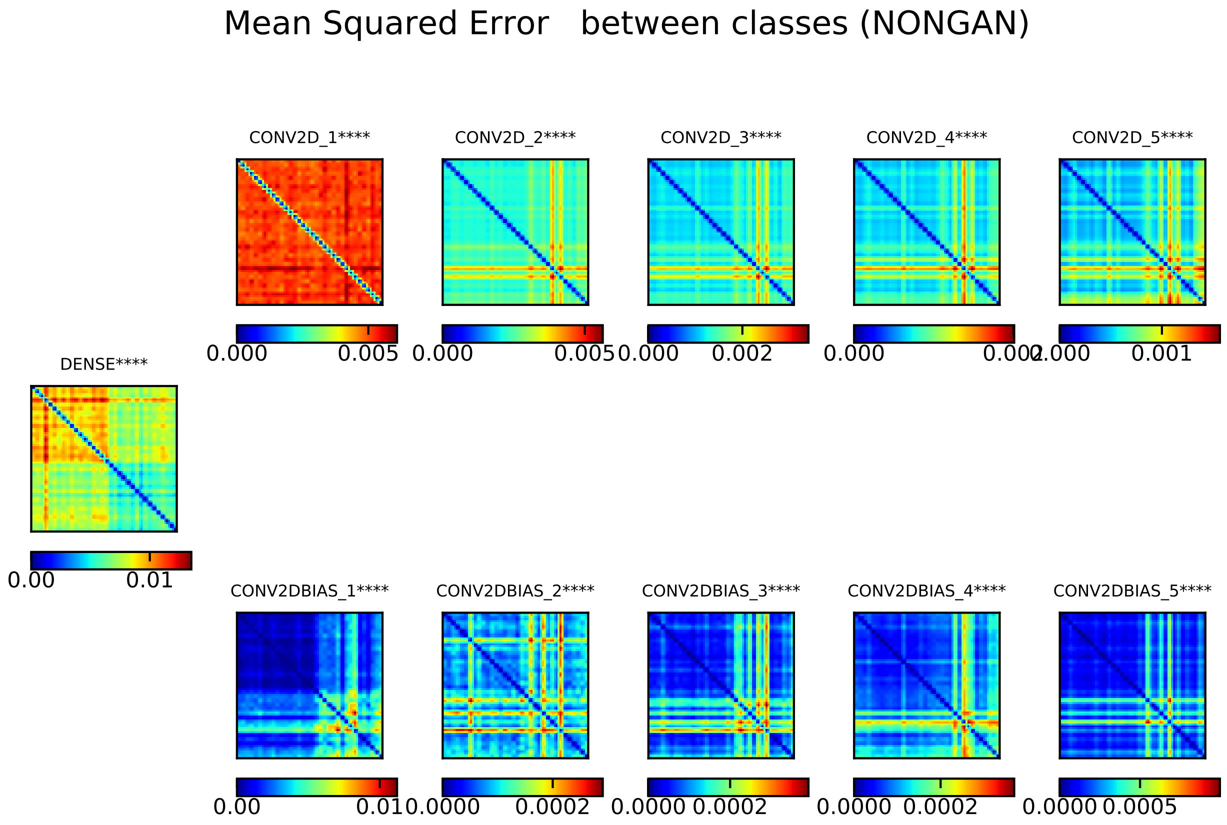


Figure 2.37: NONGAN Discriminator weight mean-squared error (MSE) distance matrices calculated for each convolutional layer (and for the learnable layer biases separately) and the terminal Dense layer. The characteristic pattern is found in the terminal dense layer only but there is more similarity amongst Things networks in the contributing bias layers.

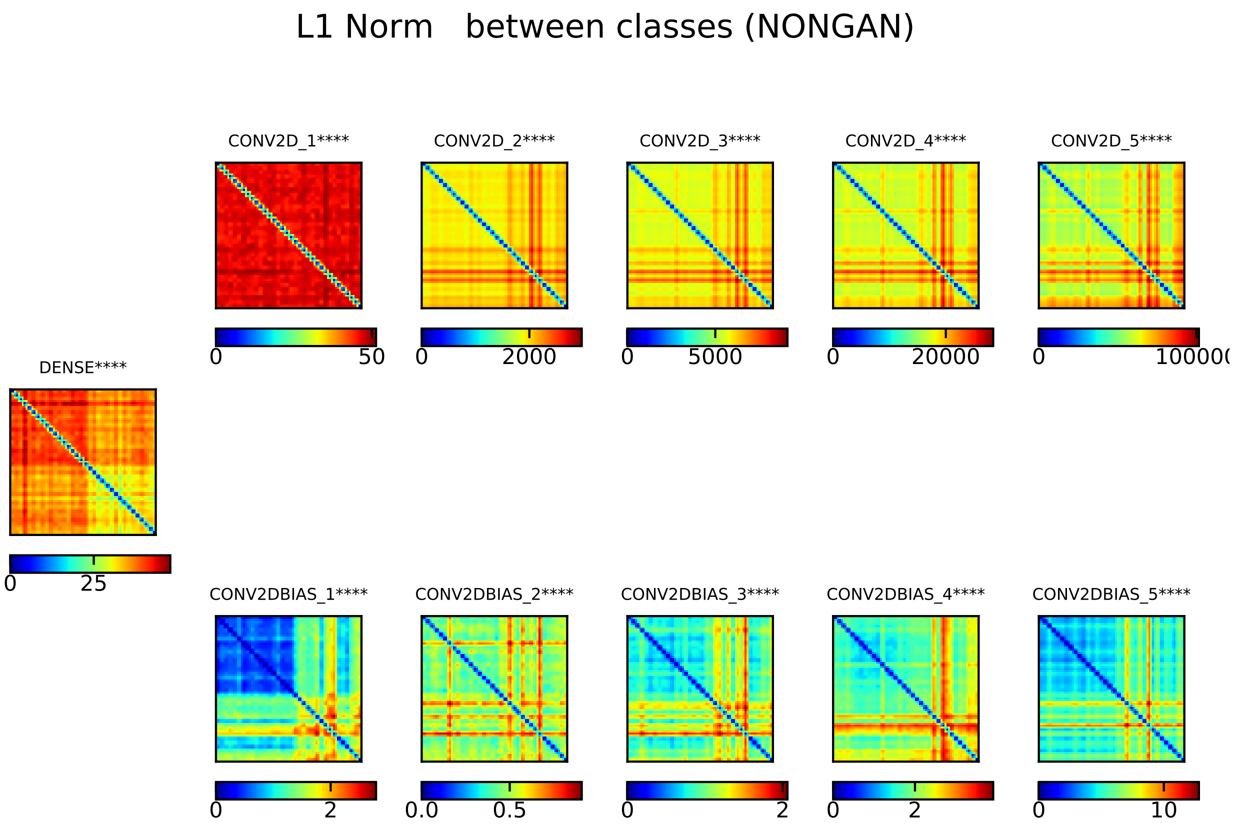


Figure 2.38: NONGAN Discriminator weight L1 Norm distance matrices calculated for each convolutional layer (and for the learnable layer biases separately) and the terminal Dense layer. The characteristic pattern is found in the Dense layer but the inverse characteristic pattern is found in the first bias layer.

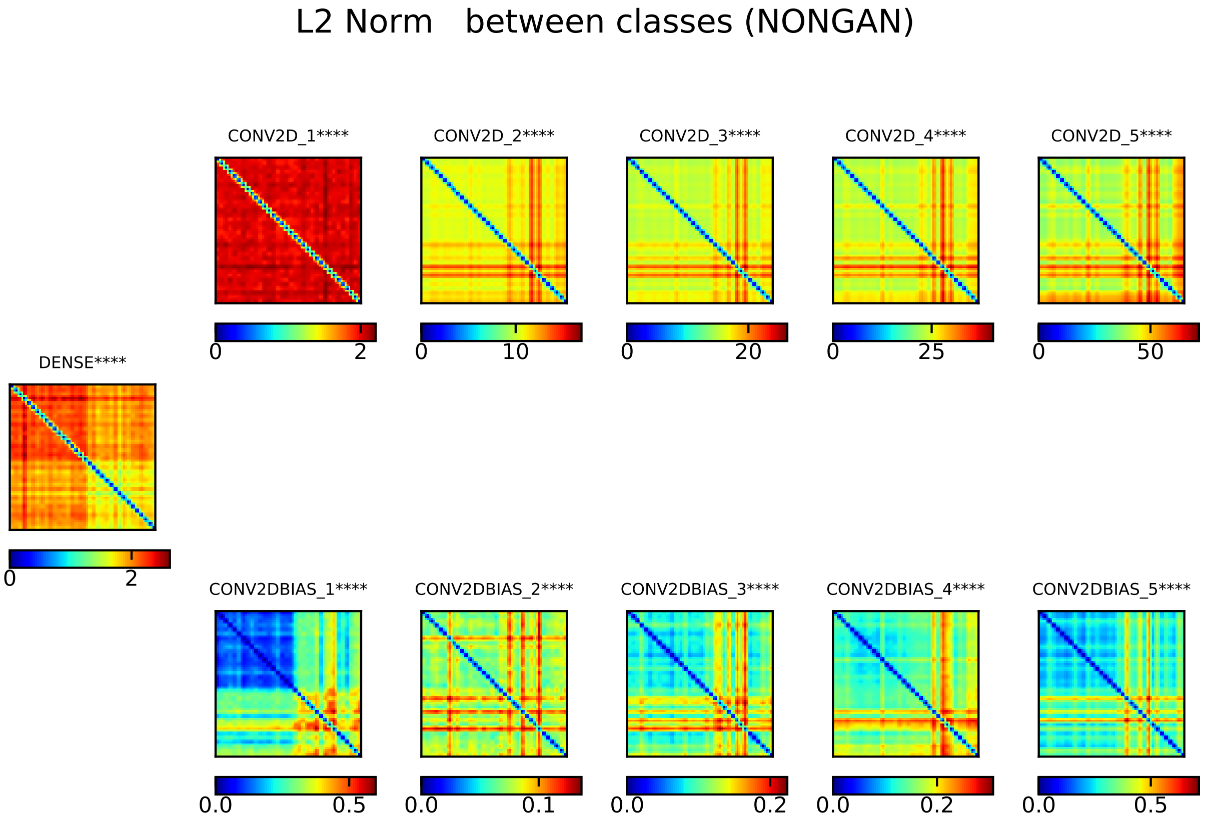


Figure 2.39: NONGAN Discriminator weight L2 Norm distance matrices calculated for each convolutional layer (and for the learnable layer biases separately) and the terminal Dense layer. As with the L1 Norm, the characteristic pattern is found in the Dense layer and the inverse pattern is found in the first bias layer.

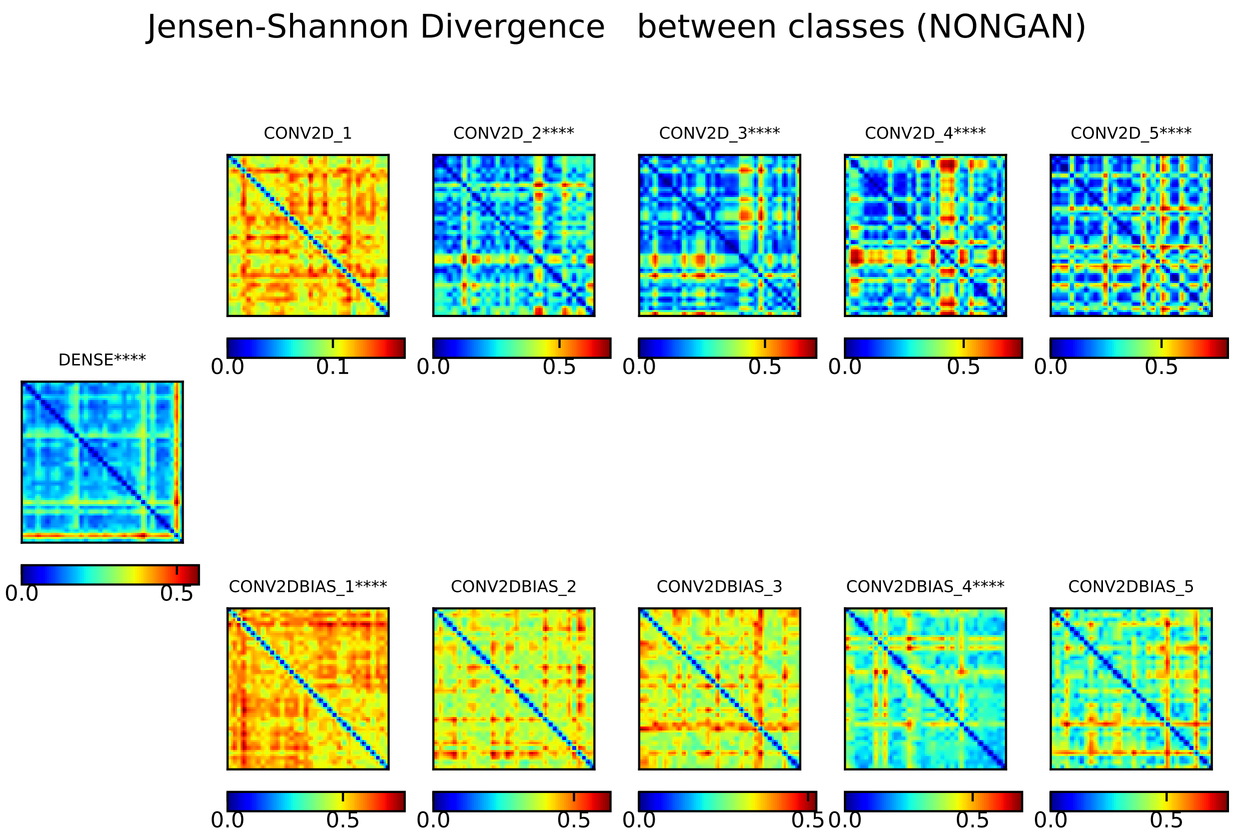


Figure 2.40: NONGAN Discriminator weight Jensen-Shannon distance matrices calculated for each convolutional layer (and for the learnable layer biases separately) and the terminal Dense layer. A clear characteristic pattern is found nowhere but the first bias layer and all subsequent (layer 2,3,4,5) convolutional layers are opposed in their patterning.

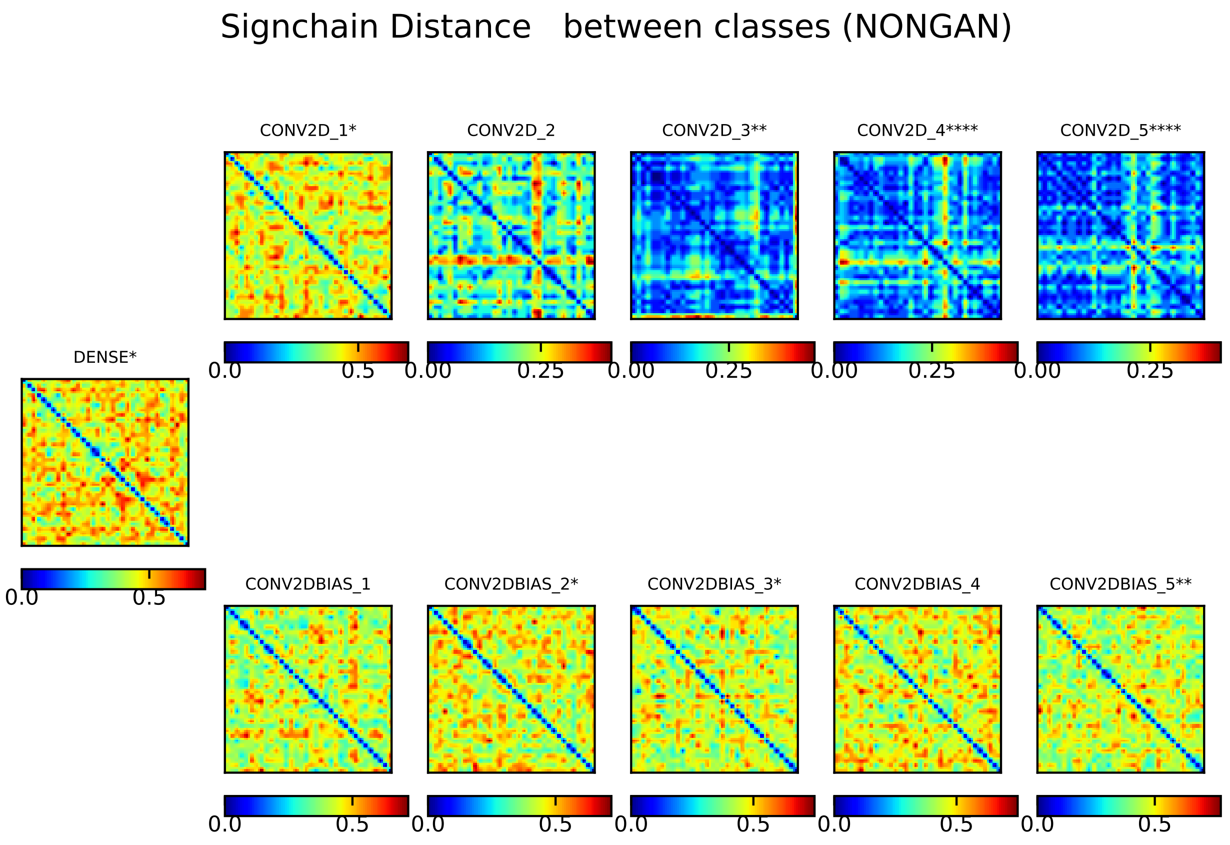


Figure 2.41: NONGAN Discriminator weight signchain distance matrices calculated for each convolutional layer (and for the learnable layer biases separately) and the terminal Dense layer. A pattern of dissimilarity pervades; the signchain distance for NONGANs is even more weakly related to the Jensen-Shannon distance.

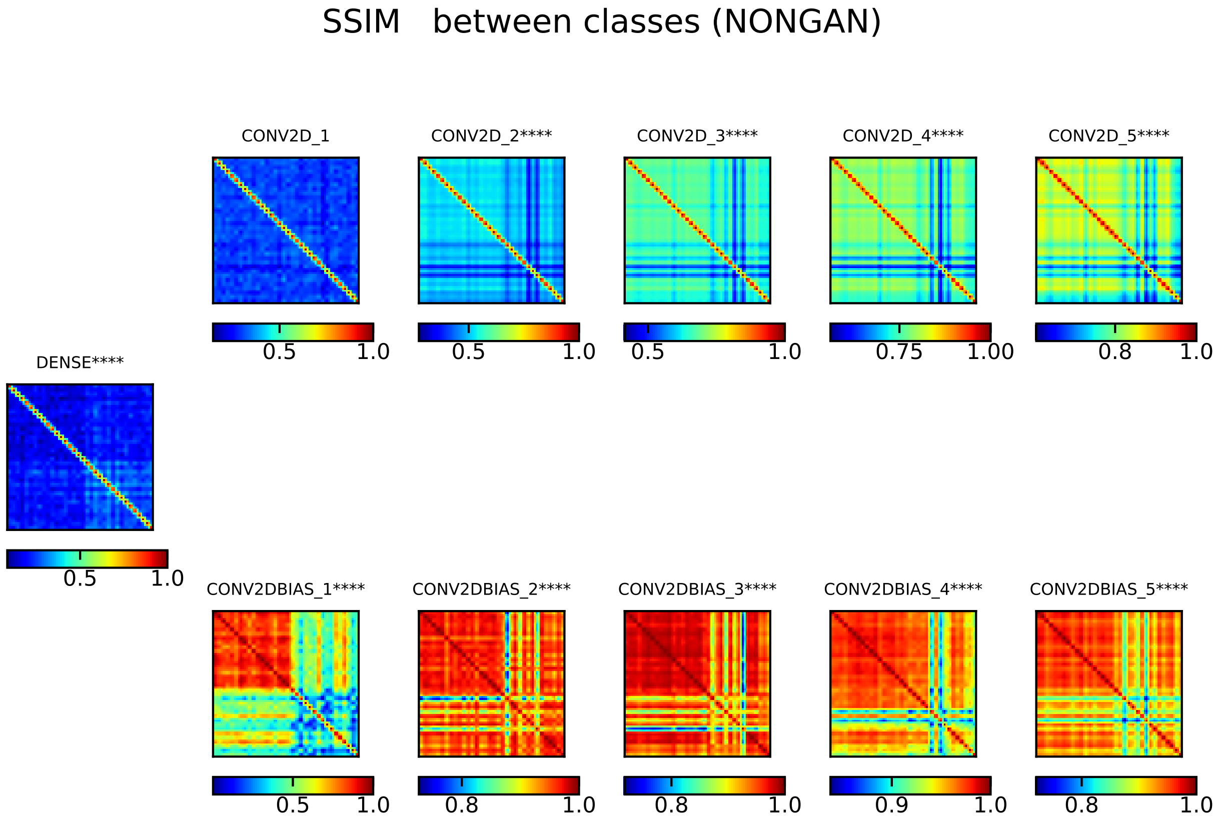


Figure 2.42: NONGAN Discriminator weight SSIM matrices calculated for each convolutional layer (and for the learnable layer biases separately) and the terminal Dense layer. The characteristic pattern appears in the first convolutional bias (but since this is a similarity it is actually the inverse characteristic pattern). Strong similarity is observed between Thing networks in all layers of the bias. There is extremely weak elevated similarity amongst Stuff by the time of the Dense layer preceding the discriminator's decision.

Characteristic Pattern (low Stuff-Stuff network variation, high Things-Things variation)	Dense Layer (MSE, L1Norm, L2Norm, SSIM)
Inverse Characteristic Pattern (high Stuff-Stuff network variation, low Things-Things variation)	1st Convolutional Layer Bias (MSE, L1, L2, SSIM)
Cross-Comparisons Most Severe Pattern (highest Stuff-Things network variation)	1st Convolutional Layer Bias (JSDivergence)

Table 2.2: Subjective presence of characteristic Things-Things vs. Stuff-Stuff quadrant separation in interclass NONGAN model comparisons

2.3.4 Intermodel parameter distances between GAN and complementary NONGAN, early vs. late specialization for loss vs. content

The immediately preceding section discussed the early vs. late vision debate as embodied in the Things vs. Stuff dichotomy in the GANsemble and NONGANsemble separately. That discussion looked at the lateness distinction in terms of specialization for content. There is another way we can look at the pressure of the choice of loss function directly, to intuit "when" specialization for the loss function itself is likely to most severely occur.

Since the GANsemble and NONGANsemble use fully commensurable architecture and training, it is appropriate to look at the GAN and NONGAN networks as pairs, and take the "difference" between these dual representations of the unrectified DCGAN architecture discriminator along parameter disparity lines. The Wilcoxon significance level is still indirectly signified in the parameter distance vectors. Indeed the test between the leading Things segment and trailing Stuff segment of each vector comports more often with visual inspection of a systematic difference because the problem of the two-dimensional dependency has been eliminated. However, the Wilcoxon test is still subject to outliers, despite the rank-based formulation of the statistic. This provides a general caution about depending on these tests absent visual or carefully automated inspection of dichotomous data.

The characteristic pattern for the Things vs. Stuff vector is still low Stuff variation and high Thing variation. As shown in Table 2.3, no easily apparent inverse characteristic pattern was observed. The characteristic pattern (again, of the relative fungibility analogously to promiscuity of Stuff networks) is observed, but only nearly comprehensively by the terminal Dense layer. The signchain distance, a weak version of the Jensen-Shannon distance behaviorally before, reveals no systematic change. The L1 and L2 norm and SSIM detect a difference in the third convolutional layer's bias, which is still reasonably late.

Note that nearly all of the dichotomy consistent or dichotomy related stereotyped patterns subjectively observed have been either with the Dense layer or the bias components of the convolutional layers. This suggests that the bias does most of the heavy lifting related to specialization for content and for loss in deep convolutional recognizers. There is a limitation to this that is inherent in the visual inspection and that is that there are many fewer components (the weight vectors have much lower dimension) in these as opposed to the main filter implementing weights of the convolutional network and so a dichotomy might not appear very strongly, or it might even hinge on a few sparsely distributed weights.

But from a relative persepective, this investigation has seemed to show that the specialization for the kind of loss happens "after" early visual circuitry and simple filters, towards the middle of the network. The specialization for content, at least so far as it is only discoverable in the presence of the gross Things vs. Stuff dichotomy, happens only at the end for the GAN and only at the convolutional extremes (suggesting survival of this information in middle layers) for the NONGAN.

If early vs. late development of the dichotomy is linked to a tradeoff in discriminative competence, then it is important to measure competence of this particular DCGAN architecture objectively, at least for the NONGAN where some form of protection against overfitting can be employed without the cost of training the highly expensive WGAN-GPs on separate folds of data or sacrificing the already precious sample count. And if Things and Stuff GANsembles and NONGANsembles should be convened separately because of their statistical distinction, it is important for establishing these committees to be able to briskly take an unknown class and corresponding network and assign it a Thing or Stuff metacategory label so that it can be properly referred.

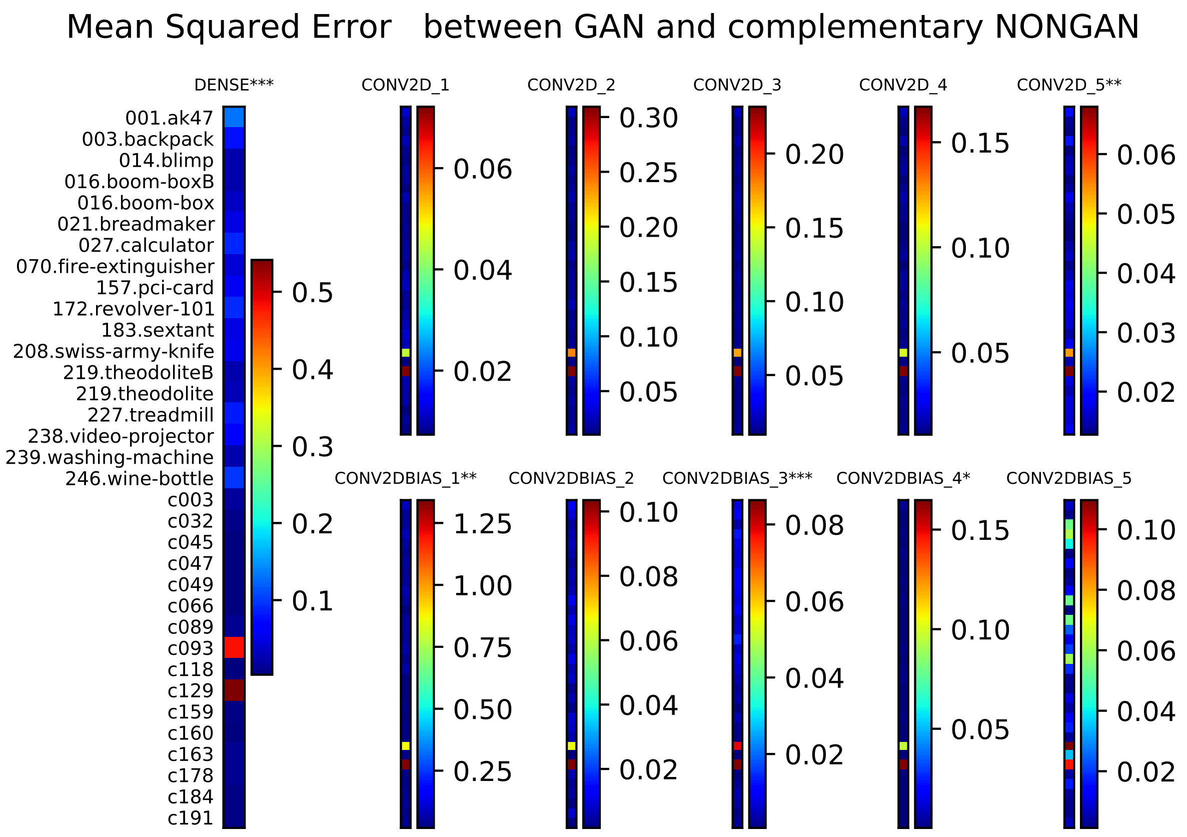


Figure 2.43: Per-network GAN vs. NONGAN discriminator layerwise MSE of the weights. As before, the synthesis failure classes c093 and c129 are detectable anomalies. The largest MSE differences found are for the final convolutional layer's bias and the terminal Dense layer prior to the decision. In both cases, the Thing GANs and their respective NONGAN complements vary more between each other. Other layers are relatively similar (cool) between GAN and NONGAN compared to the synthesis failures, irrespective of Thing or Stuff class affiliation. As before, a Wilcoxon rank-sum test is run between things and stuff and the significance is conventionally signaled with asterisks – this test is less miseleading because of the decreased dependence but is still susceptible to outliers.

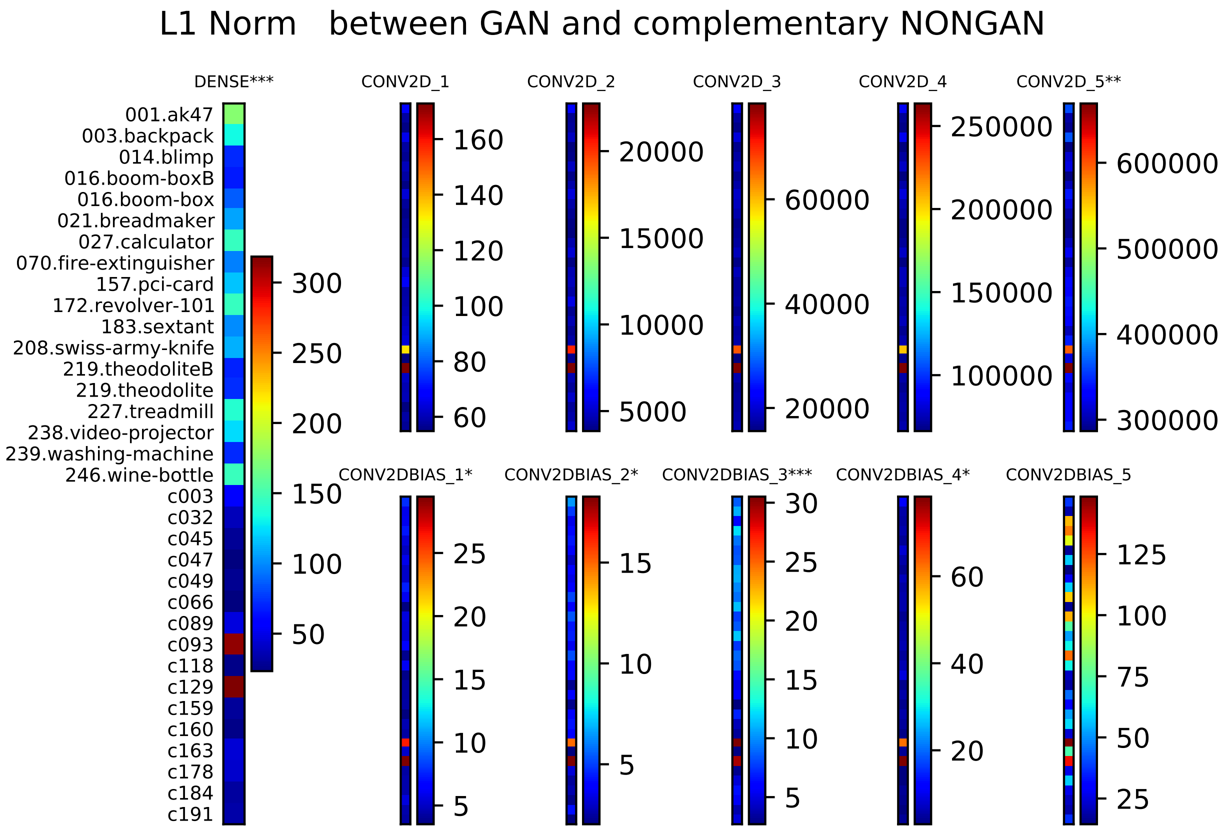


Figure 2.44: Per-network GAN vs. NONGAN discriminator layerwise L1 Norm of the weights. The largest L1 Norm differences are found in the middle (3rd) convolutional bias and the terminal Dense layer. The final convolutional bias layer is highly variable in GAN-NONGAN distance.

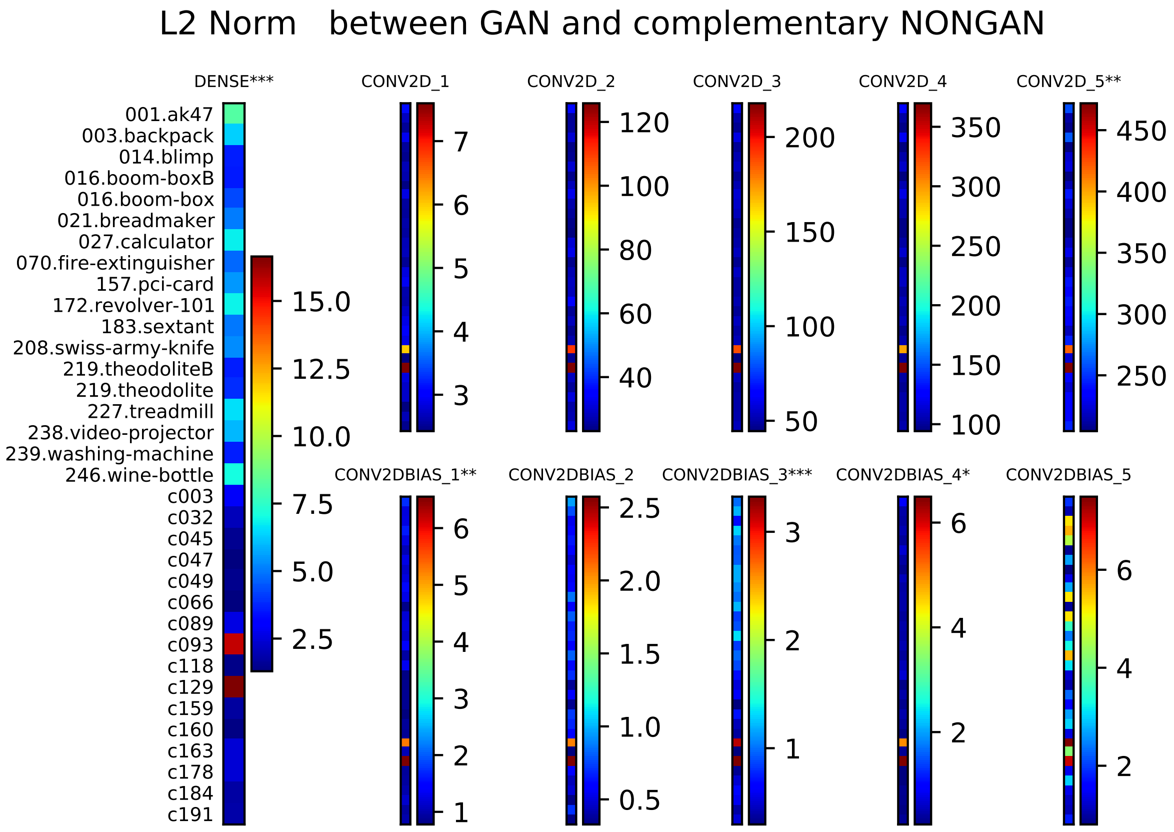


Figure 2.45: Per-network GAN vs. NONGAN discriminator layerwise L2 Norm of the weights. The largest L2 Norm differences are found in the middle (3rd) convolutional bias and the terminal Dense layer. The final convolutional bias layer is highly variable in GAN-NONGAN distance. As with the interclass distances, the L2 norm very strongly resembles the L1 norm, suggesting that including both of these measures is redundant.

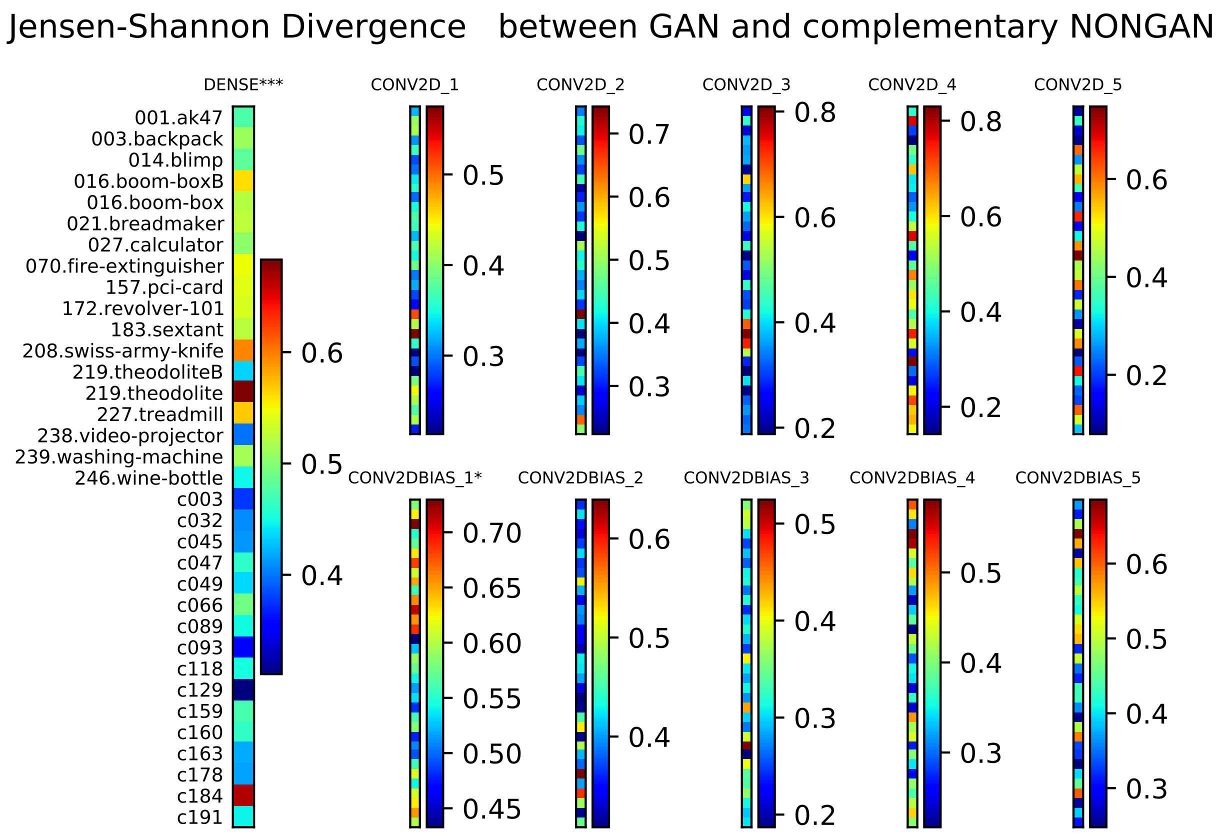


Figure 2.46: Per-network GAN vs. NONGAN discriminator layerwise Jensen-Shannon distance of the weight distributions. The only reliable Things vs. Stuff Jensen-Shannon distance difference is found in the terminal Dense layer. Otherwise the weight distributions vary wildly.

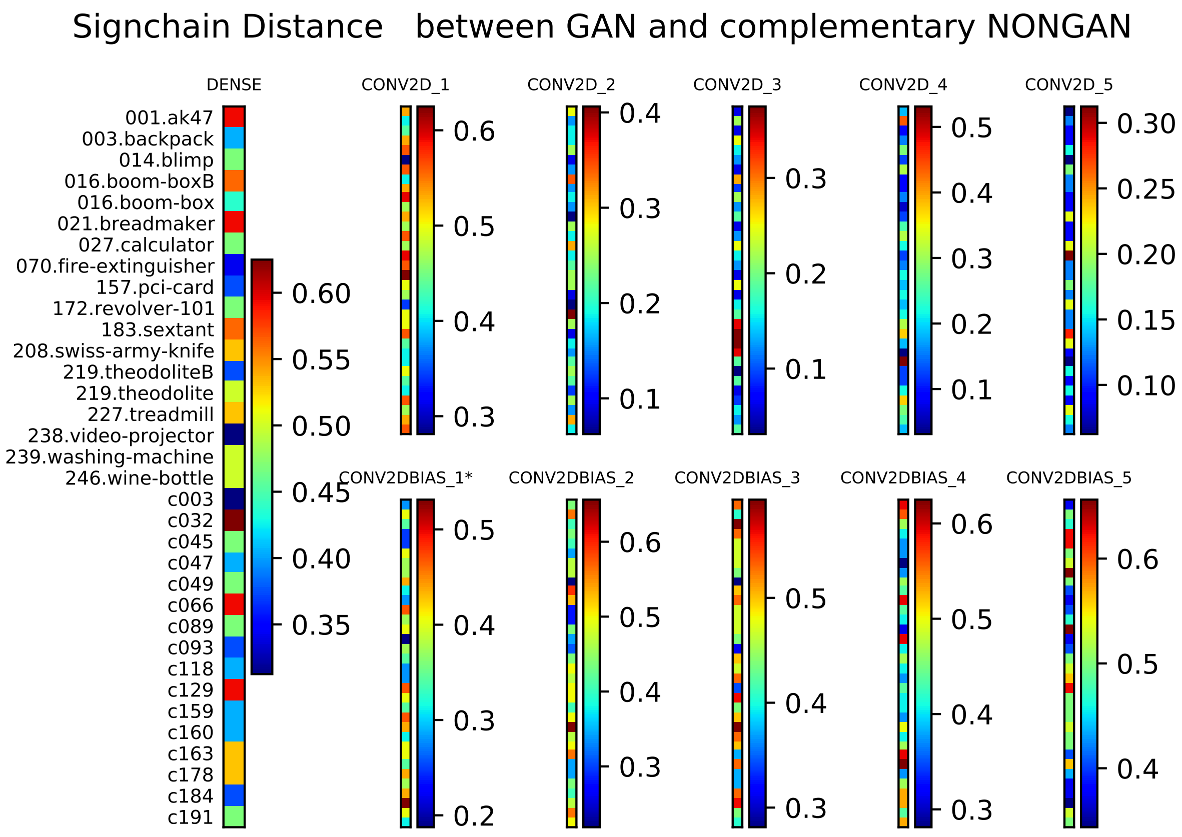


Figure 2.47: Per-network GAN vs. NONGAN discriminator layerwise Signchain Distance of the weight distributions. No reliable Things vs. Stuff Signchain distance difference is found. As with the Jensen-Shannon distance, the signchain distances vary considerably and unpredictably from network to network.

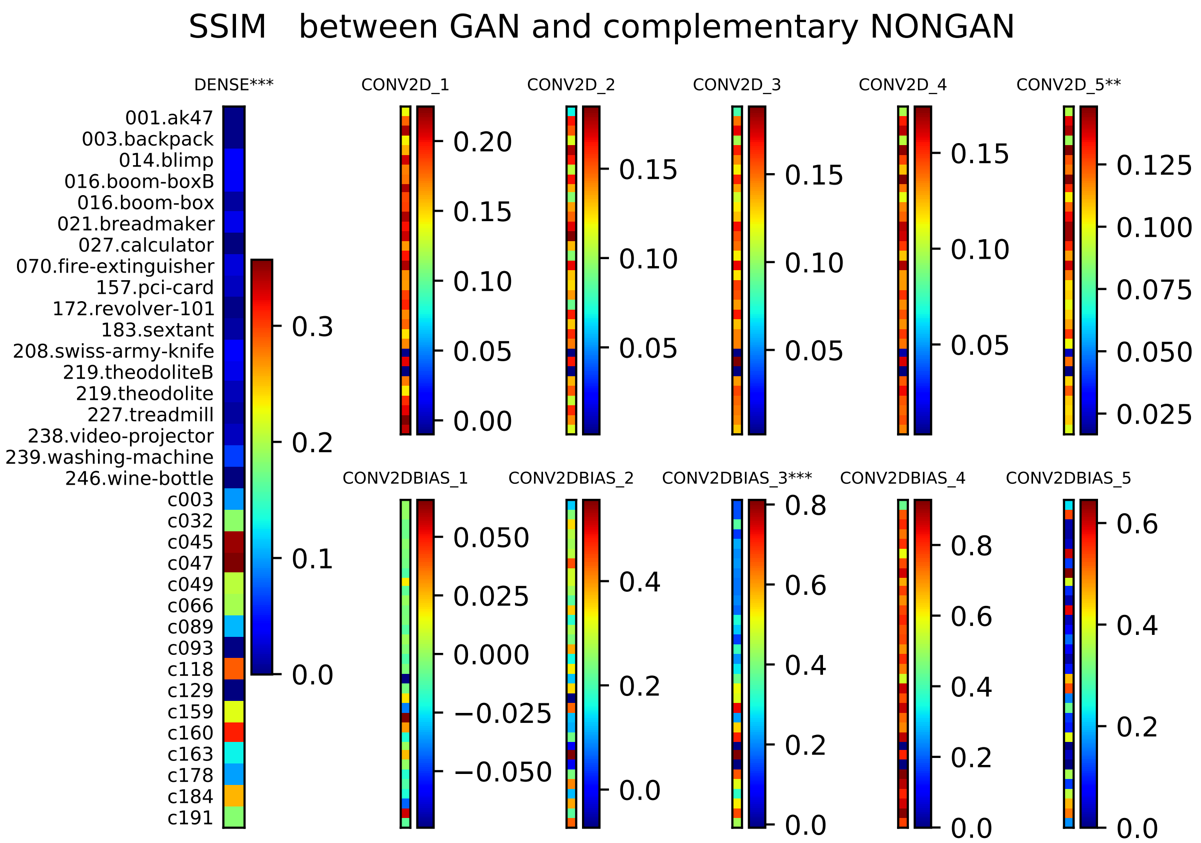


Figure 2.48: Per-network GAN vs. NONGAN discriminator layerwise SSIM (similarity) of the weight distributions. The only reliable Things vs. Stuff SSIM difference is found in the Dense layer and the 3rd convolutional bias term (as with the L1 and L2 norms). In both cases Things matched network pairs have less SSIM-similar weights than Stuff.

Characteristic Pattern (low Stuff-Stuff network variation, high Things-Things variation)	3rd Convolutional Layer Bias, (L1,L2,SSIM) 5th Convolutional Layer Bias, (MSE) Dense Layer (MSE,L1,L2,JS,SSIM)
Inverse Characteristic Pattern (low Stuff-Stuff network variation, high Things-Things variation)	(none)

Table 2.3: Subjective presence of characteristic Things vs. Stuff vector segment separation in intraclass GAN-NONGAN model comparisons

2.3.5 Performance of an explicit Thing vs. Stuff recognizer, tested or trained on held-out classes

In the context of panoptic segmentation [99] seeking to label each pixel with its identity (as in semantic segmentation) but also to carve up the scene into sensible regions and parts (more along the lines of classical image segmentation), omnibus Thing vs. Stuff classifiers are obviously useful, and at least one [5] has manifested as an "objectness" classifier meant to be used on windows of input passed over a scene. However, this method is explicitly scene-based, viewing object patches as anomalies and is implemented using a Bayesian model for combining heuristic cues for the presence of a Thing such as the density of detected edges, neigbhorhood color contrast, and the connectedness of texturally-coherent superpixels found by segmentation. Even prior to the introduction of impressive pixel-level labeling (i.e. semantic segmentation), good techniques existed for inferring texturally-coherent regions and their boundaries, such as the pointwise mutual information technique of Isola et al. [85] or the many-cued hierarchical multiscale contour detection program of Arbeláez et al. [7].

Nevertheless, in consideration of defining a possibility in between panoptic segmentation or semantic segmentation involving pixel labeling in terms of sophistication, we consider presently a direct discriminator for things and stuff that is simply a deep convolutional one not based on heuristics. If our NONGAN architecture, albeit not designed at all to be state-of-the-art and only to compare directly with GAN-origin complements, is at all competent, it should be able to exhibit a reasonable degree of accuracy on the Things vs. Stuff distinction.

This accuracy can be assessed by training a NONGAN on the same patches but with modified labels where the true class labels are replaced with 1 in the case of originating from the Things metacategory and 0 if originating from the Stuff metacategory. There are some limitations with the approach (the argmax accuracy only is considered, and there is no cross-validation of the training of the GAN, so the good performance could reflect memorization and fail to generalize), but this would establish the bare competency of the NONGAN in a limited sense.

To avoid the problem of no cross validation of NONGAN training for this analysis only, Figure 2.50 shows the accuracy of the direct Things vs. Stuff NONGANsemble trained on the original Things vs. Stuff dataset used throughout this investigation on a second, held-out dataset collected for this specific purpose from alternative classes. A mirror NONGANsemble trained on the second dataset assesses the original set of classes as well. The NONGANsemble trained on the second dataset does better, and perhaps this could be related to a lack of synthesis failure classes in Stuff on that dataset, but as we have not incurred the substantial cost (about 4 or 5 times the training time) to train a GANsemble for the second set of classes, we do not know, nor do we have available held-out GANsemble accuracy data.

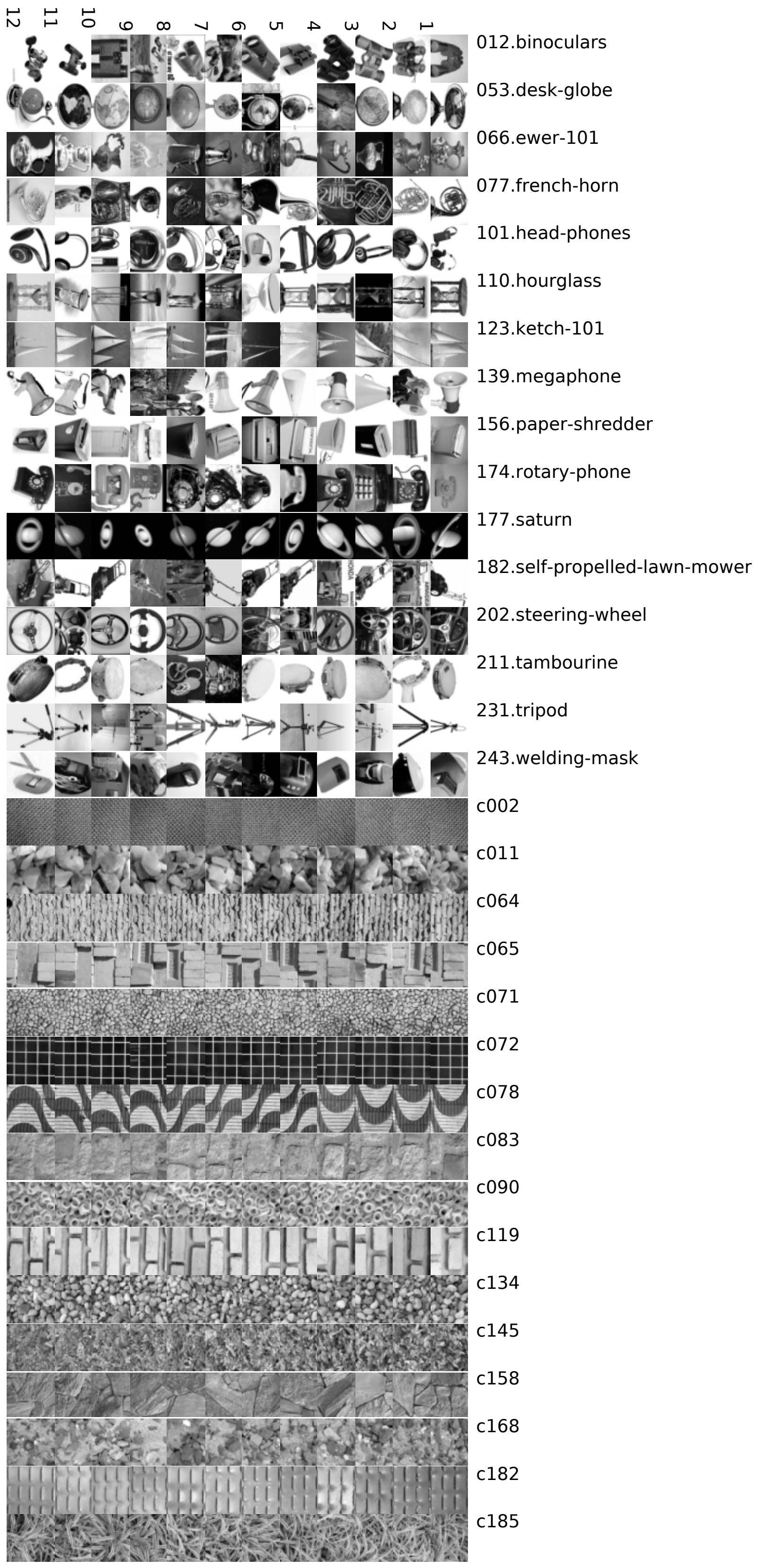


Figure 2.49: Sample Things vs. Stuff images from the second held-out/comparison dataset

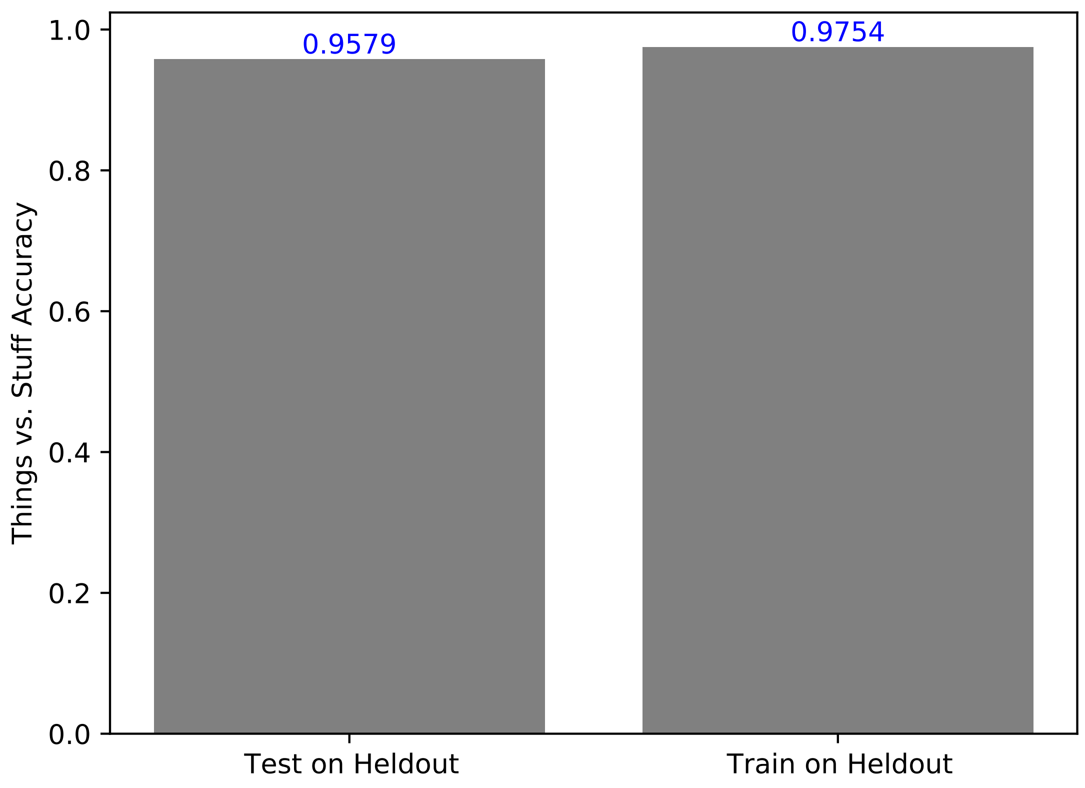


Figure 2.50: Performance of NONGAN objectness detector a) trained on original dataset, tested on heldout dataset; b) trained on heldout dataset, tested on original dataset

2.3.6 Examining out-of-vocabulary misclassifications on held-out data for GANs vs. NONGANs to confirm sensitivity to similarity

Just as successful classification by strictly naive argmax prior to baseline subtraction is a stricter expectation on success than successful classification using the joint information of the full vector of label emissions from a GANsemble through the LDA classifier, there are various levels of confidence we can have in the usefulness of the GANsemble relative to the NONGANsemble for inducing good similarity maps.

A weak test is to take the affinity matrix and confirm that there is enough sensibly graded negativity among classes to effect similarity, and a strong technique for implementing this weak test is explored in detail in Chapter 3. A strong test along the lines of the argmax for classification is examining the *specific misclassifications* of a GANsemble and NONGANsemble on held out data.

If the GANsemble misclassifies in ways humans might objectively consider forgivable (such as by seeing a treadmill when the network is actually given a self-propelled lawnmower, owing to the similar handle structure of those objects), then it is more likely to have a strong sense of similarity. Recall that a weakness of our non-cross-validated-through-GAN-training approach used for expediency is that it could very plausibly be that the GANs could be mostly memorizing sample information and are unable to contribute robust information to classifiers. Strong preservation of similarity to completely out-of-repertoire classes is thus related to generalizability of recognition, which seems itself an easier task than providing a naturalistically agreeable judgment of similarity.

The GANsemble's detailed top classification labels for each class of the second, held-out dataset are listed below. The GANsemble is actually listed twice because the argmax formulation of classification allows other Stuff particularly to be dominated by the two Stuff synthesis failure classes that are defectors on the committee emitting a different and not sensibly graded activation value when presented with any patch; the second GANsemble listing explicitly pretends these classes' networks do not exist (i.e. they are forcibly ejected from the committee for this sensitivity analysis alone).

The top classification labels are also enumerated from the perspective of the NONGANsemble.

Examining these detailed misclassifications reveals that strong similarity is only likely for the NONGAN. This is plausible since only the NONGANs have been exposed to alternative class data. Can a robust idea of similarity in general (e.g. the shared pointyness of AK-47s and sextants vs. the shared ornate shapes of theodolites and sextants) be learned from only being exposed to one class? It seems a priori unlikely.

This should not be construed just yet to suggest that the networks working in tandem in the GANsemble are likely to not generalize, nor that the single GAN-origin-discriminators are consigned even themselves to not generalize for classification. It does suggest that single GAN-origin discriminators do suffer in the similarity sensing domain relative to non-GAN-origin discriminators, which is eminently reasonable.

	GANsemble categorizes held-out dataset
012.binoculars	(c093 : 60)(c129 : 54)(c191 : 53) (c032 : 14)(c118 : 14)
053.desk-globe	(c003 : 26)(c093 : 19)(c032 : 10) (c118 : 9)(c129 : 8)
066.ewer-101	(c093 : 59)(c003 : 9)(c129 : 4) (c066 : 3)(157.pci-card : 2)
077.french-horn	(c093 : 38)(c003 : 37)(157.pci-card : 5) (c191 : 3)(c118 : 3)

101.head-phones	(c003 : 33)(c093 : 28)(c191 : 22) (021.breadmaker : 14)(c118 : 10)
110.hourglass	(c093 : 65)(c003 : 9)(c066 : 5) (c118 : 2)(c191 : 1)
123.ketch-101	(c093 : 55)(c003 : 21)(003.backpack : 14) (157.pci-card : 6)(c129 : 6)
139.megaphone	(c003 : 31)(c093 : 25)(c191 : 9) (c118 : 6)(c129 : 5)
156.paper-shredder	(c003 : 39)(c093 : 26)(c129 : 7) (c118 : 6)(c191 : 4)
174.rotary-phone	(c093 : 50)(c129 : 10)(c003 : 7) (c191 : 5)(c118 : 5)
177.saturn	(c093 : 95)(c118 : 1)
182.self-propelled-lawn-mower	(c003 : 76)(c093 : 17)(c066 : 7) (c191 : 6)(c118 : 5)
202.steering-wheel	(c093 : 73)(c003 : 8)(c191 : 6) (021.breadmaker : 3)(003.backpack : 3)
211.tambourine	(c003 : 38)(c093 : 29)(c191 : 10) (c118 : 6)(003.backpack : 5)
231.tripod	(c093 : 47)(c003 : 45)(c129 : 4) (157.pci-card : 3)(021.breadmaker : 3)
243.welding-mask	(c093 : 45)(c129 : 17)(c191 : 8) (c118 : 6)(c003 : 5)
c002	(c093 : 12)
c011	(c093 : 12)
c064	(c093 : 12)
c065	(c093 : 7)(172.revolver-101 : 3)(157.pci-card : 1) (c129 : 1)
c071	(c093 : 12)
c072	(c093 : 12)

c078	(c093 : 10)(172.revolver-101 : 1)(c129 : 1)
c083	(172.revolver-101 : 10)(183.sextant : 1)(001.ak47 : 1)
c090	(c093 : 12)
c119	(c093 : 10)(172.revolver-101 : 2)
c134	(c093 : 12)
c145	(c093 : 12)
c158	(172.revolver-101 : 8)(c093 : 2)(183.sextant : 1)(001.ak47 : 1)
c168	(c093 : 12)
c182	(172.revolver-101 : 7)(021.breadmaker : 5)
c185	(c093 : 12)

Table 2.4: Top 5 categorizations of GANsemble argmax classifier attempting to categorize held-out dataset. The synthesis failure classes dominate the other classes, often even when Things are presented.

	GANsemble (synthesis failures excluded) categorizes held-out dataset
012.binoculars	(c191 : 96)(c118 : 35)(c066 : 20)(c032 : 19)(157.pci-card : 16)
053.desk-globe	(c003 : 26)(157.pci-card : 15)(c118 : 12)(c032 : 10)(c191 : 8)
066.ewer-101	(157.pci-card : 18)(172.revolver-101 : 12)(003.backpack : 11)(c003 : 9)(c066 : 7)
077.french-horn	(c003 : 37)(157.pci-card : 19)(172.revolver-101 : 13)(001.ak47 : 6)(c118 : 5)
101.head-phones	(c191 : 33)(c003 : 33)(021.breadmaker : 16)(c118 : 14)(c066 : 10)
110.hourglass	(c118 : 16)(157.pci-card : 15)(172.revolver-101 : 15)(c066 : 11)(c003 : 9)

123.ketch-101	(157.pci-card : 25)(c003 : 21)(003.backpack : 21) (172.revolver-101 : 12)(183.sextant : 11)
139.megaphone	(c003 : 31)(c118 : 13)(c191 : 11) (172.revolver-101 : 9)(157.pci-card : 8)
156.paper-shredder	(c003 : 39)(c118 : 11)(172.revolver-101 : 10) (c191 : 7)(c032 : 6)
174.rotary-phone	(c191 : 19)(c118 : 17)(157.pci-card : 13) (c066 : 9)(c003 : 7)
177.saturn	(157.pci-card : 38)(001.ak47 : 38)(003.backpack : 8) (227.treadmill : 4)(172.revolver-101 : 4)
182.self-propelled-lawn-mower	(c003 : 76)(c191 : 10)(c066 : 9) (172.revolver-101 : 7)(c118 : 7)
202.steering-wheel	(157.pci-card : 21)(c191 : 15)(172.revolver-101 : 13) (021.breadmaker : 10)(c003 : 8)
211.tambourine	(c003 : 38)(c191 : 17)(c118 : 14) (157.pci-card : 6)(003.backpack : 5)
231.tripod	(c003 : 45)(172.revolver-101 : 14)(c118 : 14) (157.pci-card : 12)(183.sextant : 7)
243.welding-mask	(c118 : 22)(c191 : 19)(c066 : 9) (157.pci-card : 8)(172.revolver-101 : 8)
c002	(001.ak47 : 7)(172.revolver-101 : 3)(183.sextant : 2)
c011	(172.revolver-101 : 5)(157.pci-card : 3)(183.sextant : 3) (c118 : 1)
c064	(001.ak47 : 10)(c066 : 1)(172.revolver-101 : 1)
c065	(172.revolver-101 : 9)(157.pci-card : 2)(183.sextant : 1)
c071	(001.ak47 : 12)
c072	(001.ak47 : 12)
c078	(172.revolver-101 : 9)(183.sextant : 2)(c118 : 1)
c083	(172.revolver-101 : 10)(183.sextant : 1)(001.ak47 : 1)

c090	(183.sextant : 9)(172.revolver-101 : 3)
c119	(172.revolver-101 : 10)(183.sextant : 2)
c134	(001.ak47 : 9)(172.revolver-101 : 3)
c145	(001.ak47 : 8)(172.revolver-101 : 3)(183.sextant : 1)
c158	(172.revolver-101 : 9)(157.pci-card : 1)(183.sextant : 1) (001.ak47 : 1)
c168	(183.sextant : 7)(172.revolver-101 : 4)(157.pci-card : 1)
c182	(172.revolver-101 : 7)(021.breadmaker : 5)
c185	(001.ak47 : 12)

Table 2.5: Top 5 categorizations of GANsemble (synthesis failures excluded) argmax classifier attempting to categorize held-out dataset. Synthesis failure classes have been removed, but the GANsemble does not provide sensible similar miscategorizations, which doubtfully portrays the successful generalization of the GANsemble argmax to other classes or even data (given that our reported accuracies are not cross-validated through the training of the GAN).

	NONGANsemble categorizes held-out dataset
012.binoculars	(003.backpack : 57)(016.boom-boxB : 22)(238.video-projector : 19) (172.revolver-101 : 19)(208.swiss-army-knife : 15)
053.desk-globe	(219.theodolite : 15)(219.theodoliteB : 11)(238.video-projector : 9) (003.backpack : 8)(157.pci-card : 6)
066.ewer-101	(246.wine-bottle : 10)(021.breadmaker : 10)(219.theodolite : 9) (208.swiss-army-knife : 8)(070.fire-extinguisher : 7)
077.french-horn	(157.pci-card : 18)(027.calculator : 13)(219.theodolite : 12) (183.sextant : 10)(172.revolver-101 : 7)
101.head-phones	(003.backpack : 24)(246.wine-bottle : 21)(208.swiss-army-knife : 15) (021.breadmaker : 10)(238.video-projector : 8)
110.hourglass	(246.wine-bottle : 13)(219.theodolite : 12)(070.fire-extinguisher : 12) (003.backpack : 11)(227.treadmill : 7)

123.ketch-101	(227.treadmill : 42)(246.wine-bottle : 19)(014.blimp : 11)(021.breadmaker : 7)(219.theodolite : 5)
139.megaphone	(021.breadmaker : 13)(208.swiss-army-knife : 10)(238.video-projector : 8)(172.revolver-101 : 7)(246.wine-bottle : 6)
156.paper-shredder	(021.breadmaker : 43)(003.backpack : 20)(239.washing-machine : 8)(208.swiss-army-knife : 5)(027.calculator : 3)
174.rotary-phone	(003.backpack : 15)(219.theodolite : 10)(172.revolver-101 : 10)(070.fire-extinguisher : 6)(157.pci-card : 5)
177.saturn	(001.ak47 : 28)(238.video-projector : 23)(246.wine-bottle : 19)(014.blimp : 6)(003.backpack : 5)
182.self-propelled-lawn-mower	(227.treadmill : 46)(208.swiss-army-knife : 41)(027.calculator : 8)(070.fire-extinguisher : 5)(219.theodolite : 4)
202.steering-wheel	(227.treadmill : 10)(003.backpack : 9)(070.fire-extinguisher : 8)(246.wine-bottle : 7)(238.video-projector : 7)
211.tambourine	(021.breadmaker : 23)(238.video-projector : 14)(003.backpack : 14)(208.swiss-army-knife : 10)(227.treadmill : 5)
231.tripod	(001.ak47 : 18)(246.wine-bottle : 16)(208.swiss-army-knife : 13)(172.revolver-101 : 11)(227.treadmill : 8)
243.welding-mask	(003.backpack : 40)(238.video-projector : 9)(246.wine-bottle : 8)(001.ak47 : 5)(070.fire-extinguisher : 5)
c002	(c066 : 9)(070.fire-extinguisher : 1)(027.calculator : 1)(157.pci-card : 1)
c011	(c184 : 3)(246.wine-bottle : 2)(208.swiss-army-knife : 1)(014.blimp : 1)(021.breadmaker : 1)
c064	(227.treadmill : 4)(c049 : 3)(c066 : 2)(219.theodolite : 1)(001.ak47 : 1)
c065	(001.ak47 : 3)(c184 : 2)(246.wine-bottle : 2)(239.washing-machine : 2)(208.swiss-army-knife : 1)
c071	(c066 : 7)(c089 : 4)(c178 : 1)
c072	(001.ak47 : 5)(227.treadmill : 3)(c032 : 1)(219.theodoliteB : 1)(003.backpack : 1)

c078	(001.ak47 : 3)(239.washing-machine : 2)(219.theodolite : 1) (c159 : 1)(014.blimp : 1)
c083	(c118 : 5)(239.washing-machine : 2)(c047 : 1) (014.blimp : 1)(021.breadmaker : 1)
c090	(c045 : 2)(183.sextant : 2)(c184 : 2) (c178 : 1)(070.fire-extinguisher : 1)
c119	(c191 : 5)(c047 : 2)(001.ak47 : 1) (208.swiss-army-knife : 1)(014.blimp : 1)
c134	(c089 : 7)(219.theodolite : 1)(c045 : 1) (c184 : 1)(027.calculator : 1)
c145	(c066 : 4)(c159 : 3)(219.theodolite : 1) (001.ak47 : 1)(c184 : 1)
c158	(c118 : 5)(021.breadmaker : 3)(208.swiss-army-knife : 2) (c045 : 1)(001.ak47 : 1)
c168	(c032 : 4)(172.revolver-101 : 3)(c191 : 2) (219.theodolite : 1)(c047 : 1)
c182	(c118 : 7)(021.breadmaker : 5)
c185	(c066 : 5)(c089 : 2)(027.calculator : 2) (227.treadmill : 1)(001.ak47 : 1)

Table 2.6: Top 5 categorizations of NONGANsemble attempting to categorize held-out dataset. Stuff is more susceptible to thing labeling than Things are to stuff labeling. Compared to the GANsemble, somewhat sensible similarities between classes in the vocabulary of the GANsemble and classes in the held-out dataset are occasionally suggested. For example, desk globe and french horns are claimed by theodolite networks, ewers by wine bottles, tripods by AK-47s, and lawnmowers by treadmills. This suggests that even the information from single network firing of NONGANsembles can be used to build image similarity spaces whereas this is not practical for discriminators trained with the GAN loss and exposed only to one class, but does not rule out similarity spaces being created from ***joint*** information from multiple network activations on the GANsemble committee.

2.3.7 Differential activation levels during and after training of GANs and NONGANs

The previous sections consider GAN and NONGAN activations in a steady-state, after the 15,000 training units have elapsed. Nothing, however, has been shown about the differential *development* of GAN and NONGAN sensitivities, and in particular the acculturation of the level of firing of the networks.

To remedy the lacunarity inherent in our treatment so far, we introduce a "differential trace" procedure. Three familiar classes (hereafter termed the *discrimination triad*) are chosen: a target class (the boom-box), a comparison Thing class (the AK-47), and a comparison Stuff class (the brick wall class c191).

A GAN and a NONGAN are trained from scratch for the now-established standard 15,000 training units, each targeting the target boom-box class in their proprietary manners. The GAN seeks to create convincing boom-box fakes and also a competent fake detector, and the NONGAN attempts to mold itself into a competent boom-box detector, although aided in its quest by being exposed to other classes.

At each training unit along the way, the activation of the network is polled for each of the three classes in the triad, but naturally with no corresponding gradient update to the feedforward passes. Loss (and thus unrectified "prediction") in a neural network is immediately visually recognizable as having the potential to be noisy: to improve our estimate, at each sample point, the 10-sample output activation will be polled. This simply means that the first 10 patches from the class are passed to the network and the 10 unrectified scalar activation values are collected from each of the two networks.

The mean 10-sample activation trace is plotted first for the GAN. Because it is indeed very noisy, a "shape-preserving" Savitzky-Golay filter of wide bandwidth (999 samples, chosen to resemble the automatically chosen plot abcissas, and using an degree 2 polynomial) has been applied to the time series. The standard deviation time series of the 10-sample trace has also been taken, with identical filtering. The process is repeated for the NONGAN.

We see in both cases the development of elevated terminal node firing strength to the target classes 10 patch sample. For the GAN, the maximum firing strength emerges very early at around training step 2000 (well before synthesis convergence) and descends, decelerating and maintaining a large separation between itself and the comparison class. The mean separation is not strong between comparsion classes and actually the Things class is more distantly separated from the target Things class. This, however, makes some sense given that the activation value is negative, which could be associated with the GAN judging that the comparison Thing class is more strongly reminsicent of a fake image of the target class than the comparison Stuff class. As was shown with the affinities, ordinary classification emulated on WGAN-GP emissions is not intepretable in the usual way.

The standard deviation time series is more strongly separated, with the standard deviation remaining roughly flat for the comparsion classes and with the Things class overlying the Stuff class. This is perhaps interpretable purely on the image statistics, which were presented with the dataset – there is simply more intersample variation in the Things class, which of course was also inherent in the Things vs. Stuff definition we have temporarily adopted.

The activation trace of the NONGAN is more exotic-looking and also substantially less in need of the Savitzky-Golay filter. Among the more minor considerations: the Things comparison class only barely overlies the Stuff comparison class nearly everywhere, including in the standard deviation of the firing, suggesting perhaps that NONGAN firing variation is less susceptible to the underlying variation in the image statistics. More notably, the graph of the comparison pair of classes and the graph of the target class are close to symmetrical about the x-axis (signifying no firing). They converge more closely when, at around training unit 10,000, the NONGAN suffers from a failure of gradient descent causing its target firing to drop towards zero and the comparison classes to rise towards zero.

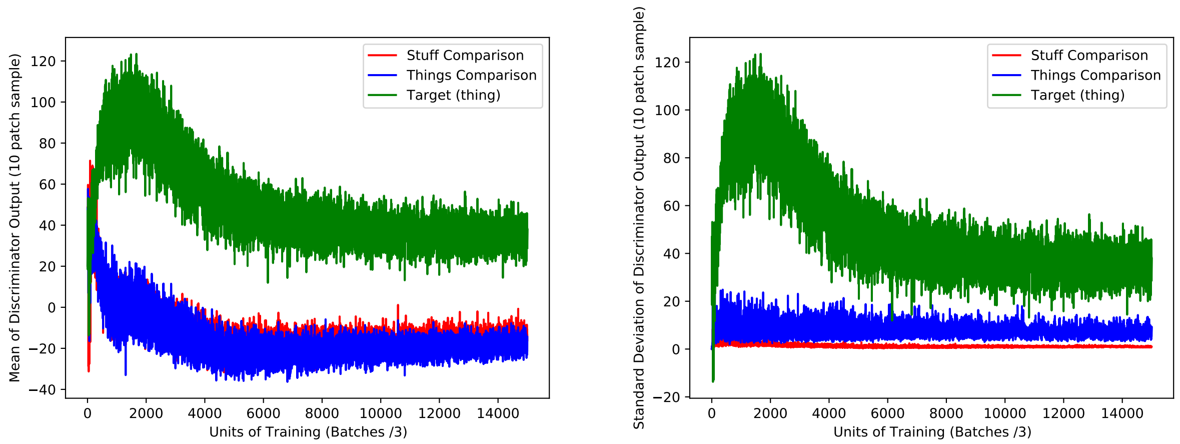


Figure 2.51: Raw differential GAN training trace of the a) mean and b) standard deviation of static real-image 10-sample output activation from a target class and two comparison classes for 15000 units of training (3 batches of 32 image presentations ea.)

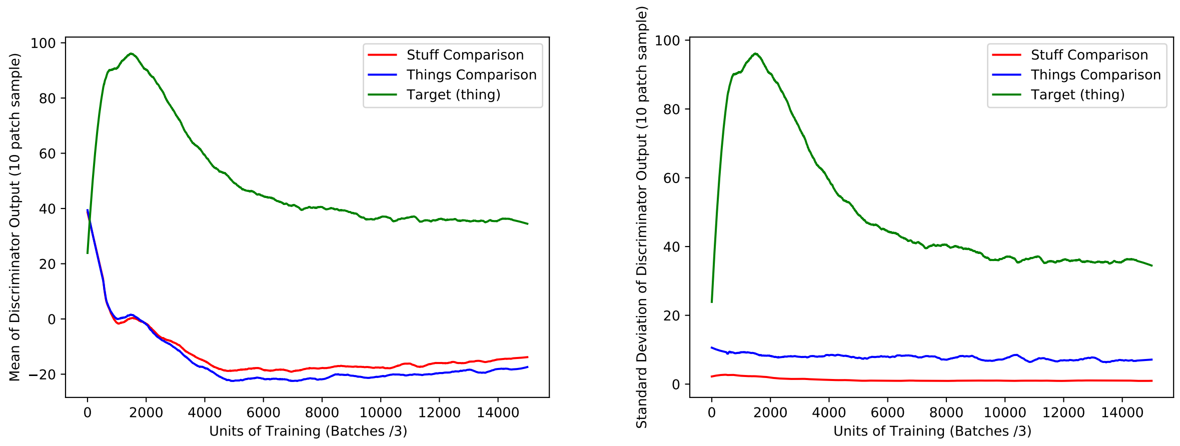


Figure 2.52: Savitzky-Golay filtered differential GAN training trace, filter window of 999 training units. The b) standard deviation of the comparison stuff and things traces are separated but not the mean a), while the network fires high to its target class

At the end of training, if we consider the average NONGAN and baseline-subtracted GAN firing strength, the firing strength of the GAN is still greater even after baseline subtraction. Furthermore a strong group difference between Things and Stuff is observed in a higher firing ratio favoring the Things. Of course, this could be affected by the precision of the image statistics of the Stuff classes, with their low-intersample variation.

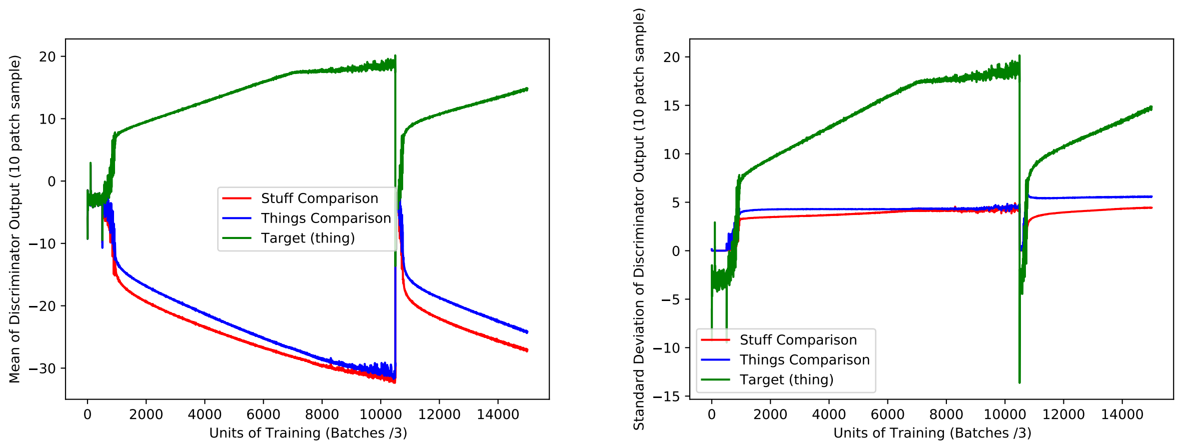


Figure 2.53: Raw differential NONGAN training trace of the a) mean and b) standard deviation of 10-sample output activation. Sample activation undergoes a discontinuity at around unit 11,000 when stochastic gradient descent fails potentially due to overfitting)

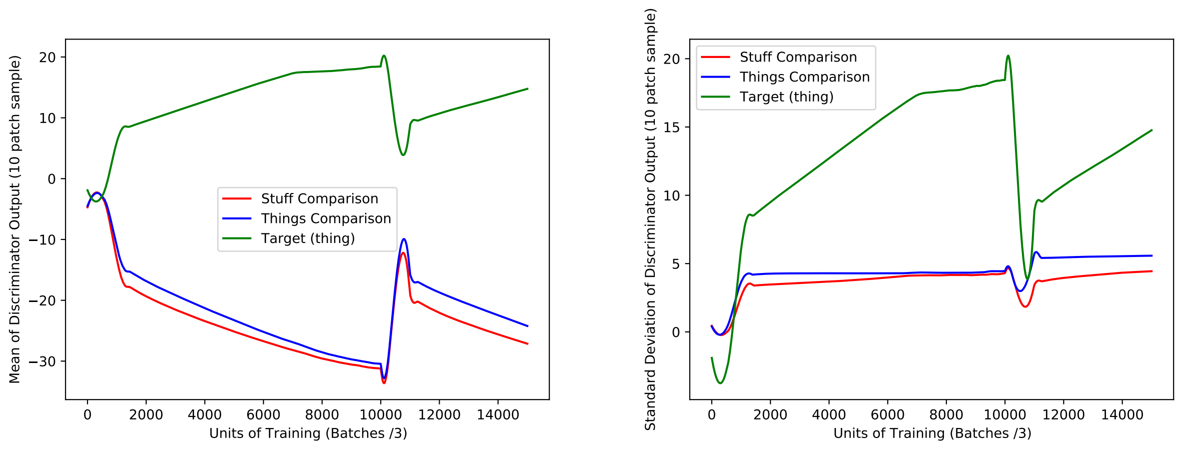


Figure 2.54: Savitzky-Golay filtered differential NONGAN training trace, showing insignificant separation of both a) mean and b) standard deviation of sample activation between thing and stuff comparison classes, still firing high to the target class

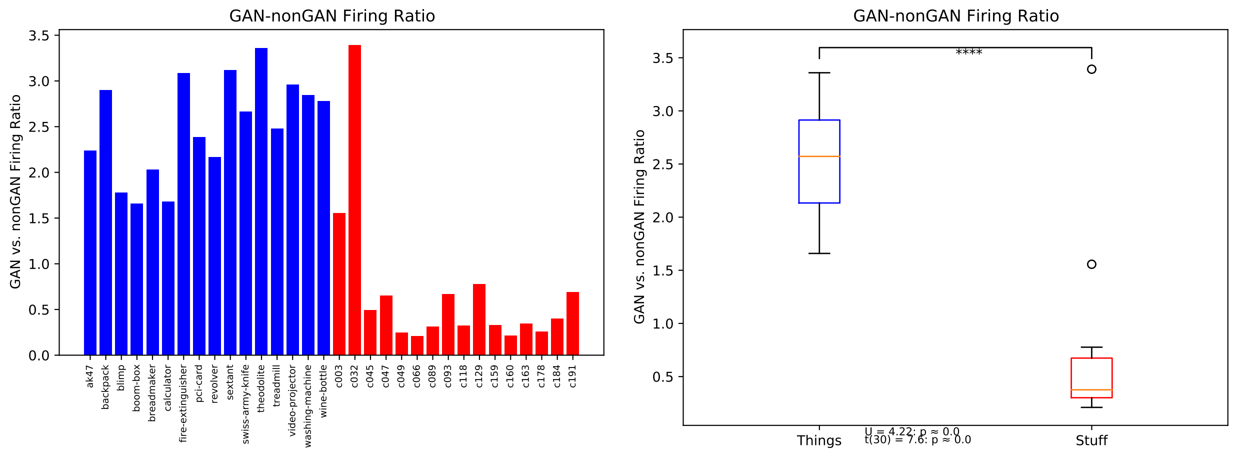


Figure 2.55: Things vs. Stuff comparison of the average GAN/NONGAN firing ratio to a network's own real reference patches with baseline subtraction for the GAN. GAN activations are higher even after baseline subtraction than their NONGAN equivalents, and Thing ratios are dependably higher than stuff ratios. This latter observation could possibly be an artifact of the baseline activation estimate presumably being more reliable for the ultimately homogeneous stuff.

If this trace was to be repeated for a large number of possible discrimination triads, it might perhaps be confidently established that the NONGAN is more susceptible to training discontinuties where the gradient updates suddenly destroy performance after too much training. The very popular strategy of early stopping with discriminator networks in general might then be revised if the element of GAN-based-discrimination that is not susceptible to this could be transplanted into non-GAN-based discriminators without a loss of performance. However, this pattern of robust GAN discriminator learning, since it is only studied for one triad, and not even until very high-quality synthesis is shown to be met, could disappear with the substitution of other classes, unlucky initialization of the GAN, or most problematically with an extension of the training time being surveilled.

This activation trace procedure is only for the action of a single network. The steady-state results shown thus far in this investigation highlight the importance of considering the added power of the joint action of the GANsemble or NONGANsemble in total. The change over time of the joint action of the committee members could be studied as well, as at every time point, there is also a correponding affinity matrix. The time evolution of GANsemble and NONGANsemble affinity matrices, as well as the model parameter matrices, could be studied with a number of methods, but it is possible to take an affinity matrix A_1 preserved at time t_1 and an affinity matrix A_2 at time t_2, compute their difference matrix $A_{1,2}$, and as this difference matrix is a scalar field, compute the gradient, yielding a vector field. One notable vector field would be the one resulting from the difference matrix of an affinity matrix before training and after training. If the Things vs. Stuff characteristic pattern were to acculturate in the final scalar field, this particular vector field difference might demonstrate coherencies (as compared to randomness) at the quadrant divides.

In Chapter 3, embeddings will be computed on these matrices – there is thus even a time evolution study that could be done of the Things vs. Stuff divide in terms of embeddings where things and stuff clouds would be able to be visualized in a video sequence as pulling

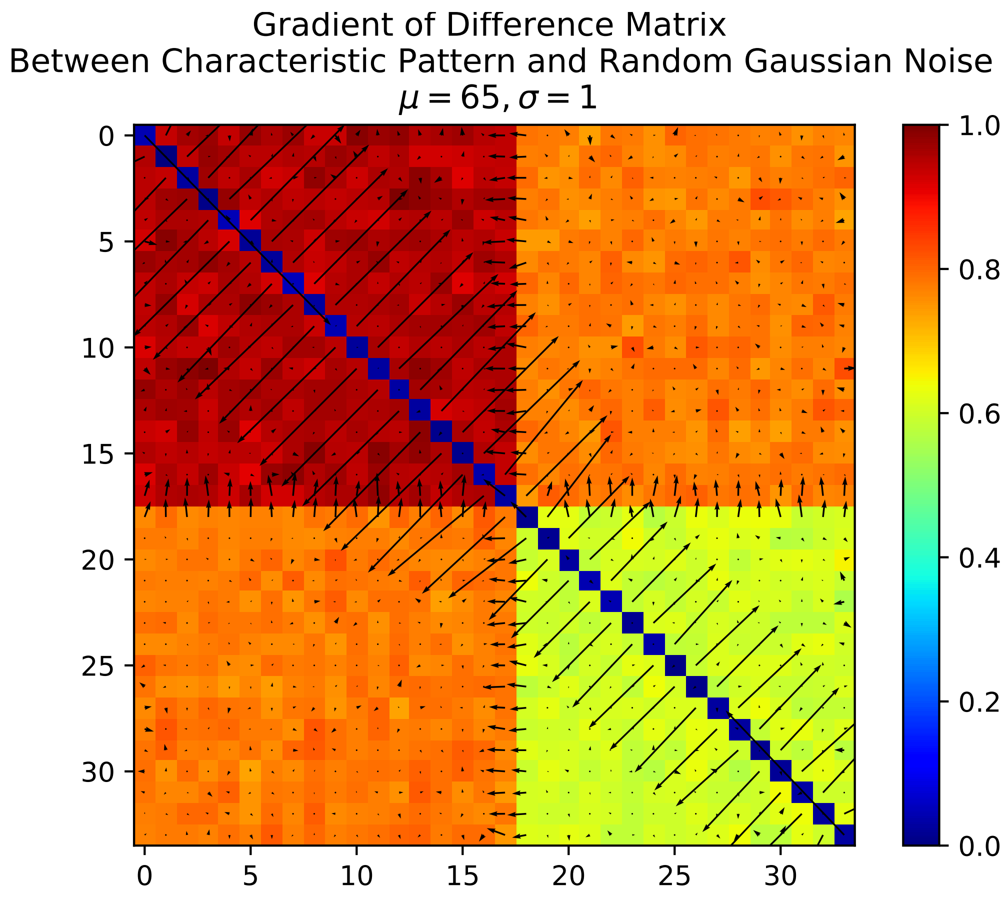


Figure 2.56: The gradient of the mock characteristic pattern matrix of *model parameters* with low sigma subtracted by random noise with the same mean and deviation. In this extreme case, the quadrants are detectable just from the magnitudes and direction at the interfaces in the vector field. Statistically, we may expect more coherence along interfaces over time and more random, cancelling action within the quadrants when examining later differencing periods in training.

apart for some recognizers throughout training but with the added local and global uncertainty caused by the stochasticity of the embedding process. Both the time evolution of matrices study and the time evolution of embeddings study were not pursued because the GAN state at a training time t would have to be sampled and retained as the quite large networks were trained – and this would have to have been preplanned and would have added considerable time and storage requirements that are beyond the scope of this project. Nevertheless, studying the time evolution of firing affinity matrices and embeddings presents an interesting extension of evaluation of training that could be used in conjunction with traditional per-model losses in neural network ensembles, when evaluated very coarsely given the computational demands.

Therefore, notwithstanding this section, the investigation-at-large has been confined to the steady-state behavior of the networks being considered.

2.3.8 Compressibility of GAN vs. NONGAN weights

The homogeneity of the image statistics of the Stuff would lead one to consider that the weights of the network might be affected differentially in terms of how varied they are. As we examined the GAN-to-NONGAN firing strength ratio, we can also take a measurement related to the compressibility of the weights. A natural hypothesis is that the NONGAN's weights, since they would tend to remember the more varied information they receive from out-of-class patches, would be less compressible.

We can take the 256-bin extemporaneous histogram of weights from the NONGANs and the GANs per-class. Recall the caveat that the extemporaneous histograms are not defined over the same discretized probability space. However, if we trust the relative comparability of resulting measurements, we can estimate the entropy of each distribution.

Entropy (in bits) is:

$$\mathbb{H}(X) = -\sum_{k=1}^{K} p(X=k)\log_2 p(X=k)$$

summing over the transformed probabilities encountered at the weight levels in the extemporaneous histogram.

The per-class difference of the estimated entropy of the GAN weight distribution and its complementary NONGAN weight distribution is plotted in Figure 2.57. It is negative for most classes, with no statistically significant difference between the Thing and Stuff classes.

This means that the NONGAN has greater estimated entropy than its GAN complement for most classes.

It is problematic to relate the compressibility of real floating-point data on disk using real-world compression algorithms because of the construction of dictionaries, varying padding and numeric representation schemes, and compressed blocks within compressed blocks, so if we take the entropy as estimated to be indicative of the compressibility, the GAN weights are generally more compressible, as expected, implying that the NONGAN networks "soak up" more information.

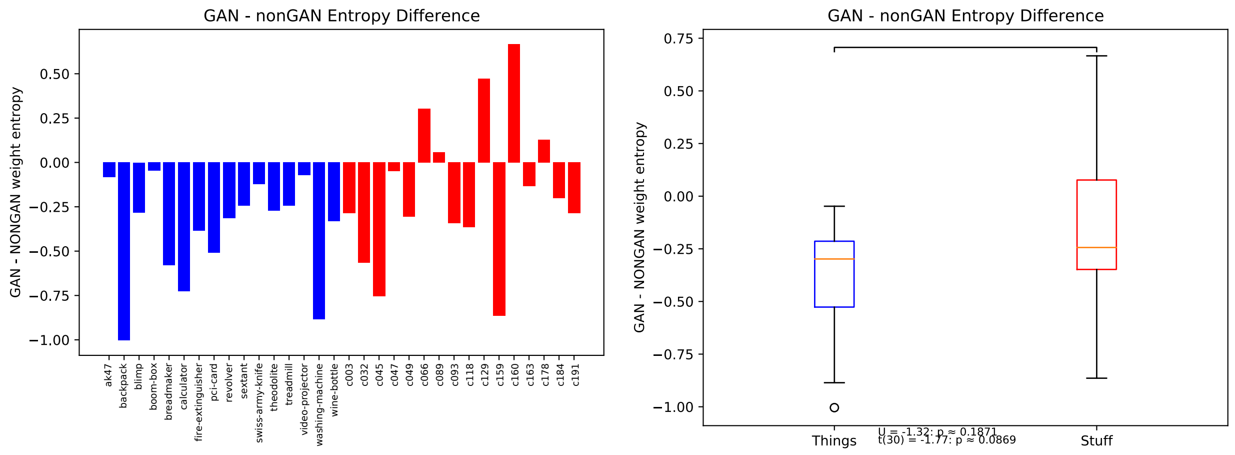


Figure 2.57: Things vs. Stuff comparison of the average entropy difference of discriminator weights. All weights from all layers are combined and then the entropy of the weight distribution based on (separate) 256-bin adaptive histograms are measured. The entropy difference of the GAN and NONGAN shows a clear negative bias, especially for things. This means that the NONGAN's entropy is higher, perhaps reflecting how its training requires being exposed to all of the other classes and memorizing statistically distinct input. There is no significant difference between Things and Stuff in this regard.

2.4 Things vs. Stuff Biases in "Gold-Standard" Reference Networks

If the Things vs. Stuff dichotomy is comprehensively real, it should be seen even in networks which are not the member networks in our GANsemble and NONGANsemble. Accordingly, attention turns to two notable networks in use for quality estimation and feature map visualization respectively.

2.4.1 InceptionV3 as a thing-biased quality oracle

Beyond the adversarial trick of pitting discriminator against generator, the GAN itself lacks an explicit objective function that could be closely associated with the perceptual quality of samples. A popular quality index to report, however, has been the Inception Score of Salimans et al. [140]. The Inception Score uses the quite large and complicated Inceptionv3 network [158] as a defacto quality oracle. Specifically, Inception Score was defined as

$$IS = e^{(\mathbb{E}_x KL(p(y|x)||p(y)))}$$

where $p(y|x)$ is the conditional label distribution, and $p(y)$ is the distribution of labels, at the output of the network, and KL denotes the familiar KL divergence.

The motivation was to have a distribution $p(y|x)$ with low entropy, and to further have $\int p(y|x = G(z))dz$ have high entropy. The secondary criterion simply expresses that across the range of noise inputs to the generator, the discriminator should be experiencing a wide range of inputs that leads to a wide range of classifications. The first criterion corresponds to desiring a peaky or overall confident label distribution per class so that within a class, the discriminator is only placing high probability on a few possible target classes.

Strictly, this criterion is only directly naturally applicable while adhering to the classes used by the oracle network trained. However, in practice, new classes you wish to fake that are (to misappropriate a term from natural language processing) out-of-vocabulary of the Inception network (evaluated on 1000 classes from the ILSVRC 2012 classification challenge [137] validation set) will be somewhat similar to some classes but not others that have graced the pretrained Inception network's sensorium.

In computing the average dichotomous Inception Score using the Inceptionv3 network, you are at the mercy of whether your synthesis quality is truly biased or the Inception score is biased by one overarching category being more similar to the Inceptionv3 class vocabulary than the other.

An ideal test case for the possibilty of this is the comparison between Things and Stuff. Most competition datasets in computer vision are either totally or near-totally Thing-based, and so it seems a priori likely that the conditional label distribution will be flat and uninformative across the range of Stuff classes. The associated hypothesis is that, in computing the Inception Score of our present dataset, the Inception Score will systematically judge the quality of Stuff synthesis as lower than Thing synthesis despite the fact that at 15,000 training units it is manifestly perceptually more advanced than perhaps any of the Thing classes, synthesis failures of stuff excepted.

The expectation $\mathbb{E}(x)$ being taken prior to the exponentiation is the average over images but it is also possible to compute this score over several splits of the data. The implementation used here very closely followed that of [20], which computes the Inception Score over ten subsets and also smooths the probabilities by adding a very small ϵ = 1e-16 prior to log transformation; the choice of a small epsilon may be particularly important here as zero probabilities are expected. To match the expected tensor dimensions expected by the network, the Thing and Stuff patches were resized to 299x299, the single gray-level channel was replicated two times to fake three-channel input, and the image was preprocessed according to the prescription of Keras (gray levels rescaled into the -1 to 1 interval).

In Figure 2.58, it can be seen that the hypothesis of bias is substantiated. There is a highly significant distance in the estimated Inception Score, tending for a higher score for Things even though the subjective attained synthesis quality of Stuff is uniformly higher. The Inception Score also fails to detect the synthesis failures as anomalies that are not acceptable quality.

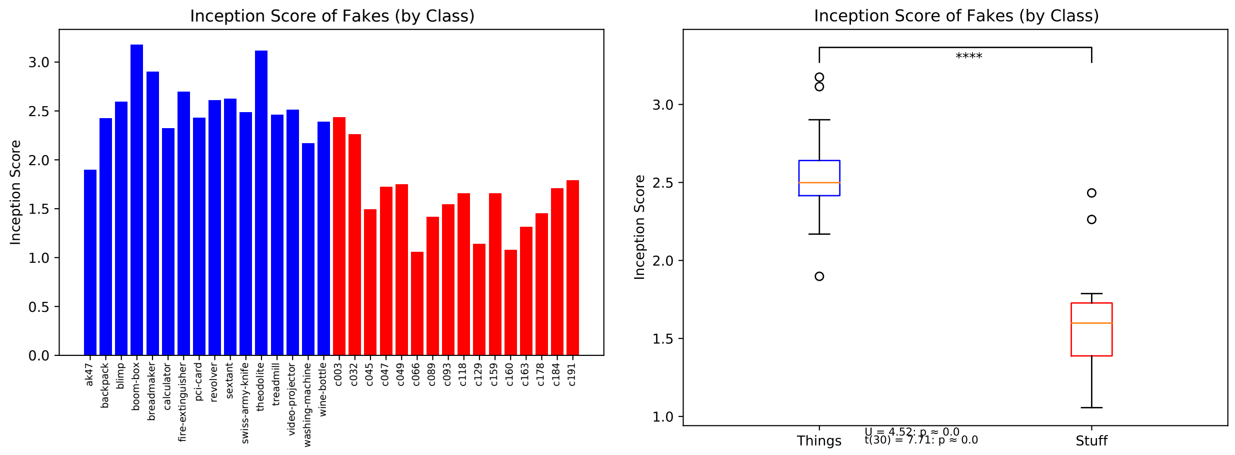


Figure 2.58: Things vs. Stuff comparison of per-class Inception score, calculated by sending the GANsemble's fakes to the Inceptionv3 network (100 samples with 10 splits) and consulting the entropy of the label distribution. Despite the almost perfect perceptual quality of Stuff patches, the things earn a systematically higher score. This is likely due to the implicit bias of the network having been trained on other Things patches.

2.4.2 Fréchet Inception Distance for Things vs. Stuff: Reals-to-fakes, fakes-to-fakes & reals-to-reals

A potentially anticorrelated measure involving the same network that, however, is not directly subject to the label distribution Things vs. Stuff entropy bias is the Fréchet Inception Distance, introduced by Heusel et al. [76] in their work on a two time-scale update rule

(TTUR) that prescribes the differential learning rate of the discriminator and the generator in GANs.

The Fréchet distance alluded to (which they note to be equivalent to the Wasserstein-2 distance, c.f. the Wasserstein-1 distance used in WGANs) is a specific case of the general Fréchet distance (the tether-like shortest distance between two paths corresponding in, for example, time), specifically the Fréchet distance between multivariate Gaussians [42]

$$d^2((\mu_1, C_1), (\mu_2, C_2)) = \|\mu_1 - \mu_2\| + \mathrm{Tr}(C_1 + C_2 - 2(CC_2)^{1/2})$$

where μ_1 and μ_2 are the means and C_1 and C_2 are the covariances of the first and second Gaussian respectively, and the final term contains a matrix square root (e.g. $BB^T = A$, which may fail if A is singular).

The Inception element of FID is integrated in that the multivariate Gaussians you are comparing are based on the mean and covariances of the activations extracted from the pooling layer prior to Inceptionv3's output. Conventionally, you take the distance between the Gaussians based on empirical means and covariances of a batch of GAN fakes and a batch of real images that generated them. Similarly to the Inception Score, the implementation closely followed [19] and included the same preprocessing prescribed for the network.

Figure 2.59 shows that the difference between Things vs. Stuff of per-class Fréchet Inception Distance between reals and fakes is highly significant, although not as overwhelmingly significant as the Inception Score difference. Importantly, since the FID measures how poorly the fakes match the reals in this context, then the difference means that the FID is, consistently with its spirit, declaring that the Thing fakes are poorer quality. However, the FID fails to uniquely identify the synthesis failures, flagging only c129 and not c093.

Although it is assuredly not the main intention of the FID, the measurement mathematically isn't restricted to comparing same-class data. So a pairwise FID affinity matrix is calculable for Reals and Fakes. The appearance of these matrices (Figure 2.60 and Figure 2.61, respectively) is close to identical on casual inspection, with more gradation actually represented in the Fake-based matrix, and the overall pattern is, surprisingly, low FID in the Things-Things quadrant. This suggests that FID is doing something strange – possibly based on the fact that the statistics of Stuff classes are more sharply defined, and so if you are taking variance-corrected distances, they really might be more distant in the space induced by summarizing the Inception coding layer's activations.

Finally, in Figure 2.62, the Inception Score of fakes per-class is plotted directly against the FID between real and fakes. Conceptually, these should be anticorrelated. However, the measures are overall positively correlated. Outliers were eliminated using a RANSAC process (RANdom SAmple Consensus [49], see `sklearn.linear_ model.RANSACRegressor`[130]) before plotting lines-of-best-fit but after taking the correlation coefficient, showing the significant influence of outliers in reversing the sign of the Stuff slope. The strength of the overall correlation would decrease considerably when the Thing and Stuff subsets are considered separately.

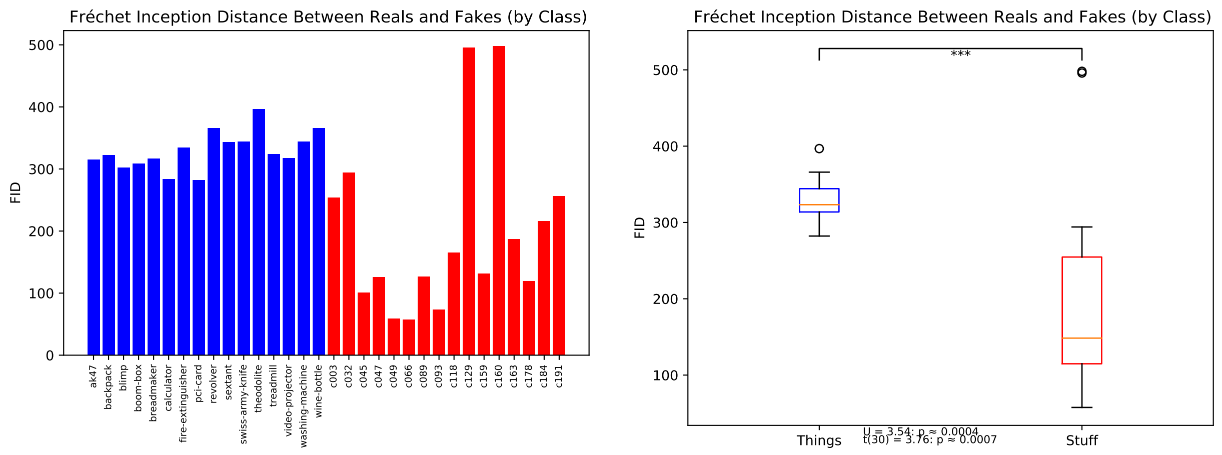


Figure 2.59: Things vs. Stuff comparison of per-class Fréchet Inception Distance, based on assessing how real and fake patches affect the final pooling layer activations of the Inceptionv3 network, specifically real and fake Wasserstein-2/Fréchet difference as modeled by a multivariate Gaussian. The Things have a higher observed distance which implies that their fakes and reals are not as similar as the fakes and reals for the Stuff patches. This is the perceptually expected result.

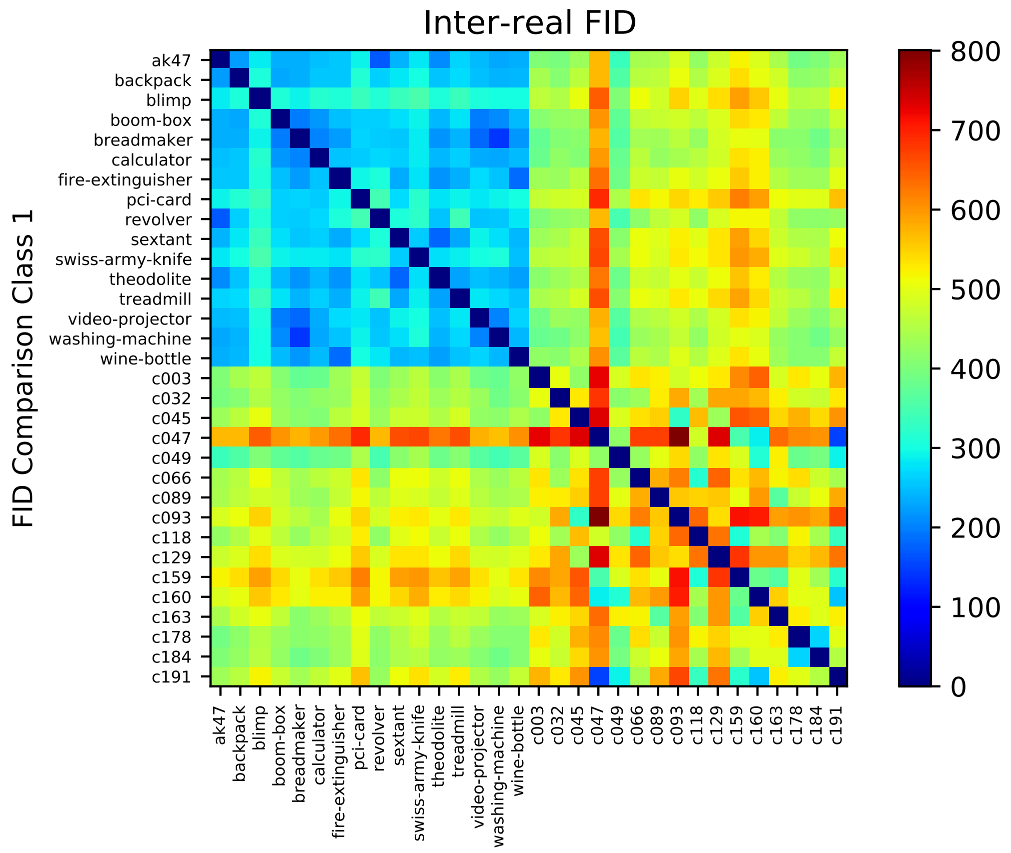


Figure 2.60: The FID-induced affinity matrix between classes for real patches, measuring interclass difference through an improper use of FID.

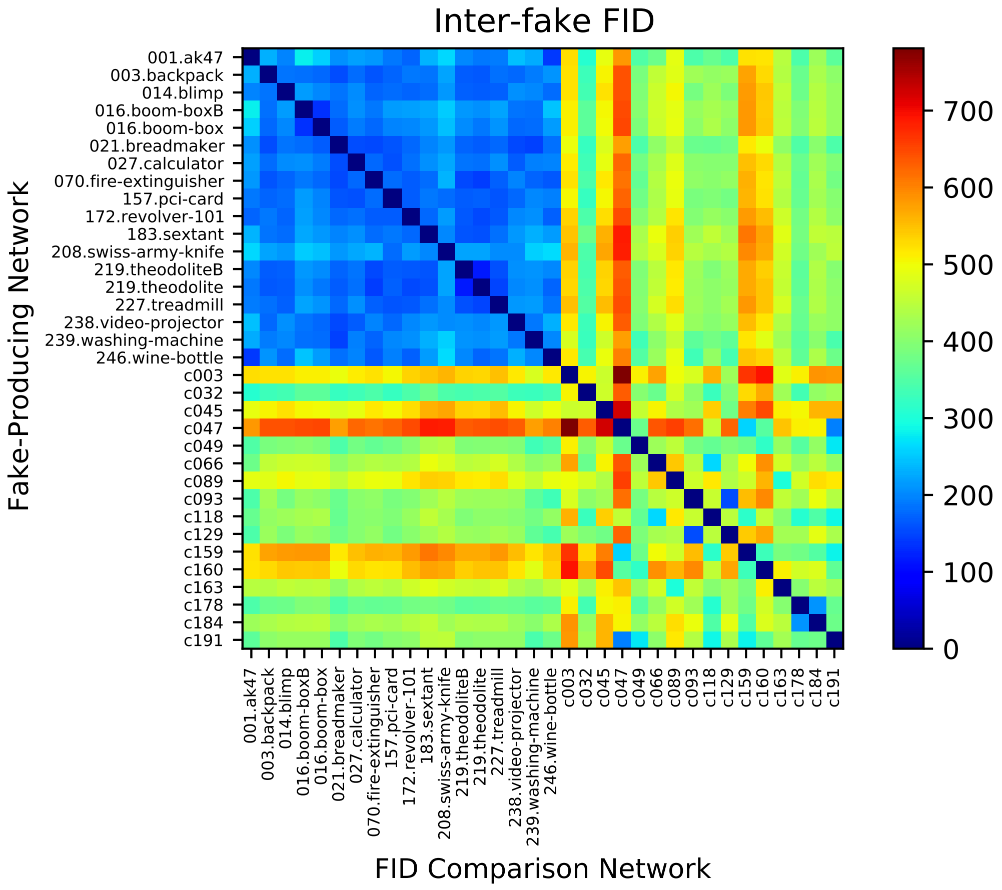


Figure 2.61: The FID-induced affinity matrix between networks for fake patches, measuring internetwork difference through an improper use of FID. As potentially expected, the interclass FID for fakes looks substantially like FID for reals.

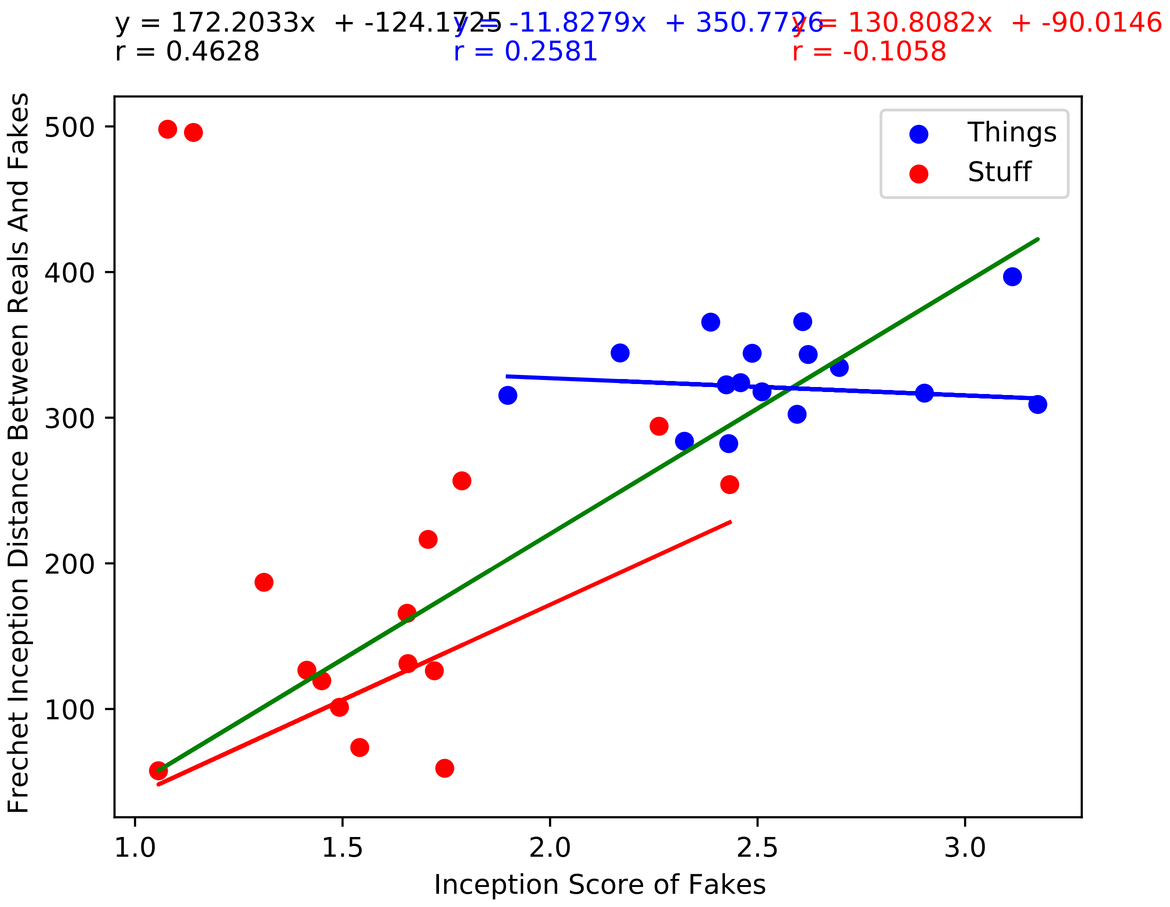


Figure 2.62: Correlation of Inception Score of Fakes and Fréchet Inception Distance between Reals and Fakes. Regression lines of best fit have been determined with outliers eliminated through a RANSAC process for the stuff alone, the things alone, and both combined. The expectation is that FID and IS would be anticorrelated.

2.4.3 Things and Stuff differentially engage late-stage filters in VGGNet

The Inceptionv3 network is a gold standard with respect to experiments in quantifying quality, but one other network of historical interest has been VGG16, sometimes used for visualizing filter specialization, owing to its relative simplicity as compared to Inception nets. Knowing where in the network filters are general purpose for a space of natural images (for example, when the filters of the first convolutional layer resemble edge detectors), and by which later stage they have been specialized (e.g. a late filter that directly resembles something like the particular instance of a class) is a subject more of interest to computational vision and neuroscience researchers. The original Inception network [157] and the associated DeepDream [4] program that created psychedelic art by optimizing images to maximally stimulate a late stage artificial neuron spurred popular interest, but landmark work by Zeiler and Fergus [183] and follow-up techniques from Simonyan et al. [149], and especially Yosinski et al. [181] focused on determining which images in the training set maximized responding at the filter channel level.

Zeiler and Fergus's approach accumulates activations of interest from a layer by attaching a "deconvnet" to the layer almost as one might attach a debugger to a piece of software. The deconvnet takes the layer of interest's output and "unpools" it (reconstitutes areas of activation in the downstream representation that were collapsed to points from regions of maxima in the upstream representation using a series of switch variables), rectifies it with ReLU, and filters (but attempting inversion by using transposed versions of the upstream filters) it. This involved process and the later refinements are focused on producing the best possible reconstruction of filter response to an input patch: presenting pictures of animal heads may discover filter channels that princpally seem to produce deconvnet maps highlighting the eyes, increasing confidence that there are explicit eye filters. In a convolutional network, the only purely visualizable filters are the first set of filters, as the path of convolutions through the network constitutes the true filter, not the contents of the filter channel at the current layer.

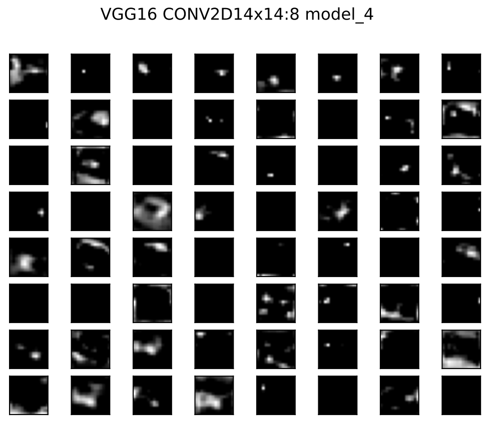
For judging relative activation coarsely, however, these involved and thoughtful inverse operations are not needed. One can derive coarse activation maps that display the activity of the filter channel output based on input patches purely by considering the activations from those channels at the end of the Conv2D block. We take the images in our dataset and map them into the proper dimensions and gray-level domain for VGGNet which is 224x224 3-channel images. The same channel duplication strategy used with the Inceptionv3 network has been used here, but the gray levels have been mapped into the [0,1] range instead of the [-1,1] range from the conventional [0,255].

In Figure 2.63a we see the first 64 of 512 filter channel outputs for the last Conv2D block with extent 14x14, as they would appear when the network was subjected to a single image from the boom-box class. In Figure 2.63b, we see the same channel visualization but for an image from the final brick wall class c191.

These images are presented so that the reader takes them to be prototypical of the Things vs. Stuff distinction as viewed through late stage VGG activations. Stuff patches fill fewer of the channels but more of their extent. Thing patches, however, induce the stereotypical filter response map result of convolution, which is a focal dot where the target shape matching the template of the filter would reside: for example, if the template was something even as high-level as a STOP sign and made the filter, it would when convolved with a scene tend to produce a focal activation at the STOP sign in the scene's activation map.

The reader is not expected to take this on faith, however. Repeating this process for all of the image in our Things vs. Stuff dataset, we can compute three averages of interest to quantify this difference.

The first average to take is the per-class average of the conditional raw average of channel



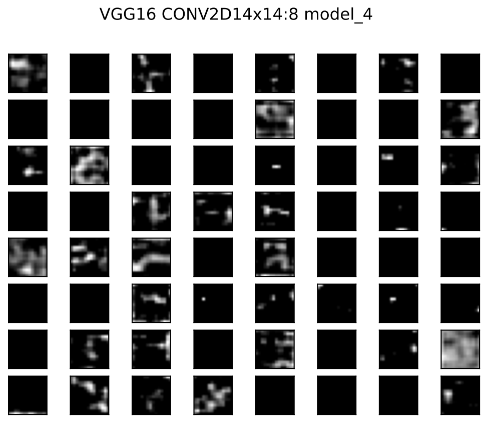


Figure 2.63: Filter channel activations (first 64 channels) from the VGG16 network's last Conv2D layer when presented with a) a boom-box image, b) a brick wall image. Analysis using the whole dataset shows that Stuff patches engage fewer channels but fill them more widely, whereas for thing patches focal activation points which resemble the stereotyped output of a simple convolutional detector are common.

activation, where raw average is taken to mean the average activation value in each channel slice of the Conv2D block tensor. The values are not strictly clipped to be in the [0,1] range following the convolution. Only active channels with some activation value above zero anywhere within the response map are included, hence the average being conditional. This is because we wish to indirectly quantify the width of the actually-realized filter activations. A more direct and accurate approach would use 8-connected components analysis or perhaps Hough transforms to explicitly find and measure the radii of candidate activation blobs, but for expediency this simple approach suffices.

The second average to take is the per-class average of the conditional binary-threhsolded average of channel activation, where the meager innovation adopted over the last average is to threshold each pixel contributing to the average as ON or OFF corresponding to its nonzero status. This average is an appropriate check if you are aiming at discovering the area active within the channel and you do not wish that estimate to be biased by especially high-activation level values from the convolution. We see that the general pattern of the raw average survives in the binary thresholded average.

The final average to take is the per-class average of the proportion of *quiescent channels* in the tensor. This is the fraction of channels that are “dead” as previously mentioned forming the condition from an activity standpoint when the probe patch is presented (i.e. with (binary-thresholded) average or sum of channel activation equaling zero). Note that this is distinct from the problems of “dead neurons” sometimes talked about with respect to learning, where the neuron is connectively dead due to the severing of incoming connections, or incurs no gradient update due to saturation of the neuron’s nonlinear activation function.

In Figure 2.65, it can be seen that there is a highly statistically significant difference between Things and Stuff as measured by the Wilcoxon test for the raw average activation for active channels, where Stuff classes activated the channel more strongly. To confirm that this is not just biased by exceptionally strong focal activation, the occupancy of the channel is shown through the per-class binary threshold average in Figure 2.66, where there is again a highly significant difference tending the same way. Interestingly, as shown in Figure 2.67, this difference is also sustained for the proportion of quiescent channels. This means that fewer channels at the end of the VGG16 network near to the 1000-class multinoulli judgment are active when presented with the Stuff patches, but when they are active they are significantly more active. Possibly this suggests a population composed of broadly-tuned and sharply-tuned filter channels.

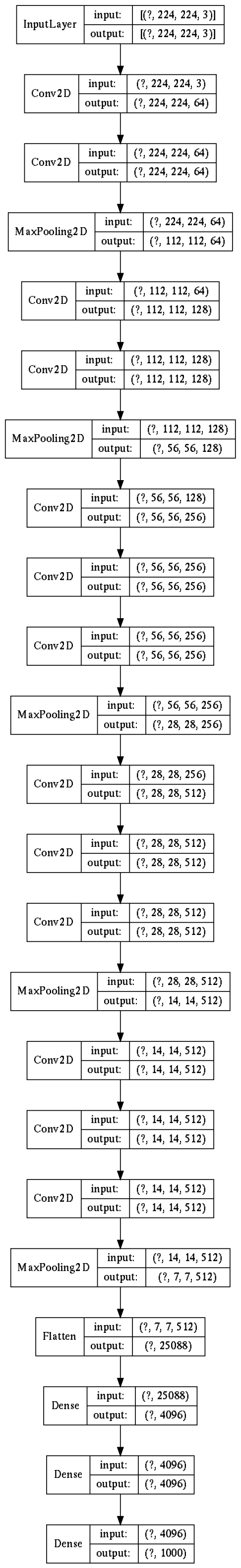

InputLayer | input: [(?, 224, 224, 3)] | output: [(?, 224, 224, 3)]
Conv2D | input: (?, 224, 224, 3) | output: (?, 224, 224, 64)
Conv2D | input: (?, 224, 224, 64) | output: (?, 224, 224, 64)
MaxPooling2D | input: (?, 224, 224, 64) | output: (?, 112, 112, 64)
Conv2D | input: (?, 112, 112, 64) | output: (?, 112, 112, 128)
Conv2D | input: (?, 112, 112, 128) | output: (?, 112, 112, 128)
MaxPooling2D | input: (?, 112, 112, 128) | output: (?, 56, 56, 128)
Conv2D | input: (?, 56, 56, 128) | output: (?, 56, 56, 256)
Conv2D | input: (?, 56, 56, 256) | output: (?, 56, 56, 256)
Conv2D | input: (?, 56, 56, 256) | output: (?, 56, 56, 256)
MaxPooling2D | input: (?, 56, 56, 256) | output: (?, 28, 28, 256)
Conv2D | input: (?, 28, 28, 256) | output: (?, 28, 28, 512)
Conv2D | input: (?, 28, 28, 512) | output: (?, 28, 28, 512)
Conv2D | input: (?, 28, 28, 512) | output: (?, 28, 28, 512)
MaxPooling2D | input: (?, 28, 28, 512) | output: (?, 14, 14, 512)
Conv2D | input: (?, 14, 14, 512) | output: (?, 14, 14, 512)
Conv2D | input: (?, 14, 14, 512) | output: (?, 14, 14, 512)
Conv2D | input: (?, 14, 14, 512) | output: (?, 14, 14, 512)
MaxPooling2D | input: (?, 14, 14, 512) | output: (?, 7, 7, 512)
Flatten | input: (?, 7, 7, 512) | output: (?, 25088)
Dense | input: (?, 25088) | output: (?, 4096)
Dense | input: (?, 4096) | output: (?, 4096)
Dense | input: (?, 4096) | output: (?, 1000)

Figure 2.64: VGG16's last high-level Conv2D block finally produces 512-channel output measuring 14x14 pixels which is where we measure the utilization of channels and the number of dead channels relating to Things vs. Stuff

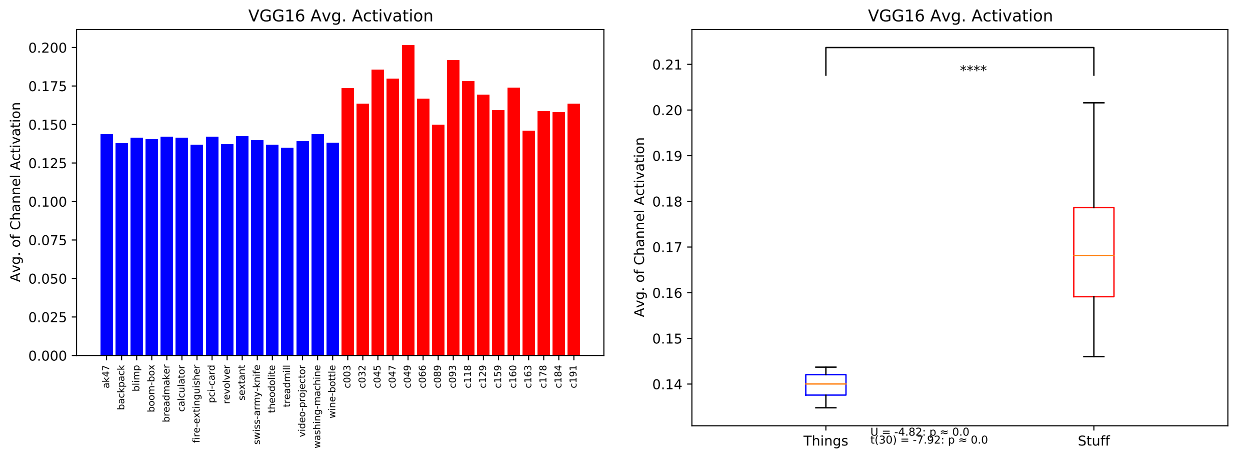


Figure 2.65: Per-class channel average of activation in the last convolutional layer of VGG16.

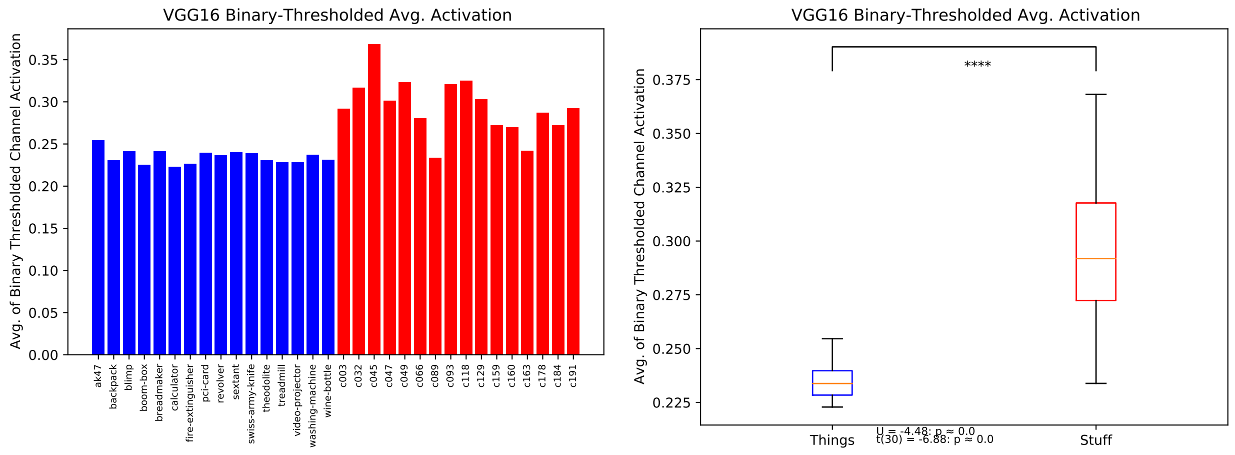
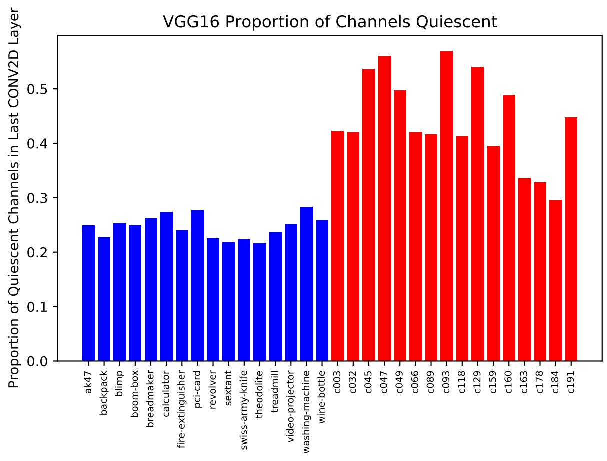


Figure 2.66: Per-class channel average of binary-thresholded activation in the last convolutional layer of VGG16.



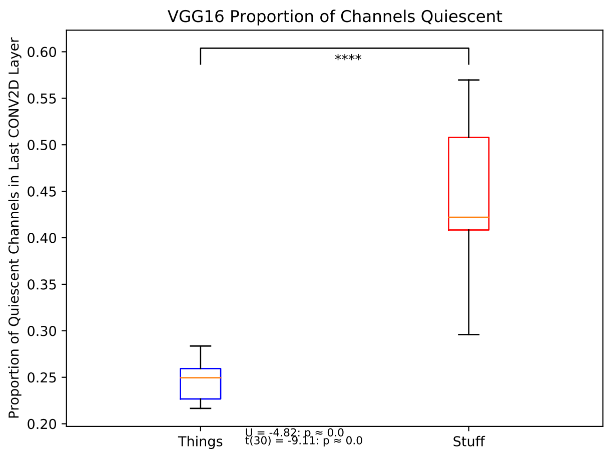


Figure 2.67: Per-class proportion of quiescent channels in the last convolutional layer of VGG16.

2.5 Recognition performance of LDA and argmax classifiers using GANsemble and NONGANsemble label emissions

Assessing the performance of the GANsemble and NONGANsemble follows closely the strategy in Section 1.4.2. The LDA classifiers are still cross-validated but they take as input the features from the GANsemble and the NONGANsemble, which are not cross-validated. The feature extraction involves no further filtering beyond what is done within the individual member networks to produce their terminal activations.

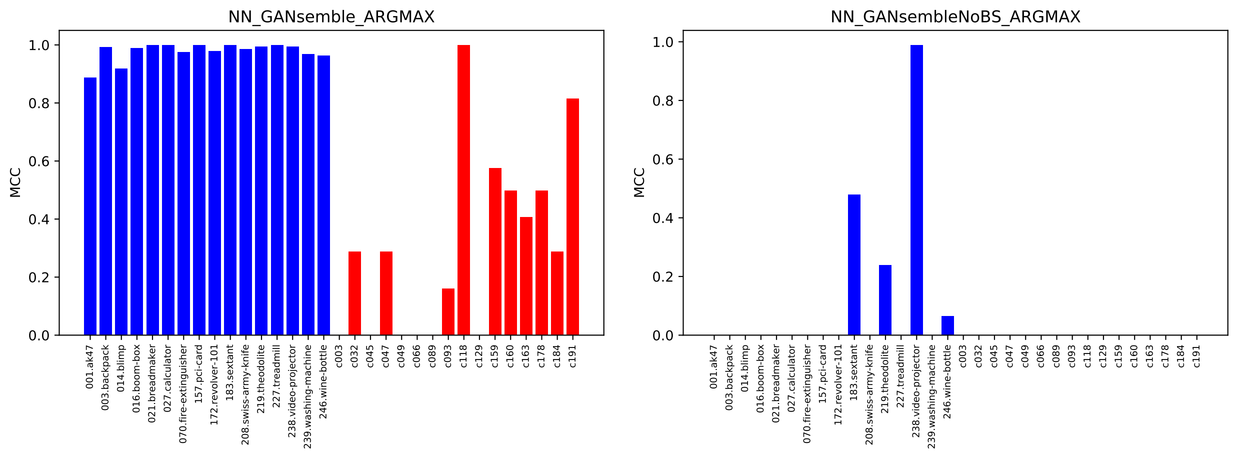


Figure 2.68: Per-class Matthews Correlation Coefficients of simple argmax classifier using GANsemble label emissions, with and without baseline subtraction

In Figure 2.68, the performance of the GANsemble according to the per-class Matthews Correlation Coefficients using the argmax classifier is shown, with and without baseline subtraction. It is apparent that baseline subtraction is critically important for Things, raising the average accuracy well above 85%. It increases the MCC for a number Stuff classes too, although not to the same extent. It is not the case that this means that Things are easier to recognize in deep convolutional recognizers in general, however. Recall that earlier in Chapter 2, the flat nature of the real and fake affinity matrices in the Stuff-Stuff quadrant

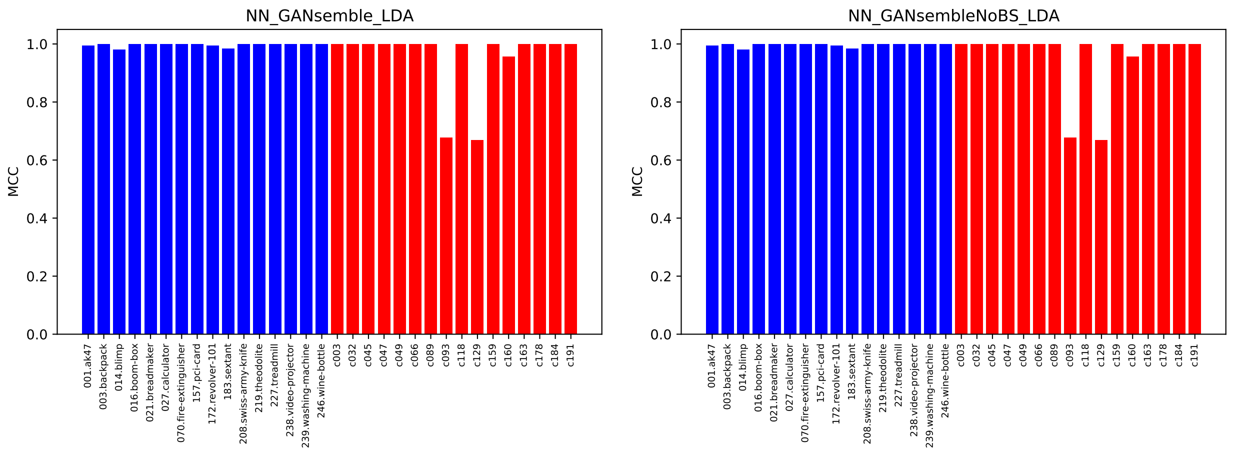


Figure 2.69: Per-class Matthews Correlation Coefficients of LDA classifier using GANsemble label emissions, with and without baseline subtraction

(QIV) suggested that similarity and recognition quality would be hampered among Stuff comparisons. The effect of this is what is shown in Figure 2.68b, but the other consideration is the inclusion of the synthesis failure classes, as mentioned in Section 2.3.6, when similarity was assessed through the out-of-vocabulary labeling on the heldout dataset. Even with baseline subtraction, nothing has been done to remove the synthesis failure classes.

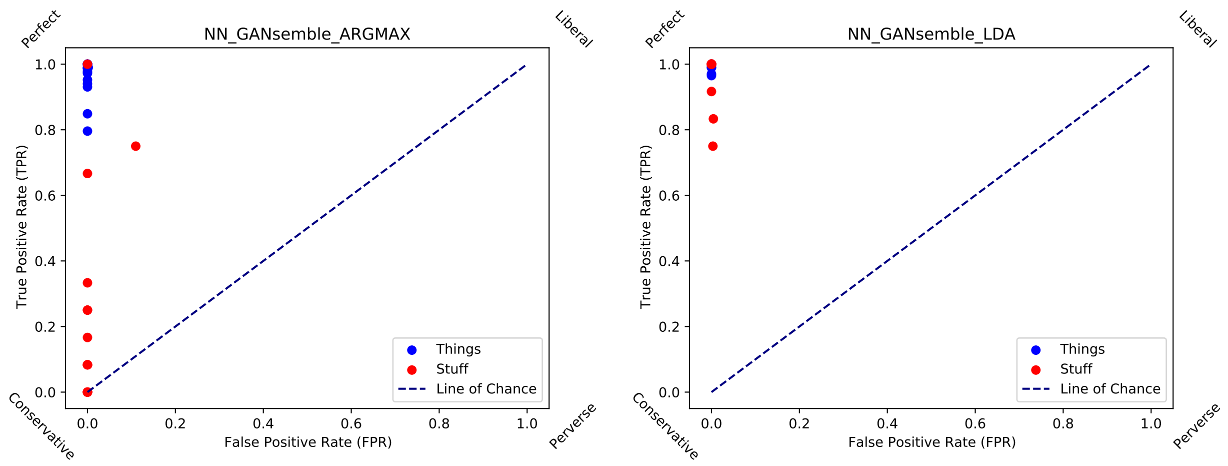


Figure 2.70: Receiver Operating Characteristic Plot of argmax and LDA classifier performance for the GANsemble. The LDA classifier has much better performance on stuff and improved performance on things. It avoids false positives scrupulously (again, excepting the synthesis failure classes). The argmax classifier also avoids issuing many false positives (the clear outlier is synthesis failure class c093).

Rather than do that, which is an ad hoc kind of measure if an automated method for discovering synthesis failures from their anomalous contribution to the parameter distance is not also endorsed, Figure 2.69 considers the case in which the argmax classifier is replaced with the LDA classifier using the same feature vector. Performance is now close to ceiling, with significantly reduced MCC only being observed for two classes, which are exactly the two synthesis failure classes. In Table 2.8, it is suggested that this inaccuracy is due to synthesis failure classes trading off when they make mistakes. Figure 2.70 shows the difference in ROC space between the argmax classifier and the LDA classifier. For the LDA classifier, nearly all of the classes are cleaving to the TPR axis, with some Stuff classes (the failure classes), exhibiting TPR as low as below 0.8. In the argmax classifier's ROC space, the anomalous synthesis failure class c093 can be seen with high false positive rate, and this can be seen directly by examining the Precision or directly the number of False Positives for the class in Table 2.7.

GANsemble ARGMAX

Class	TP	TN	FP	FN	Precision	Recall	d'	MCC
001.ak47	78	1772	0	20	1.0000	0.7959	inf	0.8872
003.backpack	149	1719	0	2	1.0000	0.9868	inf	0.9928
014.blimp	73	1784	0	13	1.0000	0.8488	inf	0.9180
016.boom-box	89	1779	0	2	1.0000	0.9780	inf	0.9884
021.breadmaker	142	1728	0	0	1.0000	1.0000	inf	1.0000
027.calculator	100	1770	0	0	1.0000	1.0000	inf	1.0000
070.fire-extinguisher	80	1786	0	4	1.0000	0.9524	inf	0.9748
157.pci-card	105	1765	0	0	1.0000	1.0000	inf	1.0000
172.revolver-101	98	1768	3	1	0.9703	0.9899	5.252731	0.9789
183.sextant	100	1770	0	0	1.0000	1.0000	inf	1.0000
208.swiss-army-knife	106	1761	0	3	1.0000	0.9725	inf	0.9853
219.theodolite	83	1786	0	1	1.0000	0.9881	inf	0.9938
227.treadmill	147	1723	0	0	1.0000	1.0000	inf	1.0000
238.video-projector	96	1773	0	1	1.0000	0.9897	inf	0.9946
239.washing-machine	79	1786	0	5	1.0000	0.9405	inf	0.9684
246.wine-bottle	94	1769	0	7	1.0000	0.9307	inf	0.9628
c003	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c032	1	1858	0	11	1.0000	0.0833	inf	0.2878
c045	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c047	1	1858	0	11	1.0000	0.0833	inf	0.2878
c049	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c066	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c089	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c093	9	1654	204	3	0.0423	0.7500	1.902106	0.1609
c118	12	1858	0	0	1.0000	1.0000	inf	1.0000
c129	0	1858	0	12	1.0000	0.0000	NaN	0.0000
c159	4	1858	0	8	1.0000	0.3333	inf	0.5761
c160	3	1858	0	9	1.0000	0.2500	inf	0.4988
c163	2	1858	0	10	1.0000	0.1667	inf	0.4072
c178	3	1858	0	9	1.0000	0.2500	inf	0.4988
c184	1	1858	0	11	1.0000	0.0833	inf	0.2878
c191	8	1858	0	4	1.0000	0.6667	inf	0.8156

Table 2.7: Example signal detection calculation table for the GANsemble argmax classifier. *Note!: The d-prime is misleadingly marked as infinite here even though false classifications were made. This is possible when the false alarm rate (FP/FP+TN) is 0, as when the number of false positives is 0. Stanislaw & Todorov 1999 [151] survey various proposed corrections for this case including adding "smoothing" to the counts, but this has not been done here.*

NN GANsemble LDA								
Class	TP	TN	FP	FN	Precision	Recall	d’	MCC
001.ak47	97	1772	0	1	1.0000	0.9898	inf	0.9946
003.backpack	151	1719	0	0	1.0000	1.0000	inf	1.0000
014.blimp	83	1784	0	3	1.0000	0.9651	inf	0.9816
016.boom-box	91	1779	0	0	1.0000	1.0000	inf	1.0000
021.breadmaker	142	1728	0	0	1.0000	1.0000	inf	1.0000
027.calculator	100	1770	0	0	1.0000	1.0000	inf	1.0000
070.fire-extinguisher	84	1786	0	0	1.0000	1.0000	inf	1.0000
157.pci-card	105	1765	0	0	1.0000	1.0000	inf	1.0000
172.revolver-101	98	1771	0	1	1.0000	0.9899	inf	0.9947
183.sextant	97	1770	0	3	1.0000	0.9700	inf	0.9841
208.swiss-army-knife	109	1761	0	0	1.0000	1.0000	inf	1.0000
219.theodolite	84	1786	0	0	1.0000	1.0000	inf	1.0000
227.treadmill	147	1723	0	0	1.0000	1.0000	inf	1.0000
238.video-projector	97	1773	0	0	1.0000	1.0000	inf	1.0000
239.washing-machine	84	1786	0	0	1.0000	1.0000	inf	1.0000
246.wine-bottle	101	1769	0	0	1.0000	1.0000	inf	1.0000
c003	12	1858	0	0	1.0000	1.0000	inf	1.0000
c032	12	1858	0	0	1.0000	1.0000	inf	1.0000
c045	12	1858	0	0	1.0000	1.0000	inf	1.0000
c047	12	1858	0	0	1.0000	1.0000	inf	1.0000
c049	12	1858	0	0	1.0000	1.0000	inf	1.0000
c066	12	1858	0	0	1.0000	1.0000	inf	1.0000
c089	12	1858	0	0	1.0000	1.0000	inf	1.0000
c093	10	1850	8	2	0.5556	0.8333	3.594529	0.6780
c118	12	1858	0	0	1.0000	1.0000	inf	1.0000
c129	9	1852	6	3	0.6000	0.7500	3.398034	0.6685
c159	12	1858	0	0	1.0000	1.0000	inf	1.0000
c160	11	1858	0	1	1.0000	0.9167	inf	0.9572
c163	12	1858	0	0	1.0000	1.0000	inf	1.0000
c178	12	1858	0	0	1.0000	1.0000	inf	1.0000
c184	12	1858	0	0	1.0000	1.0000	inf	1.0000
c191	12	1858	0	0	1.0000	1.0000	inf	1.0000

Table 2.8: Example signal detection calculation table for the GANsemble LDA classifier. Since there are very few cases unclaimed by classes when the performance is at ceiling, synthesis failure classes (c093 and c129) are mostly trading off with each other in making their mistakes.

Figure 2.71 and Figure 2.72 show the argmax and LDA classifier accuracy for the NONGAN discriminators, with and without baseline subtraction using the values from the GAN. In both cases, accuracy is shown to be perfect. Given the lack of cross-validation over GAN training, this result does not advocate for this network to be generally used for recognition because of some nebulous special properties or true perfection, but the NONGANsemble is certainly more adept than the GANsemble at classification. However, this difference is mitigated with proper use of baseline subtraction and an explicit second-stage classifier of competence (here, LDA) to a large degree. It cannot therefore be said that GAN-origin-discriminators lose the ability to be useful conventional recognizers when they are combined in ensemble. More care is needed to use them, however.

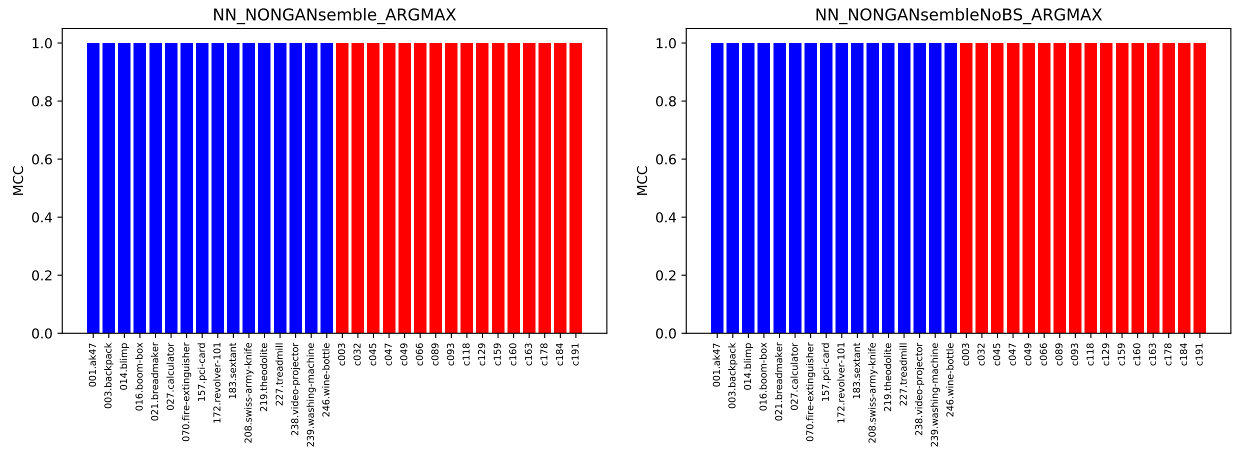


Figure 2.71: Per-class Matthews Correlation Coefficients of simple argmax classifier using NONGANsemble label emissions, with and without baseline subtraction

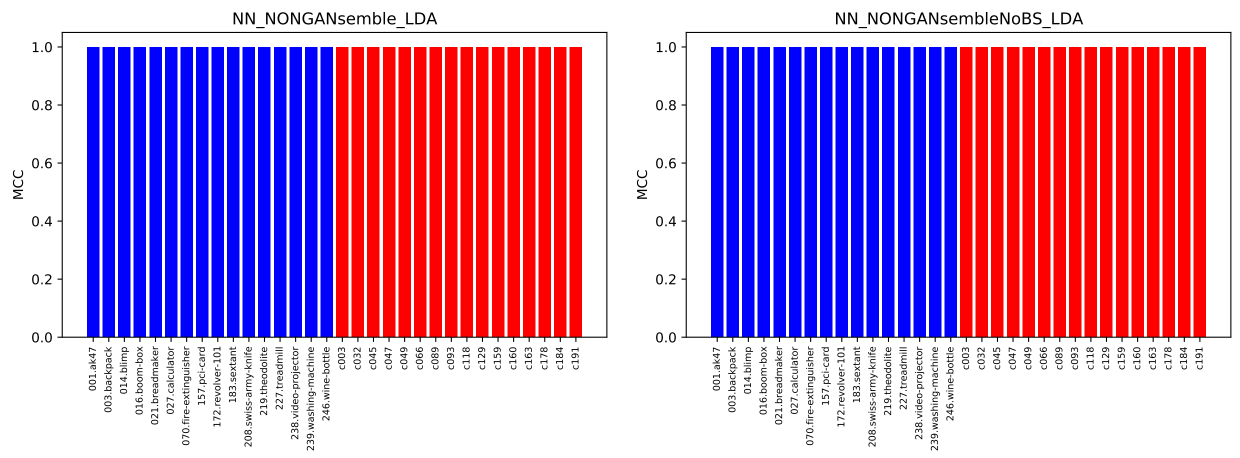


Figure 2.72: Per-class Matthews Correlation Coefficients of LDA classifier using NONGANsemble label emissions, with and without baseline subtraction

In a manner reminiscent of Table 1.4 for the primitive recognizers of Chapter 1, Table 2.9 summarizes the accuracies, averaged per-class MCCs, and resulting Wilcoxon or Mann-Whitney U statistics comparing Thing vs. Stuff recognition performance for each of the classifiers. The only statistically significant change between Things and Stuff is found in the argmax classifier for the GANsemble, whose deficiencies are not generally related to the complexity of recognizing Things and Stuff.

Therefore, while ensembles of deep convolutional networks may be too powerful recognizers to uncover Things and Stuff recognition disparities visible through mere *performance*, in the course of developing them we can see many evidences of the Things vs. Stuff distinction, including those in synthesis, similarity, and quality assessment operations.

Classifier	ACC	ACC(things)	ACC(stuff)	Avg. MCC	MCC(things)	MCC(stuff)	U(MCC)	p-val
NN_GANsemble_LDA	99.25	99.52	96.88	0.977	0.997	0.956	-0.075	0.940
NN_NONGANsemble_LDA	100.00	100.00	100.00	1.000	1.000	1.000	0.000	1.000
NN_GANsembleNoBS_LDA	99.25	99.52	96.88	0.977	0.997	0.956	-0.075	0.940
NN_NONGANsembleNoBS_LDA	100.00	100.00	100.00	1.000	1.000	1.000	0.000	1.000
NN_GANsemble_ARGMAX	88.93	96.48	22.92	0.640	0.978	0.301	4.315	0.000
NN_NONGANsemble_ARGMAX	100.00	100.00	100.00	1.000	1.000	1.000	0.000	1.000
NN_GANsembleNoBS_ARGMAX	12.14	13.53	0.00	0.055	0.111	0.000	1.206	0.228
NN_NONGANsembleNoBS_ARGMAX	100.00	100.00	100.00	1.000	1.000	1.000	0.000	1.000

Table 2.9: Summary of GANsemble and NONGANsemble LDA and naive argmax recognition performance and Things vs. Stuff MCC difference

Chapter 3

Things vs. Stuff in the Metaspace of Recognizers

The investigation for simple filterbank based recognizers argued that it is not only the dimensionality of our features we should consider but also the relative informativeness and the efficiency per bit incurred of each feature itself. In a similar way we may wish to consider not only the magnitude of the performance of each classifier, network, or metric but also their relative *behavior.* To look at the similarity of behavior in a perceptually manifest way, it would be ideal to see how feature choices, joint network activities, and choice of similarity measurement comparing models situate classes of images and networks in a shared space. It does not seem enough to place recognizers by a similarity that only includes their performance standing, as if to say that means are stratified from signchains and signchains of signchains, and that perhaps all primitive recognizers are in general stratified by their performance on challenging Things patches from the deep convolutional recognizers.

To take an example, all Things and Stuff classes in our dataset could be situated in a space induced by features employed by a recognizer. In this space, perhaps the Things and Stuff classes are separable, and perhaps similarity between similar classes is preserved. The space induced by the recognizer relying on a feature vector could for lack of better nomenclature be called the *feature space* in which classes are situated. In this chapter, we further consider the possibility that it is interesting to consider the recognizers themselves as situated in a space, which with a similar lack of creativity could be termed the *recognizer space*, as it is a space in which the recognizers owning the features placing the classes are situated. Further, for the deep convolutional recognizers we have developed in Chapter 2, the recognizers may exist in several spaces only as we have studied them already: a) in terms of the feature vectors the ensembles produce for a later stage classifier such as LDA, b) in terms of the raw affinity matrices they evince between classes prior to combination in a flattened-out decision on specific patches. We can then imagine these separate senses as points in a space of spaces in which the recognizers exist: the beginnings of a *metaspace* of recognizers.

The vectors of features (e.g. means and signchains of histograms, or label emissions of GANsembles) used by the recognizers we have considered are at times quite high-dimensional, having been formed by concatenating either scalars or vectors of numbers for each member filter or network – easily possessing dozens or hundreds of components. The vectors which embody the rows of the affinity matrices we have computed for GANs and NONGAN cross-class firing and the vectors which embody the rows of the dissimilarity or similarity matrices we compute layerwise to compare the ensembles' constituent networks in various senses are lower-dimensional but still higher-dimensional than we could ever practically visualize mentally or with pages of draftsman's plot arrays of scatter plots.

For ease of interpretation, we consider positioning of classes and networks in a plane based on their full feature vectors. This is the province of dimensionality reduction methods, which provide a low-dimensional embedding (LDE) of the input data. Much deep learning research, such as that which demonstrates that word embeddings like those induced by word2vec permit analogy completion of man::woman::king::??? [118] by simple linear arithmetic, concerns itself with the low dimensional embeddings *in networks.*

In the current situation, we concern ourselves with the low-dimensional embeddings that exist in the space created *between networks, or recognizers, or choice of features.*

While many techniques are available for extremely high-dimensional data, such as the more recently introduced t-distributed stochastic neighbor embedding (t-SNE, one of a family of stochastic neighbor embeddings introduced by [110] Hinton and collaborators) method popular for visualizing where input or synthesized images lie on lower-dimensional manifolds which exist in spaces of activations or noise-inputs in deep networks, the data we possess is not extremely high-dimensional, in the hundreds of thousands to hundreds of millions of features.

A more classical technique suitable to feature vectors of intermediate dimension was chosen for this investigation from the multidimensional scaling family of methods, which have in common that they situate data points in the low-dimensional embedding according to the minimization of a loss function called stress which measures how well the euclidean distance between data points' feature vectors is respected.

To preview this chapter:

- Metric multidimensional scaling (metric MDS using SMACOF) is advocated as the preferred method of taking Chapter 1 and Chapter 2's data and creating low-dimensional embeddings.
- Low-dimensional embeddings (LDEs) are created through multidimensional scaling of three embedding groups (not mathematical groups) of features: recognizer features, deep convolutional recognizer labeling affinity matrices, and deep convolutional parameter distance matrices.
- The idea of an encapsulation taxonomy declaring linear separability of Things and Stuff

point sets, or full, inviolate containment of one point set within another is proposed, and an algorithm based on the computation of convex hulls and ray casting intersection tests is provided. All embeddings are processed by the algorithm and the lack of members across the 3 groups in one specific encapsulation class is discussed.

- Scaled Orthogonal Procrustes superimposition is reviewed and used to align embeddings within embedding groups to least squares distance to produce an embedding-pairwise distance matrix for each embedding group.
- "Embeddings of embeddings" (which may be termed metaembeddings or acroembeddings) are introduced as the MDS induced embeddings of Procrustes-aligned embeddings. Each first stage embedding in an embedding group is a point in the space induced for the embedding groups.
- The Euclidean "meta distance" between conditional metaembeddings (separate metaembeddings created originally from only either Things or Stuff) is assessed for each embedding in each metaembedding space by using a second-round of Procrustes alignment following the second round of MDS.
- A distance matrix is created between classifier per-class rankings by Matthews Correlation Coefficient using Kendall's Tau distance and a higher-level embedding analogous to the metaembeddings is created for the rankings.
- The rankings of recognition difficulty are aggregated into a composite ranking using the Borda count method of voting and the Borda count derived composite rank is correlated to Fréchet Inception Distance and Inception Score to assess how per-class recognition disfluency across recognizers is related to synthesis disfluency using these heuristic methods, and if there is a difference between Things and Stuff.

3.1 Low-Dimensional Embeddings via Multidimensional Scaling

The simplest method in the family, sometimes known as classical MDS, involves beginning with the very well-known operation of principal components analysis, which situates a higher-dimensional dataset in a low-dimensional linear subspace by discovering the most significant principal components of the data, usually by mean-subtracting the data and computing the singular value decomposition to discover the eigenvectors and eigenvalues associated with the centered data matrix. As a result of PCA, you obtain PCA loadings (describing the linear combination of original variable values needed to synthesize a principal component coordinate) and PCA scores (representing the transformed data in the space of principal component coordinates), and thus retain the appealing advantage of easily being able to project new instances (here, corresponding classes or networks) into the space. PCA based on a specially transformed dissimilarity matrix can be used to produce the coordinates of classical MDS.

PCA is well-known to produce flat, subjectively uninformative embeddings of contrived test functions, such as the classic "swiss roll" (see e.g. in [71]) or S-shaped point clouds. In practice, it is easily discovered that it is hard to tell ahead of time without clearly predetermined criteria which choice of manifold learning algorithm is the "best" because they can highlight wildly different structure in the data – methods based on spectral graph theory may produce quite different planar embeddings of 3D point clouds than locally-varying influence of PCA (say streaky or strongly curvilinear embeddings rather than ones which allow adjacency information to uniformly fill a portion of the plane), but it seems in keeping with parsimony to prefer the next simplest method. For this study, we surrender that conspicuously mentioned advantage of being based on PCA somewhat unwillingly though because PCA is based on a linear process and we are interested utmost in preserving the natural pairwise distances of our items, than in highlighting the two directions of maximum variation – the prospect of reprojectable embeddings that can imagine new classes or networks in the space can be left as an area for future study if we are optimizing first for the quality of the embeddings. The next least sophisticated technique, metric MDS, allows a small measure more flexibility in that the stress function can correspond to a function of the pairwise distance in items (non-metric MDS is further relaxed to allow non-absolute knowledge of the value of the distance, as by reduction to ordinal information about the standing of items, but also requires an isotonic regression step to determine a monotonic function to transform the dissimilarity matrix each iteration).

The stock MDS implementation (`sklearn.manifold.MDS`,[130]) used in the following investigation was based on the SMACOF algorithm (Scaling by MAjorizing a COmplicated Function, [103]), which starts with an initial random configuration of points in the space, computes the stress from the dissimilarity matrix and the current configuration of the points and then updates the points using a process based on a specific-instantiation of the Guttman transform (a replacement of the update of the step-sized dissimilarity matrix X via the gradient of the poorness-of-fit with a square symmetric matrix-valued function C of X) that is motivated by improving on a gradient descent update of the matrix with an uninformed step size [37]. It is further parameterized by the maximum number of iterative majorization updates in the SMACOF algorithm (e.g. it may terminate early at some stress ϵ value, signifying convergence) and also the number of runs of the algorithm in total to produce the lowest stress. To produce the "best", or reasonably stable quality, embeddings with the idea that early error accumulated will propagate through later planned operations, all MDS embeddings obtained below have been run with a maximum of 1000 updates and 100 runs, using an ϵ of 0.003.

The dissimilarity matrix X has been computed from the full data or design matrix X_0 comprising the stacked original feature vectors by taking the pairwise Euclidean distance.

Clearly, different choices of distance notion here could be explored even in this facet, but the Euclidean distance between data points is the overwhelming convention.

3.1.1 Recognizer Feature LDEs

The recognizer feature LDEs depicted below show where Things and Stuff classes fall given the choice of recognizer (where recognizers, of course, have only been distinguished on which features they use). Following the investigation in Chapter 1, there are 5 primitive filterbank recognizers we have defined: the ones based on means, signchains, signchains-of-signchains, histogram bin heights, and medians. Additionally, following the completion of investigation in Chapter 2, there are two deep convolutional recognizers, the GANsemble and the NON-GANsemble that arise from the deep convolutional discriminator networks we have trained.

To produce the low-dimensional embeddings, we simply take the feature vectors which were computed to assess the classification performance in Chapter 1 and Chapter 2 and stack them for each first occurring patch for a class in the dataset for each feature vector type and subject the resulting data matrix (e.g. $X_{0,MEANS}$) to the metric MDS procedure previously described. Alternatively, the feature vectors could have been averaged prior to MDS, but this would produce feature vectors that would not be reflective of the original feature vectors. Or, the MDS configuration points within a class could have been averaged – this would have been more legitimate than the last idea but for simplicity the unique instance approach is used. The chosen rule also expresses a bias against finding separation results because individual instances could be uninformative of the class by chance. Consequently, when Thing vs. Stuff dichotomies appear in the MDS data below for recognizer features, we can be if anything more confident that the feature choice leads to a dichotomy – and where mixed results appear, the embedding might not actually be as degenerate.

Primitive

Immediately, in Figures 3.1 through 3.5 it is apparent that Things and Stuff usually exhibit, on average, but not always securely, some sort of separation that survives the low-dimensional projection of their primitive recognizer feature vectors. Systematically this separation is not linear, but often involves the cloud or point set of Stuff being contained inside the point set of Things.

The least degree of this kind of embedded separation is observed for the signchains-of-signchains features, where the tendency is almost reversed. A similar pattern is observed for the signchain. The convergence or collapse of Stuff classes to a point is most clearly observed for the means and the medians. Of course, this is mildly curious because the means are the lowest performing features and the medians along with the full histogram bin heights are among the best. So the behavioral similarity of features is not purely captured by performance.

Of course, classification performance on the Thing vs. Stuff distinction and not on ordinary, per-class recognition is most tightly related to where the Things and Stuff exist in feature space and its low-dimensional projections. This suggests, though, that there is additional information of some kind to be gained from identifying a patch as Thing or Stuff and using nonlinear classification methods even as linear methods may prove reasonably competent at the actual classification problem. It clearly suggests that omnibus Thing vs. Stuff classifiers miss out on good information using purely linear methods, and that they should have nonlinear elements that can use, at the least, quadratic or Gaussian decision boundaries.

The next most apparent pattern is that the kernel size is almost irrelevant to determining the type of separation. The kernel size (as seen at the end of Chapter 1 and Chapter 2) is indeed quite related to the quality of classification, even though it is secondary to feature selection (the means with a linear classifier are not saved by 11x11 random filters). Reducing per-class confusion then, if it is assessed by this method, involves properly separating *instances* in the low-dimensional space. If anything, where feature vectors induce a space that permits classification by attracting all members of a class to primarily a proprietary area of the space, improving recognizer success hinges on finding a way for each cloud to be discriminated from best from its nearest neighbors without losing the stability of the long-range organization of proprietary spaces to feature adjustments or transformations. Feature selection schemes, conventionally, add or mutate features in a feature vector based on improvements to classification performance, without looking at optimizing for salutary adjustments to be made in view of the underlying projection geometry; the type and severity of separation contains more information than does pure performance, even though not all of that information may be directly germane to classification performance if you are actually interested in classifying at a differently specified level of hierarchy (things vs. stuff as opposed to class vs. class). In summary, the Thing vs. Stuff distinction is strong in the face of kernel size changes, and the classification performance at a finer grain is brittle.

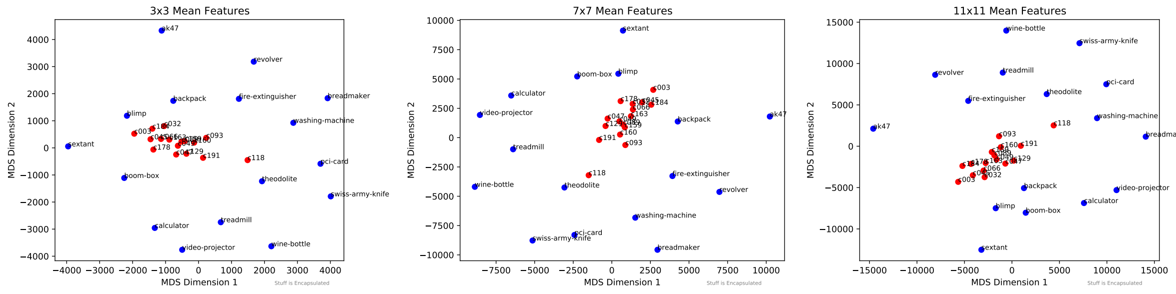


Figure 3.1: MDS embedding based on filterbank means, kernel size increasing from 3 to 11

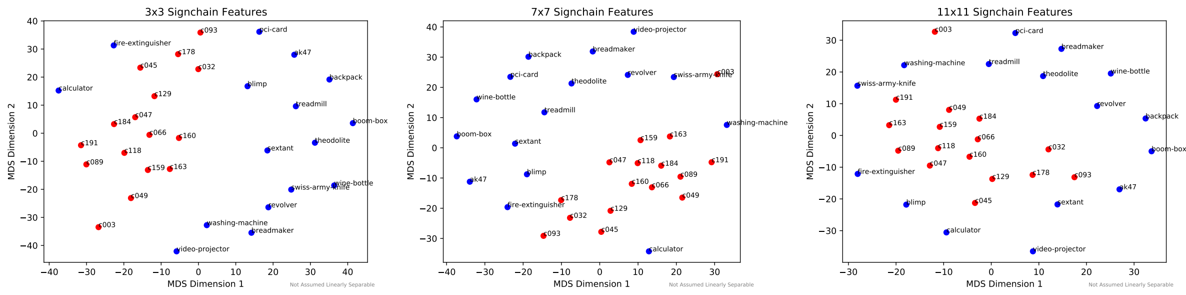


Figure 3.2: MDS embedding based on filterbank signchains, kernel size increasing from 3 to 11

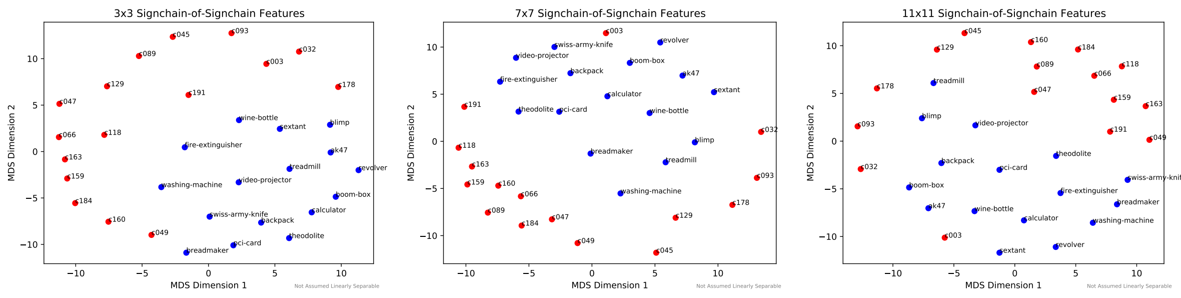


Figure 3.3: MDS embedding based on filterbank signchains-of-signchains, kernel size increasing from 3 to 11

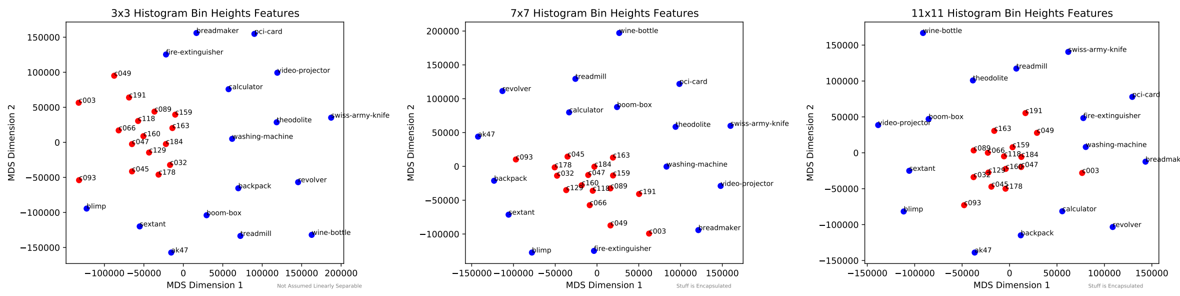


Figure 3.4: MDS embedding based on filterbank histogram bin heights, kernel size increasing from 3 to 11

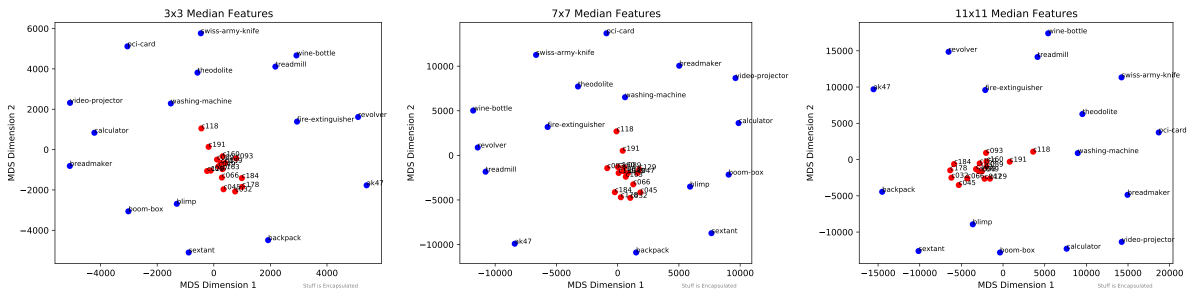


Figure 3.5: MDS embedding based on filterbank medians, kernel size increasing from 3 to 11

Deep Convolutional

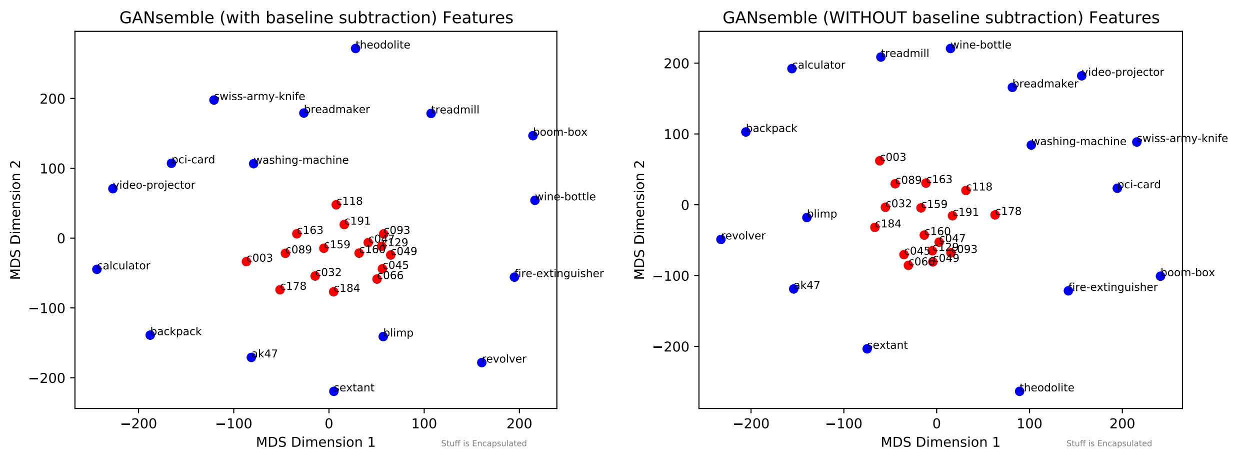


Figure 3.6: MDS embedding based on GANsemble member labels, with baseline subtraction and without

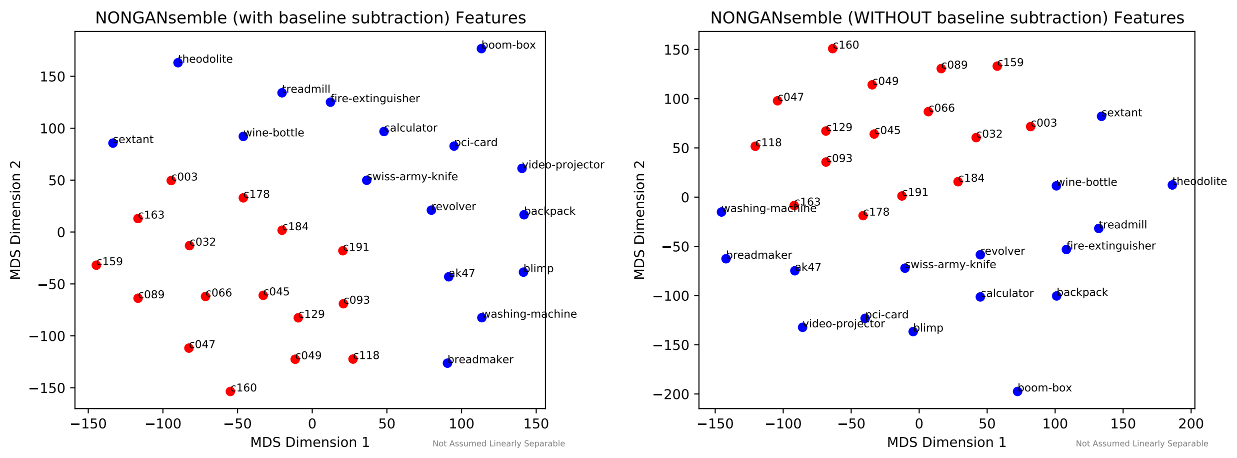


Figure 3.7: MDS embedding based on NONGANsemble member labels, with baseline subtraction and without

In Figure 3.6 and Figure 3.7, the low-dimensional embeddings of the recognizer features for the deep convolutional recognizers (the GANs and NONGANs) are shown.

The GANsemble activations for patches produce embedding geometry more reminiscent of that for the means and medians, which is a focal concentration of the Stuff classes. The NONGANsemble, meanwhile, produces an embedding that better resembles the signchains classes (but is actually expected to be more linearly separable).

This difference underscores the fact that GAN style loss and ordinary loss discriminators are indeed different (as the lens of Things vs. Stuff reveals for us), and their feature vectors created by committee members labeling patches are distinct, to the point of changing the optimal choice of consensus function, at least after dimensionality reduction is applied. In the case of the GAN activations after MDS, a Gaussian Discriminant Analysis that permits dynamic distance queries is likely preferable to a Linear Discriminant Analysis with its additional projection and its extra effort settling on a hard, constant decision boundary.

An advanced behavioral query that these types of embeddings could allow us to make in the future might be: "do direct objectness classifiers, such as those introduced using the NONGAN dichotomously on all class data in Chapter 2 occupy a position in feature space that is close to the metacategory-conditional centroid for Things, the global centroid, or neither?"

3.1.2 Ensemble Labeling Affinity LDEs (Model Output)

The labeling affinity LDEs depicted below show where things and stuff classes fall based on the internetwork firing affinity matrices. As established in Chapter 2, there are affinity matrices defined for the GANsemble processing the fake images, the GANsemble processing the original/real images, the NONGANsemble processing the fake images, and the NONGANsemble processing the real images. Further, for each of the four preceding cases, there are also transformed affinity images producing distinct embeddings according to whether baseline subtraction, no baseline subtraction, or baseline division was applied.

The low-dimensional embeddings are produced mostly as before, treating the affinity matrices themselves as the data matrices X_0 subjected to metric MDS. There is no notion of having to pick either a unique class champion arbitrarily or averaging since only the one possible affinity matrix was studied and not permutations.

The natural hypothesis is that the GANs and NONGANs are behaving quite differently and so a bifurcation between GANs and NONGANs is expected in subjectively "good" embeddings.

Figures 3.8 through 3.11 show the MDS for the real and fake GAN and NONGAN affinity matrices, with all three of their postprocessing states: no change, baseline-subtraction, and baseline-division. It can be seen clearly across all embeddings that baseline-division harms separability, as previously established multiple times throughout Chapter 2. It either draws the frontier of Things and Stuff really close, leaving far flung outliers, or it places things and stuff on a (admittedly bifurcated) line, as in the case of the GAN's affinity matrix of fakes. Overall, Things and Stuff can be separated in these projection spaces with hard decision boundaries of either a line, two lines, or a parabola, and sometimes with commodious margins. A notable exception is the GANsemble's affinity matrix for reals, where the Stuff is again included within the cloud of Things and so a GDA type classification (i.e. picking the nearest Gaussian, corrected for spread) would be best.

Finally, in the overwhelming majority of these embeddings it is important to note that the boom-box and its duplicate network are colocated (either at imperceptibly different positions, or at least with no closer intervening network) and so are the theodolite and its duplicate network. An unsurprising exception is the GANsemble fake with baseline division embedding, where this is slightly violated. This is a good, further indication that the pairwise firing-based affinity matrix retains decent similarity information since it can at least recognize same-data scenarios even through the process of an embedding. In many of the embeddings, the sextant is either close to a theodolite or a swiss army knife, AK-47, or revolver, showing evidence of a slightly less trivial level of similarity-preserving.

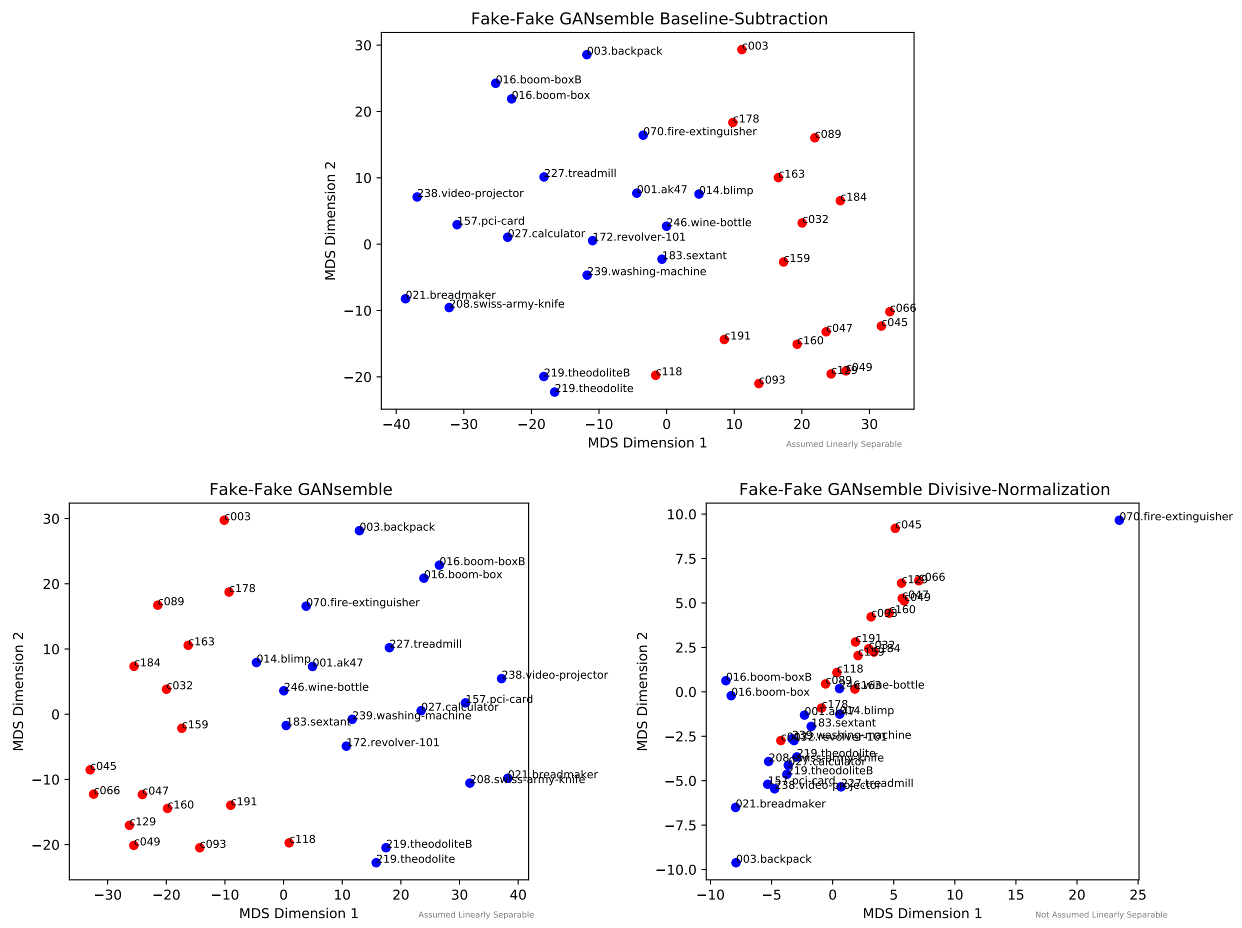


Figure 3.8: MDS embedding based on GAN affinity matrix of fake patches, with subtractive normalization of output, no normalization, and divisive normalization

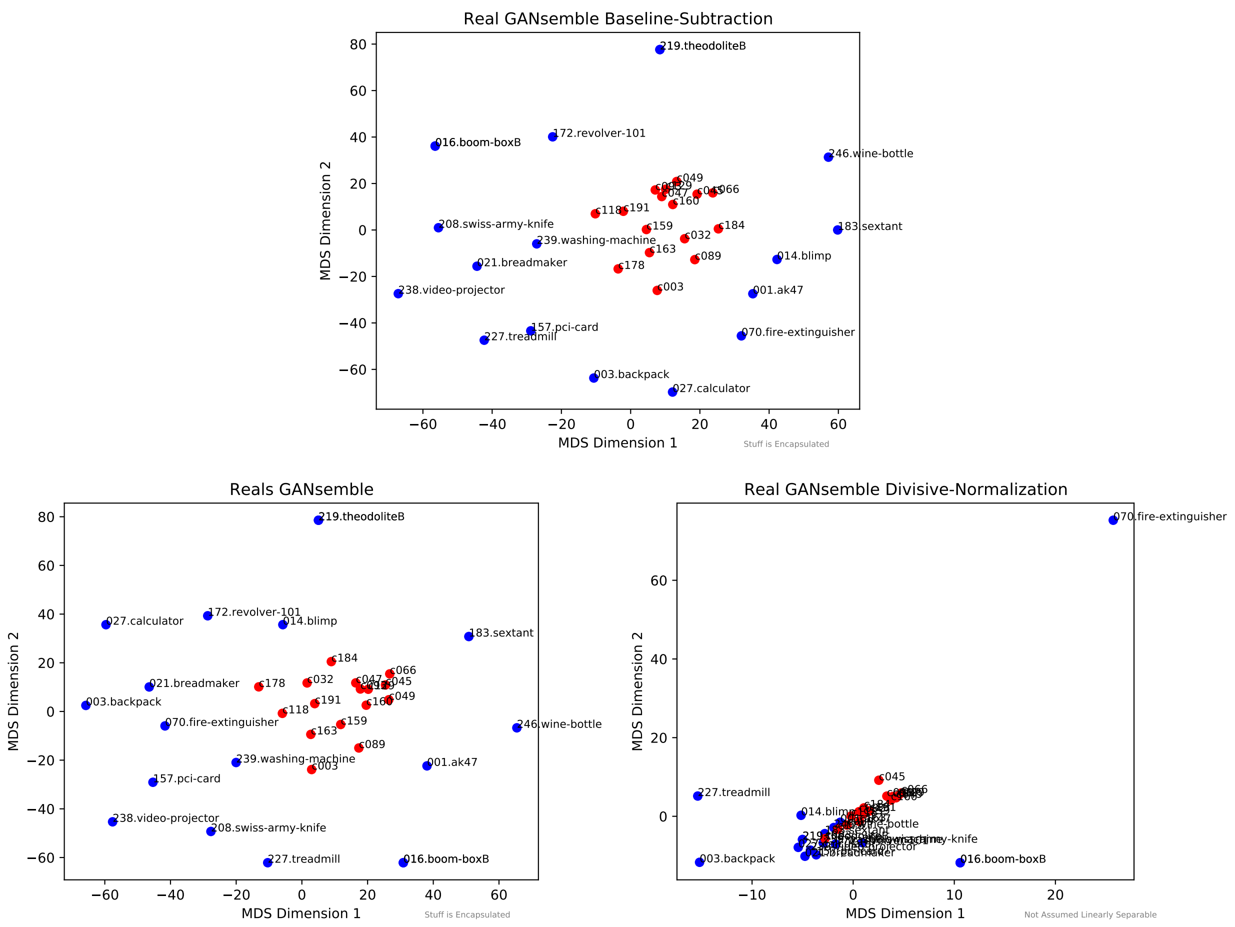


Figure 3.9: MDS embedding based on GAN affinity matrix of real patches, with subtractive normalization of output, no normalization, and divisive normalization

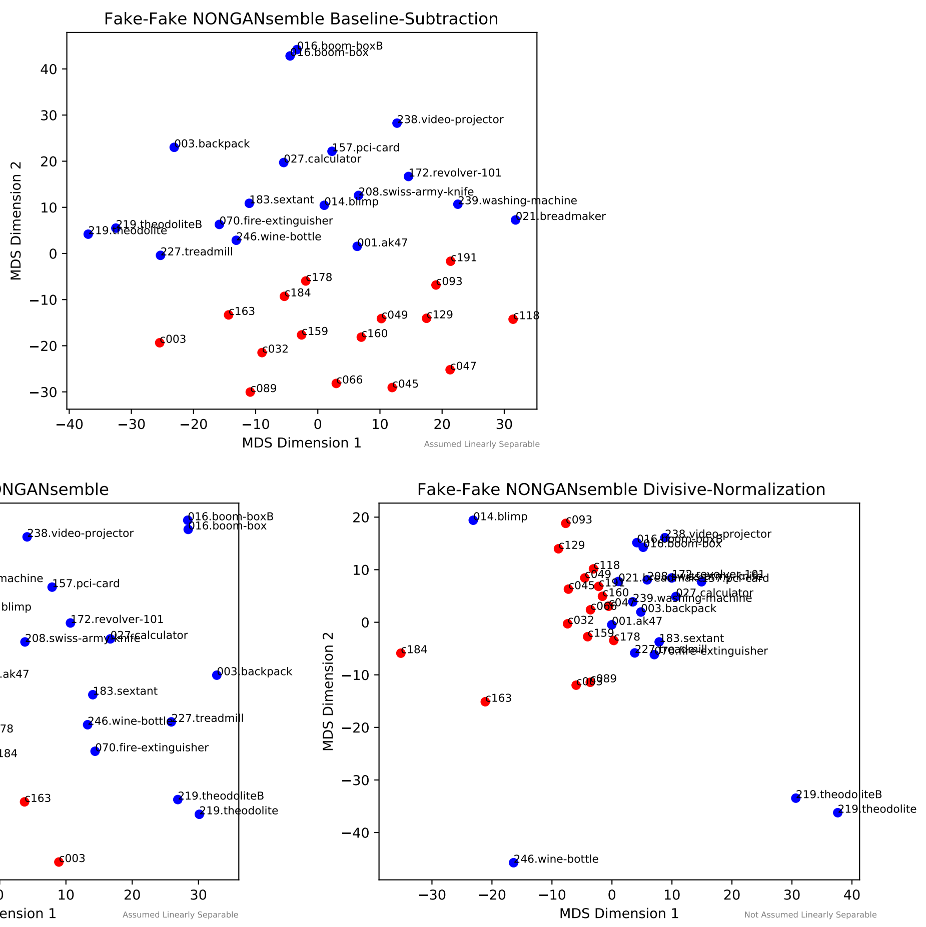


Figure 3.10: MDS embedding based on NONGAN affinity matrix of fake patches, with subtractive normalization of output, no normalization, and divisive normalization

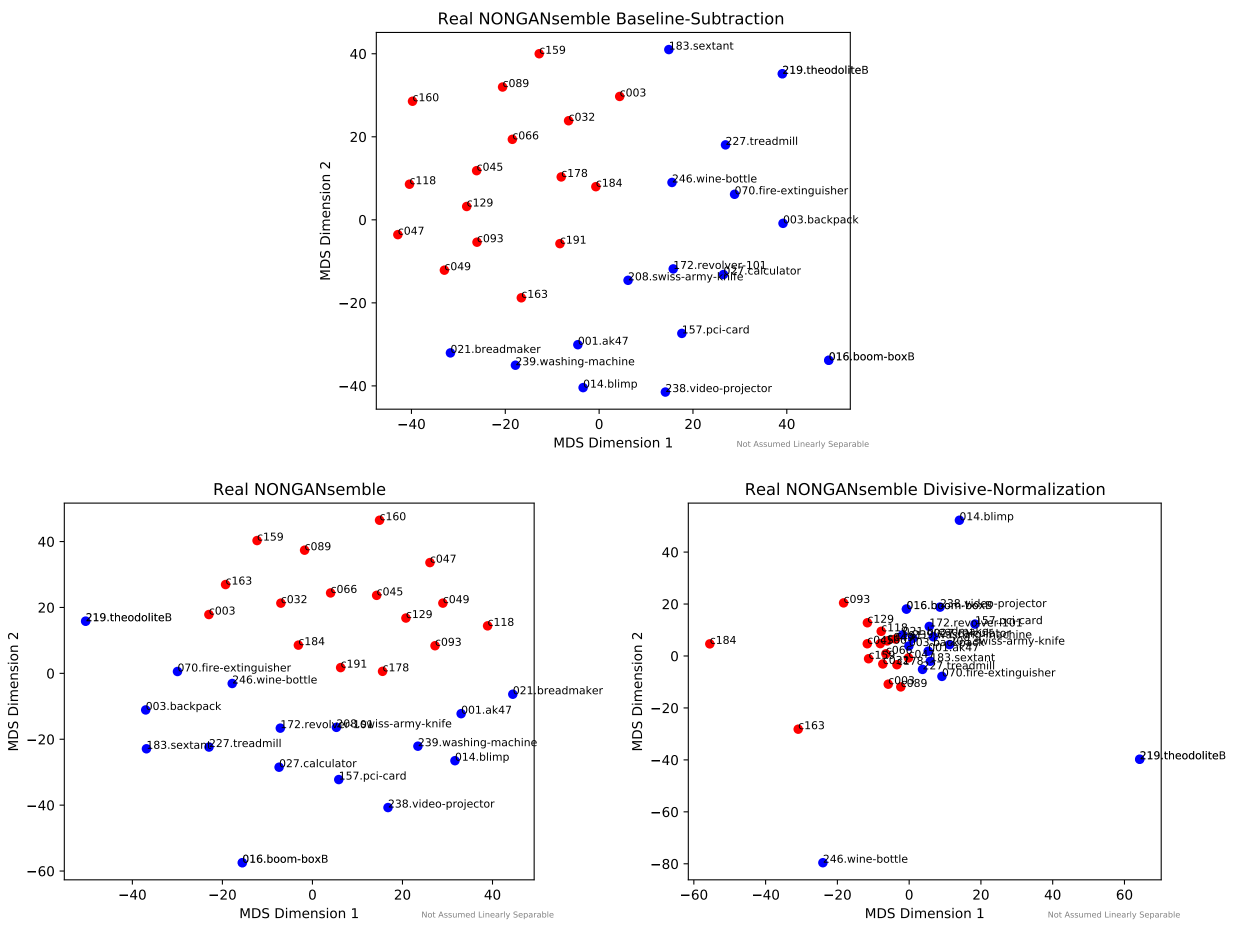


Figure 3.11: MDS embedding based on NONGAN affinity matrix of real patches, with subtractive normalization of output, no normalization, and divisive normalization

One addition is made to this family of embeddings even though it is not an affinity matrix computed from the firing of networks in the GANsemble or NONGANsemble. It is the affinity matrix that is created indirectly by the interior activations of the Inceptionv3 network when subjected to fake patches from the networks. It depicts quite vividly that the internetwork (between fakes) FID affinity spaces out Stuff far more than Things. This is distinct from the FID pattern used in its proper context, and observed in Chapter 2, between produced reals and fakes, where there were higher Thing Real-Fake FIDs – cumulatively this suggests that Fréchet Inception Distance responds very interestingly interclass with respect to the variation *within* a class, which is obviously lower for Stuff since it is based on the distance of multivariate Gaussians (which depends on mean but *also* standard deviation).

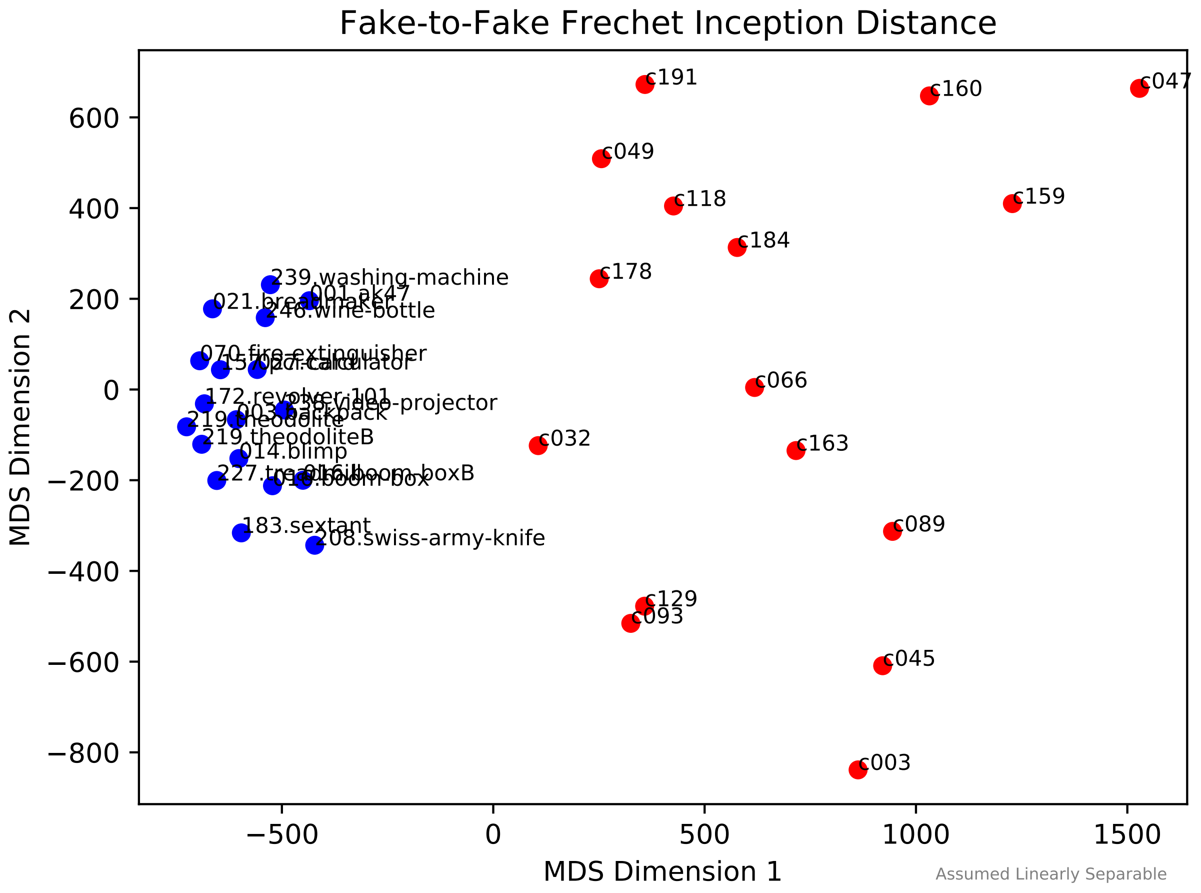


Figure 3.12: MDS embedding based on GAN internetwork fake patches Fréchet Inception Distance affinity matrix. The Frechet Inception Distance, for fake patches, induces much more variation comparing Stuff classes, which is curious given the homogeneity of stuff. Possibly this pattern results from the fact that the FID is founded on a multivariate Gaussian distance, and the difference also depends on the bandwidth of the Gaussian, which may be appreciably lower for Stuff.

3.1.3 Neural Network Interclass Weight Disparity LDEs (Model Parameters)

Analogously to the pairwise affinity matrix, the model parameter distance matrix is turned into an embedding and shown for each of the measurements defined in Section 2.2.3 in Figure 3.13 through 3.18. The embeddings are made separately for the GAN and NONGAN affinity matrix, so only the discriminator layers are considered, as the NONGAN obviously possesses no generator. The pooled layerwise matrix is composed by taking the vectors in each logical layer's affinity matrix and concatenating them so that the vector for each network is representative of all of the layers in the discriminator, and this composes the design matrix X_0 that is processed by metric MDS.

The anomalous synthesis failure classes c093 and c129 return to prominence in distorting the embeddings, especially those for the GAN's MSE, L1, and L2 Norm, and SSIM. A significant number of the embeddings are unusable for separating Things and Stuff, notably the Minkowski norms and the Jensen-Shannon distance. Approximate linear separability is evident for SSIM and for MSE, and also for the Signchain Distance (but in this case, only for the GAN).

The previous image-based analysis revealed that only certain layers expressed a Things vs. Stuff dichotomy. In this form of visualization, it can be seen that the MSE and the SSIM are the best measures (of those assessed) for disentangling Thing and Stuff discriminator networks based on all of their relevant weights. Clearly, the SSIM is a relatively-sophisticated way of looking at distance, at least on the coarse grain of trying to guess from the weights alone whether the network was trained with Things or homogeneous Stuff. But the notion of natural distance, taken between models in high-dimensional weight space, is even less meaningful, at least for discerning a difference between Things and Stuff. This could be an epiphenomenon of the sensitivity of the MSE to outliers, but is likely a problem of the MSE being a much more direct way of accumulating total model disparity on all dimensions. While the Jensen-Shannon distance shows some metacategory-centric separation, it isn't secure, and therefore it is particularly noteworthy that the signchain distance (which appeared a weaker form of Jensen-Shannon distance in the parameter matrices) separates so well, at least for GANs. It does not for the NONGANs. As previously discussed, the distributional divergences are already somewhat problematic because they are based on extemporaneous histograms, which may have different supports.

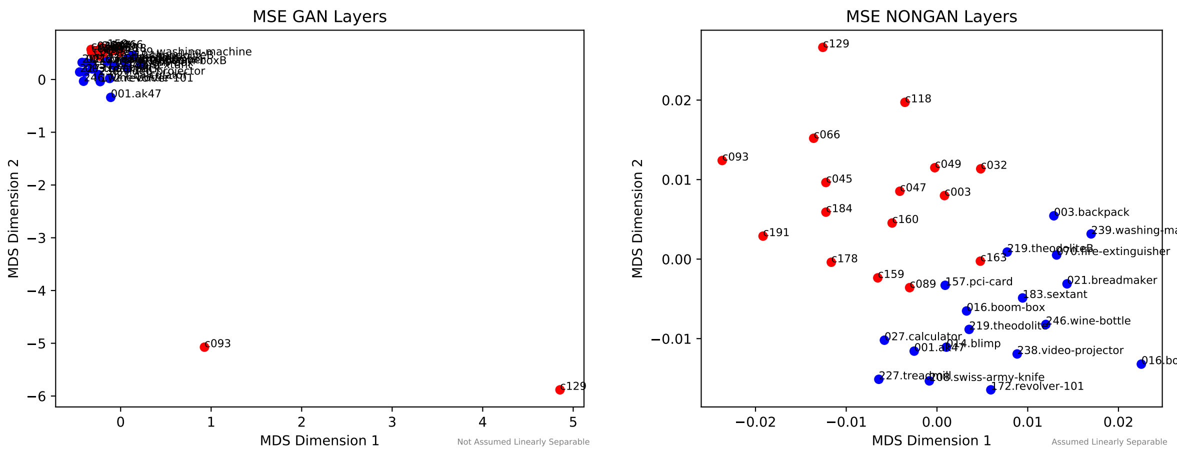


Figure 3.13: MDS embedding based on layerwise MSE between networks a) in the GANsemble b) in the NONGANsemble

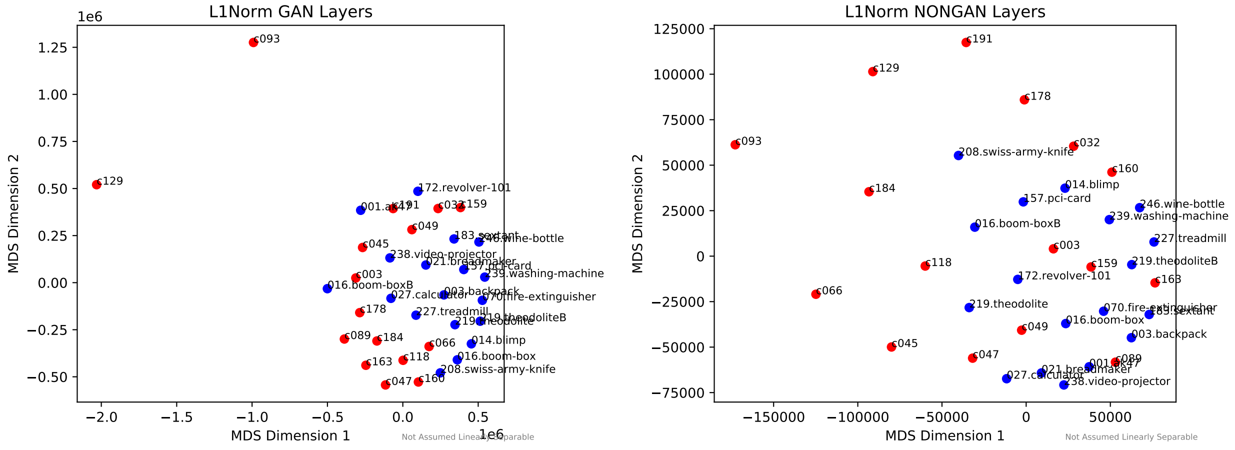


Figure 3.14: MDS embedding based on layerwise L1 Norm between networks a) in the GANsemble b) in the NONGANsemble

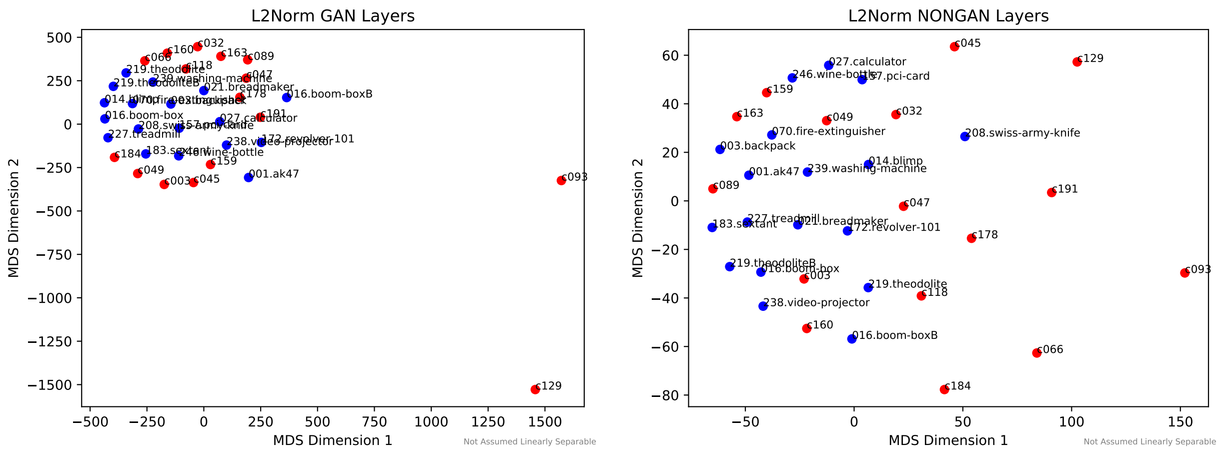


Figure 3.15: MDS embedding based on layerwise L2 Norm between networks a) in the GANsemble b) in the NONGANsemble

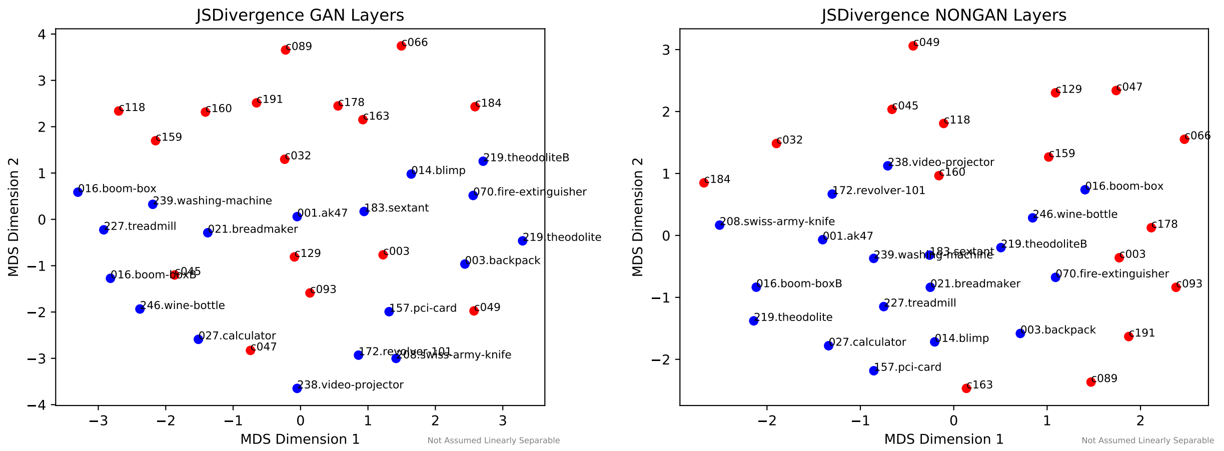


Figure 3.16: MDS embedding based on layerwise Jensen-Shannon Divergence between networks a) in the GANsemble b) in the NONGANsemble

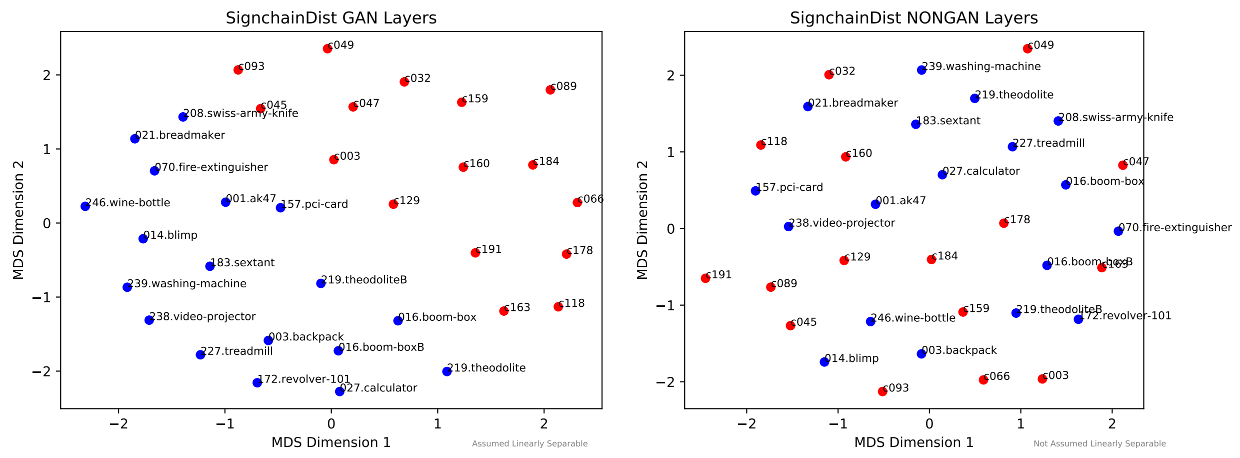


Figure 3.17: MDS embedding based on layerwise signchain distances between networks a) in the GANsemble b) in the NONGANsemble

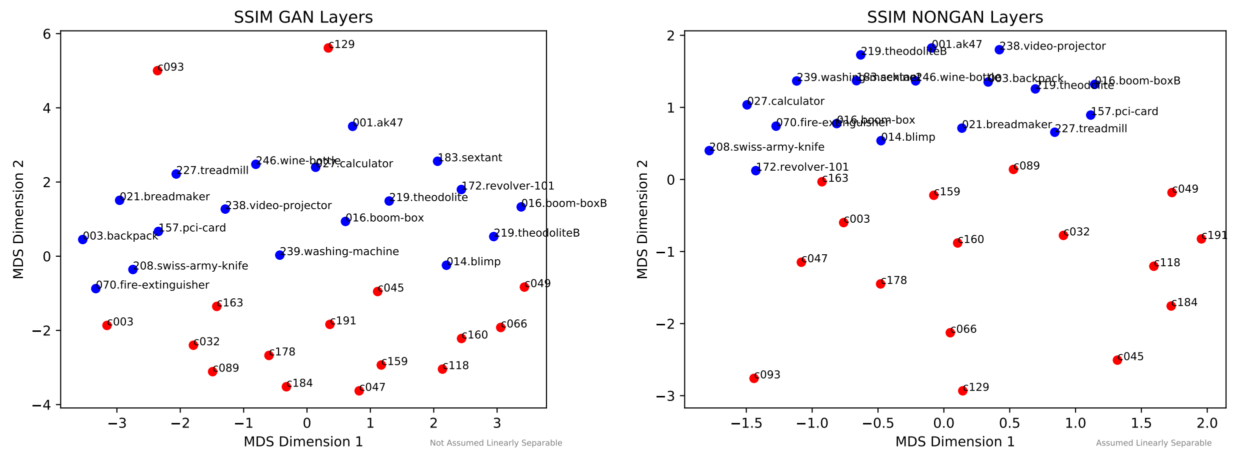


Figure 3.18: MDS embedding based on layerwise SSIMs between networks a) in the GANsemble b) in the NONGANsemble

3.2 Things vs. Stuff Encapsulation Class of LDEs

Just as we demand of recognizers various accountings of their performance, we expect some quantitative analysis of embeddings. Quantifying the empirical separability of the Things and Stuff classes demonstrating the existence of a Things vs. Stuff dichotomy displayed in a candidate embedding is obviously of foremost importance over preservation of known similarities (theodolite-theodolite, boom-box-boom-box) for our investigation of Things vs. Stuff.

There are various choices available when looking at the separation between Things and Stuff point sets. Beginning with the pairwise distance matrix p which stores the Euclidean distance between every possible pair of thing points T and stuff points S, one could report the average Euclidean distance. In addition to the average entry, two extremes can be considered.

On one extreme, focusing on the maximum separation, there is the maximin type directed Hausdorff distance (used for template matching in computer vision and elsewhere, see [64] for fast approximate Hausdorff algorithms) where

$$h(T,S) = \max_{t\in T}(\min_{s\in S}(p(t,s)))$$

describes the maximum distance from an adversarially chosen Things point to the closest Stuff point. To imagine this vividly, this is related to the farthest distance a missile would have to travel from the farthest-flung silo in an aggressor state to the nearest target in enemy territory. The distance is directed because

$h(S,T) = \max_{s\in S}(\min_{t\in T}(p(t,s)))$ the distance from the adversarially chosen far point of the S cloud to the nearest point in T need not be equivalent. Given the maximin spirit of the definition, the intuitive undirected Hausdorff distance is simply

$$\max(h(S,T), h(T,S))$$

On the other extreme, there is the minimin type minimum separating distance between the two clouds, which is

$$m(T,S) = m(S,T) = \min_{t \in T}(\min_{s \in S}(p(t,s))) = \min(p)$$

Again, to imagine this with a military planning example, this is related to the closest points of coastline in an invading and an invaded country. This is somewhat related to the max margin loss in support vector machines where you are trying to position the hyperplane such that it provides a symmetric margin maximizing this kind of distance for the support vectors of each class.

While the maximin and the minimin pairwise differences could be reported easily enough for the LDEs above, neither are extremely directly related to recognition. The embedding with minimum minimin distance could happen, for example, to be very predictably modelable by two very distinct Gaussians or separable with a linear classifier but directly abutting. In another degenerate case, a situation where Things and Stuff clouds were directly superimposed so as to make classification by conventional means very difficult but with a large constant separation between pairs could mislead if one was to erroneously believe that the best embedding was the one with the maximum minimum separation. The maximin Hausdorff distance, while very generally useful in template matching, isn't informative either for this kind of recognition; in particular, it can be driven by an outlier point far on the fringes of the LDE, such as those which might be induced in GANsemble LDEs by synthesis failure classes.

3.2.1 Automatic Estimation of Encapsulation Class Using Convex Hulls and the Point-in-Polygon Query

What we are interested in from the point of view of predicting embedding performance on separating Things and Stuff is estimating linear separability. Of course, linear separability can be inferred with the perverse misuse of a support vector machine's regularization parameter C. The parameter C being very large eliminates the softness of SVM (see [122], 14.5.2.2 for an explanation) which tolerates separability violations if the margin can be increased. If C is set very high and the SVM linearly separates the data (for example, experiences no misclassification on the data it was trained on, reminiscent of our own classification on training data with the GANsembles), for a feature selection and embedding procedure A but not a feature selection and embedding procedure B you might be more confident that A represents the better combined choice of manifold learning procedure and set of features to select for classification for unseen data – confining yourself to linear classifiers.

In addition to separability, we are also interested in the curious question of whether all Stuff might converge to a single point in Thing space (i.e. all Stuff behaves like a single Thing under some transformation) or all Things might converge to a single point in Stuff space (i.e. all Things behave like a single Stuff class) for various embeddings. The intuitive hypothesis the average person might have is that all things are a kind of stuff originating from amorphous material, but we have defined Stuff classes to be only homogeneous objects in terms of image statistics, so the opposite hypothesis seems more likely in terms of the increased expectation of more uniform behavior given more uniform input. Is it possible that all homogeneous textures behave essentially the same under some transformations? Being able to demonstrate this visually from embeddings would shore up confidence in applying separately methods that assign things and stuff categorically different outputs (e.g. Inception Score as shown in Chapter 2) and making proper range corrections for the possibility that a whole space might actually exist entirely inside another, encapsulated.

To taxonomize our embeddings by probable (for our present purposes, not by any means as a guarantee) separability and also encapsulation (to be sure, a strong variety of non-separability in the linear sense), the convex hulls for each of the Thing and Stuff point sets are computed for each embedding. Things and Stuff are assumed here to be likely separable if the convex hulls do not intersect. They are assumed to not be separable if they do. Things are assumed to be fully encapsulated in Stuff if no Stuff points lie within the Things convex hull but all Things points lie within the Stuff convex hull. Stuff is assumed to be fully encapsulated in Things if all Stuff points lie within the Things convex hull but no Things points lie within the Stuff convex hull.

The word “inside” is chosen precisely here, as for simplicity’s sake in calculation we consider the tests only *inside* the polygon corresponding to the hull, not *on* the polygon, and assume that the subcomponents of our algorithm are defined in such a way as to effectively erase that distinction. For points in the set contributing to the hull, if the “convex hull” determination algorithm only included the extremal points (e.g. it first detects and removes points when three adjacent processed points are collinear), then an “in-or-on-polygon technique” would need to do more to fulfill the “on-polygon” subquery than simply detect if the query points match vertex points for the hull, if the ”in-polygon” technique also happened to exclude the points *on* the polygon. It would also need to determine whether a point lies on the line segments which constitute the simplices of the convex hull (e.g. by using the wedge product of vectors from the query point to the simplex edgepoints to determine that the area of the triangle between the three points is zero (establishing collinearity), and by using the dot product or some other means to determine that it is included within the bounds of the line segment).

The point-in-polygon (PIP) subproblem for the convex hull can be solved by a raycasting method called the even-odd rule. A ray R with a random direction originating from the query point P is tested for intersection with every simplex of the hull using a ray-line segment intersection operation. If the number of intersections is odd, the ray has passed through at least one simplex without a corresponding intersection – it is inside the polygon, escaping. The case of a point lying outside the polygon shooting its ray into space and also the case of into and then out of the polygon is consistent with an even number of intersections. This method works for the "strictly inside" variant of the query regardless of whether the ray-intersection test includes the origin of the ray as is sensible (that is, a point lying on a simplex shooting its ray into the ether, say perpendicularly to the simplex, would be deemed outside if the intersection at the origin of the ray with the line segment were excluded as a proper intersection). An efficient method that sorts the coordinates of the vertices and potentially avoids some operations is described in [3], which also relates the even-odd rule to the Jordan Curve Theorem.

The PIP status of each query point can be accumulated in a list of PIP results for each of the Things and Stuff sets.

The determination of following types from sums of these PIP query vectors emerges:

- Type I: If the sum of Stuff points in the Thing hull is 0 and the sum of Thing points in the Stuff hull is 0, then linear separation is assumed.
- Type II: If both sums are nonzero, or the single nonzero sum is not equal to the appropriate total, then linear separation is not assumed, and neither is encapsulation. (This is the fallthrough case for Types I,III, and IV.)
- Type III: If the sum of PIP results for Stuff points in the Thing hull is equal to the number of the points in the set (all points) but the sum of PIP results for Thing points in the Stuff hull is 0 (no points), then the Stuff cloud is fully encapsulated in the Thing cloud.
- Type IV: If the sum of PIP results for Thing points in the Stuff hull is equal to the

```
def separabilityType(thingsLDEs, stuffLDEs):

    convex_hull_things = ConvexHull(thingsLDEs)
    convex_hull_stuff = ConvexHull(stuffLDEs)

    appendFirst = lambda npa: np.append(npa,npa[0])

    hullpos_things = thingsLDEs[appendFirst(convex_hull_things.vertices)]
    hullpos_stuff = stuffLDEs[appendFirst(convex_hull_stuff.vertices)]

    stuffInsideThings = pointsInOrOnPolygon(stuffLDEs, hullpos_things)
    thingsInsideStuff = pointsInOrOnPolygon(thingsLDEs, hullpos_stuff)

    stuffIsEncapsulated = False
    if np.sum(stuffInsideThings) == np.size(stuffInsideThings) and np.sum(thingsInsideStuff) == 0:
        stuffIsEncapsulated = True

    thingsAreEncapsulated = False
    if np.sum(thingsInsideStuff) == np.size(thingsInsideStuff) and np.sum(stuffInsideThings) == 0:
        thingsAreEncapsulated = True

    linearlySeparable = False
    if np.sum(stuffInsideThings) == np.sum(thingsInsideStuff) == 0:
        linearlySeparable = True

    separationString = 'Not Assumed Linearly Separable'

    if linearlySeparable:
        separationString = 'Assumed Linearly Separable'
    elif stuffIsEncapsulated:
        separationString = 'Stuff is Encapsulated'
    elif thingsAreEncapsulated:
        separationString = 'Things are Encapsulated'

    return linearlySeparable, stuffIsEncapsulated, thingsAreEncapsulated, separationString
```

Figure 3.19: Python code for determining separability type given things-only LDE coordinates, stuff-only LDE coordinates, a stock convex hull determination routine, and a membership function that returns whether a point lies in a polygon.

number of the points in the set (all points) but the sum of PIP results for Stuff points in the Thing hull is 0 (no points), then the Thing cloud is fully encapsulated in the Stuff cloud.

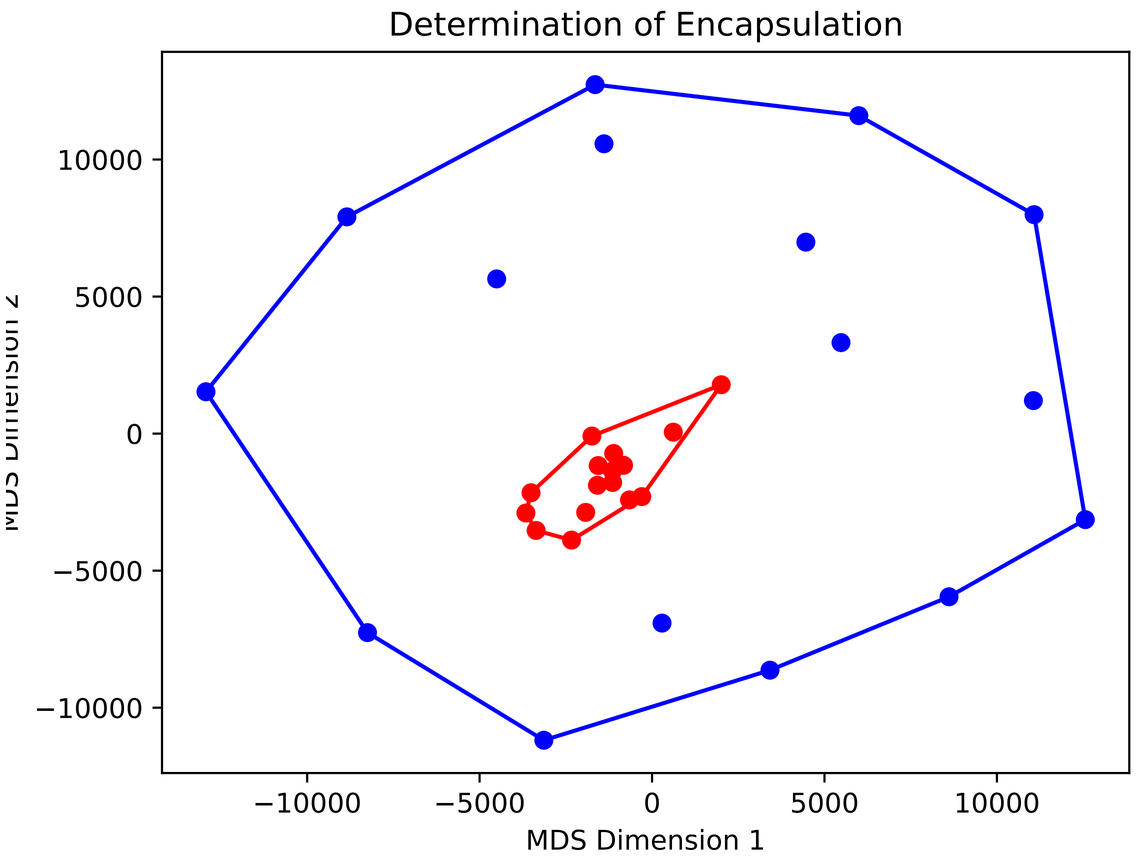


Figure 3.20: After computing convex hulls for Things and Stuff, repeated PIP (point-in-polygon) queries can determine that all Stuff points lie within the Things hull, but no Things points lie within the Stuff hull, illustrating Type III separability

3.2.2 Failure to observe Type IV separation (things converging to a point in stuff space)

The subjective conclusions of the preceding sections covering the three embedding groups can now be backed up by the algorithm just described. Each table in this section taxonomizes the embeddings within a group (i.e. recognizer feature, affinity matrix, model parameter (dis)similarity matrix), according to whether they are Type I, II, III, or IV.

In the first group, the recognizer feature embeddings, none of the embeddings are adjudged likely to be linearly separable. Kernel size, as mentioned, is not the deciding factor, except in a marginal case (if this encapsulation case was a cell, it would be exocytosing!) of the 3x3 kernel pure histogram features. Means, medians, and histogram bin heights are grouped with the GANsemble in showing all Stuff as in one contained location in the region of space for Things. Signchain and signchain-of-signchain features, as well as the label emissions of the NONGANsemble are not assumed to be linearly separable. If you are trying to separate Things and Stuff, the lowest-performing-in-their-category mean and GANsemble features are actually better, assuming it is not linear separability you are pursuing.

Encapsulation Class	Members Apparent
Type I: Assumed Linearly Separable	(none)
Type II: Not Assumed Linearly Separable	NONGANSEMBLE NONGANSEMBLENoBS 3x3_SC 3x3_SOS 3x3_HISTS 7x7_SC 7x7_SOS 11x11_SC 11x11_SOS
Type III: Stuff is Encapsulated	GANSEMBLE GANSEMBLENoBS 3x3_MEAN 3x3_MEDIAN 7x7_MEAN 7x7_MEDIAN 7x7_HISTS 11x11_MEAN 11x11_MEDIAN 11x11_HISTS
Type IV: Things are Encapsulated	(none)

Table 3.1: Things vs. Stuff Encapsulation Class of Recognizer Feature LDEs

In the second group, the firing affinity matrix embeddings, there are empirically linearly separable embeddings. They are only the affinity matrices calculated for Fakes, and also the Fake-to-Fake FID matrix. Of course, for the former, non-FID matrices, this does not make significant comment on the true nature of Things and Stuff, only likely on the competency gap that still exists in synthesis that manifests as high-quality syntheses for Stuff (and perhaps the overtrained boom-box producing network) alone. The odd Fake affinity matrices out are the deficient divisive normalization transformations of the matrix. All other embeddings, Reals for GANs and Real for NONGAN are not assumed to be linearly separable on the Things vs. Stuff distinction.

The GANsemble affinity matrix for real images with proper post-processing methods (i.e. not baseline-division) show Stuff classes being encapsulated in the hull of Things classes.

Separation Type	Members
Type I: Assumed Linearly Separable	FF_GAN FF_GAN_SUB FF_NONGAN FF_NONGAN_SUB FID_FAKEFAKE
Type II: Not Assumed Linearly Separable	FF_GAN_DIV FF_NONGAN_DIV RR_GAN_DIV RR_NONGAN RR_NONGAN_DIV RR_NONGAN_SUB
Type III: Stuff is Encapsulated	RR_GAN RR_GAN_SUB
Type IV: Things are Encapsulated	(none)

Table 3.2: Things vs. Stuff Encapsulation Class of Ensemble Labeling Affinity LDEs

In the third and final group, the model parameter distance embeddings, there are also empirically linearly separable embeddings. These are the MSE and SSIM for the NONGAN and the Signchain Distance for the GAN, as was subjectively adjudicated before, with some justification as to why this was the case for the MSE and SSIM (also almost separable for the GAN, but this is harmed by synthesis failures for the GAN). It is unknown why Signchain

Distance is separable, with a large, clear margin and Jensen-Shannon Distance is not. All other embeddings hit the default case, Type II of neither being linearly separable or having one metacategory embedded in the other.

Encapsulation Class	Members Apparent
Type I: Assumed Linearly Separable	SSIM_NONGAN MSE_NONGAN SignchainDist_GAN
Type II: Not Assumed Linearly Separable	SSIM_GAN MSE_GAN L1Norm_GAN L1Norm_NONGAN L2Norm_GAN L2Norm_NONGAN JSDivergence_GAN JSDivergence_NONGAN SignchainDist_NONGAN
Type III: Stuff is Encapsulated	(none)
Type IV: Things are Encapsulated	(none)

Table 3.3: Things vs. Stuff Encapsulation Class of Neural Network Weight Disparity LDEs

Now that the number of instances of each class of embedding separation has been tabulated, even the casual reader will note that in over 40 sometimes very different types of embeddings, Type IV was never observed. So there is now an open conjecture of whether any practically motivated set of features results in strong encapsulation of all Things classes within the cloud of Stuff classes. The faux-philosophical question of "Is all Stuff really a Thing?; Or are all Things really Stuff?" has a tentative answer that Stuff in terms of both primitive and deep convolutional recognizers all tend to behave like a single Thing more than the other way around. That the results tend this way should, once again, be unsurprising given the homogeneous definition of Stuff and the individualistic concept of Things, but it is comforting to see that confirmed visually after an improbable number of mathematical operations.

3.3 Inducing Feature-Based "Embeddings of Embeddings" via Procrustes Alignment and Second-Round MDS

With so many computed LDEs, it is desirable to be able to compare them on the basis of more than just their membership in an encapsulation class. Within each coarse category (recognizer feature, affinity, model parameter distance) of embedding, we seek to easily visualize their degree of relatedness. This suggests employing a second-round of MDS, but predicated on the dissimilarity matrix of the embeddings themselves and not on the features directly. In other words, what we desire is an "embedding of an embeddings", or a *metaembedding*, based on an appropriate notion of the "distance" between embeddings. In fact the term "meta-embedding" has come into use recently in NLP [30] to denote merely composite embeddings computed, usually, by simply averaging corresponding configurations across a set of embeddings. This does not cohere with the popular reflexive sense of the meta- prefix (as in metafiction or metahumor) as it would better apply to "embeddings of embeddings" – perhaps we will be left with referring to our high-level or parent embeddings as something clumsier or less immediately cognizable like *matriembeddings* or *acroembeddings* (because they teeter above daughter embeddings) in the wake of this prior art.

3.3.1 Procrustes Alignment of Embeddings And MDS of Procrustes Distance Matrix

In any case, the natural notion of distance is the L2 norm, but we are looking to minimize this distance between corresponding points (i.e. identifying the same classes or networks) in each embedding. Minimizing the pointwise differences between paired datasets by a specific alignment procedure is minimizing the "Procrustes distance" between the point sets.

Practitioners of shape analysis such as those in anthropology or paleontology are familiar with the Generalized Procrustes Analysis [65], which builds a consensus reference shape that is an intermediary between a list of shapes being compared. By contrast, classical Procrustes analysis chooses one data set as the reference shape and aligns the other set to it by finding an optimal transformation matrix T which when multiplied with the data set B best superimposes on the data set A. The specific procedure used by our stock implementation (`scipy.spatial.procrustes`, [170]) is based ultimately on the Ph.D. thesis work of Schonemann [143], who provides an algorithm for finding the optimal orthogonal procrustes transformation, meaning one that only used rotations and reflections. Schonemann worked at the University of Illinois, where others studied the burgeoning literature of factor analysis rotations and Hurley and Cattell developed a popular computer program called PROCRUSTES, cheekily named after the bandit of Greek myth. They humorously opined [82] in 1962:

> This program lends itself to the brutal feat of making almost any data fit almost any hypothesis! Because of this possible proclivity we gave the code name Procrustes to this program, for this reference describes what it does, for better or worse. To publish widely a program which permits any tyro, by pressing a computer button, to seem to verify any theory, is as irresponsible as loosing opium on the open market.

making comparisons to the blind application of the Varimax rotation. Schonemann showed himself that his work was a more generalized version of that of Green [67] which also attempted to find a minimization of the sum of squares distance.

Since the unencumbered term "Procrustes analysis" can then be considered somewhat historically ambiguous, the specific instantiation we use apparently proceeds as follows:

- The data matrix A and B are mean subtracted, moving the data to the origin. The

Frobenius norm is taken of each transformed matrix. The matrices are divided by their respective norms, producing transformed matrices A' and B'.

- Then the Orthogonal Procrustes transformation matrix R that transforms B' and returns a scalar s is computed.
  - To compute the Orthogonal Procrustes matrix R and scale factor, the singular value decomposition of $(B'^T A)^T$ is taken.
  - The transformation matrix R is produced by multiplying the resulting left singular vectors matrix u by the right singular vectors matrix v.
  - The scale s is the sum of the singular values w.
- Now that R and s are computed:
  - The transformed version of B', B'', is $sB'R^T$.
  - The Procrustes distance M^2 is the sum of squares, or $\sum (A' - B'')^2$.

With the Procrustes distance defined, any group of embeddings operating on the same base objects (e.g. classes through their recognizer features) can have its pairwise Procrustes distance matrix calculated. This distance matrix can then be fed as a different data matrix X to be used by the metric MDS procedure. This second round of MDS produces the metaembedding for the group of embeddings, associating a point in the metaembedding's space with each embedding.

Schematically, this is the procedure so far, up to the point of producing the metaembeddings:

$$\text{MDS(features)} \rightarrow \text{Procrustes(MDS(features))} \rightarrow \text{MDS(Procrustes(MDS(features)))}$$

Metaembeddings could potentially enable some interesting advanced queries. For example, if you were to signchain-decimate the feature vectors in your design matrices X, would the signchain-decimated features end up in close proximity to their unsignchained equivalents? If so, it would suggest that quite a lot of information that can be used for classification is encoded in the subordinal information of features.

3.3.2 Recognizer Feature Metaembeddings

In Figure 3.21, the embedding pairwise Procrustes distance matrix is shown for the various recognizer feature embeddings. Naturally, embeddings have zero distance to themselves but other embeddings with a small amount of Procrustes distance are the medians features amongst themselves, and, to a lesser extent, the means. The NONGANsemble is also somewhat similar to the NONGANsemble without baseline subtraction, whereas the GANsemble is not. This comports with the recognition results at the end of Chapter 2 and also the affinity matrix results where baseline subtraction is critical for the GANsemble but not for the NONGANsemble. NONGANsemble embeddings are very poorly alignable with GANsemble embeddings and also the primitive recognizer embeddings. Most of the similarity in a Procrustes minimizing sense is found among the various primitive features.

Recall that these distances are not related directly to similar performance, they are related to how similarly the feature vectors assign classes and networks to similar points in the low-dimensional embedding spaces.

Visually, it is inconvenient to reason about the kinship of embeddings using a heatmap visualization of the distance matrix. Figure 3.22 presents the embeddings as points in the metaembedding space. These metaembeddings reveal, for example, the earlier mentioned tendencies. For example, the NONGANsemble with and without baseline subtraction experience very diminished separation compared to the GANsemble with and without baseline subtraction, and this is seen immediately. The signchain of signchains and signchain features are grouped together regardless of feature size, and the median features are as well. Mean and Histogram features primarily are but move into different sections of the space with a change to the highest filter kernel size, 11x11.

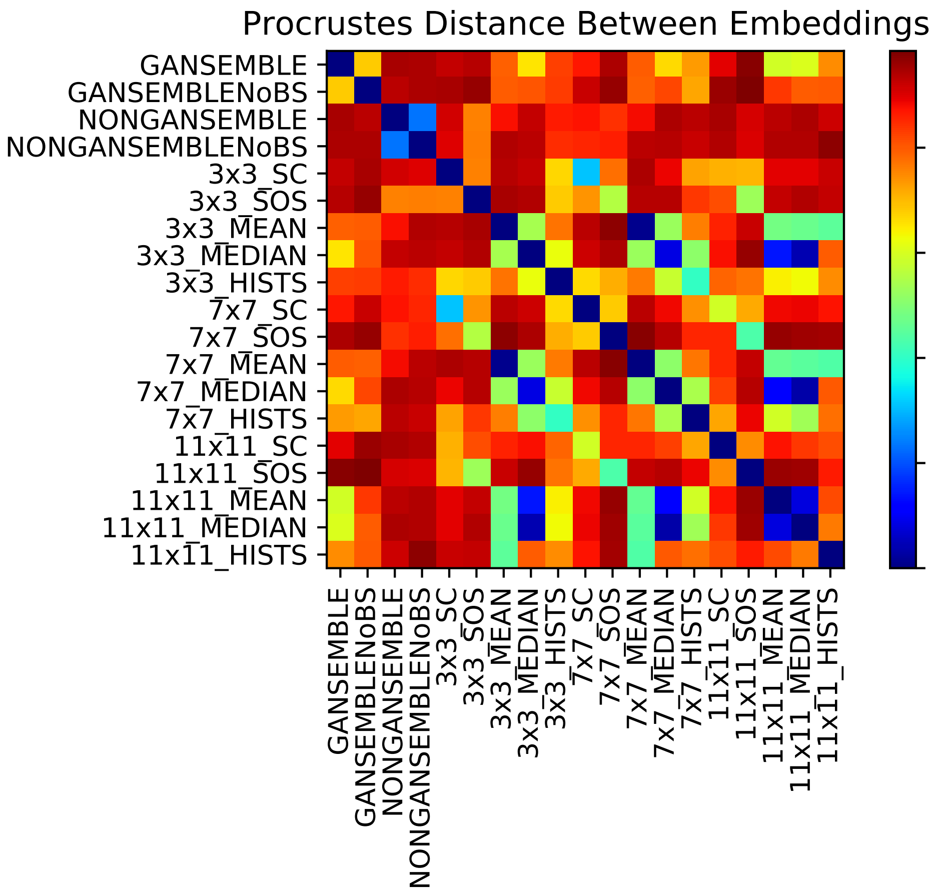


Figure 3.21: Procrustes distance matrix between embeddings of classes based on recognizer (histogram and label spectrum) features.

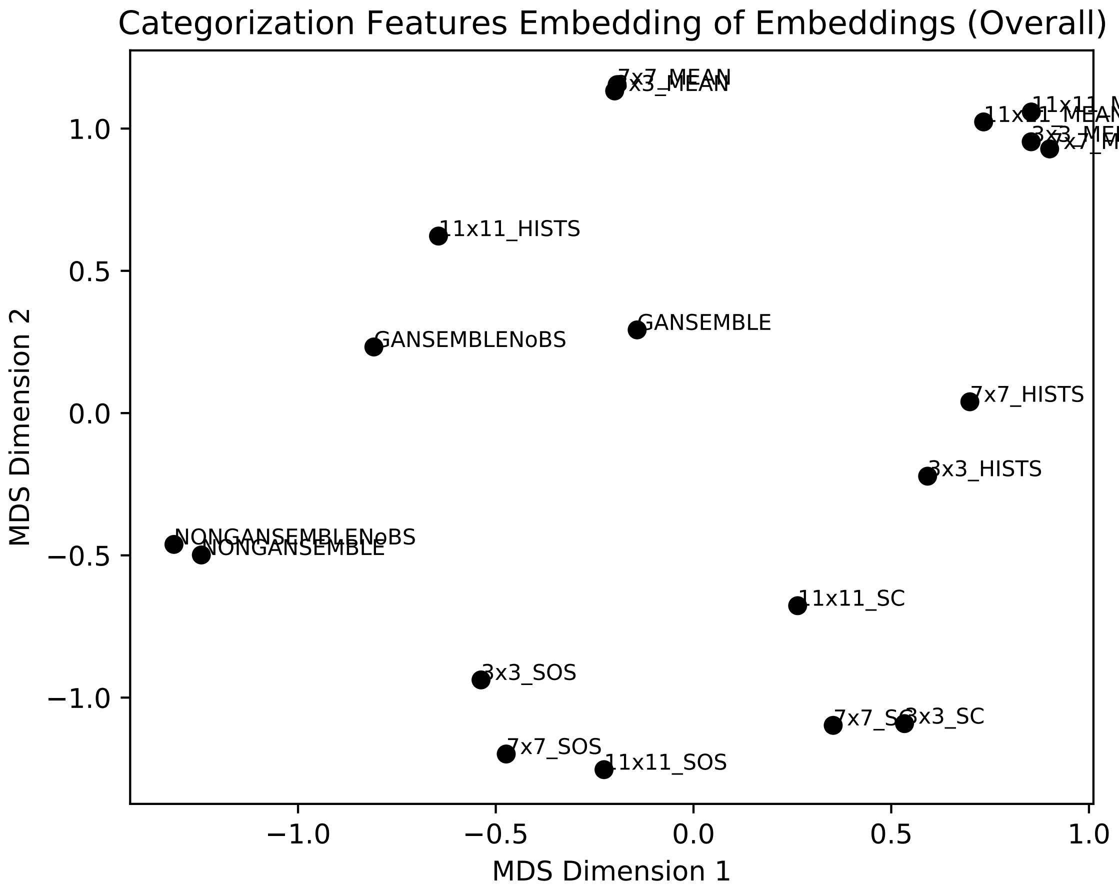


Figure 3.22: MDS metaembedding which situates recognizer features in a shared space using MDS on the Procrustes distance between first-stage (classes in a feature-based space) embeddings

3.3.3 Ensemble Labeling Affinity Metaembeddings (Model Output)

Figure 3.23 displays the pairwise Procrustes matrix for the second group of embeddings, based on the first-round MDS of affinity matrices. It can be seen that the closest similarities are of the GAN and NONGAN affinity matrices with raw activation to their corresponding affinity matrices with baseline subtraction. This too was borne earlier in examining by eye the first stage embeddings, and also realizing the effect of the affinity matrices (although the dissimilarity is higher for the NONGAN than it is for the GAN, which is somewhat curious given the importance of the baseline subtraction activation. The greatest distance seen in the matrix is between the GAN operating on reals and the NONGAN. Baseline divided matrices can also be expected to be distant from each other in a second-round embedding. Interestingly, the FID between fakes is not the most distant from any thing but the reals compared through the GAN, which suggests that the NONGAN assessed on reals and fakes and the GAN assessed on fakes "perceive" in a relatively more similar way to the Inceptionv3 net, in the sense of positioning the various classes in the space created by activations (specifically model-terminal activations for the GANs and NONGANs, and slightly earlier activations for the comparatively sprawling Inceptionv3).

Figure 3.24 shows the metaembeddings for firing affinity. The NONGAN and NONGAN subtraction conditions for reals and fakes are strongly colocated. The GAN assessing fakes and the GAN assessing fakes with subtraction are nearly coincident, and proximate to the FID metaembedding, which lies between these and the NONGANs with proper post-processing. Meanwhile the GAN and GAN with baseline subtraction operating on the reals are colocated, but distant from the majority of configurations. Flanking them are the improper transformations of the GAN and NONGAN, the divisive normalization cases. In all other cases, the tightest pairs are based on assessing the same kind of images (either only reals or only fakes) and having the same kind of network (only GAN or only NONGAN). The divisive normalization bucks the trend by showing the most closely aligned embeddings to be those that share only the network type regardless of type of images seen. Thus, dividing by the baseline creates matrices that are more discriminative of the type of network, which may relate primarily to the different levels of activation (the GAN firing ratio, which was positive for all classes, showing detectably higher activation levels even after baseline subtraction only applied to the GAN activations) – a factor which is highlighted best when the activations have not been corrected for class-differential firing regimes and the affinity matrices are therefore streaked.

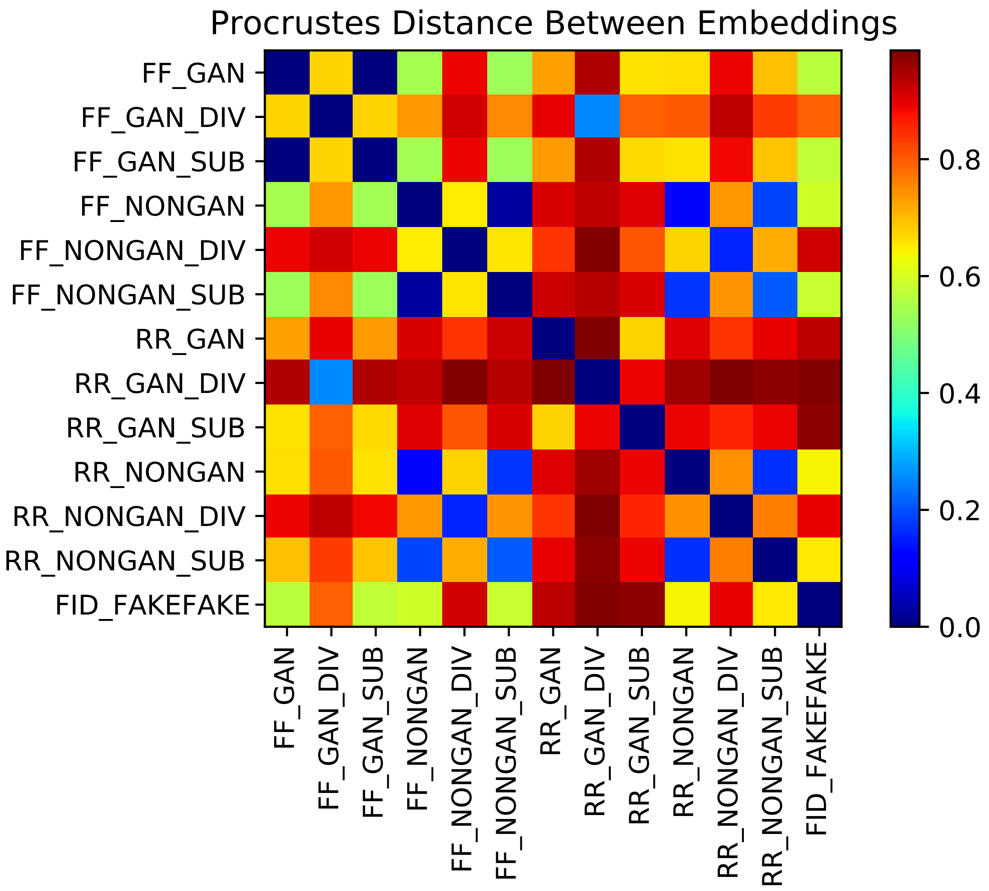


Figure 3.23: Procrustes distance matrix between embeddings based on GANsemble and NONGANsemble labeling affinity matrices.

Note that in the metaembedding space, GANs and NONGAN affinity matrices are linearly separable, and the closest vector to the probable margin (which in SVM would be probably chosen as a support vector) is the FID vector. Thus, FID distance between classes may represent something intermediate to the sense of distance separately established by GANs and NONGANs. Once again, just as the recognizer feature metaembeddings do not focus on the mere performance of the recognizer features primarily but the nearness of the feature vectors (the behavioral similarity of the operation in terms of placing classes and networks near each other), so too do the affinity metaembeddings comment on the behavioral similarity of the affinity matrices at the class positioning level and not at the performance level (which for affinity matrices and distance matrices we have considered by the end of Chapter 2 to be a

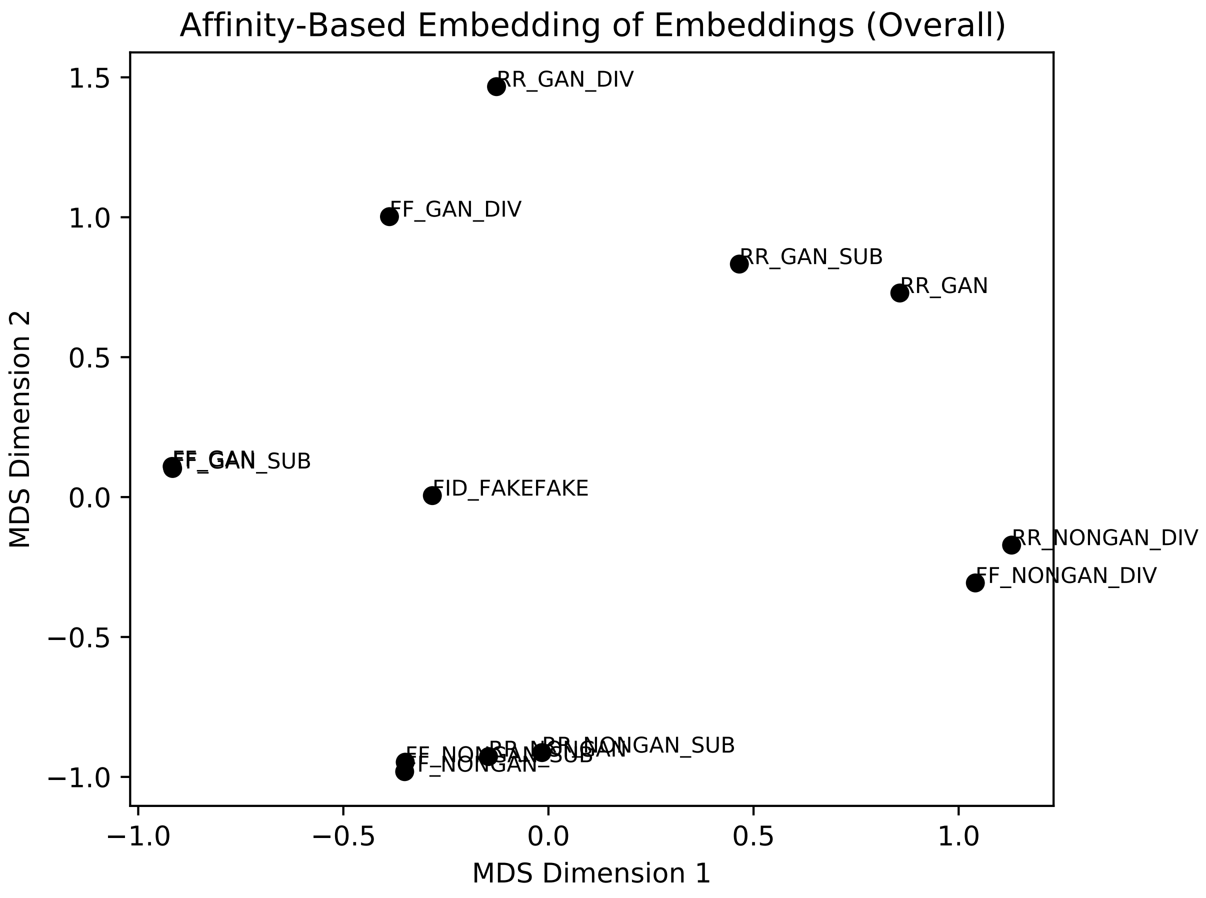


Figure 3.24: MDS metaembedding which situates affinity matrices in a shared space using MDS on the Procrustes distance between first-stage (classes in a affinity-matrix-based space) embeddings

mix of demonstrating the Things vs. Stuff dichotomy by displaying the characteristic pattern while also preserving enough graded negativity for similarity and recognition operations to be successful).

Accordingly, an advanced future query that this type of metaembedding could allow us to make is: "does a straight mixture of GANsemble and NONGANsemble joint labeling as by addition or averaging lie evenly between the GANsemble and NONGANsemble centroids?"

3.3.4 Neural Network Interclass Weight Disparity Metaembeddings (Model Parameters)

One way of assessing the balance of constituent models (not according to behavior, but according to makeup) in a chimeric mixture of models is by assessing the relative distance of a chimera to candidate parents using the model parameter distances studied in Chapter 2. The metaembeddings constructed from the various parameter distance embeddings do not, as before, comment on the recognizers (whether deep convolutional or primitive convolutional). They comment on the kinship of distance measurements themselves, demonstrating to us through the lens afforded by particular high dimensional objects (our GANs and NONGANs) how closely the measurements perform similar functions. This is without resort to the mathematical formulation of those functions, so it is an interesting method with which to taxonomize functions functionally, rather than by descent through analytical derivation.

Of course, this method of assessing nearness of functions is only valid when the functions are receiving the same input, so we should again expect to see a bifurcation on differing input, specifically in this instance a separating frontier between the GANs and NONGANs since the distances are measuring different sets of data.

Upon examination of Figure 3.26, this frontier is discoverable, cutting across from top left to bottom right and separating GAN vs. NONGAN metaembeddings. However, on a nearly opposing axis, there is a separate bifurcating frontier (it is not being called a decision boundary in the absence of a preplanned decision) which situtates metaembeddings in the upper left with points corresponding to embeddings which represent distances which are distributional (the Jensen-Shannon divergence and the signchain distance). In the lower right, the points correspond to traditional distances, with the very traditional distances, the Minkowski p-norms, situated as the most extreme alternatives to the distribution divergences, and the SSIM and for the NONGAN the MSE as intermediary alternatives.

This pattern is seen in Figure 3.25 which is the metaembedding-generating pairwise Procrustes matrix. The top left corner of the matrix shows a variety of distances, but the distributional distances, ordered last, are strongly different from the other distances, enough to make their entries look like a visually distinct augmentation to the matrix. However, the second-round of MDS is powerful enough to help see that the Jensen-Shannon divergences and signchain distances are similar enough to each other forming a bloc in the long run as compared to the other measurements, even when, from the perspective of the Signchain Distance, the lazy strategy of picking solely the nearest Procrustes neighbor with no more detailed information about other potentials would lead one to group the SSIM and the Signchain Distance most closely together.

Since these metaembeddings can be constructed even where distances are measuring slightly different data from the same general source, it would be highly interesting to create a similarity map of the exotic zoo of remotely valid distances, metrics, similarities, and correlations that have been proposed throughout mathematics. If colocations were stable for high-dimensional enough but varying source data, it would go far as to justifying the choice of distances evaluated by researchers in many areas (for example, the controversy of the redundancy of SSIM, MSE, and PSNR in image quality statistics [41][127]). The decreased emphasis on clustering metrics by their own reported performance rather than redundancy mirrors the decreased emphasis on performance, and increased emphasis on similar judgments, inherent in this procedure.

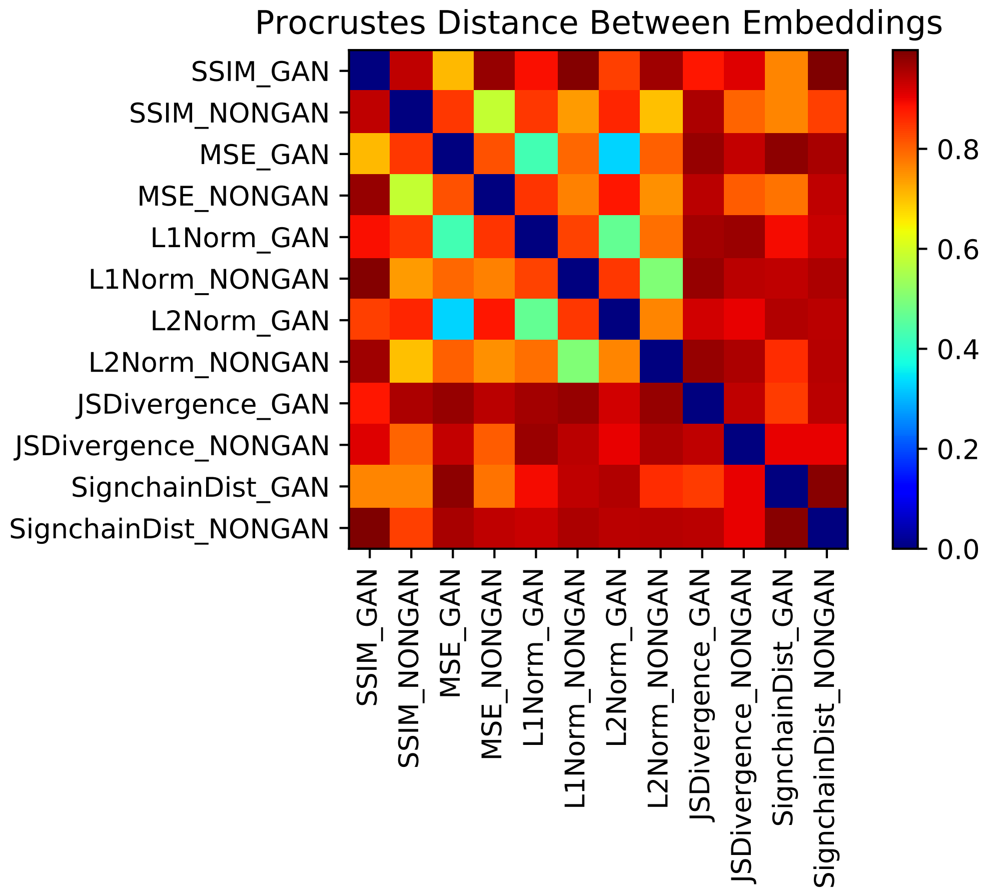


Figure 3.25: Procrustes distance matrix between embeddings based on GAN and NONGAN model parameter distances.

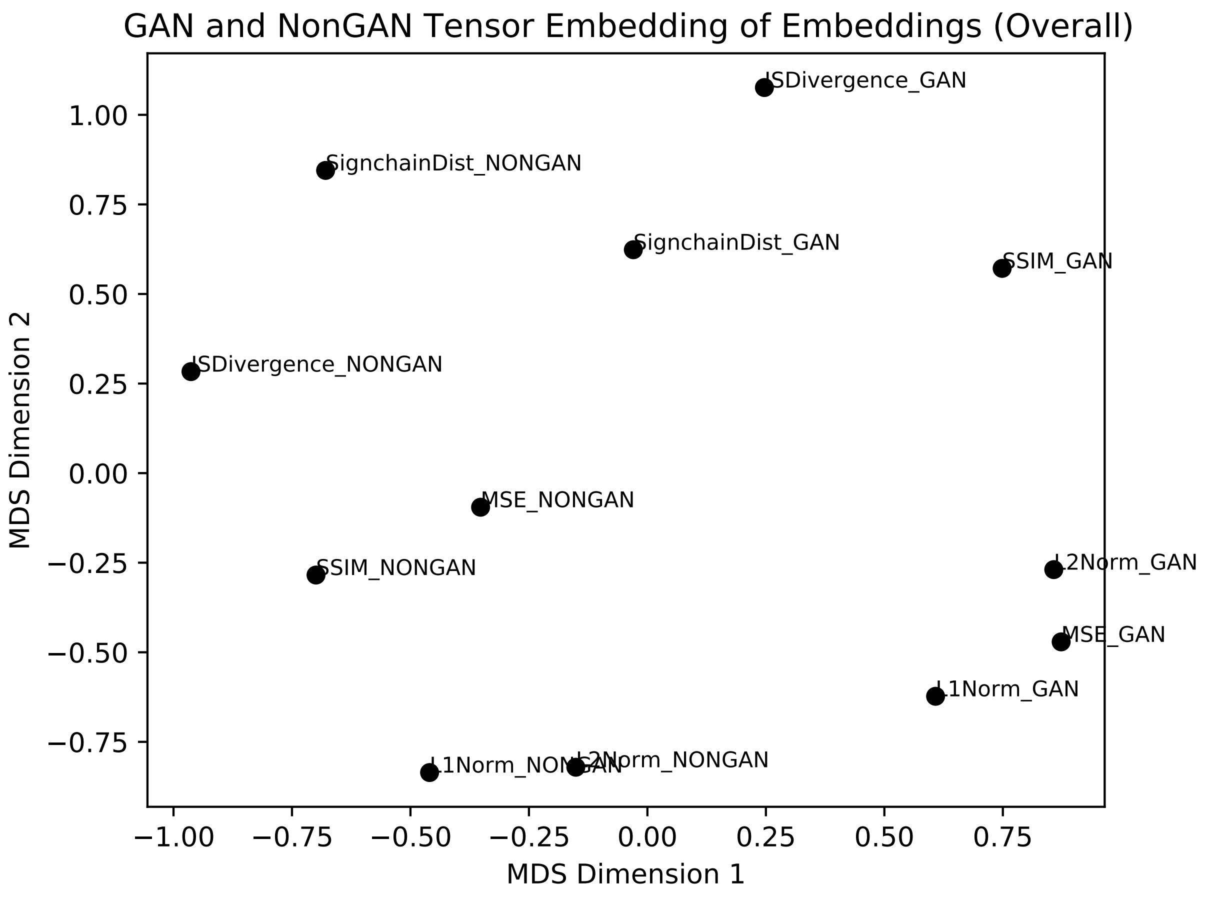


Figure 3.26: MDS metaembedding which situates notions of model distance in a shared space using MDS on the Procrustes distance between first-stage (networks in a "metric"-based space) embeddings

3.3.5 Estimating Relative Invariances of Metaembeddings to the Things vs. Stuff Distinction via Second-Round Procrustes Alignment

Since the object of this investigation is ultimately to compare Things and Stuff classes, we extend the metaembedding procedure one step further. Accordingly, two new sets of metaembeddings for each parent metaembedding are recalculated with only the things classes considered in one case, and only the stuff classes considered in the other. These could be called the *conditional metaembeddings* of the embedding group under consideration because their construction has been subject to the conditioning variable of objectness in this case.

It can be easily seen that we now have two sets of corresponding metaembedding coordinates, which seems to call out for a second round of Procrustes alignment following the second round of MDS.

Once more, schematically:

$$\begin{aligned} &\text{MDS(features)} \rightarrow \text{Procrustes(MDS(features))} \\ &\rightarrow \text{MDS(Procrustes(MDS(features)))} \rightarrow \text{Procrustes(MDS(Procrustes(MDS(features))))} \end{aligned}$$

Once A'_{Things} and B''_{Stuff} are superimposed, the pairwise Procrustes distance matrix is not particularly interesting, given that it is one entry. What we are interested is the vector of pointwise disparities themselves, which is specifically the Euclidean distances between corresponding thing metaembedding and stuff metaembedding points.

When these distances are known, it will be known, for example, which recognizers were most vs. least invariant to the Things vs. Stuff distinction, not in terms of recognition performance, but in terms of recognizer labeling behavior.

Much past this point, one may object that the accumulating error of the Procrustes alignment process and the instability of the multidimensional scaling procedure would accumulate to the point where these results are uninformative. Indeed the misalignment is what we rely upon to make the metaembeddings and comparisons of conditional metaembeddings in the first place so it is not escapable, and the second concern is a point well-taken that is only slightly helped by the fact that we have baked into the metric MDS procedure many iterations to try to get the highest quality results. A full Monte Carlo estimation of the stability of conditional metaembedding alignment through iterated optimizations is somewhat beyond the scope of the investigation.

Fortunately, there is no more natural way forward chaining together MDS-based embedding and Procrustes-based alignment beyond situating the three embedding groups (recognizer feature, network affinity, and model parameter) somehow in a shared space. With only three groups of metaembeddings operating on somewhat dissimilar base objects, such a delve into the space of the space (metaspace) of recognizers seems profitless for the time being, and we stop at mentioning its existence as a theoretical object.

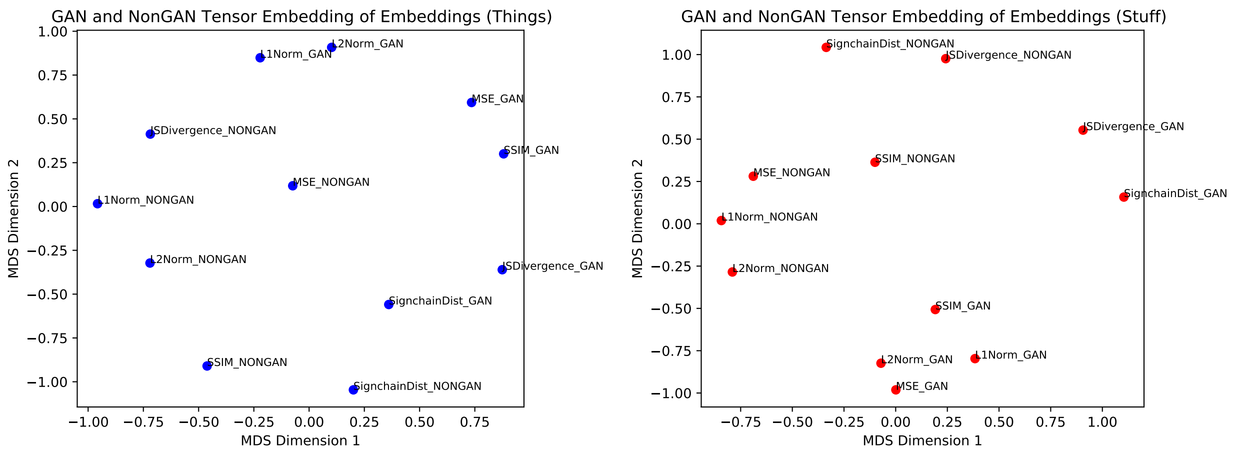


Figure 3.27: Conditional (Things and Stuff classes respectively) metaembeddings ultimately based on model parameter distances, shown separately

Figure 3.27 shows the conditional metaembeddings (for Things and for Stuff, respectively) for the distance embeddings. The "unconditional" metaembedding was shown previously, in

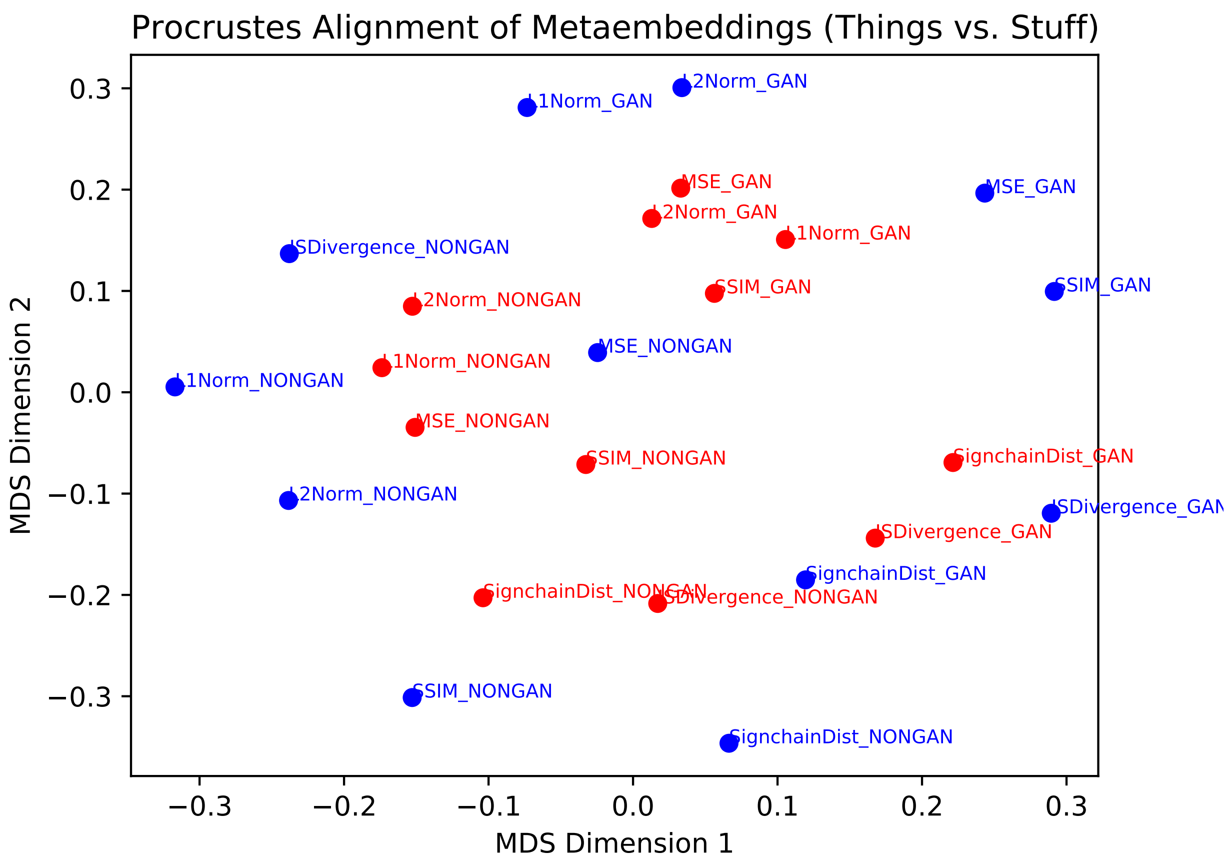


Figure 3.28: Second-round Procrustes alignment of conditional (Stuff and Things) model parameter distance metaembeddings into a shared space.

Figure 3.25. When the conditional metaembeddings are aligned through a second, chancier round of Procrustes superimposition, the result is shown in Figure 3.28. For completeness, Figures 3.30 and 3.29 show the conditional metaembeddings and their second-round Procrustes superimposition for recognizer features and affinity matrices metaembeddings respectively.

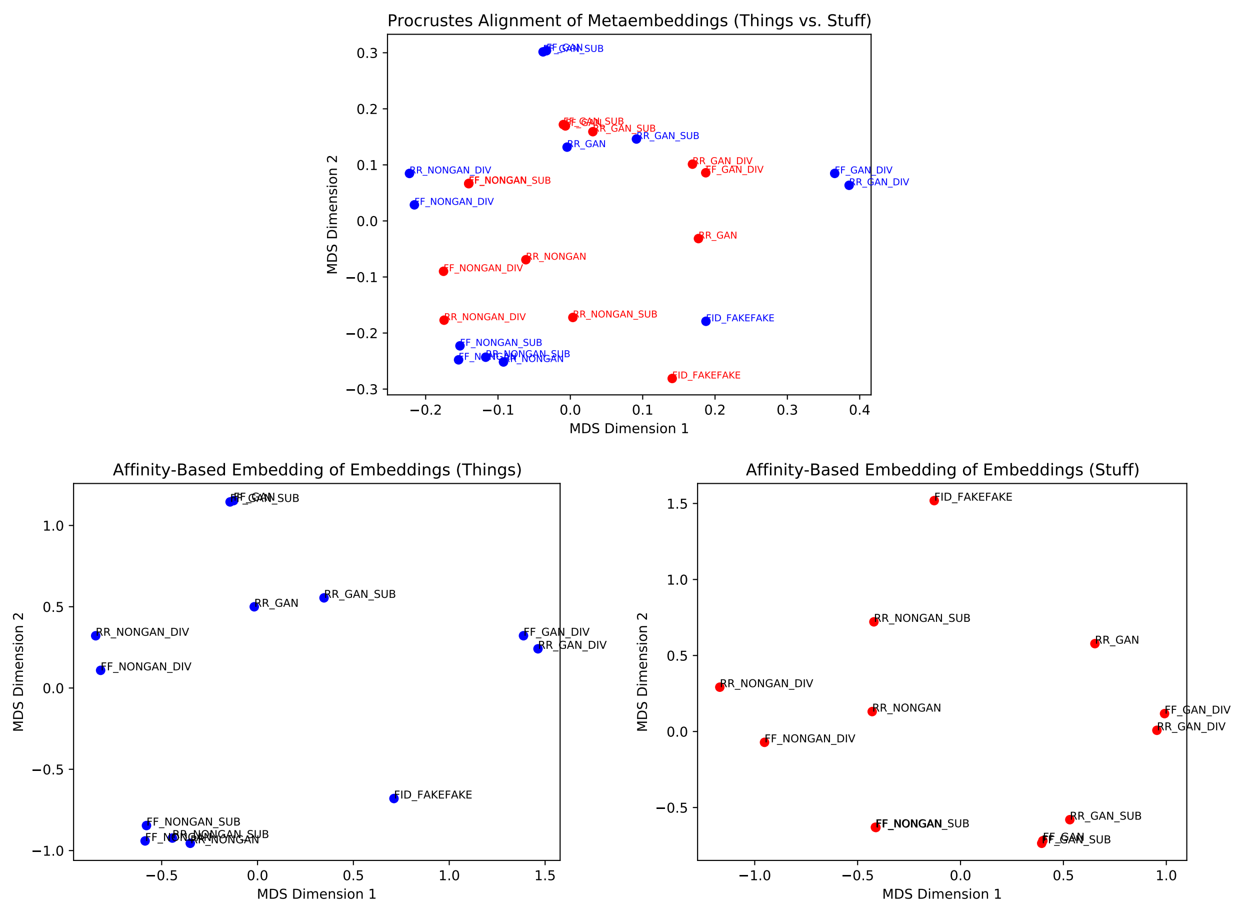


Figure 3.29: Second-round alignment, affinity metaembedding

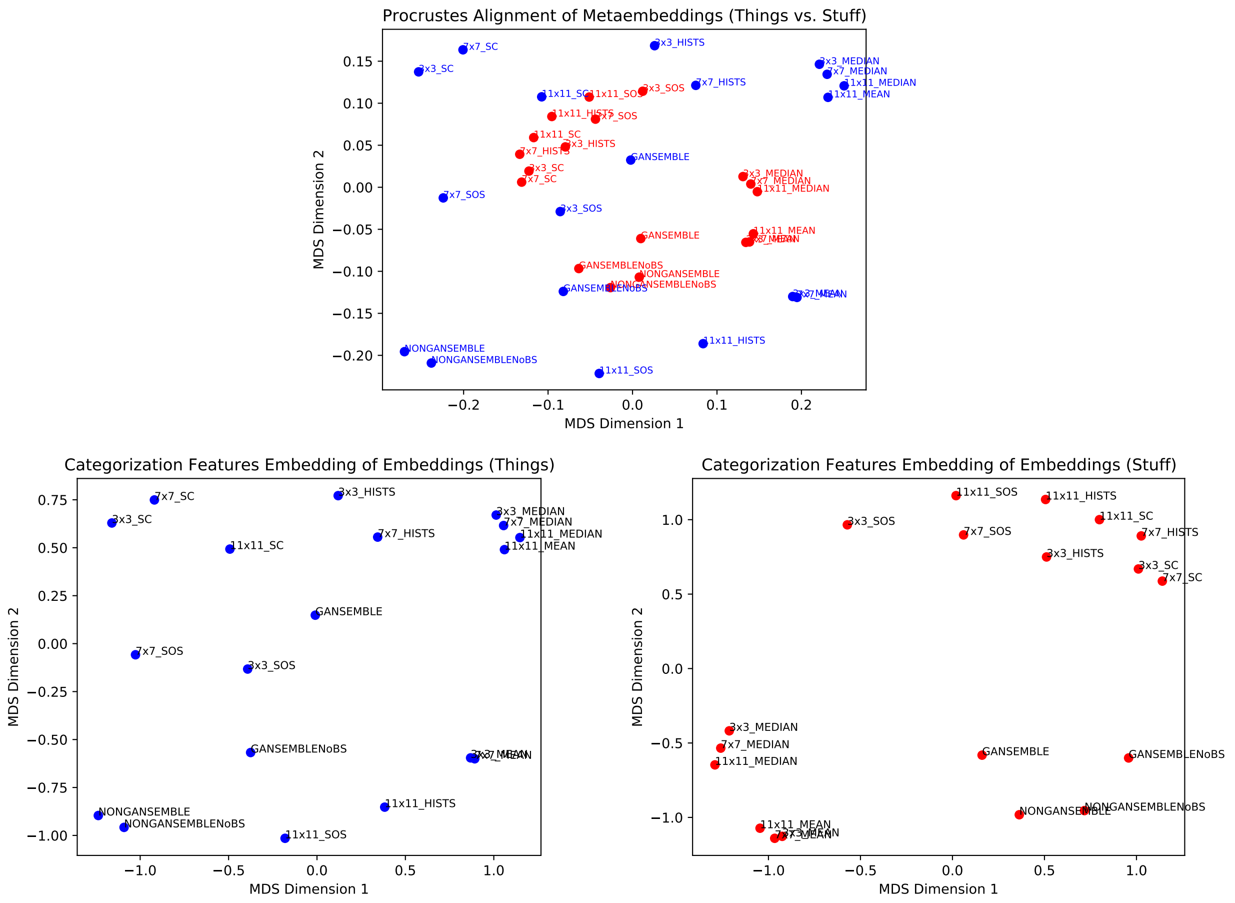


Figure 3.30: Second-round alignment, recognizer feature metaembedding

It can be seen that at this point the Procrustes process does not always align points in the Things cloud closest to their correspondents in the Stuff cloud. However, some large degree of specific misalignment still resulted in the useful and theoretically and empirically explicable metaembeddings. The accumulation of optimization error and the constraint on affine transformations in Procrustes superimposition and similar limitations in MDS threaten to make this alignment somewhat less valid, however.

Pending those concerns, in Figure 3.31, the Euclidean distances between natural correspondents from each conditional metaembedding post the superimposition are listed, and sorted by least "meta-distance". If the process of two rounds of MDS and two rounds of Procrustes superimposition were perfect or not sufficiently error-accumulating, this visualization would show which distances were least affected by the Things vs. Stuff dichotomy. We might assume, perhaps incorrectly, that the most extreme values (the largest deviations, or the furthest outliers) are the most reliable for observing authentic order preservation of degree of invariance.

The most invariance in the distances is the MSE of the GAN, and the lower half of these meta distances primarily belong to natural distances (L1Norm, L2Norm) whereas the higher half (associated with *less* invariance to the dichotomy as purely according to this method) is associated with distributionally-oriented distances (the Jensen-Shannon Distance and the SSIM, which is, while not a divergence, a highly distributionally-founded distance due to its calculation of local statistics such as the mean, variance, and covariance). GAN metadistances are more interior to the sorted value curve, but not as a definite rule, and the middle of this curve should be expected to elicit greater metadistance uncertainty by the informativeness of extreme values. It is difficult to support this technique using the evidence of the previous confusion matrices, because, if anything, the MSE for the NONGAN presented layerwise quadrant differences that were more pronounced than those for the distributional differences. A proper confirmation or refutation might only arise by taking into consider-

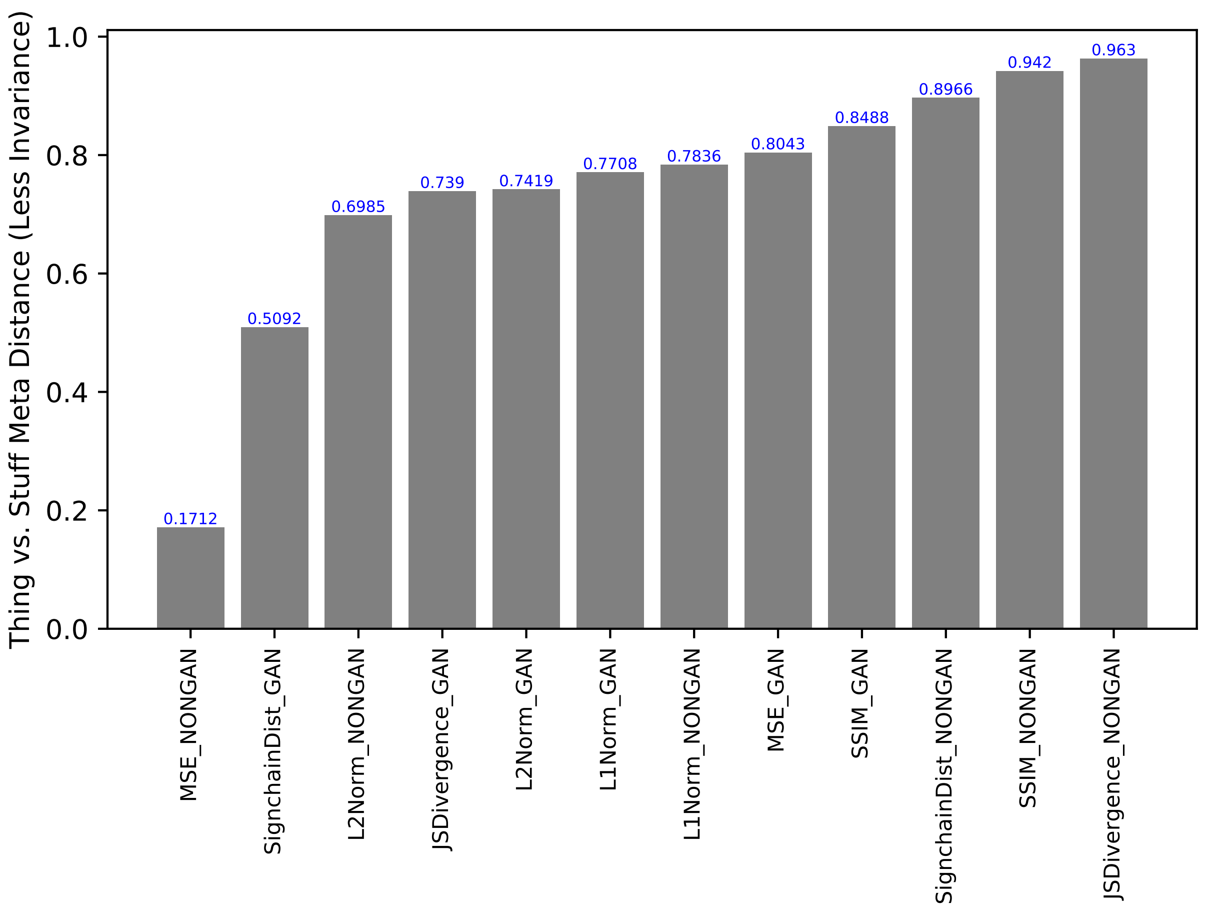


Figure 3.31: Euclidean (aka L2 Norm) distance between corresponding points in aligned thing and stuff-based metaembeddings of GANsemble and NONGANsemble model parameter distances, sorted ascending

ation distance matrices based on the entire set of weights (as was ultimately used for the metadistance calculation) instead of layerwise subsets of the weights (which is displayed in the distance matrices, which have quite significantly different attained ranges at times, especially for the L1 and L2 norms). If the highest invariance expressed by metadistance was actually the MSE of the GAN, this would make better visual sense, since the MSE of the GAN induced perhaps the subjectively flattest (mostly driven by synthesis failure anomaly) distance matrices. For the primitive recognizers, the pattern is that smaller kernel sizes are more invariant to the distinction. Of course, smaller kernel sizes are somewhat less competent at recognition, as we have seen in Chapter 1, and so a greater projected invariance to

Things vs. Stuff does not constitute a practical reason to argue for them.

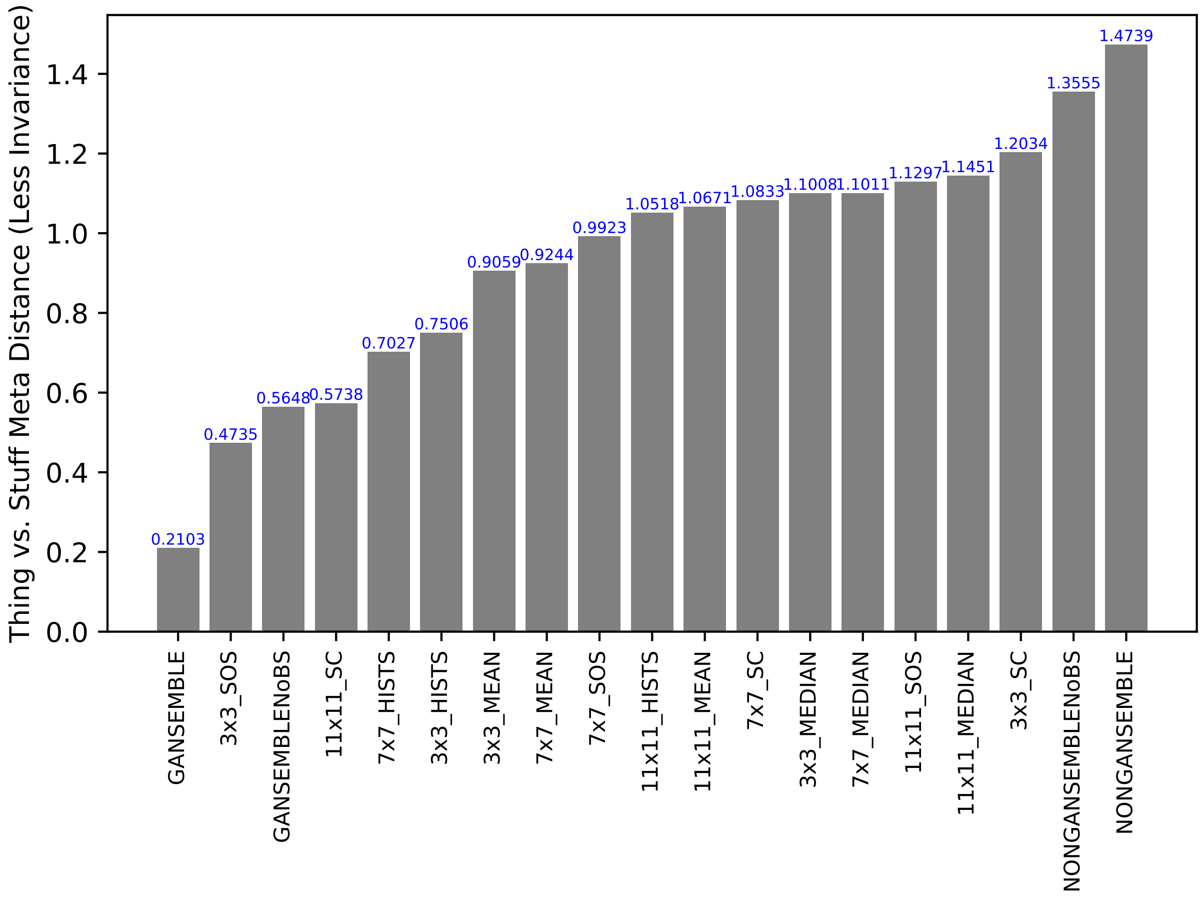


Figure 3.32: Euclidean (aka L2 Norm) distance between corresponding points in thing and stuff-based metaembeddings of recognizer features following second-round Procustes alignment, sorted ascending

This case is made with greater confidence though via the recognizer feature metaembeddings, where the tails of the sorted metadistances (Fig. 3.32) are the GANsemble and NONGANsemble respectively. This argues that the GANsemble features might be most invariant to the Things vs. Stuff distinction and the NONGANsemble features might be least invariant. Possibly confirmatory evidence for this is provided in the affinity matrices presented in Chapter 2. In the affinity matrices for GANs, the Stuff-Stuff quadrant was mostly flat, but the Thing-Thing quadrant was not strongly uniformly delineated as might be found in the idealized characteristic pattern (of parameter distance matrices); additionally, the

Thing-Stuff cross quadrants were not as strongly related in appearance to the Thing-Thing quadrant. For the NONGANsemble, this changes, and in revisiting these affinity matrices, it can be seen that the Stuff-Stuff quadrant is the qualitatively dissimilar quadrant from the other three. That this pattern coheres to the results we obtain through metadistance estimation is very impressive if it is not an accident; it is made more impressive owing to the fact that one may recall from earlier in this chapter that the embeddings for recognizer features were based on the features emitted for the first-encountered unique patch, and not all the patches in the class. If the first-stage embeddings were to be improved by making sure they represented consensus configurations averaged over all the patch instances from a class, this pattern might even more strongly develop.

Finally, in Figure 3.33, sorted metadistances are shown for the affinity matrices themselves. The most invariant is the GAN operating on the reals with subtraction and the least invariant is the GAN operating on the reals with division. The next most invariant are mostly fakes. The next least invariant are mostly reals. More of the "proper" (that is, absent baseline division) affinity matrices of the NONGAN rank lower on invariance, and more of the proper affinity matrices of the GAN (especially the reals, along with the FID between fakes) rank higher on invariance. This is consistent with the last set of metadistances. Recall once more that only one patch contributed to each feature in the recognizer feature metadistances, here we have the joint firing information across the networks and their classes represented. The GANsemble in the last set of metadistances is the firing on a real patch from each class, and the reals have the lowest metadistance (highest projected Things vs. Stuff invariance) in this set of metadistances; of course, the GANsemble's firing profile in total is a mixture of its behavior on reals and fakes, but the invariance to Things vs. Stuff is lower on Fakes. This observation, at least, makes perfect sense, because the Things and Fakes differ on quality and therefore discriminability in the sense that for this level of training, the Things were less competently synthesized than the stuff.

It is reasonable from this to assume that the GANsemble is more robust (for real images) to the Things vs. Stuff distinction than the NONGANsemble. Perhaps it is also reasonable to assume, purely from the metadistances and without much extra supporting information, that the NONGANsemble is more invariant to the distinction for fake images (when synthesis quality has not settled, and the inherent difficulty of synthesis of stuff still holds power).

This brief example is not enough to prove the validity or reliability of metadistances, but in a majority of the three cases, the metadistances matched the expectations we have gleaned in Chapter 2.

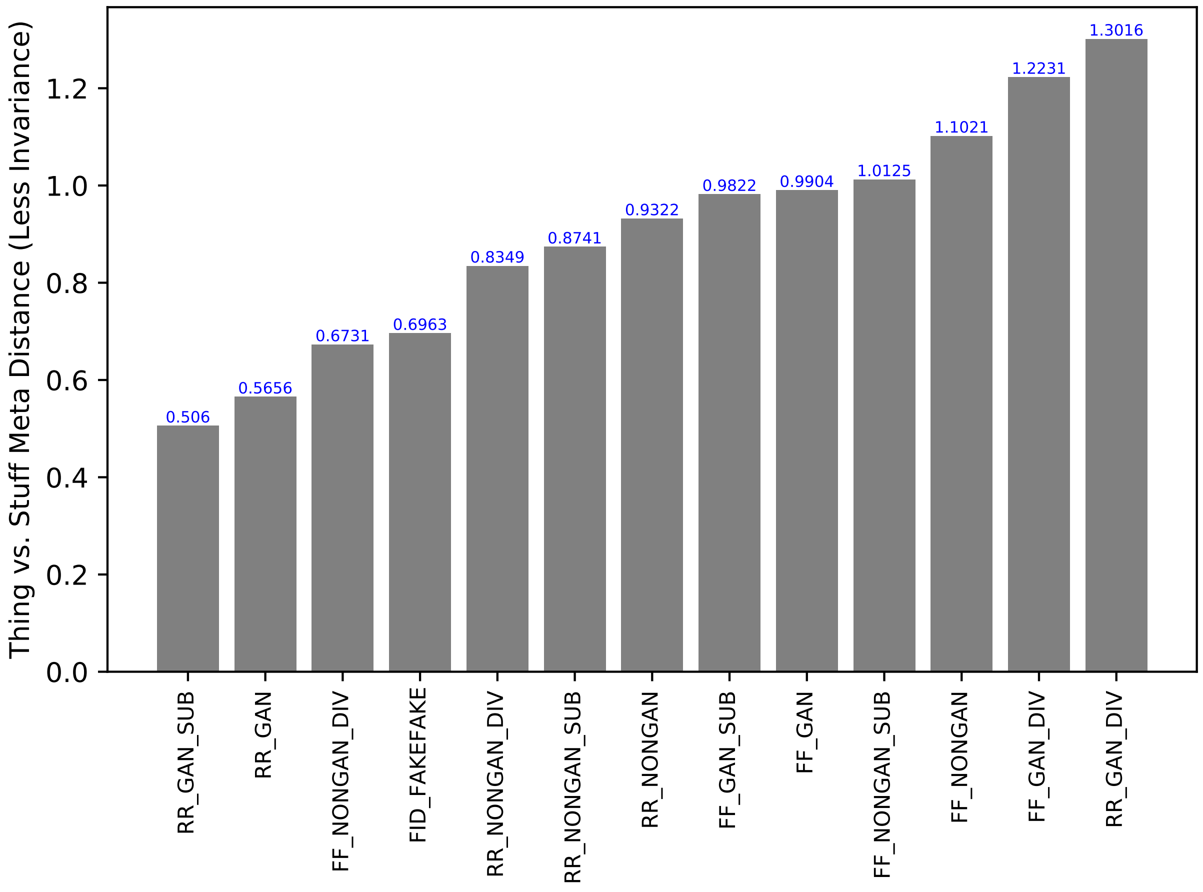


Figure 3.33: Euclidean (aka L2 Norm) distance between corresponding points in aligned thing and stuff-based metaembeddings of GANsemble and NONGANsemble labeling affinities, sorted ascending

3.4 Inducing Outcome-Based Metaembeddings Based on Classifier per-class MCC Rankings

Just as signchains showed that we could throw away some level of information about filterbank activations and still extract some meaningful information, the performance of the argmax and LDA classifiers after they act can be examined in terms of relative behavior even following some quite significant loss of information – a resort to only rank-based information about performance.

3.4.1 Kendall's tau correlation between recognition ease rankings

Consider one classifier such as the LDA classifier using medians information from the 11x11 kernel-size filters. For each class we produced a sub-classification quality index, the MCC.

The easiest classes for the classifier to recognize (as we confirmed, Stuff classes) had the highest MCC values observed within that classifier. Classifiers varied wildly in performance, so the range of MCC values makes per-class standing in terms of recognition ease incommensurable in an absolute sense.

Classifiers' per-class *rankings* of recognition ease are commensurable, however. The similarity between rankings of classes in common can be measured by how many pairs the rankings disagree on, known as the Kendall τ distance. When one ranking is considered the authoritative ranking, the distance relates how many swaps are still required to bring the unfavored ranking into the "proper order", as by the bubble-sort algorithm. The associated Kendall's tau rank correlation coefficient is calculated as:

$$\tau = \frac{c-d}{\binom{n}{2}} = \frac{c-d}{\frac{n(n-1)}{2}} = \frac{2c-2d}{n^2-n}$$

where c is the number of in-order, or concordant, pairs, and d is the number of out-of-order, or discordant pairs, involved in the Kendall τ distance.

For all of our rankings associated with various classifiers, we can compute the pairwise Kendall's τ rank correlation matrix, pictured in Figure 3.34. In this matrix, it can be seen that the pairings of intermediate Kendall's τ are all in the center, where the primitive recognizer statistics reside. The area of highest rank correlation is where classifier rankings from the mean features are compared. The mean features induce rankings that agree very strongly – of course, this is no testament to the strength of the rankings on informativeness. At the other end of the performance spectrum, the deep convolutional recognizer rankings are all (with the exception of the known-anomalous GANsemble with the argmax classifier) reasonably similar.

Once more, the similarity of the rankings is not proportional to their quality. Obviously the quality of the means rankings are poorly informative because the use of means induced at-chance or worse classifiers, as was shown at the end of Chapter 1. But the GANsemble and NONGANsemble recognizers are also expected to be poorly informative rankings. At the end of chapter 2, it was shown that these classifiers were performing at or near ceiling, with some nuisance variation primarily caused by the synthesis failure classes. Since our rankings are formulated on the basis of the per-class MCCs and these are at ceiling, they do not provide much meaningful (rather than accidental) information for ranking the recognition difficulty of textures in an attempt to find a Things vs. Stuff dichotomy. Therefore it can be summarized that to create valid rankings, one needs classifiers of *intermediate* competence.

For completeness, we shall include all of the classifiers from Chapter 1 and 2, but better quality results would likely be gained by removing uninformative classifiers. Nevertheless, we simultaneously desire a method where uninformative classifiers do not efface all useful information, and, if anything, since it is more honest to bias ourselves against finding a Things vs. Stuff distinction, the addition of a range of classifiers which anomalously figure the recognition ease of Things or which declare an equal level of ease works towards that goal.

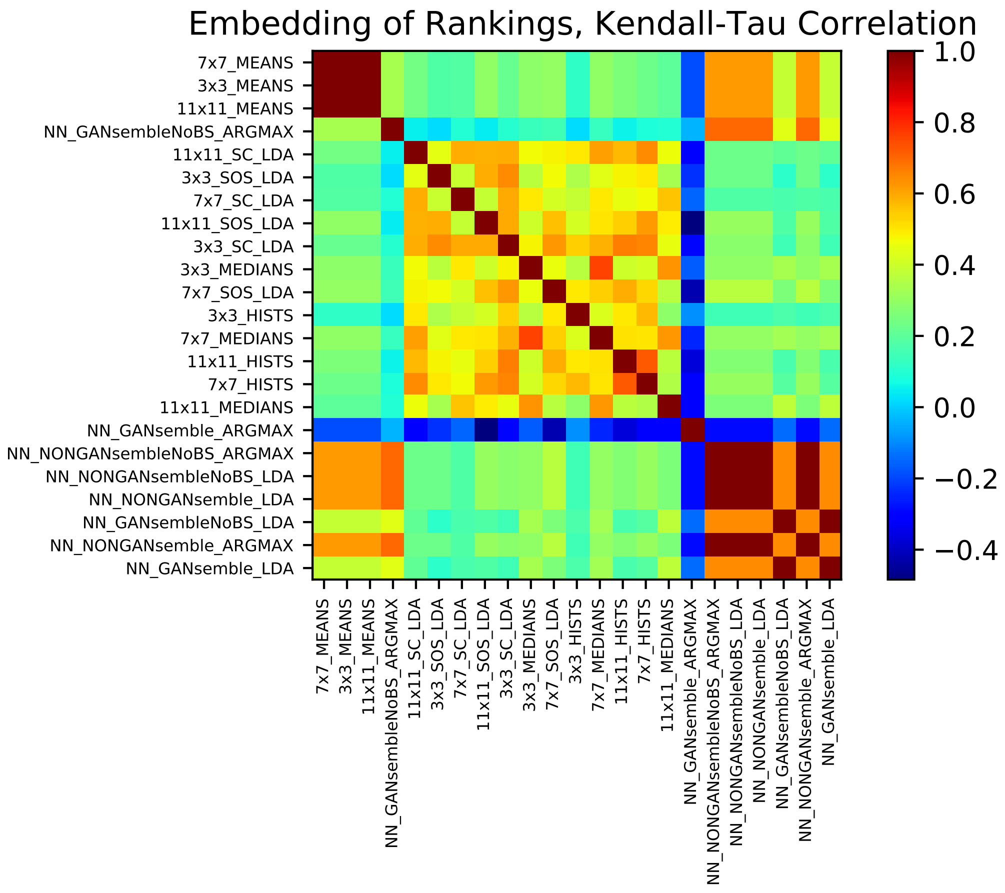


Figure 3.34: Kendall's τ rank correlation coefficient matrix between each classifier's ranking. This is the number of concordant pairs pairs minus the number of discordant pairs divided by the binomial coefficient appropriate for the number of items, distinct from the "Kendall's tau distance" which is just the number of discordant pairs. *Hotter values indicate ranking pairs which agree more.*

3.4.2 Higher-order embedding of rankings via MDS

As with the affinity matrices we have seen up to this point, then this matrix can be processed by metric MDS as an input data matrix, producing an embedding. Though this relates recognizers, just as with the metaembeddings, the resulting embedding is *not* a metaembedding because the rankings used to construct it are not embeddings (even as the rankings do relatively position classes, albeit in a one-dimensional sense). Analysis of the distance between thing and stuff conditional embeddings of rankings we will then omit, although a comparable analysis is clearly possible.

Figure 3.35 shows the embedding of rankings. Visible are the means tightly off on their own, the other histogram features of intermediate informativeness clustered together, although not tightly by kernel size primarily or method primarily (as the specific rankings, also encompassing all patches are not as tightly separated by measure as performance), the GANsemble and NONGANsemble classifiers of most kinds, and then distantly as singletons the GANsemble with the argmax classifier, in both its incarnations, without baseline subtraction and with baseline subtraction. If the exclusion of the synthesis failure classes, pursued only for Section 2.3.6, was replicated here, then it could be expected that the rankings produced by the GANsemble using argmax with baseline subtraction and exclusion might be closer to the rankings of the other GANsemble and NONGANsemble classifiers.

In summary of the lesson of most of the preceding work it can be said that there are actually a number of distinct levels of analysis of recognizers. For primitive ones based on mere features, there is the separation caused by the features, the performance under a classifier, and the separation in the internal projections possibly used by that classifier. For deep convolutional ones based on committees there is the separation in the list of activation emissions of a set of networks evaluating a particular patch, but there is also the separation between the recognizers that latently exists, up to a point of training, which can be viewed in various ways, which for a GAN include performance on fakes, reals, and discrepancy measures like the FID. Beyond that for networks there are the disparities between the recognizers model weights themselves. And even after the classifiers have done their work there is the difference the recognizers produce in the classification performance, and in the ordinally-restricted iteration of that, the rankings. There is more to analyze than mere performance.

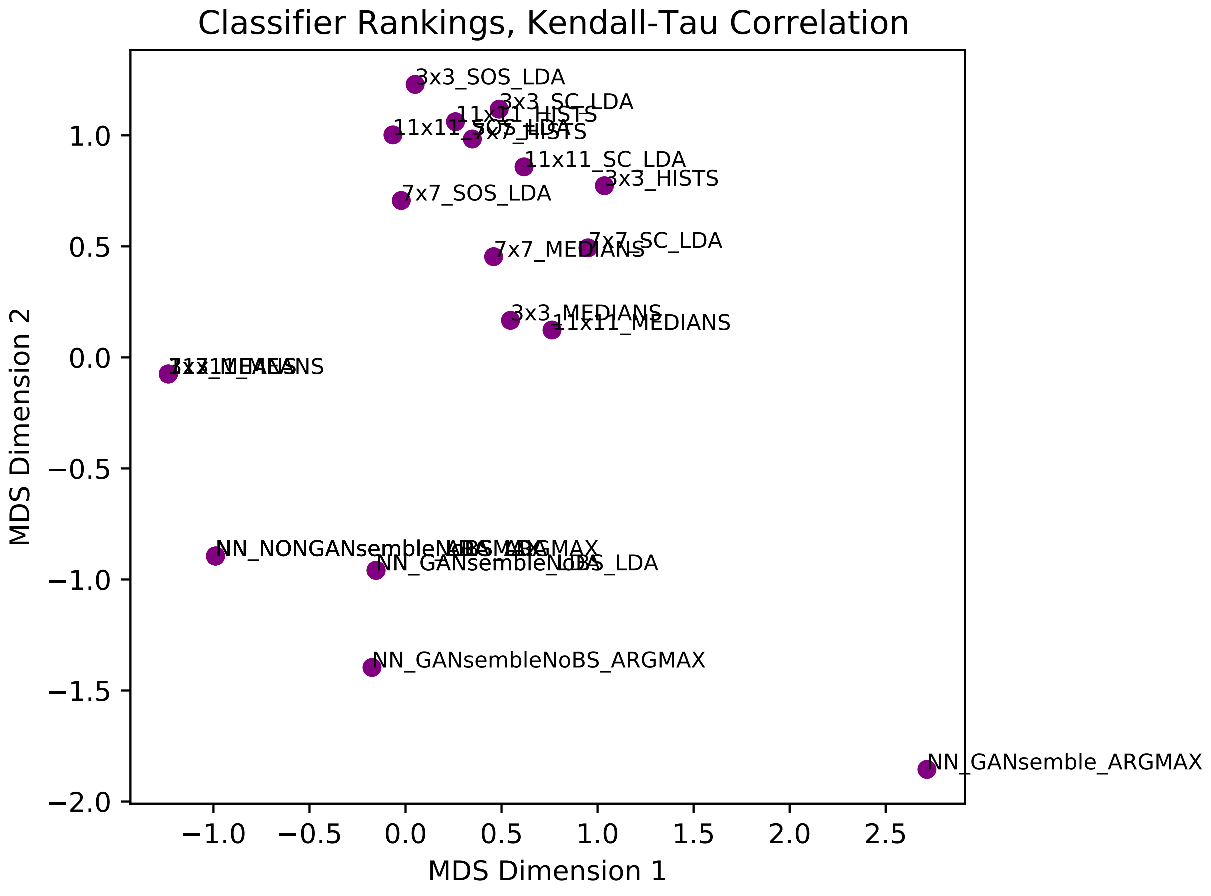


Figure 3.35: Higher-order ranking embedding based on Kendall's tau rank correlation coefficient. The means, other histogram features, and deep convolutional recognizers not based on argmax of the GANsemble form clusters.

3.5 Ease of recognition index composited from primitive and deep convolutional classifier performance rankings

More interesting than being able to cluster the behavior of rankings is the fact that having accumulated a number of differently founded rankings of recognition ease for the same classes freely admits creating a consensus ease ranking composited from the individual rankings.

3.5.1 Combining rankings by the Borda count method

The commonly accepted way to achieve consensus is, of course, to put it to a vote. We can borrow from the surprisingly technical study of election methods and social choice (e.g. [139]) the Borda count. In the context of elections, the Borda count is the prototypical positional voting system (which means that candidates are rank ordered by each voter and seats are allocated first to commonly high rank individuals). Although the Borda count is often described as a method which give voters with an identical ballot a way to express a graded precedence for a list of candidates in common, the voting scenario is substantially identical to the problem of determining an average global ranking from the individual rankings of classifiers.

The Borda count assigns for a ranking an opposite-order number of points. Specifically, a 0-indexed Borda count assigns 0 Borda votes to the candidate ranked last by the rater (the numerically highest rank value), and n votes to the most preferred candidate (the numerically lowest rank value, of highest rank), where n is the number of candidates, with subsequent candidates receiving $n-1, n-2, n-3...$ Borda votes. The consensus winner can be determined by counting who has the most votes. Voting systems are often evaluated based on how they satisfy arbitrarily complex criteria of desirability (e.g. [179]).

The Borda count lacks some desirable criteria, as all voting systems must to some voters, but these mostly are irrelevantly for our purposes restrained to arbitrary distinctions – the Borda count is not guaranteed to elect the majority first-ranked or the Condorcet (plurality of pairwise elections) winner. The Borda count's susceptibility to manipulation in the real world is more relevant. Several facets of this are still irrelevant in our context – the classifiers will not be indulging in strategic voting for candidates they don't support nor will they truncate their ballots by returning partial rankings. But one consideration is semi-relevant: with the Borda count, a political party fielding many, possibly weak candidates is less harmed than in conventional systems (the effect of adding additional candidates of homogeneous appeal (clones) is studied by Tideman [160] in the motivation of his ranked-pairs graph theoretical voting method) and more likely to win some seats and dilute the power of other parties. Note that this is a problem of candidates not a problem of voters.

In a mostly unrelated way, the voter "blocs" formed by our very similar classifiers (e.g. the signchains) may gain extra influence over the composite ranking than they would have as a result of their inclusion; it could be thus a priori reckoned less fair to have the kernel variations included if we expect they induce similar rankings. Along the same lines, the informativeness of the composite ranking could be harmed by how ties are handled, which figures in significantly for the worst and best classifiers that hit floor and ceiling performance. For simplicity, all classifiers previously described have been included, and worse, with assigned ranks artificially made to be unique by doling them out in the case of equality of MCCs in the canonical appearance order of classes (the somewhat lexicographic order split on things and stuff used in the majority of figures). In a refined analysis it might be deemed more appropriate to have the composite rankings only restricted to classifiers which have intermediate performance so that some difference can be discovered, and in "fair" proportion to the family (e.g. one argmax neural network, one filterbank signchains at the best performing kernel size) of classifier.

Table 3.4 nevertheless models the necessary computation within a classifier for the familiar, high-performing primitive recognizer using the LDA classifier on the medians of 11x11 random noise filterbank features. The ranks are assigned with ties going to the first-occurring class (even as it might be fairer to properly tie them), and the table shows classes by ascending MCC. Ascending MCC is associated with increasing rank number (lower rank) and lower numbers of Borda votes.

11x11 LDA MEDIANS

Class	Rank	Borda	MCC
c093	2	30	-0.004161
c032	1	31	-0.004161
theodolite	3	29	0.029075
revolver	4	28	0.044750
blimp	5	27	0.045267
washing-machine	6	26	0.057511
ak47	7	25	0.067089
c129	8	24	0.067851
fire-extinguisher	9	23	0.088940
sextant	10	22	0.095333
c118	11	21	0.097343
pci-card	12	20	0.104857
wine-bottle	13	19	0.119508
calculator	14	18	0.124689
boom-box	15	17	0.141094
treadmill	16	16	0.171279
video-projector	17	15	0.216622
breadmaker	18	14	0.220134
swiss-army-knife	19	13	0.231428
c178	20	12	0.301785
c047	21	11	0.415670
backpack	22	10	0.478174
c184	23	9	0.486231
c191	24	8	0.562411
c163	25	7	0.577606
c160	26	6	0.607829
c049	27	5	0.664514
c066	28	4	0.677994
c003	29	3	0.694437
c159	30	2	0.726441
c045	31	1	1.000000
c089	32	0	1.000000

Table 3.4: Example computation of Borda votes given MCC for the 11x11 medians LDA classifier, with ties in rank going to the first occurring class

Class	Borda Votes	Ease Ranking	Difficulty Ranking	Avg. MCC (across classifiers)
blimp	123.0	32	1	0.314636
fire-extinguisher	182.0	31	2	0.323676
revolver	183.0	30	3	0.322915
boom-box	188.0	29	4	0.332154
ak47	212.0	28	5	0.323538
calculator	217.0	26	6	0.335903
sextant	217.0	27	7	0.339548
theodolite	262.0	25	8	0.337631
treadmill	281.0	24	9	0.341005
washing-machine	284.0	23	10	0.332816
wine-bottle	291.0	22	11	0.333519
pci-card	303.0	21	12	0.365585
swiss-army-knife	323.0	20	13	0.368268
c093	347.0	19	14	0.306469
c047	365.0	18	15	0.336530
c032	377.0	17	16	0.362817
video-projector	378.0	16	17	0.413662
c178	389.0	15	18	0.315585
backpack	395.0	14	19	0.436127
breadmaker	409.0	13	20	0.427023
c129	449.0	12	21	0.356371
c163	480.0	11	22	0.363049
c003	486.0	10	23	0.514773
c184	493.0	9	24	0.366618
c049	499.0	8	25	0.486428
c118	548.0	6	26	0.467162
c160	548.0	7	27	0.521319
c045	551.0	5	28	0.625492
c066	574.0	4	29	0.585853
c159	581.0	3	30	0.505439
c089	595.0	2	31	0.619319
c191	614.0	1	32	0.496561

Table 3.5: Combined Borda Count ease of recognition votes and final composite difficulty ranking based on all classifiers. Note that the average MCC observed per class does not always cohere with the composite ranking.

Once all classifiers have the number of Borda votes per class tabulated, then the number of Borda votes per class can be added across the classifiers, producing the combined Borda vote and composite ranking table reproduced as Table 3.5.

It can be seen that sorting by per-class MCC averaged over classifiers is not equivalent to the Borda-induced ranking. The five hardest classes to classify, according to our index are the blimp, fire extinguisher, revolver, boom-box, and ak47, all Things. The five easiest classes to classify are c191 (one of the brick wall classes), c089 (round river rocks), c159 (mixed aggregate), c066 (streaky stucco), and c045 (quilted dot). c093 and c129, the synthesis failure classes, are among the hardest Stuff to classify. The hardest Stuff to classify was c093 (truncated domes, a failure class), c047 (high contrast brick wall), c032 (a granite-like veiny rock), c178 (grape ivy), and c129 (organic plant "cellular" shapes, a failure class). The 5 easiest things to classify were breadmakers, backpacks, video projectors, swiss army knives, and Peripheral Component Interconnect expansion (PCI) cards. Video projectors, backpacks, and breadmakers were easier to categorize than some Stuff classes. There is not a systematic pattern among the Things which are easier and the Things which are hard, but this is partially reflected in the lack of separation of average MCC which constitutes a sanity check on the rankings (most are around 0.33). Video projectors, backpacks, and breadmakers tend to have highly stereotyped shapes, even more than boom-boxes and washing machines, and this lack of variance may partially account for the ease of recognition. The easiest stuff was notably inorganic, and lacking in the radial, proto-stuff like shapes that characterized the synthesis failure classes.

We can avoid dwelling too long on the appropriateness of adjustments to the Borda count though because the main hypothesis is definitely supported. A full evaluation of these methods would require many more classes to be assessed, and a careful eye taken to balancing the number and diversity of samples. Because of the prohibitive cost in time to train and fully analyze the GANs and NONGANs as was done in Chapter 2, the number of classes was purposely kept small; the objective was not to build a good recognizer or the best continuous evaluator of textural complexity. Certainly with a wider number of sensible classifiers included, this method could make a good estimate of the class-level complexity of textures. It remains to be seen how best to estimate an individual patch level of texture complexity; of course, these methods could be applied to classes consisting of a single patch with varying levels of noise, but they are so expensive that patch-level complexity estimates will eventually have to be regressed from a known library of complex textures.

It is, as a rule though, easier to recognize Stuff than it is to recognize Things, a fact that should be surprising to few but which has been verified in great detail over the last three chapters.

3.5.2 Does composite ease of recognition correlate with our proxy for quality of synthesis?

Our original objective in training the class-specific GANs was to assess recognition as a proxy for synthesis quality. At the end of Chapter 2 we discovered that, at the natural "fair" point of barely-competent for both Things and Stuff in training, this was not practicable because performance reaches a ceiling, or close to it. One possible remedy is to consider the other intuitive point, that at which all the Stuff classes were competent. This was not the road traveled by but since we have the fortune of composite rankings that are informative (even as they include the uninformative rankings in their formulation), the question can be asked of whether the difficulty of synthesis is related to the difficulty of recognition. If it is strongly related, then that constitutes the first real suggestion of a unitary, underlying notion of texture complexity that does not have to be the aggregate of many computational challenges.

We refine that inquiry to ask if the synthesis quality and recognition ease are *straightforwardly* related, which is operationalized in particular as linearly related; quality is invariably judged for lack of a satisfying alternative using the same Inception Score and Fréchet Inception Distance we cast doubt on before in Chapter 2.

Figure 3.36 plots our Borda composite score of recognition ease against the Inception score of the fakes produced by the GANsemble (at its current 15,000 unit reference level of training). The correlation coefficients are calculated as is and then linear regression equations are found separately for Things, Stuff, and all classes following a RANSAC pass as was done for the regressions in Chapter 2. What can be seen is that overall ease may appear to be inversely related to quality! However, when you inspect the Thing and Stuff clouds separately, this pattern very barely holds ($r \approx 0.13$) for Things and is reversed, again very weakly ($r \approx 0.22$) for Stuff – this is because the correlation coefficient calculation is pre-RANSAC. So, especially given our limited sample of classes ($n = 32$), it is hasty to conclude that ease of recognition and quality of synthesis are inversely related even though this is the discovered pattern, if anything.

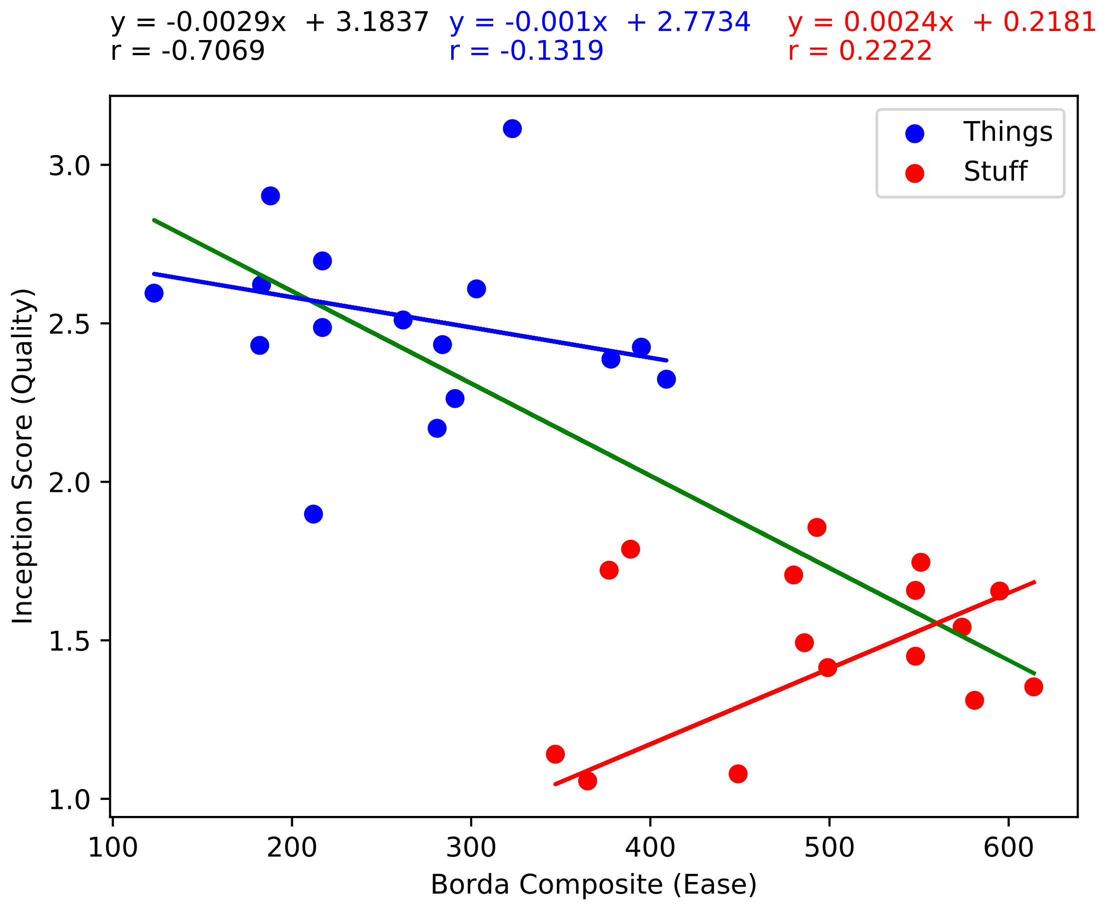


Figure 3.36: Borda votes towards ease of recognition are plotted against Inception Score as a proxy for image quality. The correlation coefficient is calculated and then RANSAC is used *(in an attempt to)* to remove outliers before a line of best fit is determined separately for Things, Stuff and all classes. The combined regression line recalls Simpson's Paradox in that the overriding negative correlation (ease of recognition inversely related to quality) is much stronger.

The reader will object that we have already shown the horrendous bias against Stuff in Inception Score, and note that if we throw Stuff out, the correlation coefficient is very weak. Perhaps FID, which is less subject to the problem of bias because it does not depend on the label distribution at the end of Inceptionv3 but does instead use multidimensional Gaussian distance as estimated from the activations of a previous layer, is a fairer test.

Figure 3.37 reproduces the same regression plot as before, but substituting Inception Score for the *Inverted* Frechet Inception Distance (the FID multiplied by -1). FID is inverted because a large FID between Reals and Fakes for a class is indicative of the Fakes not attaining the quality of reals, with the critical assumption of course, that the fakes have not surpassed the Reals in subjective quality. The need for RANSAC, averted in the last plot, can be seen again with the two classes with anomalously high FID throwing off the pre-RANSAC correlation coefficient for the Stuff. The correlation coefficients of Things and Stuff, post-RANSAC, would be obviously modest.

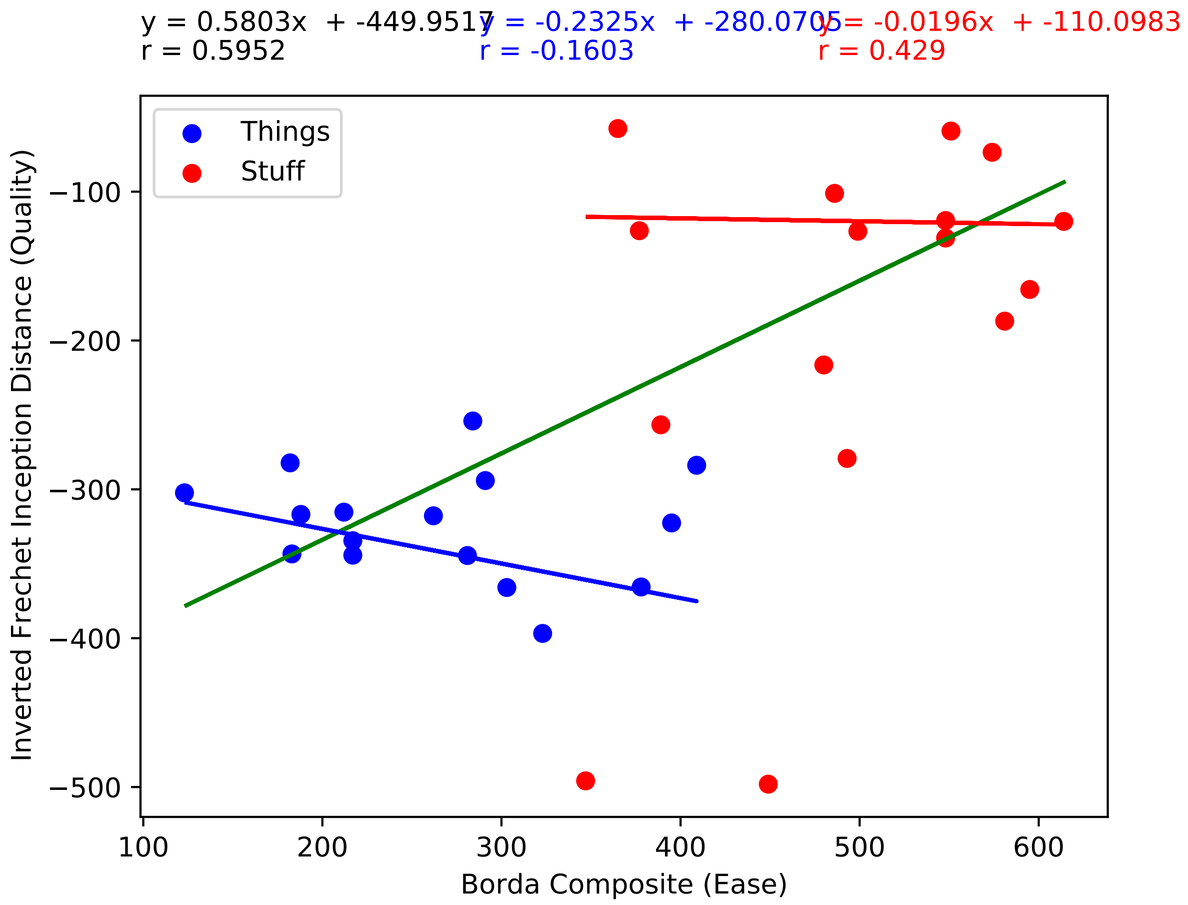


Figure 3.37: Borda votes towards ease of recognition are plotted against *Inverted* (FID is mulitiplied by -1 rather than taking the multiplicative inverse) Fréchet Inception Distance as a proxy for image quality. The correlation coefficient is calculated and then RANSAC is used to remove outliers before a line of best fit is determined separately for Things, Stuff and all classes. For the possibly less biased FID, synthesis quality is positively related to ease of recognition for a class.

The combined correlation coefficient, properly corrected, would be stronger but still not excessively high. However, clearly on an overall basis, the ease of recognition is positively related to quality as measured by (inverted) FID. Because FID, but not Inception Score, actually matches the qualitative synthesis results we observed at the beginning of Chapter 2, it is obviously more credible.

However, since to get a reasonably strong correlation coefficient we have to forget the Things vs. Stuff distinction, this relationship merits more investigation with a more numerous set of classes. Presumptively, the ease of recognition is though *somewhat linearly related* to the quality of synthesis, specifically in the case of partially trained, class-specific WGAN-GPs operating on single-channel image data.

Conclusion

At this point, a preponderance of the available evidence argues that the intuitive and arguably completely obvious distinction between Things and Stuff has functional impacts within and between filterbank based recognizers, whether of primitive or advanced sophistication. That is, there exist detectable systematic differences in the output of (particularly, class-specific) recognizers as well as the constituency of recognizers which are conditional on the class belonging to the very-high-level category of Thing or Stuff. Quality-control analyses currently in widespread use, such as the Inception Score as a proxy for quality, are susceptible to strong training biases that future researchers should be aware of, and the convening of heterogeneous committees of recognizers that include both Things and Stuff must responsibly handle this distinction if they are to be maximally effective. Even though results in these areas are already impressive, more refined knowledge of the distinction could improve advanced recognition and synthesis tasks such as panoptic segmentation and dynamic video inpainting [55]. To fail to address the problem of bias may misdirect the choice of architecture or training tricks in deep-learning type computer vision research generally, because synthesis quality measures are a salient factor used to assess the worthiness of algorithmic and architectural (Marr level 2) innovations current researchers perceive of and propose. Indirectly, this suggests that computational neuroscience, vision science, and cognitive neuroscience researchers studying visual systems should be aware of the possibility that brains at some stage incorporate separate recognition or retrieval regions, representations, processes, or criteria based on whether a patch of the image received from the retina resembles heterogenous (particularly object-suggesting) or homogeneous texture, simply because it appears from this investigation that to maintain some level of segregation would prevent producing some of the unusual situations that result (e.g. the incorrectness of Inception Score, the group promiscuity and susceptibility to synthesis failure outliers of Stuff detectors serving on committees) when measurements of Things and Stuff are presupposed to be directly commensurable.

In Chapter 1, it was shown that it was overwhelmingly significantly easier to recognize Stuff patches as opposed to Thing patches using a small set of sensible histogram statistics of constant random noise filters as features and a classifier admitting a linear decision boundary. The kernel extent of the filters themselves was shown to be far less critical than the choice of statistic. The effect of activation map outliers was significant enough to make mean-derived features almost entirely uninformative in the eyes of the classifier and to elevate even less-than-ordinal information about the shape of the convolutional response distributions above outlier-biased information about the location of the distributions. Less-biased median activation map statistics were shown to perform comparable or favorably to including the entire histogram. This showed that Stuff recognition is in a sense cheaper than Thing recognition in terms of the amount of information needed to perform competent classification. It suggests that not all bits are created equal with respect to information about filter response distributions and that the efficiency and representational precision of features should always be considered alongside the dimensionality of feature vectors. The effect of outliers potentially provides some caution in light of the fact that even modern deep neural networks sometimes involve pooling filter activations to find a maximum ("max pooling"), and this too is a quantity that can be affected by outliers in convolutional features. This investigation also reinforces that shape information – even highly tortured shape information – about filter response distributions can also be discriminative. Since at this point it was possible that the Things vs. Stuff distinction could exist in primitive, historical-type recognizers characteristic of 20th century machine and computer vision and that this distinction might be remedied in the sophisticated deep learning models of today which eschew feature engineering, some investigation into the distinction's survival in these new models had to be undertaken.

In Chapter 2, an ensemble of WGAN-GP networks and a non-synthesis oriented equivalent was developed to represent the more sophisticated, currently realizable class of recognizers, where filterbanks are learned across different scales and through many rounds of stochastic gradient descent. Since after sensible corrections were made, Things and Stuff were categorized about equally well because the incompletely cross-validated accuracy of recognition was at ceiling, this distinction did not exist as clearly. However, in the course of developing and examining the networks to make these sensible corrections, systematic distinctions between Things and Stuff were repeatedly made. For one thing, when considering the pairwise joint affinity of class-specific network activations, Stuff networks were shown to be, as a group, more promiscuous to other classes. This susceptibility remained when examining Stuff recognizer network performance on "out-of-vocabulary" classes from a held-out dataset. For another, when comparing the networks' actual parameter weights in a layerwise fashion using a small number of different notions of dissimilarity or distance, characteristic patterns of Thing vs. Thing and Stuff vs. Stuff comparison separability appeared in specific layers. The location of the dichotomy-evidencing layers changed depending on whether the GAN-origin (generation-focused, trained only on ascertaining Real and Fake for images originating from the target class) or non-GAN-origin (discrimination-focused, trained on the entire set of real and fake images) discriminators were considered, suggesting a developmental difference existing in how "early" or "late" artificial synthetic vision specializes for content as opposed to form. Images of the weights themselves, in some layers, are perceptually different between Things and Stuff as supported by the locally-sensitive MSSIM measurement currently used for ordinary image quality inquiries. On that note, Things and Stuff's divergent behavior is not exclusive to *these particular* deep convolutional recognizers; it can also be seen in famous networks used for standard computation of synthesis quality estimates and feature map visualization, the Inceptionv3 and VGG networks, that they systematically cause different behavior. Based on Inceptionv3, the Inception Score is biased against assigning high quality index to Stuff images as a result of its Thing-intensive training and the alternate Fréchet Inception Distance based on a massively multivariate Gaussian estimated from activations slightly earlier in the network spreads Stuff classes very far from each other as opposed to Thing classes. VGG16's final convolutional layer channel activations show that presented Stuff patches as a rule occupy more activation of the filter channels they do activate, but they activate a smaller proportion of the available channels than Thing patches. In terms of subjective synthesis results, the fakes of the GANsemble studied evidence earlier (in terms of training time) subjectively competent synthesis of Stuff as compared to Things, with the group difference measuring in the hundreds of thousands of image presentations. A transfer learning experiment which retrained the barely competent networks to a Thing or a Stuff target showed that both Thing and Stuff trained networks more easily specialized to the Stuff target. Synthesis failure classes were observed, but only unilaterally for stuff, and it is visually suggested that their synthesis objectives are unfortunately close to early synthesis results for Stuff, or "proto-Stuff". Finally, an explicit, omnibus Things vs. Stuff classifier, derived from the untuned-for-purpose non-GAN-discriminator, was able to exhibit high accuracy on this distinction whether it was trained on held-out classes and tested on the original classes or trained on the original dataset and tested on the held-out classes, suggesting that the Things vs. Stuff distinction (previously called "objectness") is directly sensable by neural networks specialized to look for this distinction in presented texture patches.

A number of collateral results, including the fact that the joint-affinity pattern and member activation vectors of ensembles should be baseline-subtracted for GAN-origin-discriminators to account for the Real-Other-Fake tripolarity (cf. the traditionally-encountered Same/Different bipolarity) were also obtained. The examination of the "GANsemble" vs. the "NON-GANsemble", separately from the motivating Thing vs. Stuff distinction, represents one of the first thorough demonstrations of the survival of general-purpose discriminability in GANs severally and in ensembles *designed for discrimination*, if not actually the first demonstration. Since GAN-origin discriminators and non-GAN-origin discriminators issue systematically different judgments and develop divergently (including in how they treat target-consistent vs. target-inconsistent input with the provision of extra time and in the information as measured by entropy they soak up during training), and the former retain some (admittedly reduced) ability for discrimination and similarity, this may have quite important general implications for deep computer vision, because it suggests NONGAN and GAN network judgments can be combined somehow to glean extra information that may improve knowledge about unknown patches and furthermore suggests that GAN-origin-discriminators could be separately trained on individual classes in total isolation, which potentially leaves the door open to class-level embarassingly parallel training of massive image models if suitable second-stage recognizers (e.g. ensemble consensus functions) can provide similar performance to that observed here and if organizations in custody of those models also enjoy a massive amount of pre-segregated training data and the distributed computing environment to efficiently elicit committee judgments.

In Chapter 3, metric multidimensional scaling was employed to create embeddings representing recognizer features, joint recognizer-on-recognizer activation based “affinities”, and the model parameter (i.e. weight) distances between the deep convolutional recognizers. These embeddings most often visually confirm a Things vs. Stuff dichotomy in those particular senses studied in Chapters 1 and 2. A taxonomy of how Thing and Stuff class or network specific configuration points in the embeddings are separable or strictly encapsulated was arrived by comparing intersection of the convex hulls of Thing and Stuff points. One possible encapsulation class in the taxonomy, the one where all Thing points behaved more like a single Stuff point in apparent convergence, was simply not observed at all, across all three of the groups. Pairwise scaled orthogonal Procrustes superimposition of embeddings and a second round of MDS was used to create higher level embeddings of embeddings (or metaembeddings) that showed how recognizers themselves (both primitive and deep convolutional) were arranged in an induced-low-dimensional space. This confirmed findings about the dissimilarity of the GAN and the NONGAN under baseline normalizations, natural notions about the relatedness of different disparity notions themselves (i.e. SSIM, MSE, the two most-common Minkowski p-norm distances, the Jensen-Shannon distance, and the signchain distance arbitrarily defined in Chapter 1 which is shown here and in Chapter 2 to produce similar but weaker results to the Jensen-Shannon Distance), and the effective kinship of the primitive and deep convolutional recognizer features (i.e. confirming also that signchain and signchain-of-signchain features are similar and that GANsemble and NONGANsemble features do not project analyzed textures as similarly as do mean and median primitive features, even as the effectiveness of these features is differentiated dramatically). Using the notion of a conditional metaembedding which is dependent on calculation incorporating only Things or Stuff classes and the Euclidean distance residuals second-round of Procrustes superimposition yielded "metadistances" that weakly seem to suggest which recognizers and distances are empirically suggested most invariant and least invariant to the Things vs. Stuff distinction, although this process is subject to the accumulation of multiple levels of optimization error and should be viewed with some suspicion. The per-class rankings of the Matthews Correlation Coefficients used to more fairly and stably assess the performance of both the primitive and sophisticated convolutional recognizers in Chapters 1 and 2 as compared to bare accuracy were compared through the use of the Kendall's tau rank correlation coefficient and MDS was used once more to compute embeddings of rankings (that is, of similar high order to the proper metaembeddings) that reproduced the patterns of *results* or performance similarity between the recognizers as seen at the end of each Chapter (e.g. that the naïve argmax classifier behaves very anomalously to all of the other classifiers, performing better on Things given the promiscuity of the Stuff classifiers and the susceptibility of the committee's classification to confident pronouncements made by the networks declaring for synthesis failure classes). The MCC rankings are also combined, using the Borda count method across all of the studied classifiers, to produce one possible composite ranking of recognition difficulty. Class-level ease of recognition was shown to be moderately linearly related to the shown-to-be-fairer of the two gold-standard objective quality indices of synthesis quality, the FID, and almost segregated Things from Stuff in the sense that the most difficult to recognize classes across recognizers were Things, with synthesis failures in Stuff also ranking poorly.

Throughout the dissertation, the peripheral methodology developed to show the Things vs. Stuff distinction, such as metaembeddings, KIS, and the quadrant-based analysis of pairwise affinity matrices of recognizer activation and model parameters will remain generally relevant to the future study of any severe dichotomy in recognizers, not just the objectness dichotomy which was studied. And of course, this dissertation provides an enduring plan for attempting to replicate these findings with more numerous classes, better quality and less problematic image databases, and an expanded view of Stuff as encompassing all views of material, not just homogeneous patches of material from similar but non-identical sources, which was the working definition adopted here. It is reasonable that some of these distinctions may become weaker or may not survive the expansion of the working definition, but it seems likely that at least some of the differences will persist. For simplicity, the studied images were also single-channel, as opposed to the three color-channel images which are ubiquitous because of (admittedly normative, considering the varieties of human color vision anomaly) human trichromacy. Many of the methods that have been described in this dissertation can be straightforwardly adapted for color vision but it is possible that this too could impact the strength of the observed dichotomy.

Based on just this data, the secure Things vs. Stuff distinction at the class-level supports the notion, advanced above, that there may be a general level of processing difficulty (computational disfluency) associated with image categories. To further develop this notion, the methodology developed in this dissertation will have to be turned on a more numerous and less confounded sample of naturalistic image classes, and the notion of computational fluency will likely have to be built out to encompass other fundamental computations of image processing, including retrieval and segmentation, as mentioned during the motivation of the inquiry into recognition itself. Computational disfluency should probably not be forever limited to be founded on difficulty encountered by *artificial* vision systems except to be more easily estimable, and to initially establish a presumptive ordering of classes of texture on disfluency-based complexity – it is possible that through later behavioral experiments of memorability and susceptibility to visual illusion the difficulty subjects encounter in making judgments can be used as sparse data to estimate a *truer and more general* perceptual difficulty ranking of broad classes of texture.

Concerning the generality of these findings, the main threats to validity are the problem of backgroundedness, the problem of unbalanced sample intensity, the problem of insufficient breadth or representativeness of categories, the stability of the the metaembedding process and the WGAN synthesis process, and the infeasibility that was cited as a reason to not fully cross-validate GAN-based recognizer training.

To provide an estimate of the sensitivity of these results in light of each methodological shortcoming:

- **Lack of performance cross-validation through the expensive GAN training process**: this is perhaps one of the less critical objections, because the performance of deep convolutional recognizers was in the case of most of the studied innovations already at ceiling, and Things vs. Stuff differences were emphasized on dimensions other than straight performance in the deep convolutional case; this methodology proved unable to discover that discriminability in deep convolutional recognizer was directly harder for Things. The surprising lower competency for Stuff is impacted somewhat by the presence of the synthesis failure classes in the naïve argmax classifier, so it is difficult to sustain the direct result encountered for the primitive recognizers without hobbling the LDA of the ensemble classifier by introducing noise or some other difficulty. If that particular facet of the investigation was pursued, then the cross-validation objection would become more important. We know that the non-GAN-discriminators as parameterized by architecture and loss function are not totally incompetent because the omnibus Thing vs. Stuff classifier was reasonably competent, trained on entirely held-out *classes*, not just instances.

- **Stability of the WGAN-GP synthesis process**: There was an alternative point of bare competence we could have examined to assess LDA-corrected Thing vs. Stuff raw recognition performance without the introduction of noise or some other post-hoc measure. We could have trained the networks to the earlier-occurring point of bare synthesis competence on Stuff, since this was lower. However, this was not the election (of the two which were possible) that I happened to make, because I was more concerned in showing subjective synthesis results given the important focus on GAN-origin as opposed to non-GAN-origin discriminators. The stability as a problematic factor *per se* of WGAN optimization is a lesser worry because the WGAN-GP process improves so much over the ordinary GAN process in assuring subjective convergence (in every case, of course, excepting the two retained failure classes) and escaping mode collapse. The greater factor is the choice in the amount of training time allotted, which an experimenter would make differently if they had focused on just securely obtaining the recognition difficulty result in Chapter 2 congruent to Chapter 1, as opposed to much more comprehensively piling evidence of various kinds on a Things vs. Stuff dichotomy (read: the intent of this investigation).

- **Insufficient breadth or representativeness of categories**: It is certainly the case that particular categories of Things may be more Stuff like. We could take the example of traditional canvas camping tents, patches of which are large expanses of undifferentiated texture. It is also certainly the case that Stuff textures that are authentically homogeneous may exist which have repeating features so coarse that they are more Thing-like in how they present in the signals studied in this dissertation. This is a difficulty that can only be addressed by using more of the Caltech256 and USPTex datasets, or better, improved datasets without the idiosyncrasies (e.g. cartoon character and 1990s era technology specific classes, overrepresentation of brick walls and featureless texture classes) of these datasets. In this case too, the investigation was limited by the cost in calendar time and expense of training WGANs. All pairwise affinity matrix and embedding operations are also affected by an increased number of classes; for example, the number of entries in a pairwise matrix grows with the square of the number of classes or networks, and the serial calculation of the whole matrix becomes slower and clarity of visualization arguably suffers as a result of including more classes in a first foray.

- **Stability of the metaembedding process**: We saw in Chapter 3 that conclusions from the metaembeddings directly cohered with what was seen in the contributing raw data presented in Chapters 1 and 2, so the error introduced by Procrustes alignment and metric MDS was minimal enough for the size of our dataset that it can be considered as not a factor. However, it should be noted that part of this may have been due to the Monte Carlo -esque protective measure of trying to get the *best-quality* obtainable embedding out of metric MDS by sustaining the cost of many iterations within the algorithm and repetitions of it. The security of the estimate of the baseline used for baseline-subtraction also included this Monte Carlo assumption, that 100 fake samples is enough to capture the *asymptotic* average baseline with high precision, almost certainly an invalid assumption, but necessary to quickly calculate the pairwise matrices. It can be safely recommended that future researchers use at least the number of Monte Carlo iterations of metric MDS and fake generation if they attempt to replicate these results, or extend them to a better-founded dataset. In the latter half of Chapter 3, doubt was cast on the validity of the metadistances. In 2 of the 3 cases, the outer metadistance results (lowest and highest, or the tails of the categorical distributions) were easily explainable, but the fact that not all of the results were explainable urges some caution in faith in metadistance in reflecting true invariance to the Things vs. Stuff distinction after the accumulated optimization error or fundamental instability of two rounds of metric MDS and two rounds of Procrustes superimposition.

- **Problem of backgroundedness**: This is a more severe problem, because this is a subjectively observable confound between metacategories (see Figure 3.38a). Things vary in backgroundedness. Stuff does not. The proper way to address this is to take background-subtracted images and artificially give them false backgrounds as was suggested when the datasets were introduced, since there could be an implicit bias in retaining only the backgrounded Thing images from a dataset. However, in an attempt to be completely accurate to the dataset and more "naturalistic" to the images encountered in computer vision (as opposed to the real world), and a concern that fair false backgrounds would have to themselves be justified, this was not pursued. A future study should examine the robustness of these findings to this factor, however, the Thing vs. Stuff dichotomy was evidenced strongly in many cases in many facets in this investigation, and as an absolute difference of dominance in some cases. While there was substantial per-class variation in Thing backgroundedness, there was often not a close comparison between Things and Stuff that could have tracked the backgroundness proportion on a per-class basis. Presumptively it can then be assumed that this nuisance factor alone does not explain any of the major results. Figure 3.38b regresses Borda composite ranking from backgroundedness and shows a weak negative association between ease and proportion of images backgrounded at best. This would suggest greater backgroundedness of a class leads to decreased ease of recognition, whereas we see that on the whole, Stuff experiences greater ease of recognition. So in some cases, post-hoc regressions could be expected to show that the dichotomy was found *in spite of* the confound, not because of it, as one might expect. This is a position of greater comfort, if the pattern holds across other variables of interest (recall that some of our differences, however, involved subjective categorizations, and aren't subject to regression analysis as a sanity-check).

- **Problem of unbalanced sample intensity**: Arguably, this is the most severe threat to the validity of the observation of dichotomy. There were only 12 samples per class for USPTex and variably more than that for the Caltech256 class. To keep the results directly and straightforwardly replicable by someone with access to the dataset, the classes themselves instead of reasoned or randomized subsets of the Caltech256 classes were used. It was considered that the restrained definition of Stuff classes adopted out of necessity (i.e. the character of USPTex) for the study as truly homogeneous patches from distinct areas of a common source constrains the effective diversity of sample patches anyway, and if there were 100 patches from USPTex as opposed to 12, these would still involve very little change in diversity to make up the gap with the diversity of Stuff classes. This factor is very much a limitation of the dataset and a working definition of truly homogeneous patches and a future investigation should attempt to replicate these results with a considerably more liberal definition of Stuff incorporating more individuated examples of material texture for each class. Since the GANs receive equal amounts of training (recall the unconventional decision to use "units of training" as opposed to epochs) and thus image presentations, that is not so much an issue but there is the question of the impact on the development of the weights within the networks. Some comfort is offered by examining the GAN and NONGAN per-class entropy differences across Things and Stuff. These are not diametrically opposed, suggesting at least that the choice of GAN vs. NONGAN was much more of a factor than Thing vs. Stuff in terms of the theoretically expected compressibility of the neural network weights. Once more, the restrictive definition of Stuff as fully homogeneous in this investigation is something that requires renovation in future study so that the results most generally describe the essence of all Stuff, but it is entirely likely that a whole new dataset will have to be compiled with this objective in mind.

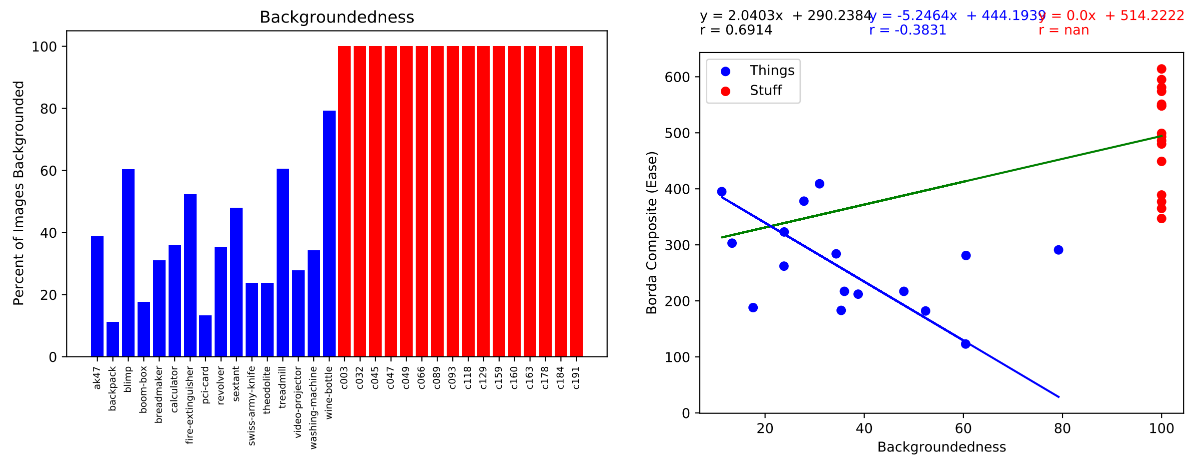


Figure 3.38: a) "Backgroundedness" proportion by class, b) Borda composite index of difficulty regressed as a possible function of backgroundedness. Only Things varied in backgroundedness, although they varied substantially between classes. The united front Things possess in most points of difference highlighted in Chapters 1 and 2 are more systematic than backgroundness being an overwhelming factor would permit (since it varies). However, there are weak correlations of backgroundedness with the terminal/output measurement of Borda composite difficulty.

With the most salient potential limitations on the generality of these findings enumerated, attention briefly turns to the expansion of the notion of computational disfluency to other fundamental operations not previously expanded upon.

One such operation is repair, considered broadly. When a signal becomes degraded during processing or within a memory system, it is possible to attempt to restore the original signal by, for example, retrieving a similar stored signal from a content-addressable memory as a stand-in or applying a learned transformation that is known to attempt to undo a degradation, destructive processing step, or failure of a sensory system to satisfactorily capture additional information. In deep learning research even quite ambitious processes have proven successful: inferred colorization of grayscale images [186], inferred "superresolution" of unsatisfactorily detailed images [172], inferred air-medium photography color and sharpness from underwater images [106], and inferred sound from video sequences with no audio [129].

Pre-deep-learning, research into less venturesome image restoration focused on undoing simple degradations, leading to (chiefly Bayesian) models of deblurring or denoising (e.g., [148]). There is a possible distinction to be drawn between learned methods that require known, paired reference images that are undegraded with methods which heuristically work well and require no paired input data. It is generally difficult to conceptualize looking finely at the disfluency-type complexity of restoration, but in aiming simply to measure relative instance-level or class-level complexity rather than to solve restoration problems practically we are not confined to generalizable methods. Consider a proposal of deblurring and denoising by a program of "evolutionary texture matching", which would use the crossover and mutation-based gradient-free stochastic optimizers of genetic algorithms and evolutionary strategies (see [46] and [36] for good high-level overviews of evolutionary computation more broadly) to directly evolve image pixels as genes with the fitness objective function specified as either the minimization of MSE or the maximization of SSIM. Evolutionary texture matching would be a completely practically useless method for texture repair because the gene-level/image-level solution arrived at by evolutionary processes is not generalizable, tailored completely to the instance and requiring at all times the reference image, but discrete evolutionary time (i.e. generations) would provide a consistent measurement of the difficulty of reaching some SSIM or MSE threshold of quality of coherence with the reference. For this reason, evolutionary texture matching could be used to characterize the relative complexity of repairing different textures or classes of texture. Once an initial degradation of blur, noise, or a mixture of the two was used to create the initial degraded image from a reference, the initial MSE and SSIM could be collected. Classes of image differ significantly on their susceptibility to identical strength degradations (see Appendix A), so initial degradation must be characterized to be factored in later for fair comparisons of rate of repair. From there, the images can be subjected to evolutionary optimization and the classic "best-so-far" curves [36] of evolutionary improvement can be charted. Over a small number of random initializations, the average best-so-far curve can be computed for each instance patch, and these can be averaged by class to produce per-class average best-so-far curves. Then the curves can be divided by their initial degradation values to determine the relative rate of evolutionary improvement by class. It can be assessed whether Things and Stuff remain separable within evolutionary repair rates, or whether there are more and less difficult types of Stuff to repair that, for once, encapsulate Things. This subject deserves its own report, since the choice of evolutionary optimizer may induce different results; evolutionary hyperparameters (refer to [36]) such as mutation rate, mutation type, crossover operator, population size, selection operator (e.g. roulette-wheel or linear stochastic ranking), and population structure (fully-mixed or island model) stand to produce different orderings and perhaps grossly different growth trajectories. This kind of study could even benefit the evolutionary computation literature at large: there are few dependable benchmark functions of intermediate complexity which also are naturalistic – evolutionary methods on one end are regularly evaluated on contrived mathematical test functions generally used to test single-objective optimizers (e.g. the Ackley, Bohachevsky, Easom, Griewank, Rastrigin, Rosenbrock, Bukin, Himmelblau, Styblinski-Tang, and Eggholder functions, see [46], Appendix A.5) and on the other extreme in complicated and experimental deep neuroevolutionary scenarios (e.g. [155]) involving their own novel sources of complexity (i.e. methodological innovations) that complicate interpretation of evolver competency.

Of course, removing stimuli from perceptual consideration also interposes difficulty which could be characterized to rank textures.

The effaceability complexity of stimuli can be characterized in perceptual disappearance phenomenon type visual illusions. One disappearance phenomenon which has long been known is Troxler's fading, but it works only for stimuli which are roughly psychologically equiluminant with the field behind them – there is not the general ability to disappear nearly any patch of a particular size, and so the illusion cannot be used to assess the vast majority of possible imagery. Motion-induced blindness (Bonneh et al. 2001 [15], but originally studied with a physical apparatus by Grindley and Townsend 1966 [69]), is a kind of transient disappearance phenomenon wherein even a highly-salient target stimulus (a bright-colored dot, a photo of an object, a visual texture with high contrast) or group of these, situated in the periphery can disappear from conscious experience if the correct conditions (rough subject fixation in the center of a display, and motion of a distractor field in the background) are met. Induced stimulus suppression of this "blindness" type also arises in binocular rivalry, inattentional blink, visual crowding, the application of TMS, various flash suppression displays, flicker fusion, perceptual filling-in of patches on dynamic noise masks, and various types of change blindness, among other examples (for a fine-grained or comparative survey, see Breitmeyer 2015 [18]). The research amassed on MIB thus far demonstrates that it is a distinct disappearance phenomenon from many others currently known to us (including PFI, CFS, Troxler's fading, CA, BR) and neurophysiological and eye movement evidence conclusively shows (Bonneh et al. 2010 [16], Kloosterman et al. 2015 [100]) that MIB is, rather than possibly being some local adaptation effect happening at the retina, associated with areas beyond early visual processing (i.e. beyond V1, through dorsal and ventral stream areas, and into frontal cortex).

MIB disappearance events are involved with changes in activation across the visual system. Libedinsky, Savage, and Livingstone 2009 [107] provided physiological evidence of MIB in nonhumans: single unit recording in V1 was performed with awake, behaving macaque monkeys. A response anticipatory increase in V1 neural activity was observed (for both actual and "illusory") disappearances. Donner, Sagi, Bonneh, and Heeger 2013 [39] observed a decrease in activity in V4 coincident with disappearance. Nuruki et al. 2013 [128] found that TMS disruption targeted at the rPPC lengthened stimulus disappearance. Disruption targeted at the ventral stream areas (V5/MT) however, shortened stimulus disappearance below baseline and thus decreased disappearance chance. For Scholvinck and Rees 2010 [142], target disappearance was concomitant with an increase in activity in V1 and V2, and also in contralateral MT. Physical target disappearance caused a reduction in activity rather than an increase. Observers with higher GABA concentrations (van Loon et al. 2013 [167]) in various brain areas tended to experience slower perceptual shifts, which in the case of MIB, were the switches between disappearance and appearance. This pattern was not found around DLPFC, perhaps suggesting that MIB emerges most completely before this late decisional stage.

Some information is known about the typical time course of MIB and what stimulus factors impact it. The disappearance of target stimuli is easily experienced by normal observers (though the time course of disappearance is variable from observer-to-observer and from epoch-to-epoch), and it lasts as long as several ($\sim$ 5) seconds (Bonneh et al. 2001 [15]). From this same report, we know that disappearance paradoxically increases with contrast, that disappearance decreases with target size, and with target speed (unlike Troxler's fading, small movements of the target may be tolerated without a reappearance), increases with speed of the distractor field (or mask), and decreases when the target is either too central or too peripheral. Interestingly, we also know from the same report that targets tend to disappear and reappear in groups, when they form good Gestalts. If the distractor field is at too different a perceived depth in a stereoscopic version of the illusion, the disappearance rate lowers (Graf et al. 2002 [66]).

Various theories of MIB, some exotic, have been proposed. The original local attentional load interpretation (Bonneh et al. 2001 [15]) argued that winner-take-all circuits engaged to eliminate stimuli when moving parts of the scene demanded more attentional processing than salient, but unmoving partially-processed entities. A second interpretation focused on the hemispheric asymmetry research (Funk & Pettigrew 2003 [54]) undertaken with MIB and argued that corrective processes in a "pattern-seeking" left hemisphere were trying to factor out inconsistencies in the scene. Related interpretations argued that MIB was a epiphenomenon of an overzealous module for processing occlusion (Graf et al. 2002) or for discounting organic damage or disruption (scotoma) to the retina (New & Scholl 2008 [125]).

Overwhelmingly, research has focused on effacing a common type of simple disappearance target stimulus (small, bright yellow circles with crisp boundaries). Historically, MIB displays vary from study to study depending on which factors are being investigated in the experimental design. However, the most common MIB display (similar to Libedinsky et al. 2009 and Graf et al. 2002) for popularizing the illusion consists of four elements: a plain black background field, a fixation dot, 3 disappearance targets (typically bright yellow circles subtending no more than a very few degrees of visual angle) arranged in a centered and inverted triangle (the two targets on the same height line lie above the center of the display flanking on each size and the third lies below on the center line), and a distractor field of blue crosses undergoing a rigid group rotation at a brisk speed (0.2 Hz). However, you can disappear photographs of faces and texture patches (albeit with somewhat more difficulty, and with an initial desaturation prior to disappearance for the complex stimuli only or "color drain" that I have informally observed).

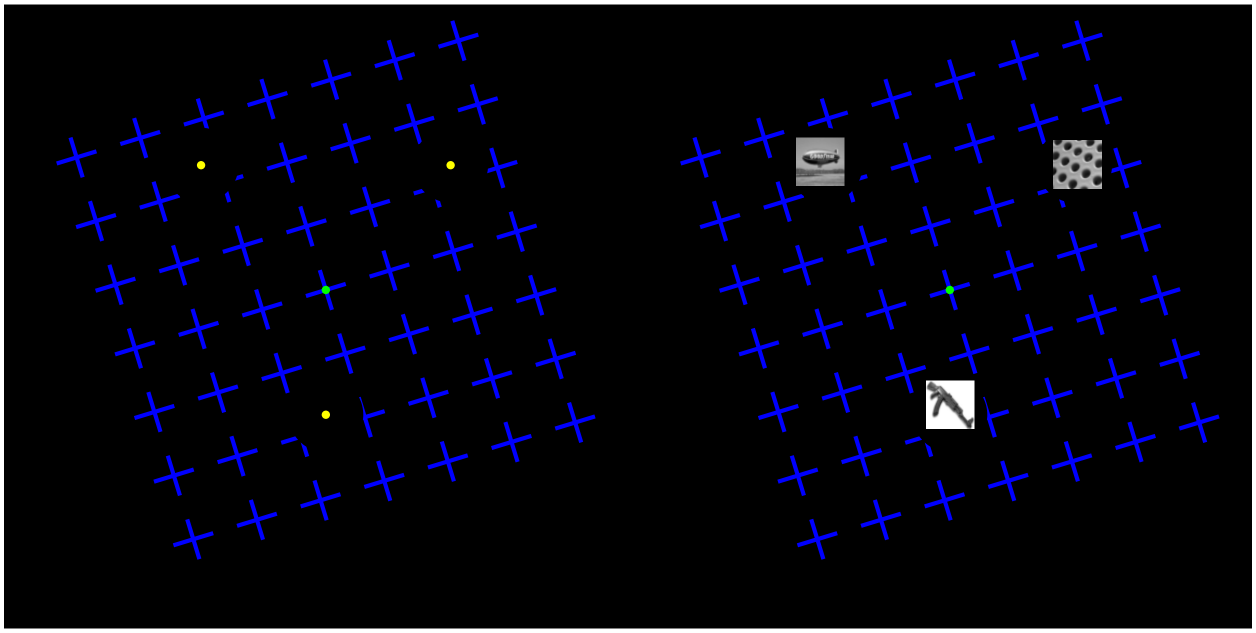


Figure 3.39: The typical motion-induced blindness display is characterized by a fixation dot the subject looks at while peripheral disappearance targets, usually sharply defined bright yellow circles, disappear due to the coherent rotating action of the distractor field of blue crosses. In a potentially interesting Thing vs. Stuff behavioral experiment, Thing and Stuff patches, grayscale as well as color, could be the disappearance targets, or arbitrary images like full-color face photographs.

A simple behavioral experiment could show if there is any consistent difference in the mean time-to-disappearance or time-to-desaturation between Thing-like or Stuff-like patches, potentially offering a new psychophysical source of texture complexity information. The class-level time-to-disappearance for Stuff classes will be expected to be greater than those in the object Things set if MIB disappearance is related to region occupancy (a la channel occupancy in VGG16), and the opposite pattern would be observed if MIB disappearance actually tracked low-level computational disfluency of recognition and synthesis, as measured in this dissertation. This method provides instance-level complexity data, unlike the recognition study. It might also happen that no significant difference would be observed, preventing texture ranking.

Looking beyond computational disfluency as an instance-level or class-level quantity, features themselves can be viewed as exhibiting disfluency-type complexity. For example, the amount of latent space distance or evolutionary time an optimizer must traverse on average in a competent generator (such as Nvidia's StyleGAN2, which currently produces faces [94] sophisticated enough to regularly fool human observers, via e.g. thispersondoesnotexist.com, using data from the Flickr-Faces-HQ dataset and exploration and training encompassing an estimated "51 single-GPU years" (!) of computing with commercial-grade GPUs) to fool a binary discriminator that a high-level feature has flipped could be used to characterize the complexity of the high-level indecomposable features of apparent male/female sex, high/low facial attractiveness, or high/low trustworthiness. It is possible that the complexity of trustworthiness is higher along these lines than the complexity of gender. Characterizing the complexity of high-level, psychologically relevant features themselves is an exciting prospect for cognitive science and experimental psychology because it could be eventually used to predict the difficulty of discovering effects in human behavioral studies, or the probability of producing quality synthetic but naturalistic stimuli for those studies.

It seems reasonable that difficulty of visual processing within, for example, the Things metacategory may found to be related to the sort of subjective feature dimensions (such as sharp angular features or large circular cavities) previously described. Across domains, especially those domains that are not simple visual search, difficulty seems more likely to be related to the presence of problematic critical objects and the conjunction of these presences, rather than the number of objects to process or the improbability of a configuration – that is, problems might profitably be thought of as having implicit, unknown feature vectors that merit analysis using the type of fuzzy logic similarity computations discussed throughout Chapter 2 and 3. This is perhaps best illustrated explicitly in cognitive science in the subfield of human and machine problem solving. A good example is the innovative Virtual Tools domain recently proposed by Allen, Smith, and Tenenbaum [6], which studies problem solving directly in a game that has human and reinforcement learning agent participants drop one of several "tools" into a highly simplified world of physically-simulated objects in an attempt to launch or support a ball-shaped object into a goal region. The ingenious element of the formulation of the challenge is the constraint of using provided tools rather than designed tools, and the only allowed action being the initial placement of the tool into the world with gravity and collision taking over thereafter – this makes the priors of Bayesian agents incredibly easy to characterize because the tool use strategies over time resemble Gaussians of tool position uncertainty conditioned on tool index. The Tools challenge levels vary substantially in difficulty as measured by completion rate in twenty attempts, with the very hardest levels involving dropping objects so that their edges effect a catapult, raze a tower, or form a bridge just right. There is substantial variability in difficulty which interacts with the central obstacle of each level, such as whether you are trying to topple the top block of a pyramid or the base block or whether you are trying to authentically create a bridge or create an unstable bridge that allows the ball to get below it, but the gross features are largely indicative of level difficulty.

Analogous to the problem of visual recognizers of different complexity and formulation having trouble on different patches, and therefore fair difficulty indices needing to incorporate *consensus* difficulty, in the Virtual Tools challenge, humans and DQN agents as well as various flavors of Allen et al.'s bespoke simulation-incorporating hypothesis-focused SSUP ("Sample, Simulate, Update") agent diverge in performance on some levels and do not on others. In a level called `Prevention A`, you have to arrest a falling bar by interposing a thin platform so that the falling bar does not fall directly into the goal region before the ball moves down an incline. Humans do much better on this level, perhaps because they have accumulated intuitive theory notions of inclined planes vs. straight drops and the ability to block falls that the somewhat clever experimentalist SSUP model has to discover and the relatively-more-exhaustive empiricist Deep Q Network "discovers" only by accident. Another human-advantaged level involves a tool placement exclusion region that forces you to trigger a chain reaction of dropping your tool to create a cascade of falling balls, which is obvious from context. A level all agents do well at involves unlidding a box enough to permit a ball in because the challenge is highly multiply realizable (you can hit either the center or the edge of the box with all of the different tools) and a rarer example of a puzzle humans are inferior at is `Falling A`, where the RL agents probably improve over humans because humans use their tools above or near a falling bucket rather than destabilizing the bucket by balancing it poorly on a ball, revealing functional fixedness. Difficulty is necessarily best estimated as a fuzzy composite across multiple dimensions: strategies, distinct perceivers, the manifest presence of generally problematic obstacle classes, and the specific, sometimes accidental conjunction of features which may make the obstacles much more effective than they usually are. Perhaps unusually, as for puzzle game levels, so for visual texture patches, in this particular respect.

In insisting on a notion of complexity that is not founded in the absolute improbability of scene generation but in the relative improbability of easy processing of those scenes by *actually realized* perceivers, a more difficult challenge is being taken on. The Thing vs. Stuff dichotomy suggests that "objectness" strongly loads on the early "principal components" of true visual complexity measurement, even if it does actually turn out that there is no clean, unitary difficulty index completely determining the resource-intensiveness of *prima facie* (but not actually) disparate processes like recognition, synthesis, retreival, and segmentation. But this is presumably just one conspicuously unsubtle determiner of this kind of complexity.

Estimating difficulty is not an easy pursuit, but it is a *worthy* pursuit. The potential importance of *deep* estimation of visual difficulty handily transcends being able to predict the training time of classes to efficiently budget beforehand the provision of computing resources for industrial enterprises or to most safely arrange hierarchical or heterarchical committees of recognizers on-demand from an available distributed pool. In education and communications, crude estimates of readability (e.g. the Gunning-Fog and Flesch-Kincaid indices [97]) have long been used to drive design of instruction materials. Once they were in sufficiently wide use, some large number of technical writers were forced, wisely or unwisely, to take notice of them and to accept constraint by these "standards" in their work product, despite the arguably shallow founding of these indices. When mid-level features of complexity can be dependably used by artificial vision systems to estimate affective reactions to environments, the design and ordering of the environments themselves will change to suit the perceptual bottlenecks and processing quirks of perceivers. This goes beyond simple estimates of local saliency, and sort of considers the inverse scenewide map which opposes saliency.

Future vision systems with the ability to predict the onset of difficulty, whether from low-level indices of visual processing or by direct discriminative models of different varieties of difficulty or consternation, could be used to better understand from street-level imagery the tentativeness or risk-taking behavior of drivers traversing unfamiliar roadways, the annoyance of clothing store customers which prevents them from browsing through an unorganized display where disparate colors, styles, and types of clothing have been arranged pell-mell, the level of distraction to educational pursuits interposed by the physical disorganization of a home, or the level of wonder experienced the first time one walks through a new, detailed, and exotic virtual reality world.

At that late point, the environmental evaluation capabilities of artificial vision systems could be reckoned to be much more closely matched to those of natural vision systems than even we observe today with networks that can already imagine scenes, and better integrated with the apperceptual mechanisms of the brains that back them.

Bibliography

[1] E. H. Adelson. On seeing stuff: the perception of materials by humans and machines. In *Human vision and electronic imaging VI*, volume 4299, pages 1–12. International Society for Optics and Photonics, 2001.

[2] E. H. Adelson, J. R. Bergen, et al. *The plenoptic function and the elements of early vision*, volume 2. Vision and Modeling Group, Media Laboratory, Massachusetts Institute of Technology, 1991.

[3] T. Akenine-Moller, E. Haines, and N. Hoffman. *Real-Time Rendering.* A. K. Peters, Ltd., USA, 3rd edition, 2008.

[4] C. O. Alexander Mordvintsev and M. Tyka. Deepdream - a code example for visualizing neural networks, July 2015. https://ai.googleblog.com/2015/07/deepdream-code-example-for-visualizing.html.

[5] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2189–2202, 2012.

[6] K. R. Allen, K. A. Smith, and J. B. Tenenbaum. Rapid trial-and-error learning with simulation supports flexible tool use and physical reasoning. arXiv preprint arXiv:1907.09620, 2020.

[7] P. Arbelez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.

[8] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. arXiv 1701.07875, 2017.

[9] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016.

[10] A. R. Backes, D. Casanova, and O. M. Bruno. Color texture analysis based on fractal descriptors. *Pattern Recognition*, 45(5):1984–1992, 2012. http://fractal.ifsc.usp.br/dataset/USPtex.php.

[11] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

[12] J. Bevins. libnoise: glossary. `http://libnoise.sourceforge.net/glossary/index.html`, 2005.

[13] S. Bhatia and R. Dahyot. Using wgan for improving imbalanced classification performance. In *AICS*, pages 365–375, 2019.

[14] H. Blum et al. *A transformation for extracting new descriptors of shape*, volume 4. MIT press Cambridge, 1967.

[15] Y. S. Bonneh, A. Cooperman, and D. Sagi. Motion-induced blindness in normal observers. *Nature*, 411(6839):798, 2001.

[16] Y. S. Bonneh, T. H. Donner, D. Sagi, M. Fried, A. Cooperman, D. J. Heeger, and A. Arieli. Motion-induced blindness and microsaccades: cause and effect. *Journal of vision*, 10(14):22–22, 2010.

[17] S. Bozinovski. Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica*, 44(3), 2020.

[18] B. G. Breitmeyer. Psychophysical blinding methods reveal a functional hierarchy of unconscious visual processing. *Consciousness and Cognition*, 35:234–250, 2015.

[19] J. Brownlee. How to implement the frechet inception distance (fid) for evaluating gans, Oct 2019.

[20] J. Brownlee. How to implement the inception score for evaluating gans, Oct 2019.

[21] G. J. Burghouts and J.-M. Geusebroek. Material-specific adaptation of color invariant features. *Pattern Recognition Letters*, 30(3):306 – 313, 2009.

[22] P. Burt and E. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on communications*, 31(4):532–540, 1983.

[23] C. S. Calude, E. Calude, and M. Dinneen. A new measure of the difficulty of problems. Technical report, Department of Computer Science, The University of Auckland, New Zealand, 2006.

[24] F. W. Campbell and J. G. Robson. Application of fourier analysis to the visibility of gratings. *The Journal of physiology*, 197(3):551, 1968.

[25] J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[26] M. Carandini and D. J. Heeger. Normalization as a canonical neural computation. *Nature Reviews Neuroscience*, 13(1):51–62, 2012.

[27] G. J. Chaitin. Information-theoretic limitations of formal systems. *Journal of the ACM (JACM)*, 21(3):403–424, 1974.

[28] D. Chicco and G. Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):6, 2020.

[29] F. Chollet et al. Keras. `https://keras.io`, 2015.

[30] J. Coates and D. Bollegala. Frustratingly easy meta-embedding – computing meta-embeddings by averaging source word embeddings. arXiv preprint arXiv:1804.05262, 2018.

[31] J. Cohen. *Statistical power analysis for the behavioral sciences.* Lawrence Erlbaum Associates, 1988.

[32] T. M. Cover and J. A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing).* Wiley-Interscience, USA, 2006.

[33] C. Cusano, P. Napoletano, and R. Schettini. Evaluating color texture descriptors under large variations of controlled lighting conditions. *JOSA A*, 33(1):17–30, 2016.

[34] K. J. Dana, B. Van Ginneken, S. K. Nayar, and J. J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions On Graphics (TOG)*, 18(1):1–34, 1999.

[35] E. R. Davies. *Computer and machine vision: theory, algorithms, practicalities.* Academic Press, 2012.

[36] K. A. De Jong. *Evolutionary computation: a unified approach.* MIT press, 2006.

[37] J. De Leeuw and P. Mair. Multidimensional scaling using majorization: Smacof in r. 2011.

[38] E. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. arXiv preprint arXiv:1506.05751, 2015.

[39] T. H. Donner, D. Sagi, Y. S. Bonneh, and D. J. Heeger. Retinotopic patterns of correlated fluctuations in visual cortex reflect the dynamics of spontaneous perceptual suppression. *Journal of Neuroscience*, 33(5):2188–2198, 2013.

[40] M. Dorigo. Optimization, learning and natural algorithms. *PhD Thesis, Politecnico di Milano*, 1992.

[41] R. Dosselmann, X. D. Yang, et al. *A formal assessment of the structural similarity index.* University of Regina. Department of Computer Science, 2008.

[42] D. Dowson and B. Landau. The fréchet distance between multivariate normal distributions. *Journal of multivariate analysis*, 12(3):450–455, 1982.

[43] V. Dumoulin and F. Visin. A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285, 2018.

[44] B. Efron. Nonparametric estimates of standard error: The jackknife, the bootstrap and other methods. *Biometrika*, 68(3):589–599, 1981.

[45] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038. IEEE, 1999.

[46] A. P. Engelbrecht. *Computational intelligence: an introduction.* John Wiley & Sons, 2007.

[47] R. Epstein and N. Kanwisher. A cortical representation of the local visual environment. *Nature*, 392(6676):598–601, 1998.

[48] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2):298–305, 1973.

[49] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[50] D. Freedman and P. Diaconis. On the histogram as a density estimator: L 2 theory. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 57(4):453–476, 1981.

[51] W. T. Freeman, E. H. Adelson, et al. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906, 1991.

[52] T. M. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991.

[53] K. Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130, 1988.

[54] A. P. Funk and J. D. Pettigrew. Does interhemispheric competition mediate motion-induced blindness? a transcranial magnetic stimulation study. *Perception*, 32(11):1328–1338, 2003.

[55] C. Gao, A. Saraf, J.-B. Huang, and J. Kopf. Flow-edge guided video completion. In *Proc. European Conference on Computer Vision (ECCV)*, 2020.

[56] L. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Advances in neural information processing systems*, pages 262–270, 2015.

[57] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. arXiv preprint arXiv:1811.12231, 2019.

[58] E. Giacomello, P. L. Lanzi, and D. Loiacono. Doom level generation using generative adversarial networks. arXiv preprint arXiv:1804.09154, 2018.

[59] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv preprint arXiv:1311.2524, 2014.

[60] J. Glenn and D. Binkley. An investigation of hierarchical bit vectors. *New Topics in Theoretical Computer Science*, page 143, 2008.

[61] I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. arXiv preprint arXiv:1701.00160, 2017.

[62] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*, volume 1. MIT Press, 2016.

[63] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[64] M. T. Goodrich, J. S. Mitchell, and M. W. Orletsky. Approximate geometric pattern matching under rigid motions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4):371–379, 1999.

[65] J. C. Gower. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51, 1975.

[66] E. W. Graf, W. J. Adams, and M. Lages. Modulating motion-induced blindness with depth ordering and surface completion. *Vision research*, 42(25):2731–2735, 2002.

[67] B. F. Green. The orthogonal approximation of an oblique structure in factor analysis. *Psychometrika*, 17(4):429–440, 1952.

[68] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007. http://www.vision.caltech.edu/Image_Datasets/Caltech256.

[69] G. Grindley and V. Townsend. Further experiments on movement masking. *The quarterly journal of experimental psychology*, 18(4):319–326, 1966.

[70] K. Groulx, C. Chubb, J. D. Victor, and M. M. Conte. The features that control discrimination of an isodipole texture pair. *Vision research*, 158:208–220, 2019.

[71] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. arXiv preprint arXiv:1704.00028, 2017.

[72] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught, and J. Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[73] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

[74] D. J. Heeger and J. R. Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238, 1995.

[75] J. Hegdé and D. C. Van Essen. Selectivity for complex shapes in primate visual area v2. *Journal of Neuroscience*, 20(5), 2000.

[76] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. arXiv preprint arXiv:1706.08500, 2018.

[77] G. Hinton, N. Srivastava, and K. Swersky. Neural networks for machine learning lecture 6e rmsprop. 14(8), 2012.

[78] D. D. Hoffman and W. A. Richards. Parts of recognition. *Cognition*, 18(1-3):65–96, 1984.

[79] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[80] B. K. Horn. The binford-horn line-finder. `https://dspace.mit.edu/handle/1721.1/5795`, 1973.

[81] D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574, 1959.

[82] J. R. Hurley and R. B. Cattell. The procrustes program: Producing direct rotation to test a hypothesized factor structure. *Behavioral science*, 7(2):258, 1962.

[83] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

[84] P. Isola, J. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976, 2017.

[85] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson. Crisp boundary detection using pointwise mutual information. In *ECCV*, 2014.

[86] ITU Radiocommunication Assembly. Characteristics of composite video signals for conventional analogue television systems. 2005.

[87] R. Jacobs, E. Baumgartner, and K. Gegenfurtner. The representation of material categories in the brain. *Frontiers in Psychology*, 5:146, 2014.

[88] M. K. Johnson and E. H. Adelson. Retrographic sensing for the measurement of surface texture and shape. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1070–1077. IEEE, 2009.

[89] D. Jurafsky and J. H. Martin. Speech and language processing (3rd ed. draft). October 2019. https://web.stanford.edu/ jurafsky/slp3/5.pdf.

[90] P. Kanerva. *Sparse distributed memory*. MIT press, 1988.

[91] N. Kanwisher, J. McDermott, and M. M. Chun. The fusiform face area: a module in human extrastriate cortex specialized for face perception. *Journal of neuroscience*, 17(11):4302–4311, 1997.

[92] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196, 2018.

[93] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. arXiv preprint arXiv:1812.04948, 2019.

[94] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of stylegan, 2020.

[95] P. J. Kellman and M. H. Cohen. Kinetic subjective contours. *Perception & Psychophysics*, 35(3):237–244, 1984.

[96] A. Khosla, A. S. Raju, A. Torralba, and A. Oliva. Understanding and predicting image memorability at a large scale. In *International Conference on Computer Vision (ICCV)*, 2015.

[97] J. P. Kincaid, R. P. Fishburne Jr, R. L. Rogers, and B. S. Chissom. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch, 1975.

[98] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[99] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic segmentation. arXiv preprint arXiv:1801.00868, 2019.

[100] N. A. Kloosterman, T. Meindertsma, A. M. van Loon, V. A. Lamme, Y. S. Bonneh, and T. H. Donner. Pupil size tracks perceptual content and surprise. *European Journal of Neuroscience*, 41(8):1068–1078, 2015.

[101] E. Koutsofios and S. C. North. Drawing graphs with dot, 1996.

[102] G. Kylberg. *Kylberg Texture Dataset v. 1.0.* Centre for Image Analysis, Swedish University of Agricultural Sciences, 2011.

[103] J. D. Leeuw, I. J. R. Barra, F. Brodeau, G. Romier, and B. V. C. (eds. Applications of convex analysis to multidimensional scaling. In *Recent Developments in Statistics*, pages 133–146. North Holland Publishing Company, 1977.

[104] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International journal of computer vision*, 43(1):29–44, 2001.

[105] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766, 2013.

[106] J. Li, K. A. Skinner, R. M. Eustice, and M. Johnson-Roberson. Watergan: Unsupervised generative network to enable real-time color correction of monocular underwater images. *IEEE Robotics and Automation Letters*, page 11, 2017.

[107] C. Libedinsky, T. Savage, and M. Livingstone. Perceptual and physiological evidence for a role for early visual areas in motion-induced blindness. *Journal of Vision*, 9(1):14–14, 2009.

[108] L. Luzi, R. Balestriero, and R. G. Baraniuk. Ensembles of generative adversarial networks for disconnected data. 2020.

[109] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3, 2013.

[110] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[111] J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *J. Opt. Soc. Am. A*, 7(5):923–932, May 1990.

[112] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.

[113] D. Marr. Early processing of visual information. *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, 275(942):483–519, 1976.

[114] D. Marr. Vision: A computational investigation into the human representation and processing of visual information, 1982.

[115] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980.

[116] D. Marr and T. Poggio. Ai memo 357: From understanding computation to understanding neural circuitry.

[117] J. L. McClelland, D. E. Rumelhart, P. R. Group, et al. Parallel distributed processing. *Explorations in the Microstructure of Cognition*, 2:216–271, 1986.

[118] T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013.

[119] K. D. Miller and D. J. MacKay. The role of constraints in hebbian learning. *Neural computation*, 6(1):100–126, 1994.

[120] MIT Vision and Modeling Group. Mit vistex. `http://vismod.media.mit.edu/vismod/imagery/VisionTexture/vistex.html`, 1995.

[121] B. Mohar. Isoperimetric numbers of graphs. *Journal of combinatorial theory, Series B*, 47(3):274–291, 1989.

[122] K. P. Murphy. *Machine learning: a probabilistic perspective.* MIT press, 2012.

[123] V. Nagarajan, C. Raffel, and I. J. Goodfellow. Theoretical insights into memorization in gans. In *Neural Information Processing Systems Workshop*, 2018.

[124] A. K. Nain. Implementation of wasserstein gan with gradient penalty, May 2020. https://keras.io/examples/generative/wgan_gp.

[125] J. J. New and B. J. Scholl. perceptual scotomas a functional account of motion-induced blindness. *Psychological Science*, 19(7):653–659, 2008.

[126] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in neural information processing systems*, pages 3387–3395, 2016.

[127] J. Nilsson and T. Akenine-Mller. Understanding ssim. arXiv preprint arXiv:2006.13846, 2020.

[128] A. Nuruki, R. Oliver, G. Campana, V. Walsh, and J. C. Rothwell. Opposing roles of sensory and parietal cortices in awareness in a bistable motion illusion. *Neuropsychologia*, 51(13):2479–2484, 2013.

[129] A. Owens, P. Isola, J. McDermott, A. Torralba, E. H. Adelson, and W. T. Freeman. Visually indicated sounds. arXiv preprint arXiv:1512.08512, 2016.

[130] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[131] B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

[132] J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision*, 40(1):49–70, 2000.

[133] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv 1511.06434, 2016.

[134] H. Ramsauer, B. Schfl, J. Lehner, P. Seidl, M. Widrich, L. Gruber, M. Holzleitner, M. Pavlovi, G. K. Sandve, V. Greiff, D. Kreil, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter. Hopfield networks is all you need. arXiv preprint arXiv:2008.02217, 2020.

[135] W. Richards. Quantifying sensory channels: generalizing colorimetry to orientation and texture, touch, and tones. *Sensory Processes*, 3(3):207–229, 1979.

[136] I. Rock. Anorthoscopic perception. *Scientific American*, 244(3):145–153, 1981.

[137] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[138] N. C. Rust and J. J. DiCarlo. Selectivity and tolerance (invariance) both increase as visual information propagates from cortical area v4 to it. *Journal of Neuroscience*, 30(39):12978–12995, 2010.

[139] D. G. Saari. Mathematical structure of voting paradoxes. *Economic Theory*, 15(1):1–53, 2000.

[140] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[141] F. Sanglard. *Game Engine Black Book: DOOM.* CreateSpace Independent Publishing Platform, 12 2018.

[142] M. L. Schölvinck and G. Rees. Neural correlates of motion-induced blindness in the human brain. *Journal of Cognitive Neuroscience*, 22(6):1235–1243, 2010.

[143] P. H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[144] C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[145] L. Sharan, R. Rosenholtz, and E. H. Adelson. Accuracy and speed of material categorization in real-world images. *Journal of Vision*, 14(10), 2014.

[146] M. F. Shlesinger, J. Klafter, and G. Zumofen. Above, below and beyond brownian motion. *American Journal of Physics*, 67(12):1253–1259, 1999.

[147] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[148] E. P. Simoncelli. Bayesian denoising of visual images in the wavelet domain. In *Bayesian inference in wavelet-based models*, pages 291–308. Springer, 1999.

[149] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034, 2014.

[150] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[151] H. Stanislaw and N. Todorov. Calculation of signal detection theory measures. *Behavior research methods, instruments, & computers*, 31(1):137–149, 1999.

[152] K. D. Stephens and D. D. Hoffman. On visual texture preference: Can an ecological model explain why people like some textures more than others? *Perception*, 45(5):527–551, 2016.

[153] R. A. Stepien. New method for analysis of nonstationary signals. *Nonlinear biomedical physics*, 5(1):3, 2011.

[154] C. Stringer, M. Pachitariu, N. Steinmetz, M. Carandini, and K. D. Harris. High-dimensional geometry of population responses in visual cortex. *Nature*, 571(7765):361–365, 2019.

[155] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arXiv preprint arXiv:1712.06567, 2018.

[156] P. Sun, C. Chubb, C. E. Wright, and G. Sperling. The centroid paradigm: Quantifying feature-based attention in terms of attention filters. *Attention, Perception, & Psychophysics*, 78(2):474–515, 2016.

[157] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[158] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[159] A. Tero, S. Takagi, T. Saigusa, K. Ito, D. P. Bebber, M. D. Fricker, K. Yumiki, R. Kobayashi, and T. Nakagaki. Rules for biologically inspired adaptive network design. *Science*, 327(5964):439–442, 2010.

[160] T. N. Tideman. Independence of clones as a criterion for voting rules. *Social Choice and Welfare*, 4(3):185–206, 1987.

[161] I. Tolstikhin, S. Gelly, O. Bousquet, C.-J. Simon-Gabriel, and B. Schlkopf. Adagan: Boosting generative models, 2017.

[162] S. Ullman. Visual routines. `https://apps.dtic.mil/dtic/tr/fulltext/u2/a133634.pdf`, 1983.

[163] I. Ustyuzhaninov, W. Brendel, L. A. Gatys, and M. Bethge. Texture synthesis using shallow convolutional networks with random filters. 2016.

[164] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014.

[165] J. van Doorn, A. Ly, M. Marsman, and E.-J. Wagenmakers. Bayesian rank-based hypothesis testing for the rank sum test, the signed rank test, and spearman's . *Journal of Applied Statistics*, 47(16):2984–3006, 2020.

[166] P. van Emde Boas. Preserving order in a forest in less than logarithmic time and linear space. *Information processing letters*, 6(3):80–82, 1977.

[167] A. M. van Loon, T. Knapen, H. S. Scholte, E. S. John-Saaltink, T. H. Donner, and V. A. Lamme. Gaba shapes the dynamics of bistable perception. *Current Biology*, 23(9):823–827, 2013.

[168] M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–691. IEEE, 2003.

[169] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *International journal of computer vision*, 62(1-2):61–81, 2005.

[170] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[171] V. Volz, J. Schrum, J. Liu, S. M. Lucas, A. Smith, and S. Risi. Evolving mario levels in the latent space of a deep convolutional generative adversarial network. arXiv preprint arXiv:1805.00728, 2018.

[172] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, C. C. Loy, Y. Qiao, and X. Tang. Esrgan: Enhanced super-resolution generative adversarial networks. arXiv preprint arXiv:1809.00219, 2018.

[173] Y. Wang, L. Zhang, and J. Van De Weijer. Ensembles of generative adversarial networks. 2016.

[174] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[175] H. S. Warren. *Hacker's Delight*. Pearson Education, 2nd edition, 2013.

[176] J. S. Weszka, C. R. Dyer, and A. Rosenfeld. A comparative study of texture measures for terrain classification. *IEEE transactions on Systems, Man, and Cybernetics*, (4):269–285, 1976.

[177] F. Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.

[178] A. Witkin. Scale-space filtering: A new approach to multi-scale description. In *ICASSP '84. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 9, pages 150–153, 1984.

[179] D. R. Woodall. Monotonicity of single-seat preferential election rules. *Discrete Applied Mathematics*, 77(1):81–98, 1997.

[180] C. B. Yann LeCun, Corinna Cortes. mnist. `http://yann.lecun.com/exdb/mnist/`, 1998.

[181] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. arXiv preprint arXiv:1506.06579, 2015.

[182] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[183] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. arXiv preprint arXiv:1311.2901, 2013.

[184] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pages 2528–2535. IEEE, 2010.

[185] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915, 2017.

[186] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. arXiv preprint arXiv:1603.08511, 2016.

[187] S. Zhou and R. J. Mondragón. The rich-club phenomenon in the internet topology. *IEEE Communications Letters*, 8(3):180–182, 2004.

[188] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

Appendix A

Thing and Stuff Susceptibility to Blur and Noise Degradation As Perceived by MSE and SSIM Measurements of Quality Preservation

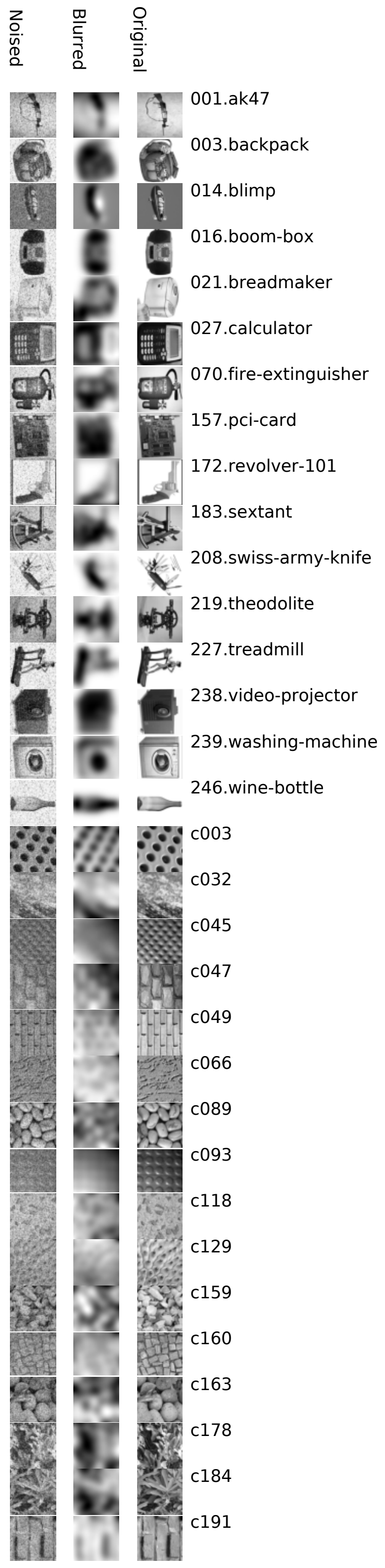


Figure A.1: Sample images from each class and their degraded (blurred and noised) equivalents. Gaussian blur is $\sigma > 1$, Gaussian noise has a mean of 1 with variance of 0.5, and Poisson, salt and pepper, and Speckle noise with a constant seed are added to create a further distorted image.

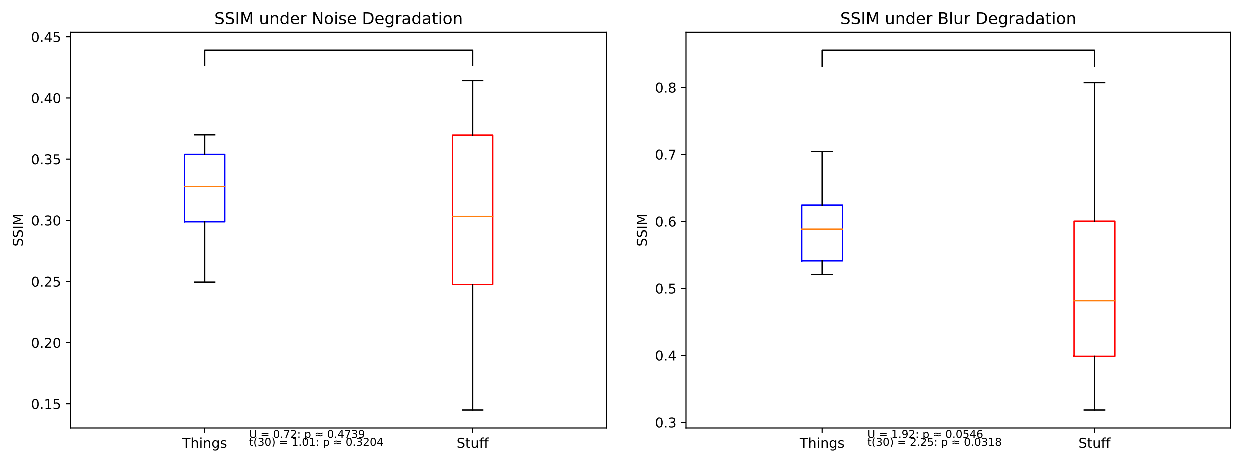


Figure A.2: Boxplots of Things vs. Stuff per-class average SSIM value following a) comprehensive noise and b) gaussian blur degradation

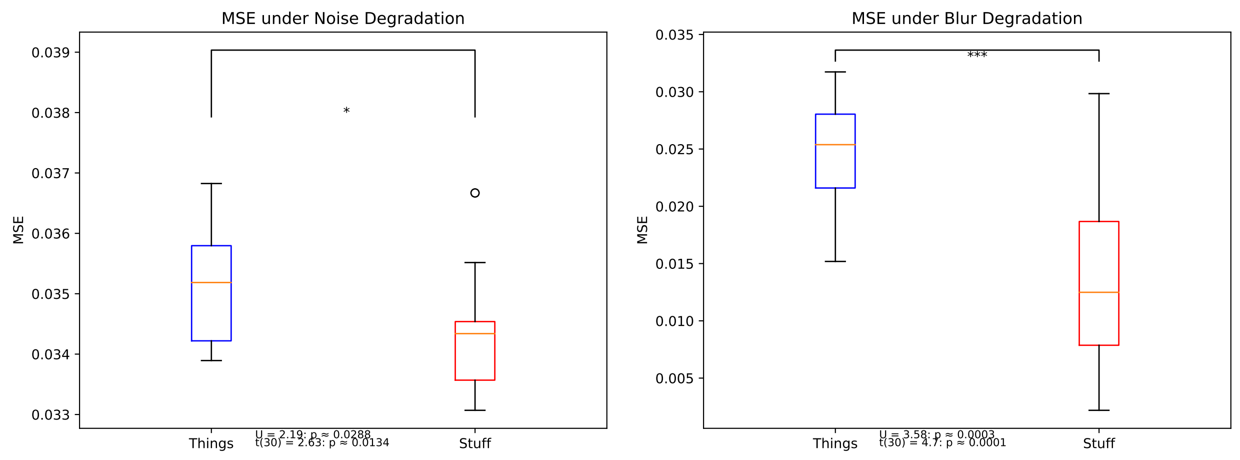


Figure A.3: Boxplots of Things vs. Stuff per-class average MSE value following a) comprehensive noise and b) gaussian blur degradation. A highly significant difference was found for blur and a significant difference was found for noise.

Class	SSIM (Noise)	SSIM (Blur)	MSE (Noise)*	MSE (Blur)***
001.ak47	0.290071	0.622159	0.034074	0.020014
003.backpack	0.356969	0.525428	0.033894	0.031732
014.blimp	0.289445	0.632064	0.033907	0.018948
016.boom-box	0.353024	0.542287	0.035034	0.027933
021.breadmaker	0.249472	0.704395	0.036825	0.015187
027.calculator	0.369878	0.520704	0.035747	0.028326
070.fire-extinguisher	0.349414	0.542153	0.034225	0.027950
157.pci-card	0.331589	0.597943	0.036483	0.022124
172.revolver-101	0.320476	0.601347	0.035287	0.024577
183.sextant	0.356505	0.521855	0.034204	0.026583
208.swiss-army-knife	0.301681	0.633590	0.036322	0.024359
219.theodolite	0.362882	0.537789	0.035085	0.028403
227.treadmill	0.352758	0.561325	0.035675	0.030406
238.video-projector	0.306313	0.620334	0.035942	0.022637
239.washing-machine	0.276506	0.630854	0.035510	0.017116
246.wine-bottle	0.323609	0.578884	0.034227	0.026184
c003	0.275599	0.626444	0.033616	0.023993
c032	0.347602	0.399484	0.033313	0.014600
c045	0.217904	0.598689	0.033069	0.006137
c047	0.223425	0.605224	0.034264	0.005437
c049	0.299617	0.548634	0.034509	0.010396
c066	0.414196	0.318380	0.033531	0.014461
c089	0.397123	0.395882	0.034433	0.029837
c093	0.144926	0.807133	0.034836	0.002188
c118	0.279294	0.440426	0.036669	0.008444
c129	0.191005	0.654931	0.034415	0.004293
c159	0.361263	0.414731	0.034599	0.018150
c160	0.325780	0.428560	0.034519	0.010520
c163	0.306720	0.522677	0.034162	0.015516
c178	0.396957	0.368437	0.033582	0.023680
c184	0.394858	0.375693	0.033356	0.020246
c191	0.255615	0.576282	0.035516	0.010152

Table A.1: Per-class average SSIM and MSE following identical strength noise and blur degradation. Things vs. Stuff significantly (according to a Wilcoxon rank-sum test) differed on their resistance to blur in the eyes of mean-squared error (stuff was seen by the MSE as degrading less). SSIM is particularly affected for both metacategories under noise.