Title
Anytime Recognition of Objects and Scenes

Author
Karayev, Sergey

Publication Date
2014

Peer reviewed|Thesis/dissertation

Anytime Recognition of Objects and Scenes

by

Sergey Kazbekovich Karayev

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Trevor Darrell, Chair
Professor Ken Goldberg
Professor Pieter Abbeel
Dr Mario Fritz

Fall 2014

Anytime Recognition of Objects and Scenes

Abstract

Anytime Recognition of Objects and Scenes

by

Sergey Kazbekovich Karayev

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Trevor Darrell, Chair

Humans are capable of perceiving a scene at a glance, and obtain deeper understanding with additional time. Computer visual recognition should be similarly robust to varying computational budgets — a property we call Anytime recognition. We present a general method for learning dynamic policies to optimize Anytime performance in visual recognition. We approach this problem from the perspective of Markov Decision Processes, and use reinforcement learning techniques. Crucially, decisions are made at test time and depend on observed data and intermediate results. Our method is applicable to a wide variety of existing detectors and classifiers, as it learns from execution traces and requires no special knowledge of their implementation.

We first formulate a dynamic, closed-loop policy that infers the contents of the image in order to decide which single-class detector to deploy next. We explain effective decisions for reward function definition and state-space featurization, and evaluate our method on the PASCAL VOC dataset with a novel *costliness* measure, computed as the area under an Average Precision (AP) vs. Time curve. In contrast to previous work, our method significantly diverges from predominant greedy strategies and learns to take actions with deferred values. If execution is stopped when only half the detectors have been run, our method obtains 66% better mean AP than a random ordering, and 14% better performance than an intelligent baseline.

The detection actions are costly relative to the inference performed in executing our policy. Next, we apply our approach to a setting with less costly actions: feature selection for linear classification. We explain strategies for dealing with unobserved feature values that are necessary to effectively classify from any state in the sequential process. We show the applicability of this system to a challenging synthetic problem and to benchmark problems in scene and object recognition. On suitable datasets, we can additionally incorporate a semantic back-off strategy that gives maximally specific predictions for a desired level of accuracy. Our method delivers best results on the costliness measure, and provides a new view on the time course of human visual perception.

Traditional visual recognition obtains significant advantages from the use of many features in classification. Recently, however, a single feature learned with multi-layer convolutional networks (CNNs) has outperformed all other approaches on the main recognition datasets. We propose Anytime-motivated methods for speeding up CNN-based detection approaches while maintaining their high accuracy: (1) a dynamic region selection method using novel quick-to-compute features; and (2) the Cascade CNN, which adds a *reject* option between expensive convolutional layers and allows the network to terminate some computation early. On the PASCAL VOC dataset, we achieve an 8x speed-up while losing no more than 10% of the top detection performance.

Lastly, we address the problem of image style recognition, which has received little research attention despite the significant role of visual style in conveying meaning through images. We present two novel datasets: 80K Flickr photographs annotated with curated style labels, and 85K paintings annotated with style/genre labels. In preparation for Anytime recognition, we perform a thorough evaluation of different image features for image style prediction. We find that features learned in a multi-layer network perform best, even when trained with object category labels. Our large-scale learning method also results in the best published performance on an existing dataset of aesthetic ratings and photographic style annotations. We use the learned classifiers to extend traditional tag-based image search to consider stylistic constraints, and demonstrate cross-dataset understanding of style.

To my parents, Kazbek and Valeriya.

Contents

List of Figures

List of Tables

Acknowledgments

I am fortunate to have worked with consistently amazing people during my PhD. My advisor Trevor Darrell has always provided wise and encouraging guidance, and supported every direction that excited me. My close collaborator and mentor Mario Fritz is responsible for too many of "my" ideas to count. Regular meetings with Pieter Abbeel helped develop the reinforcement learning formulation of vision problems presented in this thesis. Ken Goldberg, Jitendra Malik, and Bruno Olshausen provided crucial inter-disciplinary North Stars to keep in view. Aaron Hertzmann, Holger Winnemoeller, and Aseem Agarwala introduced me to the novel recognition problem of image style, and Alyosha Efros has been tirelessly encouraging of this line of work.

The bulk of graduate school life and learning is centered on one's peers, and at Berkeley I was lucky to be among the truly best. Yangqing Jia, Jon Barron, Adam Roberts, Trevor Owens, Hyun Oh Song, Ning Zhang, Judy Hoffman, Allie Janoch, Jon Long, Jeff Donahue, Evan Shelhamer, Georgia Gkioxari, Saurabh Gupta, Bharath Hariharan, Subhransu Maji, Sanja Fidler, Carl Henrik Ek, Brian Kulis, Kate Saenko, Mario Christoudias, Oriol Vinyals, Ross Girshick, Sergio Guadarrama, and so many others.

Last but not least, I am deeply indebted to the love and support of my parents. This thesis is dedicated to them.

Chapter 1

Introduction

1.1 Motivation

Perception

It is well known that human perception is both Anytime, meaning that a scene can be described after even a short presentation, and progressive, meaning that the quality of description increases with more time. The progressive time course of visual perception has been confirmed by multiple studies (Fei-Fei et al. 2007; Vanrullen and Thorpe 2001), with some studies providing evidence that enhancement occurs in an ontologically meaningful way. For example, people tend to recognize something as an animal before recognizing it as a dog (Macé et al. 2009). The underlying mechanisms of this behavior are not well explored, with only a few attempts made to explain the temporal dynamics — for instance, a promising work by Hegde 2008 has employed the framework of sequential decision processes.

Computer applications

Meanwhile, automated visual recognition has achieved levels of performance that allow useful real-world implementation. We focus on two problem formulations: *image classification*, in which some property of the image – such as scene type, visual style, or even object presence – is predicted, and *object detection*, in which the location and category (or identity) of all objects in a scene is predicted. Solutions to the two problems are often linked, as classification can be a "subroutine" in a detection method. State-of-the-art methods for classification and detection tend to be computationally expensive, insensitive to Anytime demands, and not progressively enhanced.

Application

As real-world deployment of recognition methods grows, managing resource cost (power or compute time) becomes increasingly important. For tasks such as personal robotics, it is crucial to be able to deploy varying levels of processing to different stimuli, depending on computational demands on the robot. A hypothetical system for vision-based advertising, in which paying customers engage with the system to have their products detected in images on the internet, presents another example. The system has different values (in terms of cost per click) and accuracies for different classes of objects, and the backlog of unprocessed images fluctuates based on demand and available server time. A recognition strategy to maximize profit in such an environment should exploit all signals available to it, and the quality of detections should be Anytime, depending on the length of the queue (for example, lowering recall with increased queue pressure).

Visual Features & Classification

For most state-of-the-art classification methods, a range of features are extracted from an image instance and used to train a classifier. Since the feature vectors are usually high-dimensional, linear classification methods are used most often. Features are extracted at different costs, and contribute differently to decreasing classification error. Although it can generally be said that "the more features, the better," high accuracy can of course be achieved with only a small subset of features for some instances. Additionally, different instances benefit from different subsets of features. For example, simple binary features are sufficient to quickly detect faces (Viola and Jones 2004) but not more varied visual objects, while the features most useful for separating landscapes from indoor scenes (Xiao et al. 2010) are different from those most useful for recognizing fine distinctions between bird species (Farrell et al. 2011). Figure 1.1 presents several common visual features.

Detection

Detection methods tend to employ the same visual features and classifiers but apply them to many image sub-regions. Approaches can broadly be grouped into (A) *per-class, all-region*, (B) *all-class, all-region*, and (C) *all-class, proposed-region* methods. State-of-the-art *all-class, proposed-region* methods such as Girshick et al. 2014 and *per-class, all-region* methods such as Felzenszwalb et al. 2010 are considerably slow (on the order of seconds), performing an expensive computation on (respectively) a thousand to a million image windows. To maximize early performance gains of these methods, scene and inter-object contextual cues can be exploited in two ways. First, regions can be processed in an intelligent order, with most likely locations selected first. Second, if detectors are applied per class, then they can be sequenced so as to maximize the chance of finding objects actually present in the image. And even the most recent *all-class, all-region*, Convolutional Neural Net (CNN)-based detection methods such as He et al. 2014, which take advantage of high-performance convolutional primitives for region processing and detect for all classes simultaneously, can be sped up using our idea of cascaded classification.

1.2 Our Contributions

Costliness

Computing all features, running all detectors, or processing all regions for all images is infeasible in a deployment sensitive to Anytime needs, as each feature brings a significant computational burden. Yet the conventional approach to evaluating visual recognition does not consider efficiency, and evaluates performance independently across classes. We address the problem of selecting and combining a subset of features under an Anytime cost budget (specified in terms of wall time or total power expended or another metric) and propose a new *costliness* measure of performance vs. cost.

Learning a Policy

To exploit the fact that different instances benefit from different subsets of features, our approach to feature selection is a sequential policy. To learn the policy parameters, we formulate the problem as a Markov Decision Process (MDP) and use reinforcement learning methods. The method does not make many assumptions about the underlying actions, which can be existing object detectors and feature-specific classifiers. With different settings of parameters, we can learn policies ranging from **Static, Myopic**—greedy selection not relying on any observed feature values, to **Dynamic, Non-myopic**—relying on observed values and considering future actions. The foundational machinery is laid out in Section 2.2.

Per-class Detection

For *per-class* detection, the actions are time-consuming detectors applied to the whole image, as well as a quick scene classifier. We run scene context and object class detectors over the whole image sequentially, using the results of detection obtained so far to select the next actions. Since the actions are time-consuming, we use a powerful inference mechanism to select the best next action. In Section 2.3, we evaluate on the PASCAL VOC dataset and obtain better performance than all baselines when there is less time available than is needed to exhaustively run all detectors. This work was originally presented in Karayev et al. 2012 and all work is open source[1].

Image Classification

Classification actions are much faster than detectors, and the action-selection method accordingly needs to be fast. Because different features can be selected for different instances, and because our system may be called upon to give an answer at any point during its execution, the feature combination method needs to be robust to a large number of different observed-feature subsets. In Chapter 3, we consider several value-imputation methods and present a method for learning several classifiers for different clusters of observed-feature subsets. We first demonstrate on synthetic data that our algorithm learns to pick features most useful for the specific test instance. We demonstrate the advantage of non-myopic over greedy, and of dynamic over static on this and the Scene-15 visual classification dataset. Then we show results on a subset of the hierarchical ImageNet dataset, where we additionally learn to provide the most specific answers for any desired cost budget and accuracy level. This work was originally presented in Karayev, Fritz, and Darrell 2014 and all work is open source[2].

[1] Available at https://github.com/sergeyk/timely_object_recognition
[2] Available at https://github.com/sergeyk/anytime_recognition

Cascade CNN

We additionally investigate a novel approach for speeding up a state-of-the-art CNN-based detection method, and propose a general technique for accelerating CNNs applied to class imbalanced data. We employ the classic idea of the cascade by inserting a *reject* option between expensive convolutional layers. When a CNN processes batches of images, which is standard for many applications, the reject layers allows "thinning" of the batch as it progresses through the network, thus saving processing time. This method is applicable to both *all-class, proposed-region* methods such as Girshick et al. 2014 and *all-class, all-region* methods such as He et al. 2014. We demonstrate results — along with a variety of strong baselines – on the former method, and show that the Cascade CNN method obtains a nearly 10x speed-up with only marginal drop in accuracy. All work is reported in Chapter 4.

Recognizing Style

Lastly, in Chapter 5 we present two novel datasets and first results for an underexplored research problem in computer vision – recognizing visual style. In preparation for an Anytime approach, we evaluate several different features (including CNNs) for the task, and explore content-style correlations in our datasets. Our large-scale learning gives state-of-the-art results on an existing dataset of image quality and photographic style, and provides a strong baseline on our contributed datasets of 80K photos and 85K paintings labeled with their style and genre. In a demonstration of cross-dataset understanding of style, we show how results of a search by content can be filtered by style. This work was originally presented in Karayev et al. 2014, and all code is open source[3].

Future Directions

This thesis provides an effective foundation for Anytime visual recognition, and points the way to interesting further developments. Our MDP-based formulation of learning a feature-selection policy is empirically effective, but heuristic in nature. The recently developed framework of adaptive submodularity (Golovin and Krause 2011) could provide theoretical near-optimality results for some policies, but developing an appropriate objective for our task is not straightforward. We showed our Cascade CNN model to be effective for a region-based detection task – but the model was not trained end-to-end with the threshold layers. An even more interesting future development would add an Anytime loss layer that combines classification output from multiple levels of the network in a cost-sensitive way. We expand on these ideas in Chapter 6.

1.3 Related Work

Our work spans across several sub-fields of computer vision. Here we cover the necessary background, ordered by applicability to each chapter of this thesis.

[3]Available at https://github.com/sergeyk/vislab

1.3.1 Detection

Features

Classically, the best recent performance has come from detectors that use gradient-based features to represent objects as either a collection of local patches or as object-sized windows (Dalal and Triggs 2005; Lowe 2004). Classifiers are then used to distinguish between featurizations of a given class and all other possible contents of an image window. For state-of-the-art performance, the object-sized window models are augmented with parts (Felzenszwalb et al. 2010), and the bag-of-visual-words models employ non-linear classifiers (Vedaldi et al. 2009). In Chapter 2, we employ the widely used Deformable Part Model detector (Felzenszwalb et al. 2010).

CNNs

Most recently, best performance is obtained not with hand-designed features but with those learned on large-scale labeled datasets such as ImageNet (Deng et al. 2009) by a deep convolutional neural network (CNN) such as AlexNet (Krizhevsky, Sutskever, and Hinton 2012b). This has prompted attempts to apply these computationally expensive methods to detection (Erhan et al. 2014; Sermanet and Eigen 2014). The *R-CNN* method of Girshick et al. 2014 in particular is powerful but slow, requiring costly processing of many windows. Recent work from He et al. 2014 (SPP-net) sustained the high performance of R-CNN while decreasing the running time by an order of magnitude. Our work in Chapter 4 is evaluated in the R-CNN framework, but applies to the SPP-net method also.

Windows

Window proposal is most often done exhaustively over the image space as a "sliding window", or inexhaustively with a bottom-up segmentation approach (Uijlings et al. 2013). Some approaches use "jump windows" (hypotheses voted on by local features) (Vedaldi et al. 2009; Vijayanarasimhan and Grauman 2011), or a bounded search over the space of all possible windows (Lampert, Blaschko, and Hofmann 2008). In all state-of-the-art systems, the window proposal step is conceptually separate from the feature extraction and classification.

Using feedback

None of the best-performing systems treat window proposal and evaluation as a closed-loop system, with feedback from evaluation to proposal. Some work has been done on this topic, mostly inspired by ideas from biological vision and attention research (Butko and Movellan 2009; Vogel and Freitas 2008). One application to the problem of visual detection picks features with maximum value of information in a Hough-voting framework (Vijayanarasimhan and Kapoor 2010). Another uses nearest-neighbor lookups of image windows to sum offset vectors onto objects (Alexe, Heess, and Ferrari 2012).

Multi-class context

Most detection methods train individual models for each class. Work on inherently multi-class detection focuses largely on making detection time sublinear in the number of classes through sharing features (Fan 2005; Torralba, Murphy, and Freeman 2007). Inter-object context has also been shown to improve detection (Torralba, Murphy, and Freeman 2004). A post-processing extension to detection systems uses structured prediction to incorporate multi-class context as a principled replacement for non-maximum suppression (Desai, Ramanan, and Fowlkes 2011). In a standard evaluation setup, inter-object context plays a role only in post-filtering, once all detectors have been run. In contrast, our work leverages inter-object context in the action-planning loop.

Scene context

The most common source of context for detection is the *scene* or other non-detector cues; the most common scene-level feature is the GIST (Oliva and Torralba 2001) of the image. We use this source of scene context in our evaluation. A critical summary of the main approaches to using context for object and scene recognition is given in (Galleguillos and Belongie 2010). For the commonly used PASCAL VOC dataset (Everingham et al. 2010), GIST and other sources of context are quantitatively explored in (Divvala et al. 2009).

1.3.2 Classification

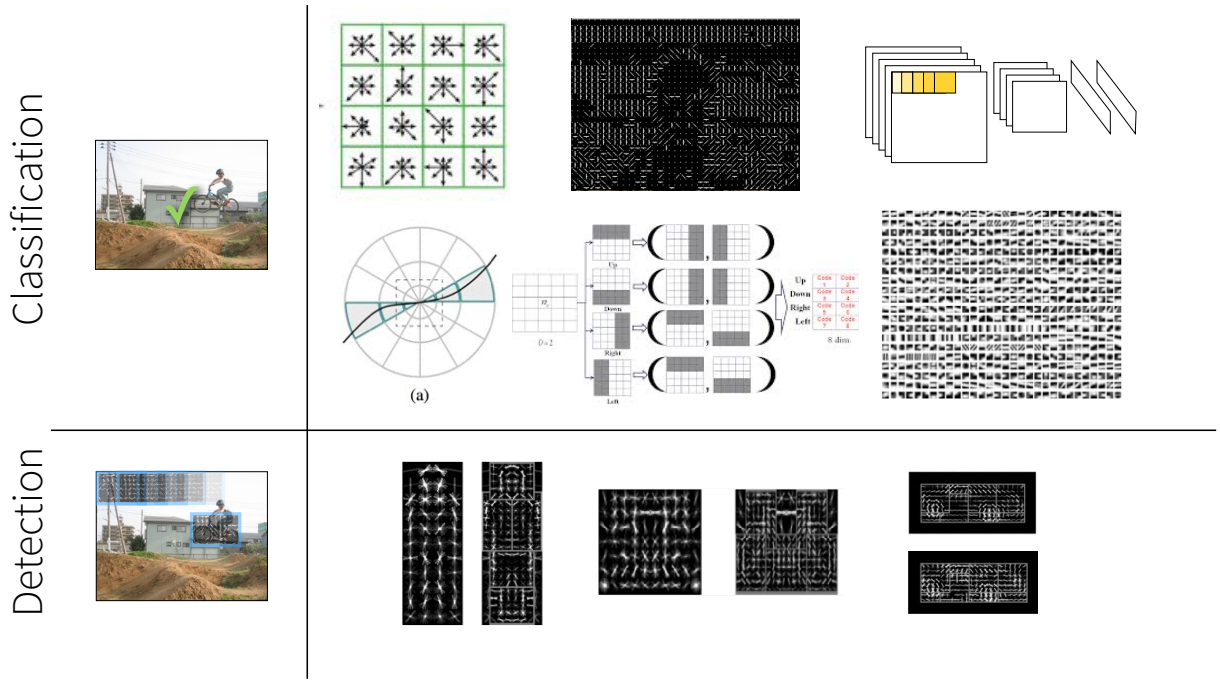


Figure 1.1: Summary of the variety of features for object detection and classification. In reading order for classification: SIFT (Lowe 2004), HOG (Dalal and Triggs 2005), CNN (Krizhevsky, Sutskever, and Hinton 2012b), Self-Similarity (Shechtman and Irani 2007), Haar basis functions (Viola and Jones 2004), basis functions learned with sparse coding (Olshausen and Others 1996). In reading order for detection: person, bicycle, and car templates for the Deformable Part Model (Felzenszwalb et al. 2010). (The features depicted were not computed on the images depicted.)

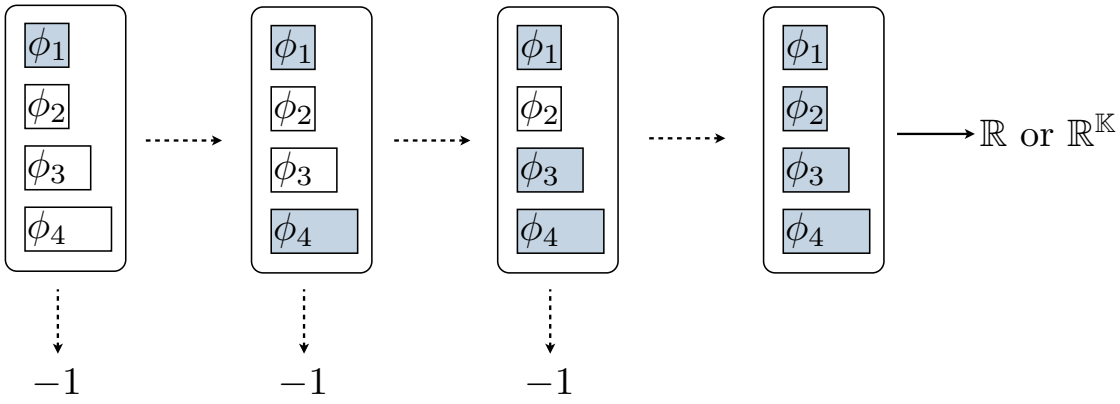


Figure 1.2: Sequential feature selection: **Cascade**. In addition to the feature computation actions, the classifier is augmented with a rejection action. The cascade is Anytime in a limited way, as only the rejection answer can be given before all features are evaluated. The fixed order of the cascade is not robust to the fact that different images benefit from different features.

Visual features

The field of computer vision has built up a small arsenal of features extracted from whole images or fixed-size patches. These features differ in computational cost and target different sources of data – for instance, the Haar wavelets feature of Viola and Jones 2004 was designed for sequential appication in face recognition datasets, while the HOG feature of Dalal and Triggs 2005 was designed for template matching in pedestrian-detection datasets. Figure 1.1 presents a sampling of the most used ones. Recently, middle layers of CNN's trained on large image categorization datasets have provided a generally applicable feature that obtains top performance on a multitude of datasets (Donahue et al. 2013a).

Feature selection

The simplest way to limit the number of features used at test time is to L_1-regularize. This method does not explicitly consider feature cost, nor is it able to evaluate features one by one, or to give an answer before all features are computed. In Figures 1.2, 1.3, 1.4, 1.5 and in the paragraphs below we explain more advanced methods, all of them treating feature selection as a sequential process. A note about the figures: the rounded rectangles represent feature sets, with shaded features ϕ representing selected features.

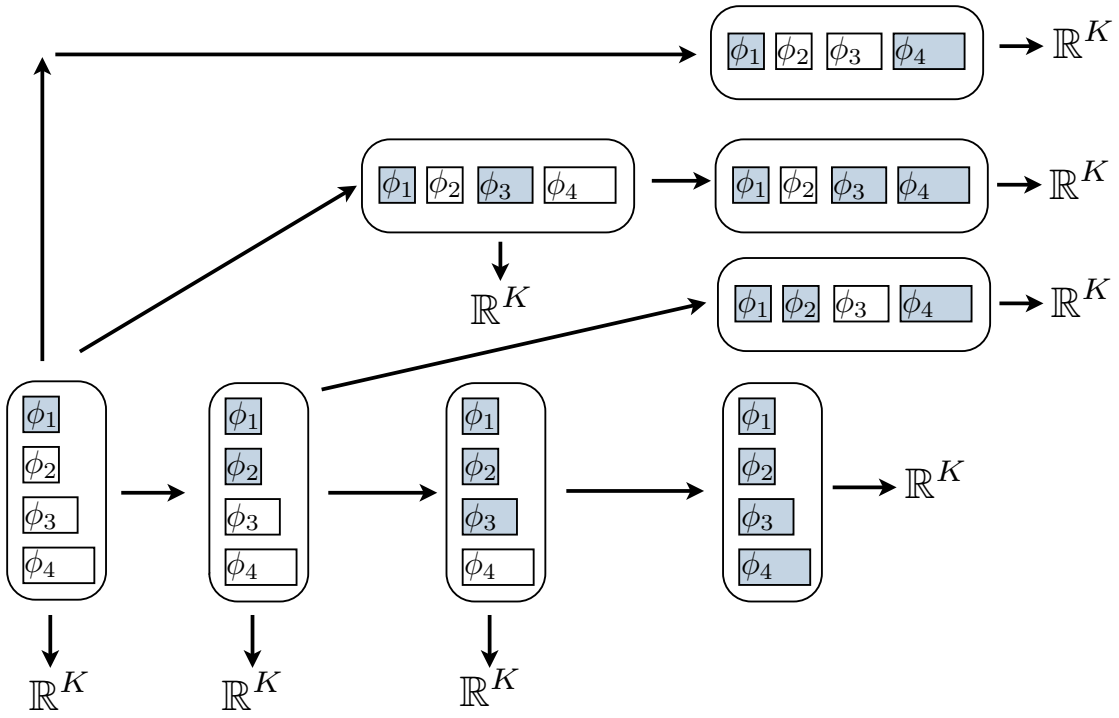


Figure 1.3: The **MD-DAG** method (Benbouzid, Busa-Fekete, and Kegl 2012) augments the traditional cascade with an additional Skip action, which allows learning a more robust policy, but does not fully cover the space of possible policies (the initial ordering sets the limit).

Cascaded methods

A well-known method to evaluate features sequentially is the cascaded boosted classifier of Viola and Jones 2004 (updated by Bourdev and Brandt 2005 with a soft threshold), which is able to quit evaluating an instance before all features are computed—but feature cost was not considered. The cost-sensitive cascade of Chen et al. 2012 optimizes stage order and thresholds to jointly minimize classification error and feature computation cost. Figure 1.2 represents this model. Xu, Weinberger, and Chapelle 2012 and Grubb and Bagnell 2012 separately develop a variant of gradient boosting for training cost-sensitive classifiers; the latter prove near-optimality of their greedy algorithm with submodularity results. Their methods are tightly coupled to the stage-wise regression algorithm. Cascades are not dynamic policies: they cannot change the order of execution based on observations obtained during execution, which is our goal.

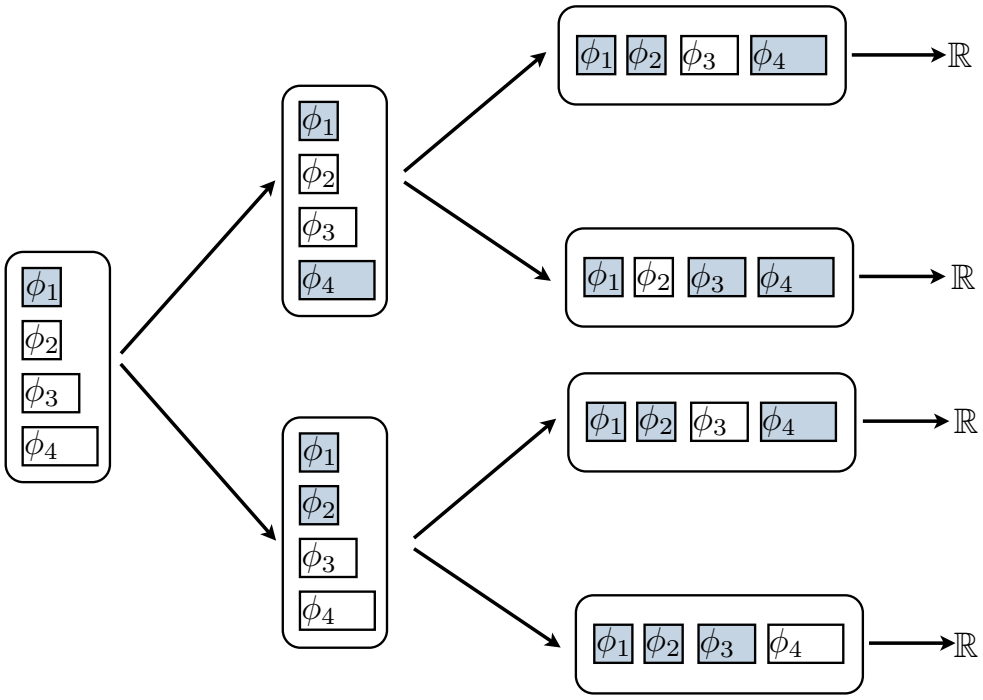


Figure 1.4: **Tree-based** methods find a tree-structured policy for computing features. Classification answers are given only at the leaf nodes. The tree structure can be found by direct optimization of some problem, such as cost-senstive classification, as in Xu, Weinberger, and Chapelle 2012, or induced by a tangential problem, as in Deng et al. 2011, who use the confusion matrix to set the structure of their Label Tree.

Dynamic methods

In contrast, *Label trees* guide an instance through a tree of classifiers; their structure is determined by the confusion matrix or learned jointly with weights (Deng et al. 2011). Xu et al. 2013 learn a cost-sensitive binary tree of weak learners using an approach similar to the cyclic optimization of (Chen et al. 2012). The state space of such tree methods is visualized in Figure 1.4. A fully general DAG – instead of a tree – over the state space is proposed by Gao and Koller 2011 under the name of *active classification*, and visualized in Figure 1.5. Their method myopically selects the next feature based on expected information gain given the values of the already selected features. Since it is based on locally weighted regression, *active classification* is highly costly at test time. Ji and Carin 2007 also formulate cost-sensitive feature selection generatively, as an HMM conditioned on actions, but select actions myopically, again at signficant test time cost.

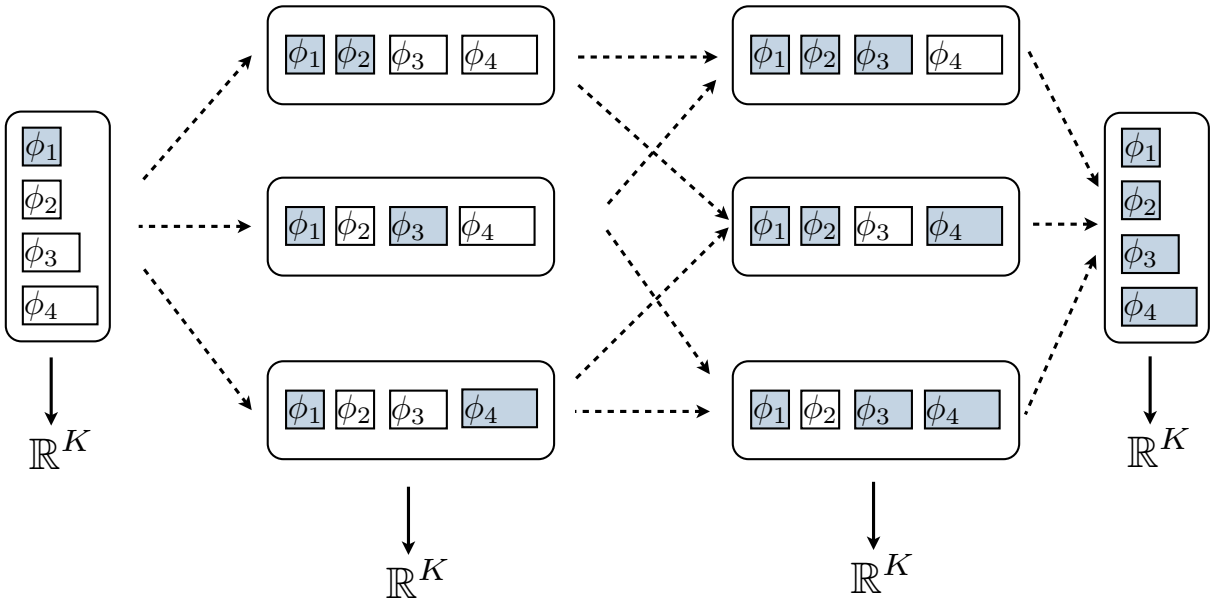


Figure 1.5: In this work and in methods such as Gao and Koller 2011, the policy is a **general DAG** over selected-feature subsets, which allows actions to be taken in an entirely flexible order. In our work, we are also able to give the classification answer from all states, making our work truly Anytime.

Reinforcement Learning

Just like active classification, our method and the three methods below can learn any possible policy. Dulac-Arnold et al. 2012 present an MDP-based solution to "datum-wise classification", with an action space comprised of all features and labels, recently extended to region-based processing (Dulac-arnold, Thome, and Cord 2014). This independently-conducted work is closely related to ours, with differences in defining the action space and learning mechanism. He, Hal III, and Eisner 2012 formulate an MDP with features and a single classification step as actions, but solve it via imitation learning of a greedy policy. Trapeznikov, Saligrama, and Castanon 2013 provides another variation on this formulation. Another notable work is the method of Benbouzid, Busa-Fekete, and Kegl 2012, graphically presented in Figure 1.3, which formulates an MDP that simply extends the traditional sequential boosted classifier with an additional *skip* action, significantly limiting the space of learnable policies. This "MD-DAG" method is able to learn only a subset of all possible policies.

Misc

Less directly related – but exciting for its novelty – is the work of (Weiss, Sapp, and Taskar 2013), who apply simple introspection to structured models for a significant speedup of human pose estimation. Another exciting direction is theoretical analysis based on adaptive submodularity (Golovin and Krause 2011). In vision, there is an application of such results to detection with humans in the loop (Chen et al. 2014). In robotics, an adaptively submodular objective was successfully formulated for the problem of grasping (Javdani et al. 2012).

Feature Combinatio Value Imputation

For SVM-based classifiers, *Multiple Kernel Learning* (MKL) provides a way to train classifiers using an automatically weighted combination of kernels (Lanckriet et al. 2004). It has been shown that MKL is outperformed by boosting single-kernel classifiers (Gehler and Nowozin 2009). Of course, if all classifiers are linear, then combining outputs of classifiers trained on different feature channel with another classifier is equivalent to training one classifier on all features at once.

The imputation problem is faced in the *collaborative filtering* literature, working on problems such as the Netflix Prize (Koren, Bell, and Volinsky 2009). Matrix factorization methods, commonly based on the Singular Value Decomposition (SVD), are often employed. Our problem is significantly different in that at training time, all values are fully observed — and the final task is classification, not simple imputation. Imputation approaches have also been explored in genomics work, where the real-world data is often missing a large portion of the observations (Hastie et al. 1999).

1.3.3 Style Recognition

Most research in computer vision addresses recognition and reconstruction, independent of image style. A few previous works have focused directly on image composition, particularly on the high-level attributes of beauty, interestingness, and memorability.

Aesthetic Rating

Most commonly, several previous authors have described methods to predict aesthetic quality of photographs. Datta et al. (Datta et al. 2006), designed visual features to represent concepts such as colorfulness, saturation, rule-of-thirds, and depth-of-field, and evaluated aesthetic rating predictions on photographs; The same approach was further applied to a small set of Impressionist paintings (Li and Chen 2009). The feature space was expanded with more high-level descriptive features such as "presence of animals" and "opposing colors" by Dhar et al., who also attempted to predict Flickr's proprietary "interestingness" measure, which is determined by social activity on the website (Dhar, Berg, and Brook 2011). Gygli et al. (Gygli, Nater, and Gool 2013) gathered and predicted human evaluation of image interestingness, building on work by Isola et al. (Isola et al. 2011), who used various high-level features to predict human judgements of image memorability. In a similar task, Borth et al. (Borth et al. 2013) performed sentiment analysis on images using object classifiers trained on adjective-noun pairs.

AVA and Attributes

Murray et al. (Murray, Marchesotti, and Perronnin 2012) introduced the Aesthetic Visual Analysis (AVA) dataset, annotated with ratings by users of DPChallenge, a photographic skill competition website. The AVA dataset contains some photographic style labels (e.g., "Duotones," "HDR"), derived from the titles and descriptions of the photographic challenges to which photos were submitted. Using images from this dataset, Marchesotti and Perronnin (Marchesotti and Perronnin 2013) gathered bi-grams from user comments on the website, and used a simple sparse feature selection method to find ones predictive of aesthetic rating. The attributes they found to be informative (e.g., "lovely photo," "nice detail") are not specific to image style.

Visual Art

Features based on image statistics have been successfully employed to detect artistic forgeries (Lyu, Rockmore, and Farid 2004). Such work focuses on extremely fine-scale discrimination between two very similar classes, and has not been applied to broader style classification. Several previous authors have developed systems to classify classic painting styles, including (Keren 2002; Shamir et al. 2010). These works consider only a handful of styles (less than ten apiece), with styles that are visually very distinct, e.g., Pollock vs. Dalí. These datasets comprise less than 60 images per style, for both testing and training. Mensink and Gemert 2014 provide a larger dataset of artworks, but do not consider style classification as its own problem.

Style vs. Content

Separate from the application domain of vision, some machine learning research has attempted to separate style from content (Tenenbaum and Freeman 2000). In particular, Neural Network researchers have provided interesting recent results: Taylor and Hinton 2009 use a Restricted Boltzmann Machine to separately consider style and content for the problem of human gait recognition, and Graves 2013 uses a Long Short-Term Memory recurrent neural network to generate realistic handwriting in a multitude of styles.

Chapter 2

Reinforcement Learning for Anytime Detection

2.1 Problem Definition

Definitions

We deal with a dataset of images $\mathcal{D}$, where each image x contains zero or more objects. Each object is labeled with exactly one category label $k \in \{1, \ldots, K\}$. The multi-class, multi-label **classification** problem asks whether x contains at least one object of class k. We write the ground truth for an image as $\mathbf{C} = \{C_1, \ldots, C_K\}$, where $C_k \in \{0, 1\}$ is set to 1 if an object of class k is present. The **detection** problem is to output a list of bounding boxes (sub-images defined by four coordinates), each with a real-valued confidence that it encloses a single instance of an object of class k. The answer for a single class k is given by an algorithm $detect(x, k)$, which outputs a list of sub-image bounding boxes B and their associated confidences.

Evaluation metric

Performance is evaluated by plotting precision vs. recall across dataset $\mathcal{D}$ (by progressively lowering the confidence threshold for a positive detection). The area under the curve yields the Average Precision (AP) metric, which has become the standard evaluation for recognition performance on challenging datasets in vision (Everingham et al. 2010). A common measure of a correct detection is the PASCAL overlap: two bounding boxes are considered to match if they have the same class label and the ratio of their intersection to their union is at least $\frac{1}{2}$. Multi-class performance is evaluated by averaging the individual per-class AP values. In a specialized system such as the advertising case study from Chapter 1, the metric generalizes to a weighted average, with the weights set by the *values* of the classes.

Policy

Our goal is a multi-class recognition policy π that takes an image x and outputs a list of multi-class detection results by running detector and global scene *actions* sequentially. The policy repeatedly selects an action $a_i \in \mathcal{A}$, executes it, receiving observations o_i, and then selects the next action. The set of actions $\mathcal{A}$ can include both classifiers and detectors: anything that would be useful for inferring the contents of the image.

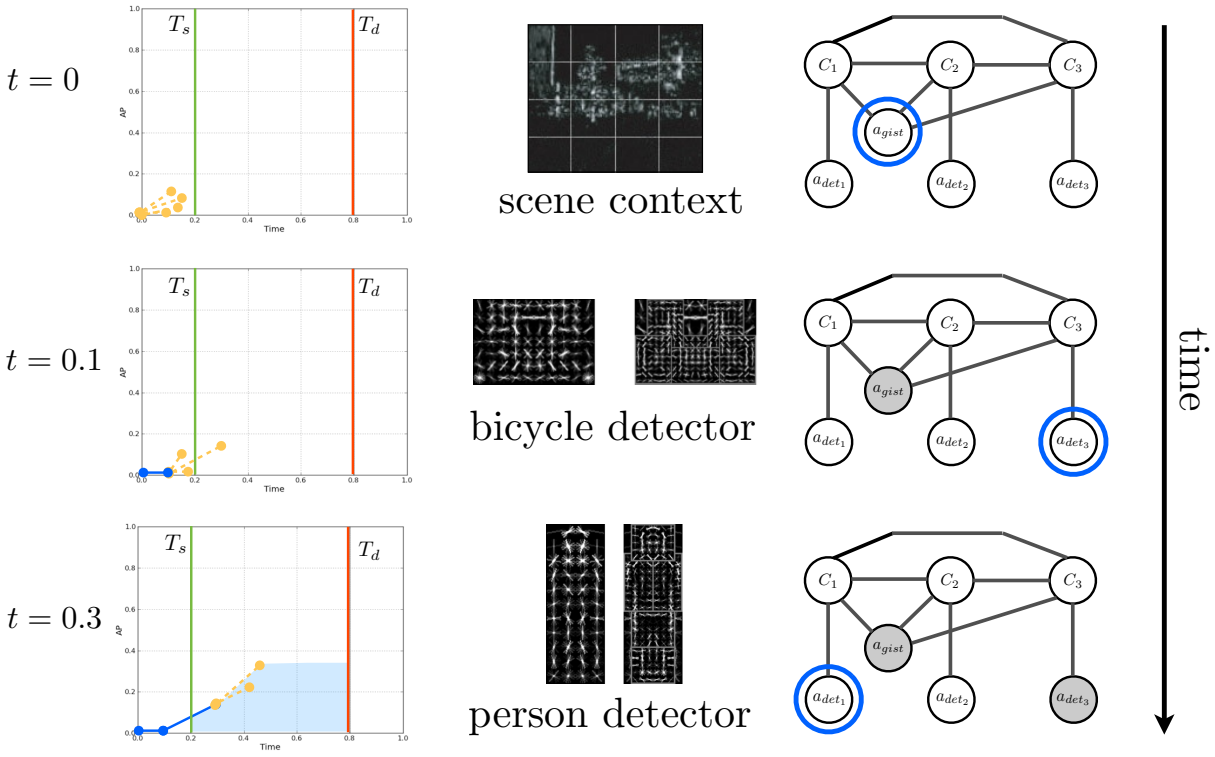


Figure 2.1: A sample trace of our method. At each time step beginning at $t = 0$, potential actions are considered according to their predicted *value*, and the maximizing action is picked. The selected action is performed and returns *observations*. Different actions return different observations: a detector returns a list of detections, while a scene context action simply returns its computed feature. The *belief model* of our system is updated with the observations, which influences the selection of the next action. The final evaluation of a detection episode is the area of the *AP vs. Time* curve between given start and end times. The value of an action is the expected result of final evaluation if the action is taken and the policy continues to be followed, which allows actions without an immediate benefit to be scheduled.

Actions

Each action a_i has an expected cost $c(a_i)$ of execution. Depending on the setting, the cost can be defined in terms of algorithmic runtime analysis, an idealized property such as number of *flops*, or simply the empirical runtime on specific hardware. We take the empirical approach: every executed action advances t, the *time into episode*, by its runtime. The specific actions we consider in the following exposition are detector actions a_{det_i}, where det_i is a detector class C_i, and a scene-level context action a_{gist}, which updates the probabilities of all classes. Although we do not showcase this here, note that our system easily handles multiple detector actions per class.

Anytime metric

As shown in Figure 2.1, the system is given two times: the setup time T_s and deadline T_d. We want to obtain the best possible answer if stopped at any given time between the setup time and the deadline. A single-number metric that corresponds to this objective is the area captured under the curve between the start and deadline bounds, normalized by the total area. We evaluate policies by this more robust metric and not simply by the final performance at deadline time for the same reason that Average Precision is used instead of a fixed Precision vs. Recall point in the conventional evaluations. Additionally, maximizing this single-number metric corresponds to maximizing Anytime performance.

2.2 Method

2.2.1 MDP Formulation

To model the **action selection** policy $\pi(x) : \mathcal{X} \mapsto 2^{\mathcal{A}}$, we employ the Markov Decision Process (MDP) to model a single *episode* of selecting actions for some instance x.

Definition 1. *The **feature selection MDP** consists of the tuple* $(\mathcal{S}, \mathcal{A}, T(\cdot), R(\cdot), \gamma)$*:*

- ***State*** $s \in \mathcal{S}$ *stores the selected action subset* $\mathcal{A}_{\pi(x)}$*, resulting observations, and total cost* $C_{\mathcal{A}_{\pi(x)}}$*.*
- *The set of **actions** $\mathcal{A}$.*
- *The **state transition** distribution $T(s' \mid s, a)$ can depend on the instance x.*
- *The **reward** function $R(s, a, s') \mapsto \mathbb{R}$ is manually specified, and depends on the actions taken and the instance x.*
- *The discount γ determines amount of **lookahead** in selecting actions: if 0, actions are selected greedily based on their immediate reward; if 1, the reward accrued by subsequent actions is given just as much weight as the reward of the current action.*

Trajectories and reward

A recognition *episode* takes an image $\mathcal{I}$ and proceeds from the initial state s^0 and action a^0 to the next pair (s^1, a^1), and so on until (s^J, a^J), where J is the last step of the process with $t \leq T_d$. At that point, the policy is terminated, and a new episode can begin on a new image. We call this a *trajectory* $\xi = (s_0, a_0, s_1, r_1, \ldots, a_{I-1}, s_I, r_I)$, where I is the total number of actions taken (and therefore features selected), s_0 is the initial state, $a_i \sim \pi(a \mid s_i)$ is chosen by the *policy* $\pi(a \mid s)$, and $s_{i+1} \sim T(s \mid s_i, a_i)$, which can depend on x. The total expected reward (value) of an MDP episode is written as

$$V_\pi(s_0) = \mathbb{E}_{\xi \sim \{\pi, x\}} r(\xi) = \mathbb{E}_{\xi \sim \{\pi, x\}} \left[\sum_{i=0}^{I} \gamma^i \, r_i \right] \tag{2.1}$$

Q-Value function

We seek π that maximizes Equation 2.1. If we had a function accurately predicting the value of taking an action in a state, we could define the policy as simply taking the action with maximum value from any state. We specify this function $Q(s, a) : S \times \mathcal{A} \mapsto \mathbb{R}$, where S is the space of all possible states, to assign a value to a potential action $a \in \mathcal{A}$ given the current state s of the decision process. We can then define the policy π as simply $\arg\max_{a_i \in \mathcal{A} \setminus \mathcal{O}} Q(s, a_i)$. The Q-function is defined and learned recursively:

$$Q^{\pi}(s^j, a) = \mathbb{E}_{s^{j+1}}[R(s^j, a) + \gamma Q^{\pi}(s^{j+1}, \pi(s^{j+1}))] \tag{2.2}$$

2.2.2 Learning the policy

Function approximation

Although the action space $\mathcal{A}$ is manageable, the space of possible states S is intractable, and we must use function approximation to represent $Q(s, a)$: a common technique in reinforcement learning (Sutton and Barto 1998). We featurize the state-action pair and assume linear structure:

$$Q^{\pi}(s, a) = \theta_{\pi}^{\top} \phi(s, a) \tag{2.3}$$

where $\phi : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}^{d_s}$ is the state featurization function, d_s is the dimensionality of the state feature vector, and θ^{π} is a vector of weights that defines the policy π.

Policy function

Specifically, the policy is defined as

$$\pi(a \mid s) = \frac{1}{Z} \exp\left(\frac{1}{\tau} \theta^T \phi(s, a)\right) \tag{2.4}$$

where Z is the appropriate normalization and τ is a temperature parameter that controls the level of exploration vs. exploitation in the policy. As $\tau \to 0$, $\pi(a \mid s)$ becomes highly peaked at $\arg\max_a Q(s, a)$; it becomes uniform as $\tau \to \infty$. In training, this parameter is turned down gradually.

Sampling Q-function

While we can't directly compute the expectation in Equation 2.2, we can sample it by running actual episodes to gather $< s, a, r, s' >$ samples, where r is the reward obtained by taking action a in state s, and s' is the following state. We then learn the optimal policy by repeatedly gathering samples with the current policy, minimizing the error between the discounted reward to the end of the episode as predicted by our current $Q(s^j, a)$ and the actual values gathered, and updating the policy with the resulting weights. This method is akin to fitted Q-iteration, a variant of generalized policy iteration (Ernst, Geurts, and Wehenkel 2005; Sutton and Barto 1998). Figure 2.2 shows a step of this process.

Gathering trajectories

During training, we gather samples starting from either a random feasible state, with probability ϵ, or from the initial empty state otherwise. Both ϵ and τ parameters decay exponentially with the number of training iterations. Training is terminated if $\pi_{\theta_{i+1}}$ returns the exact same sequence of episodes ξ on a validation set as π_{θ_i}. During test time, ϵ is set to 0.05.

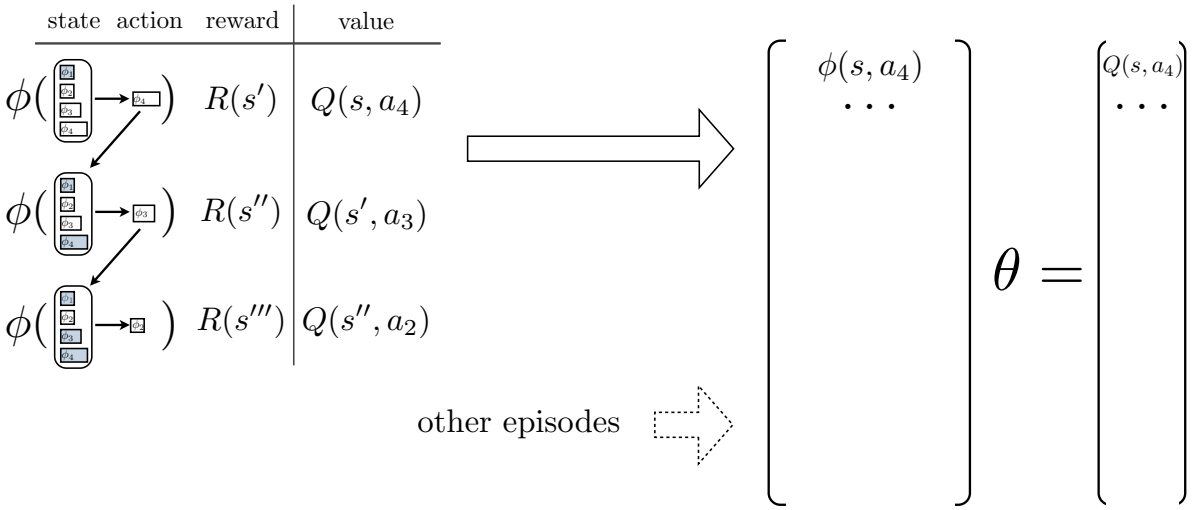


Figure 2.2: We sample $Q^\pi(s,a) = \mathbb{E}_{s'}\left[R(s') + \gamma Q^\pi(s', \pi(s'))\right] = \theta^T \phi(s,a)$ by running the policy over many images. Once an episode is complete, the Q-value at each (s,a) can be determined. To update the policy, simply minimize the prediction error of θ, and repeat.

Block-coding

To formulate learning the policy as a single regression problem, we could represent the features in block form, where $\phi(s,a)$ is a vector of size $F|\mathcal{A}|$, with all values set to 0 except for the F-sized block corresponding to a. An implementation detail: instead of block-coding $\phi(s,a)$, we learn F separate θ_f's for the features $\phi(s)$: one for each action a To prevent overfitting, we use L_2-regularized regression. The weight α of the regularization term is tied across the F separate regressions and is tuned by cross-validation on 3 folds.

Details

We run 15 iterations of accumulating samples by running 350 episodes, starting with a baseline policy which will be described in Section 2.3, and cross-validating the regularization parameter at each iteration. Samples are not thrown away between iterations.

2.2.2.1 Greedy vs non-myopic

Greedy

Note from Equation 2.2 that the $\gamma \in [0,1]$ parameter of the MDP controls the level of *discounting* of rewards of future action in computing the value Equation 2.1. In the baseline **greedy** setting, with $\gamma = 0$, rewards gained by future actions are not counted at all in determining the value of the current action. The value function is determined entirely by the immediate reward, and so only completely greedy policies can be learned in this case. This setting is used as baseline.

Non-myopic

In the **non-myopic** setting, with $\gamma = 1$, rewards gained by future actions are valued exactly as much as reward gained by the current action in determining its value. However, a slightly lower value of γ mitigates the effects of increasing uncertainty regarding the state transitions over long episodes. We set this meta-parameter of our approach through cross-validation, and find that a mid-level value (0.4) works best.

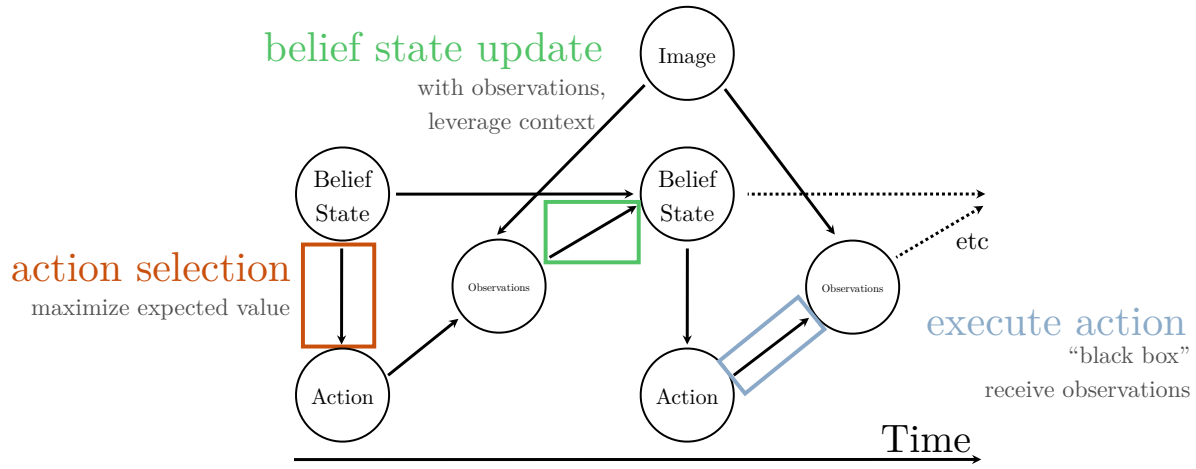


Figure 2.3: Our closed-loop method consists of selecting an action based on the belief state, executing action almost as a "black box," which makes our method very general, and then updating the state with the received observations.

Our sequential method is visually summarized in Figure 2.3.

2.2.3 Reward definition

AP of state

The policy's performance at time t is determined by all detections that are part of the set of observations $\mathbf{o}^j$ at the last state s^j before t. Recall that detector actions returns lists of detection hypotheses. Therefore, the final AP vs. Time evaluation of an episode is a function $eval(h, T_s, T_d)$ of the history of execution $h = s^0, s^1, \ldots, s^J$. It is precisely the normalized area under the AP vs. Time curve between T_s and T_d, as determined by the detections in $\mathbf{o}^j$ for all steps j in the episode.

Additive rewards

Note from Figure 2.5a that this evaluation function is additive per action, as each action a generates observations that may raise or lower the mean AP of the results so far (Δap) and takes a certain time (Δt). We can accordingly represent the final evaluation $eval(h, T_s, T_d)$ in terms of individual action rewards: $\sum_{j=0}^{J} R(s^j, a^j)$. Specifically, as shown in Figure 2.5a, we define the *reward* of an action a as

$$R(s^j, a) = \Delta \text{ap}(t_T^j - \frac{1}{2}\Delta t) \tag{2.5}$$

where t_T^j is the time left until T_d at state s^j, and Δt and Δap are the time taken and AP change produced by the action a. (We do not account for T_s here for clarity of exposition.)

2.2.4 Features of the state

State

We refer to the information available to the decision process as the *state* s. The state includes the current estimate of the distribution over class presence variables $P(\mathbf{C}) = \{P(C_0), \ldots, P(C_K)\}$, where we write $P(C_k)$ to mean $P(C_k = 1)$ (class k is present in the image). Additionally, the state records that an action a_i has been taken by adding it to the initially empty set $\mathcal{O}$ and recording the resulting observations o_i. We refer to the current set of observations as $\mathbf{o} = \{o_i | a_i \in \mathcal{O}\}$. The state also keeps track of the time into episode t, and the setup and deadline times T_s, T_d.

Open vs Closed Loc Dynamic features

An *open-loop* policy, such as the common classifier cascade (Viola and Jones 2004), takes actions in a sequence that does not depend on observations received from previous actions. In contrast, as presented in Figure 2.3, our goal is to learn a dynamic, or *closed-loop*, policy, which would exploit the signal in scene and inter-object context for a maximally efficient path through the actions. Recall from Equation 2.3 that our policy is determined by a linear function of the features of the state.

Since we want to be able to learn a dynamic policy, the observations $\mathbf{o}$ that are part of the state s should play a role in determining the value of a potential action. We include the following quantities as features $\phi(s, a)$:

$P(C_a)$	The prior probability of the class that corresponds to the detector of action a (omitted for the scene-context action).
$P(C_0\|\mathbf{o}) \ldots P(C_K\|\mathbf{o})$	The probabilities for all classes, conditioned on the current set of observations.
$H(C_0\|\mathbf{o}) \ldots H(C_K\|\mathbf{o})$	The entropies for all classes, conditioned on the current set of observations.

Additionally, we include the mean and maximum of $[H(C_0|\mathbf{o}) \ldots H(C_K|\mathbf{o})]$, and 4 time features that represent the times until start and deadline, for a total of $F = 1 + 2K + 6$ features.

Augmented MDP

Our system as any system of interesting complexity, runs into two related limitations of MDPs: the state has to be functionally approximated instead of exhaustively enumerated; and some aspects of the state are not observed, making the problem a Partially Observed MDP (POMDP), for which exact solution methods are intractable for all but rather small problems (Roy and Gordon 2002). Our initial solution to the problem of partial observability is to include features corresponding to our level of uncertainty into the feature representation, as in the technique of *augmented* MDPs (Kwok and Fox 2004).

2.2.4.1 Updating with observations

Class correlations

The bulk of our feature representation is formed by probability of individual class occurrence, conditioned on the observations so far: $P(C_0|\mathbf{o}) \dots P(C_K|\mathbf{o})$. This allows the action-value function to learn correlations between presence of different classes, and so the policy can look for the most probable classes given the observations. However, higher-order co-occurrences are not well represented in this form. Additionally, updating $P(C_i|\mathbf{o})$ presents choices regarding independence assumptions between the classes.

Update methods

We evaluate two approaches for updating probabilities: *direct* and *MRF*. In the *direct* method, $P(C_i|\mathbf{o}) = score(C_i)$ if $\mathbf{o}$ includes the observations for class C_i and $P(C_i|\mathbf{o}) = P(C_i)$ otherwise. This means that an observation of class i does not directly influence the estimated probability of any class but C_i. The *MRF* approach employs a pairwise fully-connected Markov Random Field (MRF), as shown in Figure 2.1, with the observation nodes set to $score(C_i)$ appropriately, or considered unobserved.

Learning MRF

The graphical model structure is set as fully-connected, but some classes almost never co-occurr in our dataset. Accordingly, the edge weights are learned with L_1 regularization, which obtains a sparse structure (Lee, Ganapathi, and Koller 2006). All parameters of the model are trained on fully-observed data, and Loopy Belief Propagation inference is implemented with an open-source graphical model package (Jaimovich and Mcgraw 2010).

Details

An implementation detail: $score(C_i)$ for a_{det_i} is obtained by training a probabilistic classifier on the list of detections, featurized by the top few confidence scores and the total number of detections. Similarly, $score(C_i)$ for a_{gist} is obtained by training probabilistic classifiers on the GIST feature, for all classes.

2.3 Evaluation

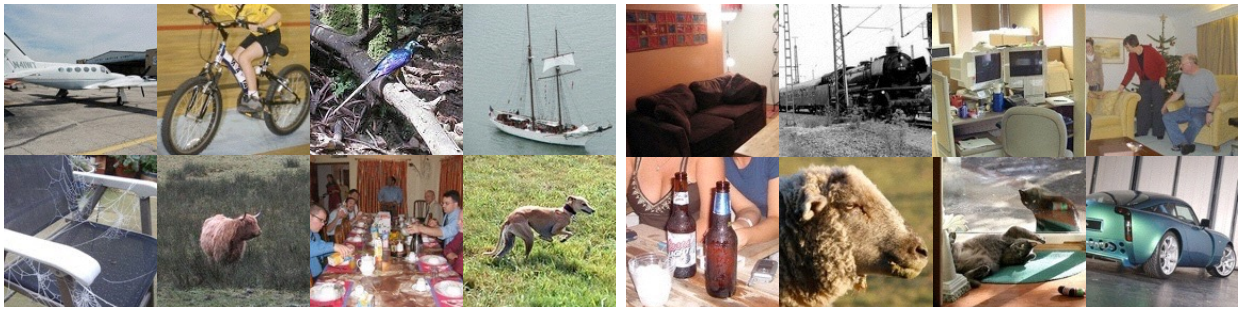


Figure 2.4: The PASCAL VOC is an object detection dataset presenting a challenging variety of image and object appearance.

Dataset

We evaluate our system on the multi-class, multi-label detection task, as previously described. Each detection episode takes an image and outputs detections with associated times, based on the order of actions taken. We evaluate on a popular detection challenge task: the PASCAL VOC 2007 dataset (Everingham et al. 2010). Example images from the dataset are shown in Figure 2.4. This datasets exhibits only a modest amount of class co-occurrence: the "person" class is highly likely to occur, and less than 10% of the images have more than two classes.

Metric

The final evaluation pools all detections up to a certain time, and computes their multi-class AP per image, averaging over all images. This is done for different times to plot the AP vs. Time curve over the whole dataset. Our method of averaging per-image performance follows (Desai, Ramanan, and Fowlkes 2011).

Detector

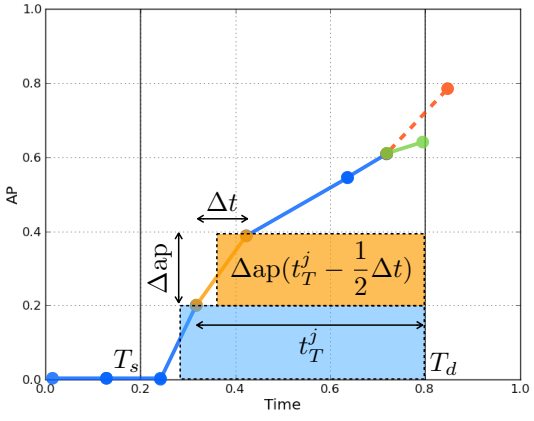
For the detector actions, we use one-vs-all cascaded deformable part-model detectors on a HOG featurization of the image (Felzenszwalb, Girshick, and McAllester 2010), with linear classification of the list of detections as described in the previous section. There are 20 classes in the PASCAL challenge task, so there are 20 detector actions. Running a detector on a PASCAL image takes about 1 second.

Training

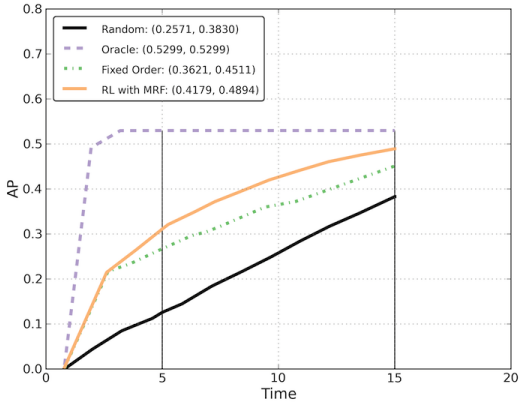
We learn weights on the training and validation sets, and run our policy on all images in the testing set. With pre-computed detections on the PASCAL VOC 2007 dataset, our training procedure takes about 4 hours on an 8-core *Xeon E5620* machine.

Settings

We test three different settings of the start and deadline times. In the first one, the start time is immediate and execution is cut off at 20 seconds, which is enough time to run all actions. In the second one, execution is cut off after only 10 seconds. Lastly, we measure performance between 5 seconds and 15 seconds. These operating points show how our method behaves when deployed in different conditions. The results are given in rows of Table 2.1.



(a) Graphical representation of the reward function.



(b) AP vs. Time curves for Random, Oracle, the Fixed Order baseline, and our best-performing policy.

Table 2.1: The areas under the AP vs. Time curve for different experimental conditions.

Bounds	Random	Fixed Order	RL	RL w/ GIST	Oracle
(0,20)	0.250	0.342	0.378	**0.382**	0.488
(0,10)	0.119	0.240	0.266	**0.267**	0.464
(5,15)	0.257	0.362	0.418	**0.420**	0.530

Baselines

We establish the first baseline for our system by selecting actions randomly at each step. As shown in Figure 2.5b, the **Random** policy results in a roughly linear gain of AP vs. time. This is expected: the detectors are capable of obtaining a certain level of performance; if half the detectors are run, the expected performance level is half of the maximum level. To establish an upper bound on performance, we plot the **Oracle** policy, obtained by re-ordering the actions at the end of each detection episode in the order of AP gains they produced. We consider another baseline: selecting actions in a fixed order based on the value they bring to the AP vs. Time evaluation, which is roughly proportional to their occurrence probability. We refer to this as **Fixed Order**.

Ours

Then there are instantiations of our method, as described in the previous section : **RL w/ Direct** inference and **RL w/ MRF** inference. As the **MRF** model consistently outperformed **Direct** by a small margin, we report results for that model only. In additional to the detector actions, we include a scene-level GIST feature that updates the posterior probabilities of all classes. This is considered one action, takes about 0.3 seconds, and brings another boost in performance. The results are shown in Table 2.1.

Trajectories

In Figure 2.5b, we can see that due to the dataset bias, the fixed-order policy performs well at first, as the person class is disproportionately likely to be in the image, but is significantly overtaken by our model as execution goes on and more rare classes have to be detected. It is additionally informative to consider the action trajectories of different policies in Figure 2.6. In contrast to the fixed order policy, our method is clearly dynamic, jumping from action to action based on the observations obtained from previous actions.

Learned weights

As an illustration, we visualize the learned weights on features as described in Section 2.2.4 in Figure 2.7. The weights are reshaped such that each row shows the weights learned for an action, with the top row representing the scene context action and then next 20 rows corresponding to the PASCAL VOC class detector actions. We note that the GIST action is learned to never be taken in the greedy ($\gamma = 0$) setting, but is learned to be taken with a higher value of γ.

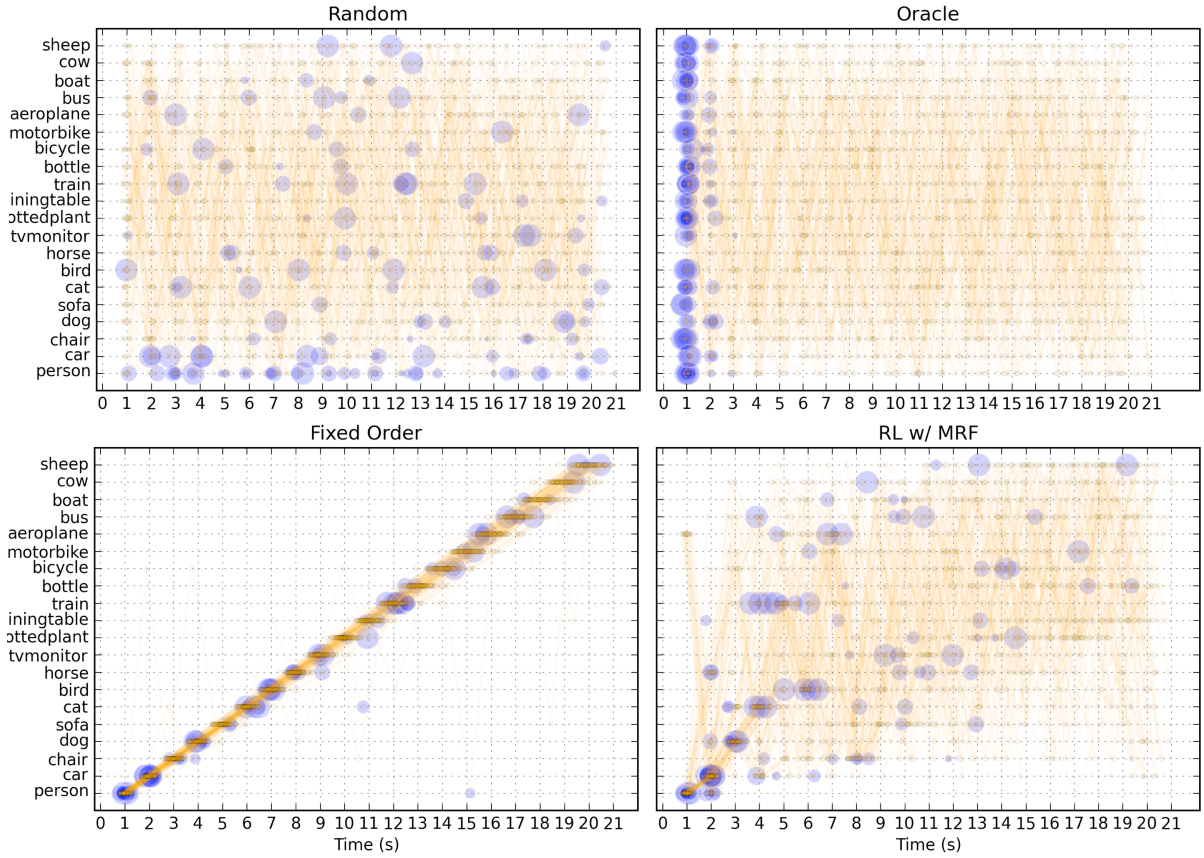
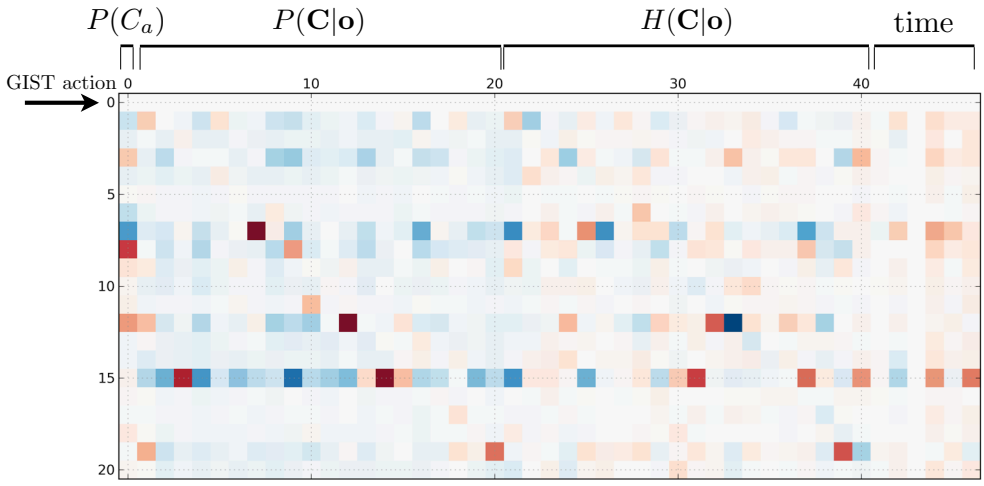
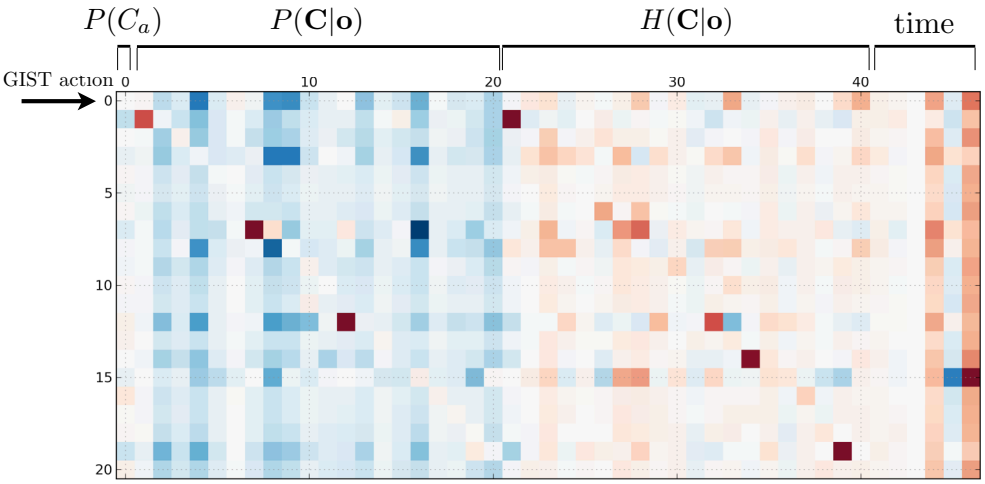


Figure 2.6: Visualizing the action trajectories of different policies. Action selection traces are plotted in orange over many episodes; the size of the blue circles correspond to the increase in AP obtained by the action. We see that the **Random** policy selects actions and obtains rewards randomly, while the **Oracle** policy obtains all rewards in the first few actions. The **Fixed Order** policy selects actions in a static optimal order. Our **RL w/ MRF** policy does not stick a static order but selects actions dynamically to maximize the rewards obtained early on.



(a) Greedy



(b) Reinforcement Learning

Figure 2.7: Learned policy weights θ_π (best viewed in color: red corresponds to positive, blue to negative values). The first row corresponds to the scene-level action, which does not generate detections itself but only helps reduce uncertainty about the contents of the image. Note that in the greedy learning case, this action is learned to never be taken, but it is shown to be useful in the reinforcement learning case.

Chapter 3

Reinforcement Learning for Anytime Classification

Motivation

In the previous chapter, running a detector was a very costly action. In other settings, such as classification problems, we may want to efficiently select a subset of fast features to compute. For Anytime performance, we should be able to effectively classify any subset of features our policy selects.

3.1 Problem definition

Definition 2. ***The test-time efficient multi-class classification problem*** *consists of*

- *N instances labeled with one of K labels: $\mathcal{D} = \{x_n \in \mathcal{X}, y_n \in \mathcal{Y} = \{1, \dots, K\}\}_{n=1}^{N}$.*
- *F features $\mathcal{H} = \{h_f : \mathcal{X} \mapsto \mathbb{R}^{d_f}\}_{f=1}^{F}$, with associated costs c_f.*
- *Budget-sensitive loss $\mathcal{L}_\mathcal{B}$, composed of cost budget $\mathcal{B}$ and loss function $\ell(\hat{y}, y) \mapsto \mathbb{R}$.*

The goal is to find a ***feature selection*** *policy $\pi(x) : \mathcal{X} \mapsto 2^\mathcal{H}$ and a* ***feature combination*** *classifier $g(\mathcal{H}_\pi) : 2^\mathcal{H} \mapsto \mathcal{Y}$ such that such that the total budget-sensitive loss $\sum \mathcal{L}_\mathcal{B}(g(\pi(x_n)), y_n)$ is minimized.*

Budget

The cost of a selected feature subset $\mathcal{H}_{\pi(x)}$ is $C_{\mathcal{H}_\pi(x)}$. The budget-sensitive loss $\mathcal{L}_\mathcal{B}$ presents a hard budget constraint by only accepting answers with $C_\mathcal{H} \leq \mathcal{B}$. Additionally, $\mathcal{L}_\mathcal{B}$ can be **cost-sensitive**: answers given with less cost are more valuable than costlier answers. As discussed in Chapter 1, the motivation for the latter property is Anytime performance; we should be able to stop our algorithm's execution at any time and have the best possible answer.

Cost

Feature costs c_f can be specified flexibly, with options including theoretical analysis, number of flops, wall clock runtime, total CPU time, or exact power expenditure. We believe that a deployment in a modern datacenter is most likely to optimize for power expenditure. In the absence of reliable ways to measure power, we use total CPU time to define the cost: if an operation is performed in parallel on multiple cores, its cost is considered to be the total cpu time on all cores.

Shared features

The features h_f can be classifier outputs, possibly multi-class; following convention, we refer to such features as *weak learners*. For a weak learner h_f, the cost c_f is composed of the cost of an underlying feature extraction $\phi_{h_f}(x)$ and the cost of subsequent classification. Once h_f is computed, its underlying feature ϕ is considered to be free for all other features h'_f that share it, if any, such that $c'_f < c_f$. Note that in state-of-the-art object recognition systems, complex features and feature encodings are often followed by linear classifiers, and feature extraction cost dominates the total cost.

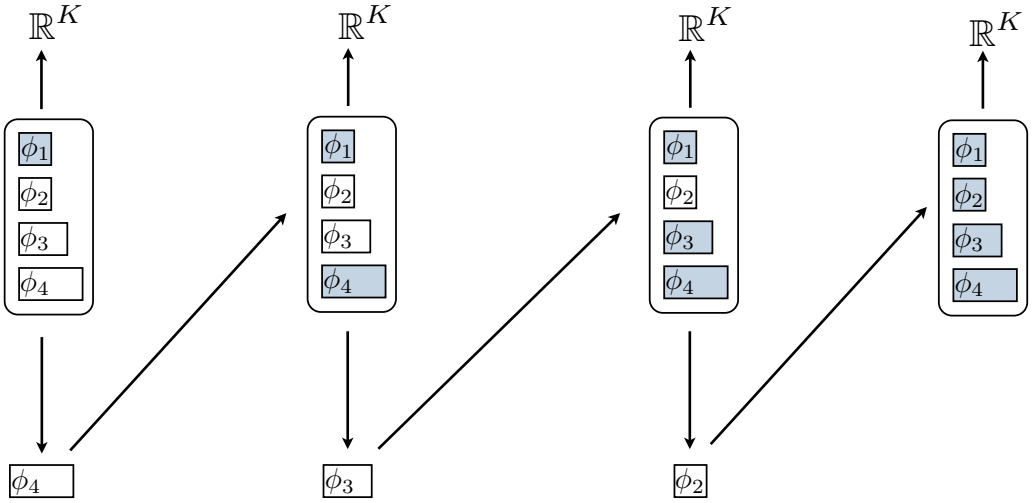


Figure 3.1: We model the problem of state traversal as a Markov Decision Process. For every state, we learn to select action of maximum expected value. The state is updated with the result of the selected action. We train a classifier on subsets of features, to give answer at any time.

Train vs. test

At training time, our computation is unbudgeted, and we can compute all features to have *fully-observed* training instances. At test time, there is a budget and so the instances we classify will only be *partially-observed*, as determined by the feature selection policy. Figure 3.1 shows the sequential process we are learning.

3.2 Method

MDP summary

We model the **feature selection** policy $\pi(x) : \mathcal{X} \mapsto 2^{\mathcal{A}}$ following the reinforcement learning approach as described in Section 2.2.1. The set of **actions** $\mathcal{A}$ is exactly the set of features $\mathcal{H}$. The policy learning approach remains the same, including implementation details. In summary, we learn the θ by *policy iteration*. First, we gather (s, a, r, s') samples by running episodes (to completion) with the current policy parameters θ_i. From these samples, $\hat{Q}(s, a)$ values are computed, and θ_{i+1} are given by L_2-regularized least squares solution to $\hat{Q}(s, a) = \theta^T \phi(s, a)$, on all states that we have seen in training.

Classifier

Three things are different: the reward definition, the state featurization function, and the additional dependence on a classifier. We defer discussion of learning the **feature combination** classifier $g(\mathcal{H}_\pi) : 2^{\mathcal{H}} \mapsto \mathcal{Y}$ to Section 3.2.3. For now, we assume that g can combine an arbitrary subset of features and provide a distribution $P(Y = y)$. For example, g could be a Naive Bayes (NB) model trained on the fully-observed data.

3.2.1 Reward definition

Loss vs. cost

The budget-sensitive loss $\mathcal{L}_\mathcal{B}$ enforces Anytime performance by valuing early gains more than later gains. To formalize this, consider Figure 3.2, which shows the entropy and the 0-1 loss of g at every point in a sequential feature selection episode for some instance x. For the best Anytime performance, we want to capture the most area above the loss vs. cost curve, up to max budget $\mathcal{B}$. Recall from (2.1) that the value of an episode ξ is defined as the sum of obtained rewards. If the reward of a single action is defined as the area above the curve that is captured as a direct result, then the value of the whole episode exactly corresponds to $\mathcal{L}_\mathcal{B}$.

Infogain

However, there is a problem with using loss directly: only the first action to "tip the scale" toward the correct prediction gets a direct reward (in the figure, it is the first action). A smoother reward function is desirable: if the classifier g can give a full distribution $P(Y = y \mid \mathcal{H}_{\pi(x)})$ and not just a prediction $\hat{y} \in \mathcal{Y}$, we can maximize the *information gain* of the selected subset instead of directly minimizing the loss of $g(\pi(x))$:

$$
\begin{aligned}
I(Y; \mathcal{H}_{\pi(x)}) &= H(Y) - H(Y|\mathcal{H}_{\pi(x)}) = \\
&= \sum_{y \in Y} P(y) \log P(y) - \\
&\quad \sum_{y, \mathcal{H}_{\pi(x)}} P(y, \mathcal{H}_{\pi(x)}) \log P(y \mid \mathcal{H}_{\pi(x)})
\end{aligned}
\tag{3.1}
$$

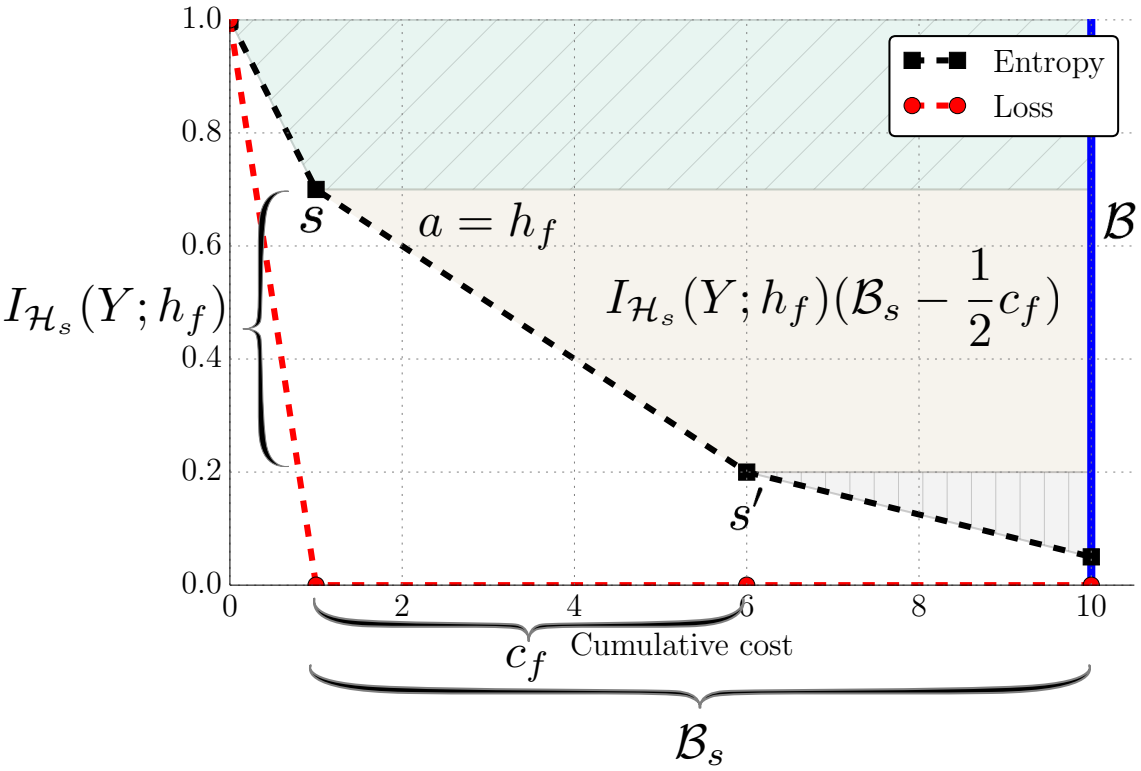


Figure 3.2: Definition of the reward function. To maximize the total area above the entropy vs. cost curve from 0 to $\mathcal{B}$, we define the reward of an individual action as the area of the slice of the total area that it contributes. From state s, action $a = h_f$ leads to state s' with cost c_f. The information gain is $I_{\mathcal{H}_s}(Y; h_f) = H(Y; \mathcal{H}_s) - H(Y; \mathcal{H}_s \cup h_f)$.

Definition

To the extent that g is unbiased, maximizing information gain corresponds to minimizing loss, and ensures that we not only make the right classification decision but also become maximally certain. Therefore, as graphically presented in Figure 3.2, we define the reward of selecting feature h_s with cost c_f with the set $\mathcal{H}_s$ computed to be $I_{\mathcal{H}_s}(Y; h_f)(\mathcal{B}_s - \frac{1}{2}c_f)$. Although we do not evaluate in this regime, note that this definition easily incorporates a **setup cost** in addition to **deadline cost** by only computing the area in between setup and deadline costs.

3.2.2 Features of the state

The featurization function $\phi(s)$ extracts the following features from the state:

- Bit vector $\mathbf{m}$ of length F: initially all bits are 1 and are set to 0 when the corresponding feature is computed.
- For each h_f, a vector of size d_f representing the values; 0 until observed.

- Cost feature $c \in [0, 1]$, for fraction of the budget spent.
- Bias feature 1.

Static vs Dynamic

These features define the **dynamic** state, presenting enough information to have a *closed-loop* (dynamic) policy that may select different features for different test instances. The **static** state has all of the above features except for the observed feature values. This enables only an *open-loop* (static) policy, which is exactly the same for all instances. Policy learned with the static state is used as a baseline in experiments.

3.2.3 Learning the classifier

Classifiers

We have so far assumed that g can combine an arbitrary subset of features and provide a distribution $P(Y = y)$—for example, a Gaussian Naive Bayes (NB) model trained on the fully-observed data. However, a Naive Bayes classifier suffers from its restrictive independence assumptions. Since discriminative classifiers commonly provide better performance, we'd like to use a **logistic regression** classifier. Nearest Neighbor methods also provide a robust classifier for partially observed data, but are slow for large datasets. We consider them in this exposition and in preliminary experiments reported in Appendix A, but do not use them in final policy learning experiments due to this problem.

Missing features

Note that at test time, some feature values are missing. If the classifier is trained exclusively on fully-observed data, then the feature value statistics at test time will not match, resulting in poor performance. Therefore, we need to learn classifier weights on a distribution of data that exhibits the pattern of missing features induces by the policy π, and/or try to intelligently impute unobserved values.

Input: $\mathcal{D} = \{x_n, y_n\}_{n=1}^N$; $\mathcal{L}_\mathcal{B}$
Result: Trained π, g

$\pi_0 \leftarrow$ random;
for $i \leftarrow 1$ **to** *max_iterations* **do**
    States, Actions, Costs, Labels $\leftarrow$ `GatherSamples(`$\mathcal{D}$, π_{i-1}`)`;
    $g_i \leftarrow$ `UpdateClassifier(`*States, Labels*`)`;
    Rewards $\leftarrow$ `ComputeRewards(`*States, Costs, Labels,* $g_i, \mathcal{L}_\mathcal{B}, \gamma$`)`;
    $\pi_i \leftarrow$ `UpdatePolicy(`*States, Actions, Rewards*`)`;
end

Algorithm 1: Because reward computation depends on the classifier, and the distribution of states depends on the policy, g and π are trained iteratively.

Joint learning

At the same time, learning the policy depends on the classifier g, used in the computation of the rewards. For this reason, the policy and classifier need to be learned jointly: Algorithm 1 gives the iterative procedure. In summary, we start from random π and g, gather a batch of trajectories. The batch is used to update both g and π. Then new trajectories are generated with the updated π, rewards are computed using the updated g, and the process is repeated.

3.2.3.1 Unobserved value imputation

Formulation

Unlike the Naive Bayes classifier, the logistic regression classifier is not able to use an arbitrary subset of features $\mathcal{H}_\pi$, but instead operates on feature vectors of a fixed size. To represent the feature vector of a fully observed instance, we write $\mathbf{x} = [h_1(x), \ldots, h_f(x)]$. In case that $\mathcal{H}_\pi \subset \mathcal{H}$, we need to fill in unobserved feature values in the vector.

Formulation

We can think of the policy as a test-time feature selection function $o(x, b) : \mathcal{X} \times \mathbb{R} \mapsto \mathbb{B}^F$, where $\mathbb{B} \in \{0, 1\}$, and $b \in [0, 1]$ is given and specifies the fractional selection *budget*. Applied to x, $o(x, b)$ gives the binary selection vector $\mathbf{o}$ which splits $\mathbf{x}$ it into observed and unobserved parts such that $\mathbf{x}^m = [\mathbf{x}^\text{o}, \mathbf{x}^\text{u}]$. X^c denotes the fully observed $N \times F$ training set. We assume that we have only sequential access to the missing-feature test instances $\mathbf{x}^m$.

Mean imputation

A basic strategy is **mean imputation**: filling in with the mean value of the feature. We simply replace the missing value in X^m with their row mean in X^c. Because our data is always zero-mean, this amounts to simply replacing the value with 0.

$$\mathbf{x}_\pi = \left[h_i(x) : \begin{cases} h_i(x) & \text{if } h_i \in \mathcal{H}_{\pi(x)} \\ \bar{\mathbf{h}}_i & \text{otherwise} \end{cases} \right] \tag{3.2}$$

Gaussian Imputation

If we assume that $\mathbf{x}$ is distributed according to a multivariate Gaussian $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \Sigma)$, where Σ is the sample covariance $X^T X$ and the data is standardized to have zero mean, then it is possible to do **Gaussian imputation**. Given a feature subset $\mathcal{H}_\pi$, we write:

$$\mathbf{x}_\pi = \begin{bmatrix} \mathbf{x}^\text{o} \\ \mathbf{x}^\text{u} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{B} \end{bmatrix} \right) \tag{3.3}$$

where $\mathbf{x}^\text{o}$ and $\mathbf{x}^\text{u}$ represent the respectively observed and unobserved parts of the full feature vector $\mathbf{x}$, $\mathbf{A}$ is the covariance matrix of (here and elsewhere, the part of X^c corresponding to) $\mathbf{x}^\text{o}$, $\mathbf{B}$ is the covariance matrix of $\mathbf{x}^\text{u}$, and C is the cross-variance matrix that has as many rows as the size of $\mathbf{x}^\text{o}$ and as many columns as the size of $\mathbf{x}^\text{u}$ (Roweis 1999). In this case, the distribution over unobserved variables conditioned on the observed variables is given as $\mathbf{x}^\text{u} \mid \mathbf{x}^\text{o} \sim \mathcal{N}\left(\mathbf{C}^T\mathbf{A}^{-1}\mathbf{x}^\text{o}, \mathbf{B} - \mathbf{C}^T\mathbf{A}^{-1}\mathbf{C}\right)$.

Complexity

After computing the complete covariance matrix Σ, which takes $O(N^3)$ time, we need to make N' test-time predictions. In the course of the predictions, we may need to compute at most $\min(N', 2^F)$ unique matrix inversions (again in $O(N^3)$). The size of the matrices being inverted is proportional to the budget b, making this method slower for larger budgets.

Instead of assuming anything, we could go directly to the source of the covariances—the actual feature values for all points in the training set. The family of Nearest Neighbor methods takes this approach. The algorithm for imputation is simple: find the nearest neighbors in the observed dimensions, and use their averaged values to fill in the unobserved dimensions. For $\mathbf{X}^m$, we find the k nearest neighbors with the highest dot product similarity $\mathbf{x}^{cT}\mathbf{x}^{\mathbf{m}}$ or lowest Euclidean distance $\|\mathbf{x}^c - \mathbf{x}^m\|$, using only the features that are observed in $\mathbf{x}^m$. For **imputation**, the unobserved values are set to the average across these nearest neighbors for that dimension. Similarly, we do **classification** by returning the mode label of the nearest neighbors.

Finding the nearest neighbors by dot product similarity is $O(NF'^2)$, and Euclidean distance is the same with an additional constant term. $F'S$ is the number of observed dimensions, which grows proportionally with budget b, making this method more expensive with increased budget.

3.2.3.2 Learning more than one classifier

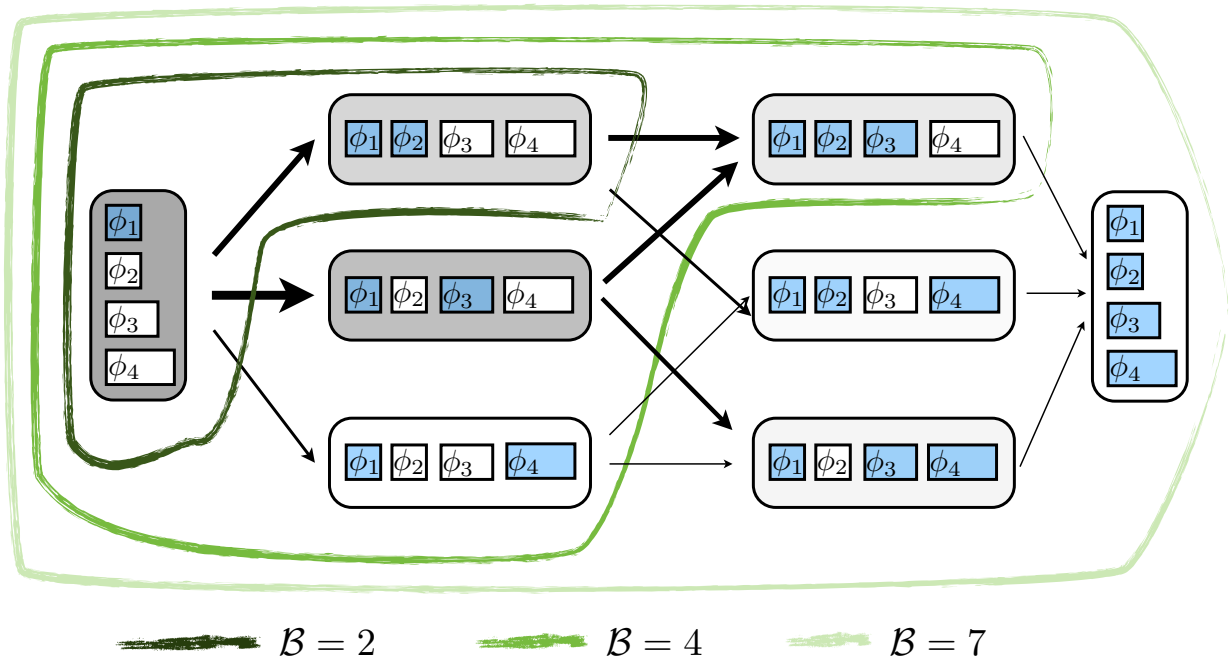


Figure 3.3: The action space $\mathcal{A}$ of the MDP is the the set of features $\mathcal{H}$, represented by the ϕ boxes. The primary discretization of the state space can be visualized by the possible feature subsets (larger boxes); selected features are colored in the diagram. The feature selection policy π induces a distribution over feature subsets, for a dataset, which is represented by the shading of the larger boxes. Not all states are reachable for a given budget $\mathcal{B}$. In the figure, we show three "budget cuts" of the state space.

Subset clustering

As illustrated in Figure 3.3, the policy π selects some feature subsets more frequently than others. Instead of learning only one classifier g that must be robust to all observed feature subsets, we can learn several classifiers, one for each of the most frequent subsets. This is done by maintaining a distribution over encountered feature subsets during training.

We use hierarchical agglomerative clustering, growing clusters bottom-up from the observed masks. In the case of training K classifiers, we need to find K clusters such that the masks are distributed as evenly as possible into the clusters. The distance measure for the binary masks is the Hamming distance; standard K-Means clustering technique is not applicable to this distance measure. Each one of K classifiers is trained with the LIBLINEAR implementation of logistic regression, with L_2 regularization parameter K-fold cross-validated at each iteration.

3.3 Evaluation

Settings

We evaluate the following sequential selection baselines:

- **Static, greedy**: corresponds to best performance of a policy that does not observe feature values and selects actions greedily ($\gamma = 0$).
- **Static, non-myopic**: policy that does not observe feature values but uses the MDP machinery with $\gamma = 1$ to consider future action rewards.
- **Dynamic, greedy**: policy that observed feature values, but selects actions greedily.

Our method is the **Dynamic, non-myopic** policy: observed feature values, and full lookahead.

Metrics

We evaluate two forms of test-time efficient performance measure: the area under the curve and the performance at max budget. (Note that all methods are trained only for the former measure.)

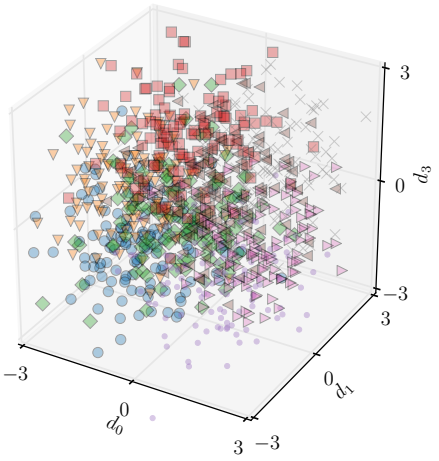
Classifier

In preliminary experiments, Logistic Regression always performed better than the Gaussian Naive Bayes classifier, and so only the former is used in the experiments below. We evaluated classification with **Gaussian** vs. **Mean** imputation, and with different number of classifiers (1, 3, and 6) clustered by feature subsets. We found that Gaussian imputation performed better than mean imputation, but not substantially, and at additional cost. Although increased number of classifiers sometimes increased performance, it also made our method prone to overfitting. In the final accounting, $K = 1$ classifiers worked best on all tasks. Results of detailed experiments on imputation and clustering are given in Appendix A. For policy experiments, we report only the best achieved imputation method.

Details

For details on the reinforcement learning implementation, see Section 2.2.1 and Section 2.3. We largely rely on classifier implementations in the `scikit-learn` package (Pedregosa et al. 2011). With pre-computed features, our training procedure takes only a few hours on an 8-core *Xeon E5620* machine for each of the experiments below.

3.3.1 Experiment: Synthetic



Feature	Number	Cost
d_i: sign of dimension i	D	1
q_o: label of datapoint, if in quadrant o	2^D	10

Figure 3.4: See text for explanation.

Setup

Following Xu et al. 2013, we first show that the policy works as advertised in a challenging synthetic example. In D-dimensional space, the data has a label for each of the 2^D orthants, and is generated by a unit-variance Gaussian in that orthant (See top left of Figure 3.4 for the 3D case). There are D cheap features that simply return the sign of the data point's coordinate for the corresponding dimension. For each orthant, there is also an expensive feature that returns the data point's label if the point is located in the corresponding orthant, and random noise otherwise.

Results

The optimal policy on a new data point is to determine its orthant with cheap features, and then take the corresponding expensive action. Note that both dynamic features and non-myopic learning are crucial to the optimal policy, which is successfully found by our approach. Figure 3.5 shows the results of this optimal policy, a random policy, and of different baselines and our method, trained given the correct minimal budget.

3.3.2 Experiment: Scene recognition

Setup

The Scene-15 dataset (**Lazebnik-CVPR-2006**) contains 4485 images from 15 visual scene classes. The task is to classify images according to scene. Following (Xiao et al. 2010), we extracted 14 different visual features (GIST, HOG, TinyImages, LBP, SIFT, Line Histograms, Self-Similarity, Textons, Color Histograms, and variations). The features vary in cost from 0.3 seconds to 8 seconds, and in single-feature accuracy from 0.32 (TinyImages) to .82 (HOG). Separate multi-class linear SVMs were trained on each feature channel, using a random 100 positive example images per class for training. We used the `liblinear` implementation, and K-fold cross-validated the penalty parameter C. The trained SVMs were evaluated on the images not used for training, resulting in a dataset of 2238 vectors of 210 confidence values: 15 classes for each of the 14 feature channels. This dataset was split 60-40 into training and test sets for our experiments.

Results

Figure 3.6 shows the results, including learned policy trajectories. For all evaluated budgets, our *dynamic, non-myopic* method outperforms all others on the area under the error vs. cost curve metric. Our results on this dataset match the reported results of Active Classification (Gao and Koller 2011) (Figure 2) and Greedy Miser (Xu, Weinberger, and Chapelle 2012) (Figure 3), although both methods use an additional powerful feature channel (ObjectBank)[1].

3.3.3 Experiment: ImageNet and maximizing specificity

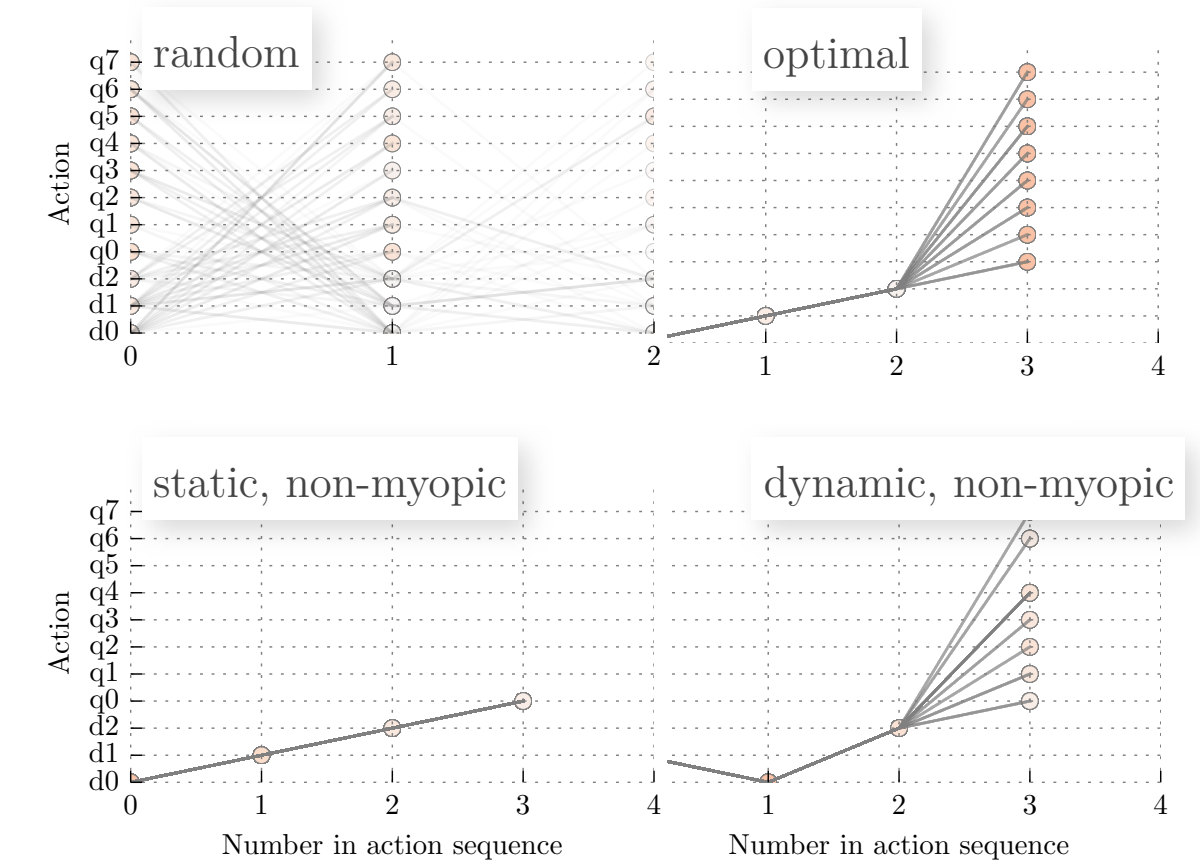
Setup

The full ImageNet dataset has over 10K categories and over a million images (Deng et al. 2010). The classes are organized in a hierarchical structure, which can be exploited for novel recognition tasks. We evaluate on a 65-class subset introduced in "Hedging Your Bets" (Deng et al. 2012). In this evaluation, we consider the situation where the initial feature computation has already happened, and the task is to find a path through existing one-vs-all classifiers: features correspond to Platt-scaled SVM confidences of leaf-node classifiers (trained on SIFT-LLC features), and each has cost 1 (Deng et al. 2010). Following (Deng et al. 2012), accuracy is defined on all nodes, and inner node confidences are obtained by summing the probabilities of the descendant nodes.

Results

We combine our sequential feature selection with the "Hedging Your Bets" method for backing off prediction nodes using the ImageNet hierarchy to maintain guaranteed accuracy while giving maximally specific answers, given a cost budget. The results are given in Figure 3.7. As the available budget increases, the *specificity* (defined by normalized information gain in the hierarchy) of our predictions also increases, while accuracy remains constant. Visualizing this on the ILSVRC-65 hierarchy, we see that the fraction of predictions at the leaf nodes grows with available computation time. This formulation presents a novel angle on modeling the time course of human visual perception.

[1] Detailed results for this and other experiments are on the project page (see front page for the link).



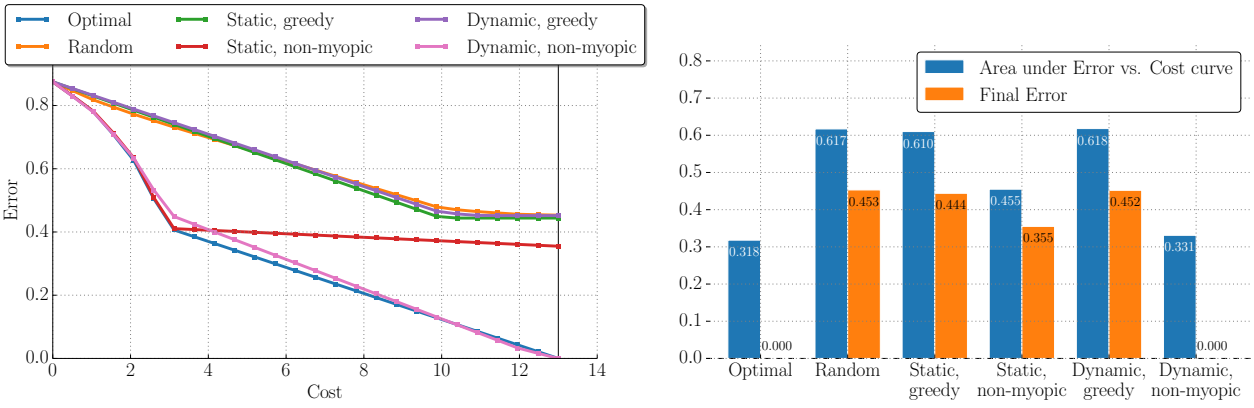
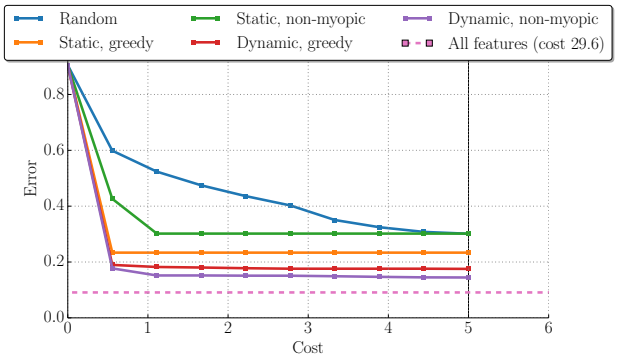
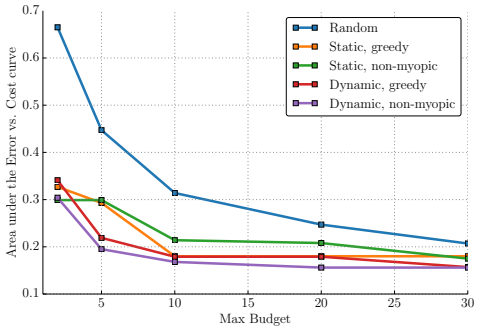


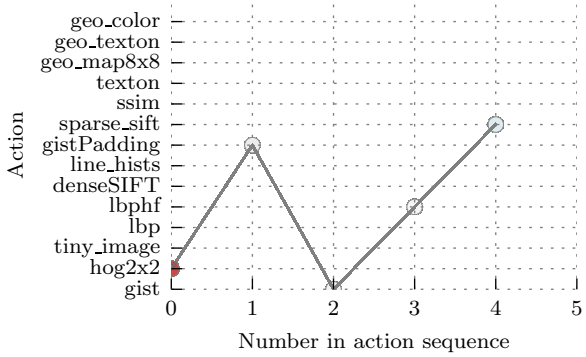
Figure 3.5: Evaluation on the synthetic example (best viewed in color). The sample feature trajectories of different policies at top: the opacity of the edges corresponds to their prevalence during policy execution; the opacity of the nodes corresponds to the amount of reward obtained in that state. Note that the *static, non-myopic* policy correctly learns to select the cheap features first, but is not able to correctly branch, while our *dynamic, non-myopic* approach finds the optimal strategy. The plots in the bottom half give the error vs. cost numbers.



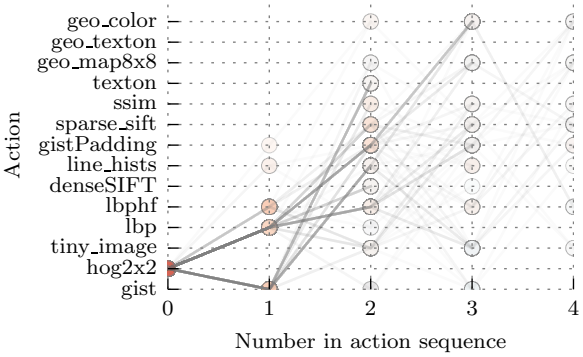
(a) Error given by policies learned for a budget = 5.



(b) Areas under error vs. cost curves of policies learned at different budgets.

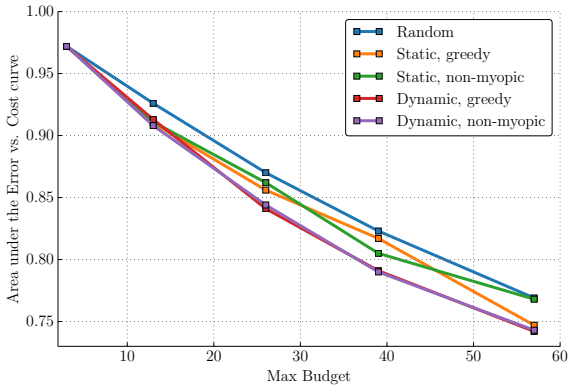


(c) Trajectory of our static policy.

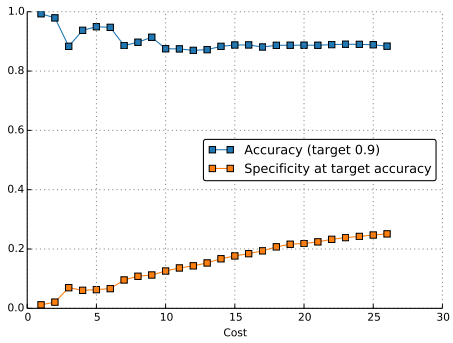


(d) Trajectory of our dynamic policy.

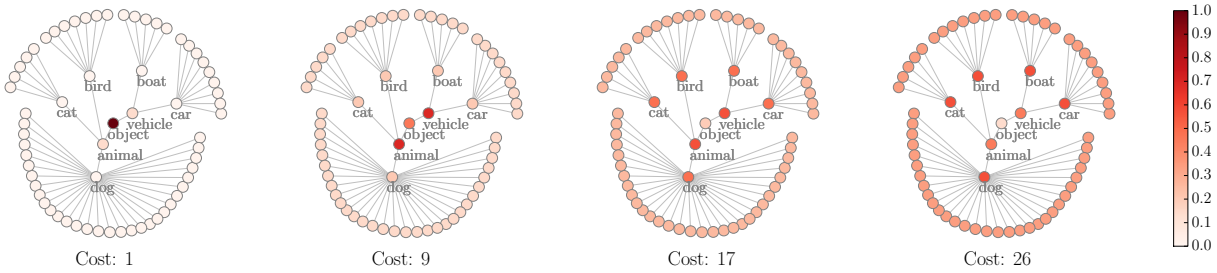
Figure 3.6: Results on Scenes-15 dataset (best viewed in color). Figure (a) shows the error vs. cost plot for policies learned given a budget of 5 seconds. Figure (b) aggregates the area under the error vs. cost plot metrics for different policies and budgets, showeing that our approach outperforms baselines no matter what budget it's trained for. Figures (d) and (e) shows the branching behavior of our dynamic policy vs the best static policy.



(a) Areas under error vs. cost curves for policies learned at different budgets. (No specificity back-off is performed here).



(b) Holding prediction accuracy constant, we achieve increased specificity with increased cost (on *Dynamic, non-myopic* policy, budget = 36).



(c) We visualize the fraction of predictions made at inner vs. leaf nodes of ILSVRC-65 at different cost points of an Anytime policy: with more computation, accurate predictions are increasingly made at the leaf nodes.

Figure 3.7: Results on the ILSVRC-65 dataset (best viewed in color). Figure (a) shows our dynamic approaches outperforming static baselines for all practical cost budgets. When our method is combined with Hedging Your Bets (Deng et al. 2012), a constant prediction accuracy can be achieved at all points in the Anytime policy, with *specificity* of predictions increasing with the cost of predictions. Figures (b) and (c) show this for the *dynamic, non-myopic* policy learned for budget = 26. This is analogous to human visual performance, which shows increased specificity at longer stimulus times.

Chapter 4

Detection with the Cascade CNN

Motivation

With the rise of Convolutional Neural Networks (CNNs) for visual recognition, the state-of-the-art in detection has shifted. Whereas in Section 2.1, a detector model was trained per class, and evaluated densely on the image, currently best-performing methods train one detector for all classes, and evaluate on a subset of possible image regions (Girshick et al. 2014). The method is still computationally expensive, taking about 10 seconds per image even with multi-threaded matrix and convolution operations. The strategy to speed detection up, however, has to change.

4.1 Method

R-CNN

We take the R-CNN ibid. method as our point of departure. As summarized in Figure 4.1, R-CNN starts with external region-of-interest proposals (ROIs), which are transformed to canonical size, and classified with a CNN, obtaining multi-class scores for each region. After all batches of ROIs have been scored, they are post-processed with non-maximal suppression and other bounding box refinement to obtain the final detections.

Our method

As summarized in Figure 4.2, we begin with the same ROI proposals, but first score all proposals with a quick-to-compute feature. We then select a batch of highly-scoring ROIs and classify them with the CNN – which is additionally sped up with cascaded structure. After the batch is scored, we optionally re-score the remaining ROIs, and repeat the process until either time or regions are depleted. The quick-to-compute feature is used to select batches of regions in a way that orders the regions most likely to contain ground truth objects earlier than other regions. This process of region selection does not reject regions flat out, but reorders them to be processed later.

Region order

We found that simply sorting the regions by score and taking batches in that sorted order results in poor performance, and can be worse than taking regions in an entirely random order, as the highest scoring regions may be highly overlapping with each other, and only result in one detection after non-maximal suppression and other post-processing of detections. Instead, we put regions in random order, set a threshold for the quick-to-compute feature such that only half of the unprocessed regions score above it, take a batch of the above-threshold regions in their order, update the threshold, and repeat.

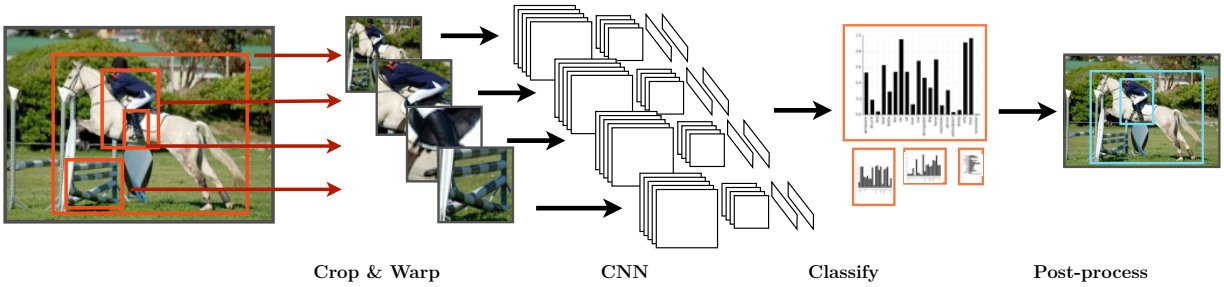


Figure 4.1: R-CNN architecture: image regions are cropped, resized, and each one fed through a CNN. The classifier outputs are post-processed to give the final detections.

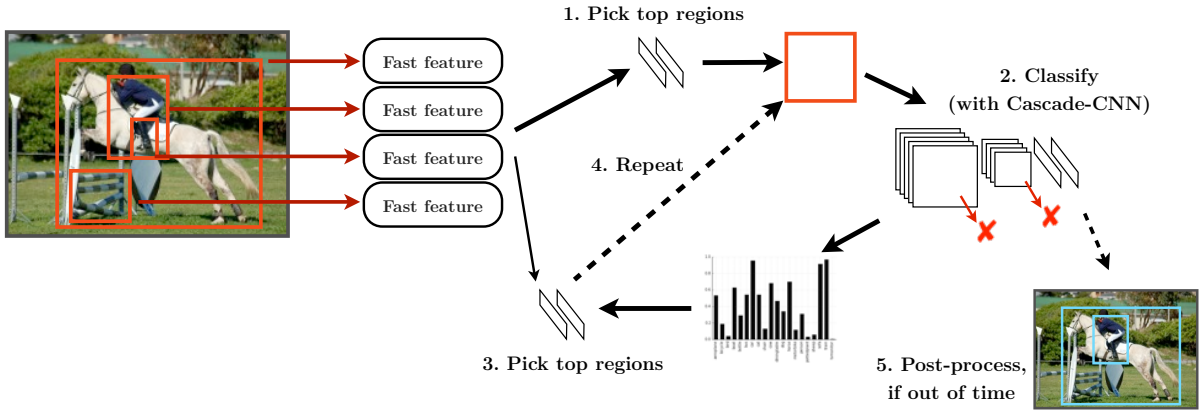


Figure 4.2: Our method scores each region of interest with a fast feature (we evaluate several), allowing us to pick promising regions first. The regions are classified with the original CNN, or a sped-up Cascade-CNN. The properties of the regions can play a role in selecting the next batch of regions.

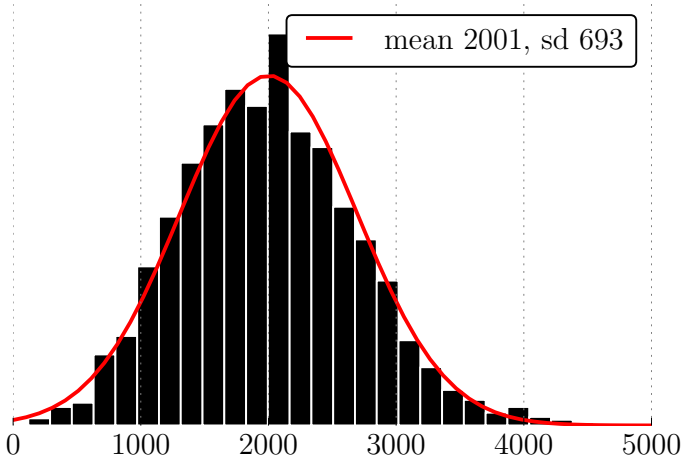


Figure 4.3: Distribution of number of regions per image.

4.1.1 Quick-to-compute feature

Region statistics

For the first source of information, we consider statistics about ROI location and overlap with other regions. For each ROI, we compute: its normalized location, its scale ($\sqrt{\text{width} \times \text{height}}$), aspect ratio ($\log \sqrt{\text{width}/\text{height}}$), and normalized counts of overlapping regions, at several PASCAL overlap thresholds (0, 0.2, 0.6). This simple feature works suprisingly well for filtering regions to process first.

Pixel gradient

In concatenation, we also consider the *pixel gradient*, back-propagated throgh a classification CNN applied to the whole image. This feature corresponds to a kind of saliency map, giving an estimate of the importance of each pixel to the final classification of the image. Our method is independently developed, but agrees with the concurrently published work of Simonyan, Vedaldi, and Zisserman 2014 quite closely. Images are resized to square and classified with an "AlexNet" Krizhevsky, Sutskever, and Hinton 2012b CNN fine-tuned on the PASCAL VOC classification dataset with multi-label loss. (Unlike the ILSVRC, the PASCAL VOC is a multi-label prediction task, with at times multiple correct labels for an image.) At test-time, the gradient at the top is with respect to the indicator function of the max-scoring class, and is back-propagated all the way to the pixels, where it is summed across channels. A couple of example images are shown in Figure 4.4. We compute an integral image on this pixel gradient map, allowing near-instant computation of sums in arbitrary regions. For each region to be evaluated, we compute the image-normalized gradient sum, sums for each of four corners, and ratio of in-region vs. out-of-region sums.

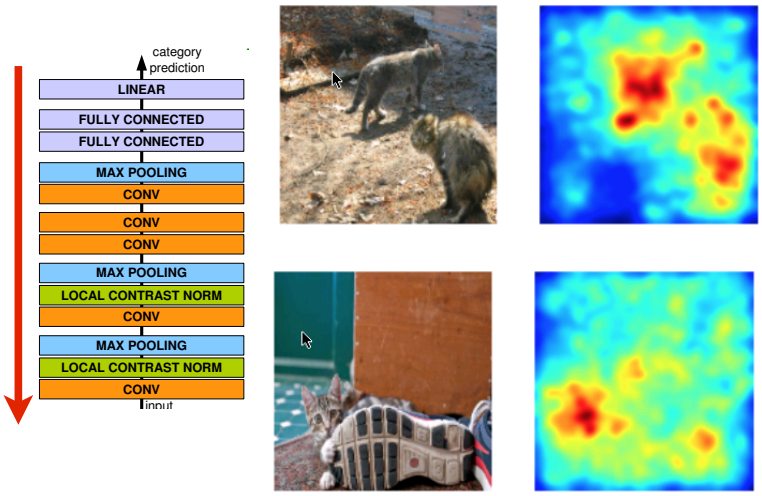


Figure 4.4: As a quick feature, we back-propagate the gradient from the top classification layer all the way to the input pixels. This induces a kind of saliency map on the image, which is useful signal for selecting image sub-regions to classify with the CNN.

4.1.2 Cascade CNN

Motivation

Despite the intentions of the region-proposal mechanism (Uijlings et al. 2013), most ROIs that are scored in R-CNN do not contain any object of interest. As in the classic cascade of Viola and Jones 2004, it would be useful to quickly reject these regions, without expending the full amount of computation on them. The problem is that the deep neural network architecture is trained to always run all the way through. We need to introduce a new primitive to enable early termination.

Method

The Cascade CNN, shown in Figure 4.5, augments the CNN network with an "Early Reject" option: after some layers, the network decides whether to keep processing the input with the next layer. Each reject layer is implemented as a fully-connected layer following a linear rectification, trained with dropout using logistic loss on foreground/background labels. We first train an AlexNet-architecture network (Krizhevsky, Sutskever, and Hinton 2012b) on the ImageNet classification dataset. This network is augmented with the Reject layers, its loss replaced with a multi-label cross-entropy loss, and fine-tuning on the PASCAL VOC training set is performed until loss stop decreasing (roughly 50000 iterations). In training, all instances pass through all Reject layers. After training, we set the reject thresholds to maintain at last 80% recall at each stage, using a large sample of regions from images in the validation set containing positive and negative examples in equal proportion.

Implementation

In the efficient implementation of CNNs we use (Caffe), there is a simple loop over each batch element in both CPU and GPU convolution code. We modify this loop to simply not perform convolution on those elements that were rejected earlier in the cascade. [1] Since most of the time is spent in the convolutional layers, we introduce a reject option after `pool1`, `pool2`, and `conv3`. The last classification layer still outputs the full multi-class scores for the surviving regions.

Timing

To estimate the saving on time by using the rejectors we timed the time spend to process 1000 regions (10 batches or 100) and the time expended in each of the first 3 layers:

- 1700 ms to process all the layers
- 270 ms (15%) in layer 1 (This includes `conv1`, `relu1`, `pool1`, `norm1`)
- 360 ms (20%) in layer 2 (This includes `conv2`, `relu2`, `pool2`, `norm2`)
- 285 ms (15%) in layer 3 (This includes `conv3`, `relu3`)

Therefore the expected "lifetimes" of regions rejected after layer 1, layer 2, and layer 3 are 0.15, 0.35, and 0.5 of the total time taken per region.

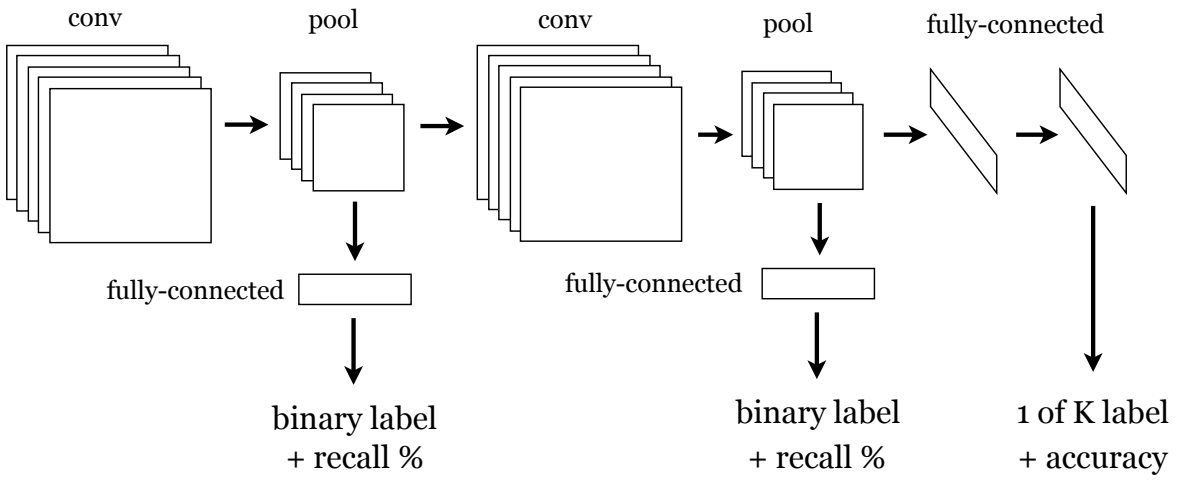


Figure 4.5: The Cascade CNN has a Reject option after computationally expensive layers, implemented as a binary prediction for reject/keep (background/foreground for our detection task). The goal of the Reject layer is to maintain high recall while culling as much of the batch as possible, so that we can avoid doing as much convolution in the next layer.

[1] While memory remains occupied in this scheme, we do not consider this a problem.

4.2 Evaluation

Dataset

We evaluate on the standard object detection benchmark: the PASCAL VOC Everingham et al. 2010. In all cases, the CNN region classifiers are trained on the PASCAL VOC 2007 trainval set. The parameters of our methods are set by training or cross-validation on the VOC 2007 val set. We evaluate on the VOC 2007 test set. The result plots and details are shown in Figure 4.6 and Table 4.1.

Implementation

The scoring function for the quick-to-compute features is trained by a logistic regression classifier onto the max PASCAL overlap with any ground truth window on the validation dataset. The classifier is optimized by stochastic gradient descent, and its regularization parameter is cross-validated. The R-CNN software was used as available in June 2014. [2] That software relies on Selective Search Uijlings et al. 2013 region proposals. Different images are proposed different numbers of regions. Figure 4.3 shows the distribution of number of regions on the validation set, with the parameters of the R-CNN. An additional parameter is the size of each batch of regions that goes through the CNN. We set batch size to 100 regions, and observe that it takes on average 500 ms to process them with the CNN. In all experiments, we use Ubuntu 12.04, Intel i5 3.2GHz CPU, and NVIDIA Tesla K40 GPU.

Table 4.1: Full table of AP vs. Time results on PASCAL VOC 2007. Best performance for each time point is in bold.

Time allotted (ms)	0	300	600	1300	1800	3600	7200	10000
Original	0	0.000	0.176	0.211	0.244	0.368	0.496	0.544
Random	0	0.000	0.295	0.381	0.426	0.504	0.536	0.544
C-CNN	0	**0.327**	0.430	0.493	0.510	0.528	0.528	-
Region Selection w/ Gradient	0	0.000	0.424	0.469	0.490	0.526	**0.542**	0.544
C-CNN, Region Selection w/ Gradient	0	0.198	**0.442**	**0.502**	**0.517**	**0.528**	0.528	-

The experimental settings are

Original
: The original order of the Selective Search regions of interest. This order is influenced by the hierarchical segmentation of their method, and so has sequences of highly overlapping regions.

Random
: A completely blind permutation of the original order.

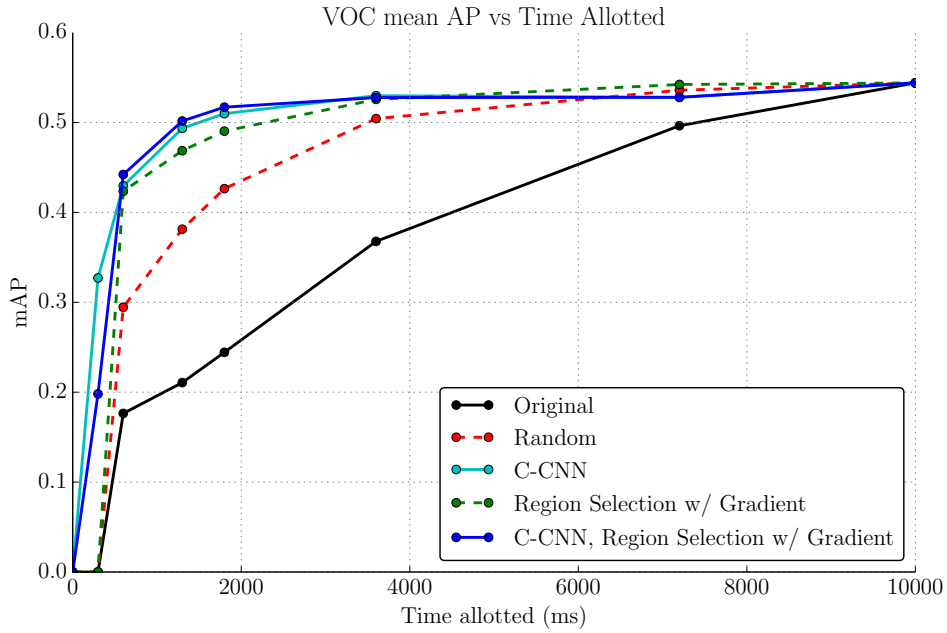
[2] https://github.com/rbgirshick/rcnn

Region Selection
The region statistics feature is always used. Additionally, we consider the Pixel Gradient feature, with *setup time* of the gradient forward-back propagation of 20 ms.

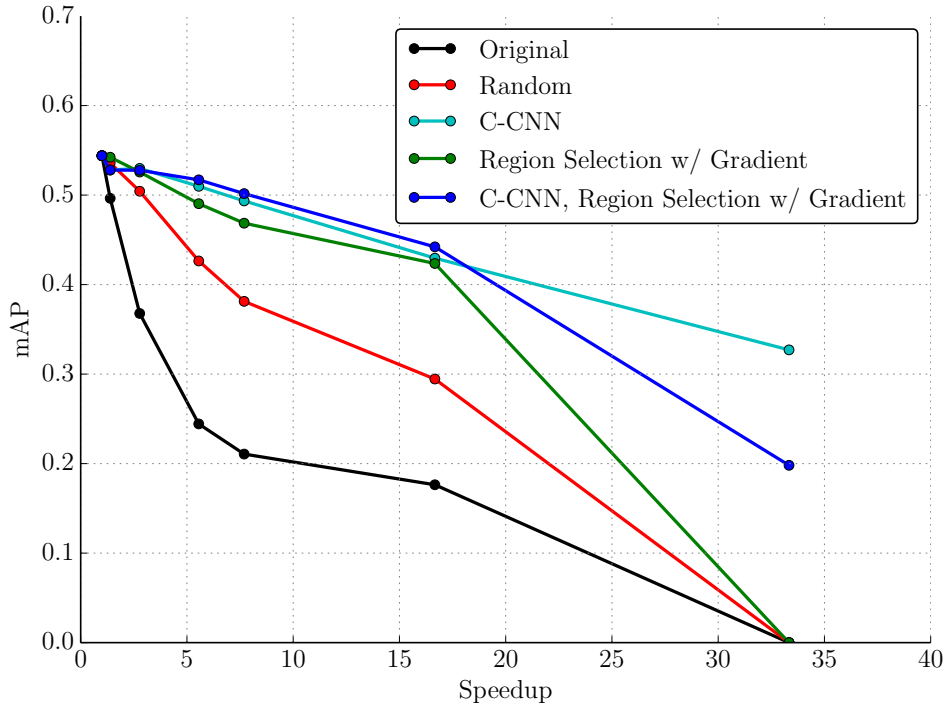
Cascade CNN
The Cascade CNN model, as described in Section 4.1.2. The first experiment (C-CNN) takes batches of regions in a random order. The next two experiments also make use of the Region Selection methodology with the quick-to-compute feature.

Analysis

Since the time to process a full batch with a non-Cascade CNN is 500 ms, there are no results for non-cascaded baselines at 300 ms. At this time, the Cascade CNN without any region ordering is best. A reason for why C-CNN with Region Selection is not as good at this point is that the region selection presents better region candidates, with fewer rejection opportunities, and thus has less coverage of the image. At 600 ms, C-CNN method have had more than one batch go through, and the Region Selection is giving it a lead over the simple C-CNN. Both method are better than the baseline non-cascaded methods for this entire duration.



(a) Plotting Mean AP vs. Time Allotted allows comparison performance at a given time budget. For example, at 1300 ms, random region selection gets about 0.42 mAP, while our best method (C-CNN with gradient-based region selection) obtains 0.50 mAP.



(b) Plotting mean AP vs. speed-up factor allows comparison of speed-ups at a given mAP point. For example, we can see that we should obtain mAP of 0.40 at around 20x speedup with our method.

Figure 4.6: Results of the Cascade CNN and other Anytime methods on the PASCAL VOC 2007 dataset.

Chapter 5

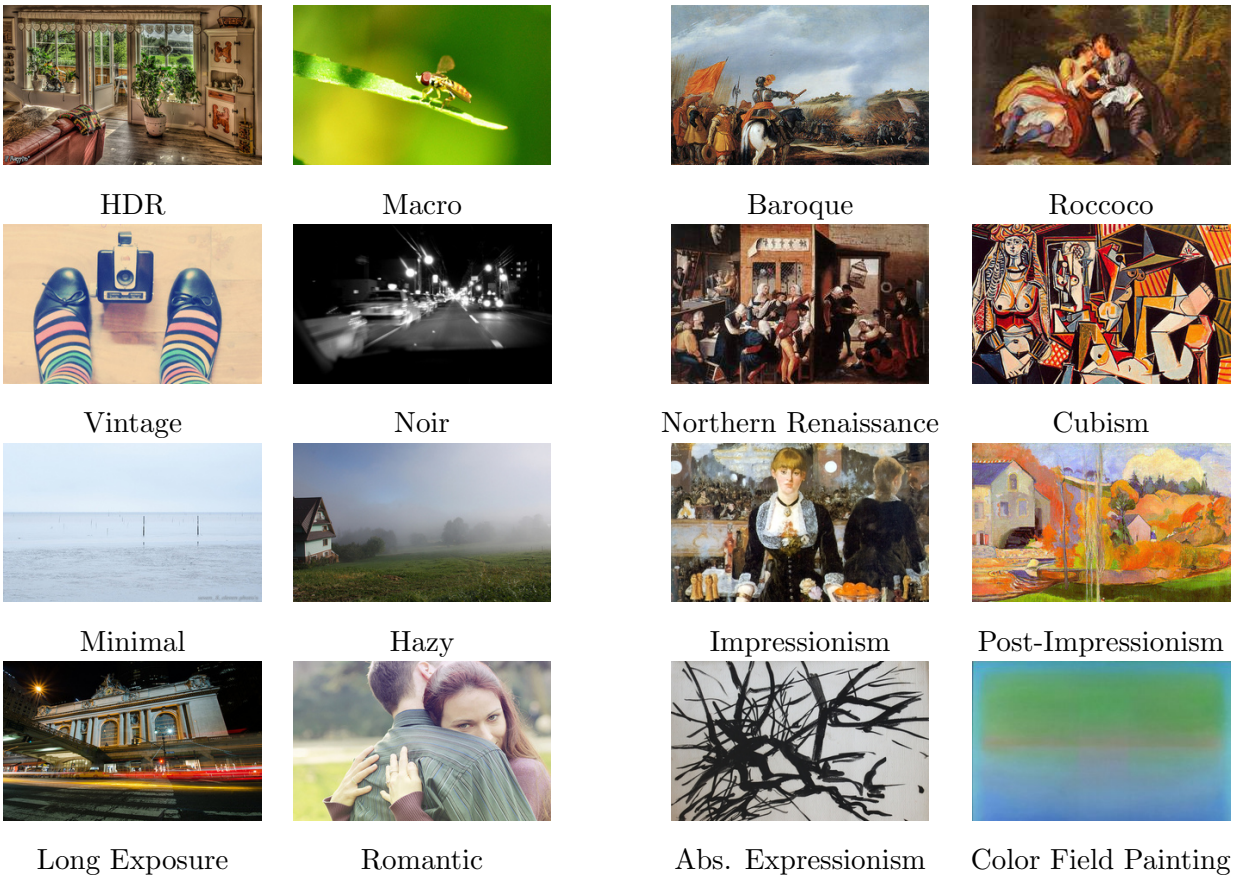
Recognizing Image Style

Motivation

Deliberately-created images convey meaning, and *visual style* is often a significant component of image meaning. For example, a political candidate portrait made in the lush colors of a Renoir painting tells a different story than if it were in the harsh, dark tones of a horror movie. Distinct visual styles are apparent in art, cinematography, advertising, and have become extremely popular in amateur photography, with apps like Instagram leading the way. While understanding style is crucial to image understanding, very little research in computer vision has explored visual style.

Definition

We define several different *types* of image style, and gather a new, large-scale dataset of photographs annotated with style labels. This dataset embodies several different aspects of visual style, including photographic techniques ("Macro," "HDR"), composition styles ("Minimal," "Geometric"), moods ("Serene," "Melancholy"), genres ("Vintage," "Romantic," "Horror"), and types of scenes ("Hazy," "Sunny"). These styles are not mutually exclusive, and represent different attributes of style. We also gather a large dataset of visual art (mostly paintings) annotated with art historical style labels, ranging from Renaissance to modern art. Figure 5.1 shows some samples. Analyzing style of non-photorealistic media is interesting as much of our present understanding of visual style arises out of thousands of years of developments in fine art, marked by distinct historical styles.



(a) Flickr Style: 80K images covering 20 styles. (b) Wikipaintings: 85K images for 25 genres.

Figure 5.1: Typical images in different style categories of our datasets.

5.1 Method

We test existing classification algorithms on these styles, evaluating several state-of-the-art image features. Most previous work in aesthetic style analysis has used hand-tuned features, such as color histograms. We find that deep convolutional neural network (CNN) features perform best for the task. This is surprising for several reasons: these features were trained on object class categories (ImageNet), and many styles appear to be primarily about color choices, yet the CNN features handily beat color histogram features. This leads to one conclusion of our work: mid-level features derived from object datasets are generic for style recognition, and superior to hand-tuned features. We compare our predictors to human observers, using Amazon Mechanical Turk experiments, and find that our classifiers predict Group membership at essentially the same level of accuracy as Turkers. We also test on the AVA aesthetic prediction task (Murray, Marchesotti, and Perronnin 2012), and show that using the "deep" object recognition features improves over the state-of-the-art results.

Applications and code

First, we demonstrate an example of using our method to search for images by style. This could be useful for applications such as product search, storytelling, and creating slide presentations. In the same vein, visual similarity search results could be filtered by visual style, making possible queries such as "similar to this image, but more Film Noir." Second, style tags may provide valuable mid-level features for other image understanding tasks. For example, there has increasing recent effort in understanding image meaning, aesthetics, interestingness, popularity, and emotion (for example, (Gygli, Nater, and Gool 2013; Isola et al. 2011; Joo et al. 2014; Khosla, Sarma, and Hamid 2014)), and style is an important part of meaning. Finally, learned predictors could be a useful component in modifying the style of an image. All data, trained predictors, and code (including results viewing interface) are available at http://sergeykarayev.com/recognizing-image-style/.

5.1.1 Data Sources

Motivation

Building an effective model of photographic style requires annotated training data. To our knowledge, there is only one existing dataset annotated with visual style, and only a narrow range of photographic styles is represented (Murray, Marchesotti, and Perronnin 2012). We would like to study a broader range of styles, including different *types* of styles ranging from genres, compositional styles, and moods. Morever, large datasets are desirable in order to obtain effective results, and so we would like to obtain data from online communities, such as Flickr.

5.1.1.1 Flickr Style

Source

Although Flickr users often provide free-form tags for their uploaded images, the tags tend to be quite unreliable. Instead, we turn to Flickr groups, which are community-curated collections of visual concepts. For example, the Flickr Group "Geometry Beauty" is described, in part, as "Circles, triangles, rectangles, symmetric objects, repeated patterns", and contains over 167K images at time of writing; the "Film Noir Mood" group is described as "Not just black and white photography, but a dark, gritty, moody feel..." and comprises over 7K images.

At the outset, we decided on a set of 20 visual styles, further categorized into types:

- **Optical techniques:** Macro, Bokeh, Depth-of-Field, Long Exposure, HDR
- **Atmosphere:** Hazy, Sunny
- **Mood:** Serene, Melancholy, Ethereal
- **Composition styles:** Minimal, Geometric, Detailed, Texture
- **Color:** Pastel, Bright
- **Genre:** Noir, Vintage, Romantic, Horror

For each of these stylistic concepts, we found at least one dedicated Flickr Group with clearly defined membership rules. From these groups, we collected 4,000 positive examples for each label, for a total of 80,000 images. Example images are shown in Figure 5.1a.

Dealing with Noise

The derived labels are considered clean in the positive examples, but may be noisy in the negative examples, in the same way as the ImageNet dataset (Deng et al. 2009). That is, a picture labeled as *Sunny* is indeed *Sunny*, but it may also be *Romantic*, for which it is not labeled. We consider this an unfortunate but acceptable reality of working with a large-scale dataset. Following ImageNet, we still treat the absence of a label as indication that the image is a negative example for that label. Mechanical Turk experiments described in subsubsection 5.2.1.1 serve to allay our concerns.

5.1.1.2 Wikipaintings

Source

We also provide a new dataset for classifying painting style. To our knowledge, no previous large-scale dataset exists for this task – although very recently a large dataset of artwork did appear for other tasks (Mensink and Gemert 2014). We collect a dataset of 100,000 high-art images – mostly paintings – labeled with artist, style, genre, date, and free-form tag information by a community of experts on the `Wikipaintings.org` website. Our dataset presents significant stylistic diversity, primarily spanning Renaissance styles to modern art movements.We select 25 styles with more than 1,000 examples, for a total of 85,000 images. Example images are shown in Figure 5.1b.

5.1.2 Learning algorithm

Multi-Class Logistic Regression via SGD

We learn to classify novel images according to their style, using the labels assembled in the previous section. Because the datasets we deal with are quite large and some of the features are high-dimensional, we consider only linear classifiers, relying on sophisticated features to provide robustiness. We use an open-source implementation of Stochastic Gradient Descent with adaptive subgradient (Agarwal et al. 2012). The learning process optimizes the function

$$\min_w \lambda_1 \|w\|_1 + \frac{\lambda_2}{2}\|w\|_2^2 + \sum_i \ell(x_i, y_i, w)$$

We set the L_1 and L_2 regularization parameters and the form of the loss function by validation on a held-out set. For the loss $\ell(x, y, w)$, we consider the hinge $(\max(0, 1 - y \cdot w^T x))$ and logistic $(\log(1 + \exp(-y \cdot w^T x)))$ functions. We set the initial learning rate to 0.5, and use adaptive subgradient optimization (Duchi, Hazan, and Singer 2011). Our setup is of multi-class classification; we use the One vs. All reduction to binary classifiers.

5.1.3 Image Features

In order to classify styles, we must choose appropriate image features. We hypothesize that image style may be related to many different features, including low-level statistics (Lyu, Rockmore, and Farid 2004), color choices, composition, and content. Hence, we test features that embody these different elements, including features from the object recognition literature. We evaluate single-feature performance, as well as second-stage fusion of multiple features.

L*a*b color histogram. Many of the Flickr styles exhibit strong dependence on color. For example, *Noir* images are nearly all black-and-white, while most *Horror* images are very dark, and *Vintage* images use old photographic colors. We use a standard color histogram feature, computed on the whole image. The 784-dimensional joint histogram in CIELAB color space has 4, 14, and 14 bins in the L*, a*, and b* channels, following Palermo et al. (Palermo, Hays, and Efros 2012), who showed this to be the best performing single feature for determining the date of historical color images.

GIST. The classic gist descriptor (Oliva and Torralba 2001) is known to perform well for scene classification and retrieval of images visually similar at a low-resolution scale, and thus can represent image composition to some extent. We use the INRIA LEAR implementation, resizing images to 256 by 256 pixels and extracting a 960-dimensional color GIST feature.

Graph-based visual saliency. We also model composition with a visual attention feature (Harel, Koch, and Perona 2006). The feature is fast to compute and has been shown to predict human fixations in natural images basically as well as an individual human (humans are far better in aggregate, however). The 1024-dimensional feature is computed from images resized to 256 by 256 pixels.

Meta-class binary features. Image content can be predictive of individual styles, e.g., *Macro* images include many images of insects and flowers. The `mc-bit` feature (Bergamo and Torresani 2012) is a 15,000-dimensional bit vector feature learned as a non-linear combination of classifiers trained using existing features (e.g., SIFT, GIST, Self-Similarity) on thousands of random ImageNet synsets, including internal ILSVRC2010 nodes. In essence, MC-bit is a hand-crafted "deep" architecture, stacking classifiers and pooling operations on top of lower-level features.

Table 5.1: Mean APs on three datasets for the considered single-channel features and their second-stage combination. As some features were clearly worse than others on the AVA Style dataset, only the better features were evaluated on larger datasets.

	Fusion x Content	$DeCAF_6$	MC-bit	L*a*b* Hist	GIST	Saliency	random
AVA Style	**0.581**	0.579	0.539	0.288	0.220	0.152	0.132
Flickr	**0.368**	0.336	0.328	-	-	-	0.052
Wikipaintings	**0.473**	0.356	0.441	-	-	-	0.043

Deep convolutional net. Current state-of-the-art results on ImageNet, the largest image classification challenge, have come from a deep convolutional network trained in a fully-supervised manner (Krizhevsky, Sutskever, and Hinton 2012a). We use the Caffe (Jia 2013) open-source implementation of the ImageNet-winning eght-layer convolutional network, trained on over a million images annotated with 1,000 ImageNet classes. We investigate using features from two different levels of the network, referred to as $DeCAF_5$ and $DeCAF_6$ (following (Donahue et al. 2013b)). The features are 8,000- and 4,000-dimensional and are computed from images center-cropped and resized to 256 by 256 pixels.

Content classifiers. Following Dhar et al. (Dhar, Berg, and Brook 2011), who use high-level classifiers as features for their aesthetic rating prediction task, we evaluate using object classifier confidences as features. Specifically, we train classifiers for all 20 classes of the PASCAL VOC (Everingham et al. 2010) using the $DeCAF_6$ feature. The resulting classifiers are quite reliable, obtaining 0.7 mean AP on the VOC 2012. We aggregate the data to train four classifiers for "animals", "vehicles", "indoor objects" and "people".

Fusion. These aggregate classes are presumed to discriminate between vastly different types of images – types for which different style signals may apply. For example, a *Romantic* scene with people may be largely about the composition of the scene, whereas, *Romantic* scenes with vehicles may be largely described by color. To enable our classifiers to learn content-dependent style, we can take the outer product of a feature channel with the four aggregate content classifiers.

5.2 Evaluation

5.2.1 Experiment: Flickr Style

Setup

We learn and predict style labels on the 80,000 images labeled with 20 different visual styles of our new Flickr Style dataset, using 20% of the data for testing, and another 20% for parameter-tuning validation. There are several performance metrics we consider. Average Precision evaluation (as reported in Table 5.1 and in Table B.2) is computed on a random class-balanced subset of the test data (each class has equal prevalence). We compute confusion matrices (Figure B.3, Figure B.4) on the same data. Per-class accuracies are computed on subsets of the data balanced by the binary label, such that chance performance is 50%. We follow these decisions in all following experiments.

Single features

The best single-channel feature is DeCAF_6 with 0.336 mean AP; feature fusion obtains 0.368 mean AP. Per-class APs range from 0.17 [*Depth of Field*] to 0.62 [*Macro*]. Per-class accuracies range from 68% [*Romantic*, *Depth of Field*] to 85% [*Sunny*, *Noir*, *Macro*]. The average per-class accuracy is 78%. We show the most confident style classifications on the test set of Flickr Style in Figure 5.5 through Figure B.2.

Analysis

Upon inspection of the confusion matrices, we saw points of understandable confusion: *Depth of Field* vs. *Macro*, *Romantic* vs. *Pastel*, *Vintage* vs. *Melancholy*. There are also surprising sources of mistakes: *Macro* vs. *Bright/Energetic*, for example. To explain this particular confusion, we observed that lots of *Macro* photos contain bright flowers, insects, or birds, often against vibrant greenery. Here, at least, the content of the image dominates its style label. To explore further content-style correlations, we plot the outputs of PASCAL object class classifiers (one of our features) on the Flickr dataset in Figure 5.2. We can observe that some styles have strong correlations to content (e.g., *Hazy* occurs with *vehicle*, *HDR* doesn't occur with *cat*).

Fusion

We hypothesize that style is content-dependent: a *Romantic* portrait may have different low-level properties than a *Romantic* sunset. We form a new feature as an outer product of our content classifier features with the second-stage late fusion features ("Fusion × Content" in all results figures). These features gave the best results, thus supporting the hypothesis.

5.2.1.1 Mechanical Turk Evaluation

Setup

In order to provide a human baseline for evaluation, we performed a Mechanical Turk study. For each style, Turkers were shown positive and negative examples for each Flickr Group, and then they evaluated whether each image in the test set was part of the given style. We treat the Flickr group memberships as ground truth as before, and then evaluate Turkers' ability to accurately determine group membership. Measures were taken to remove spam workers, as described below. For efficiency, one quarter of the test set was used, and two redundant styles (Bokeh and Detailed) were removed. Each test image was evaluated by 3 Turkers, and the majority vote taken as the human result for this image.

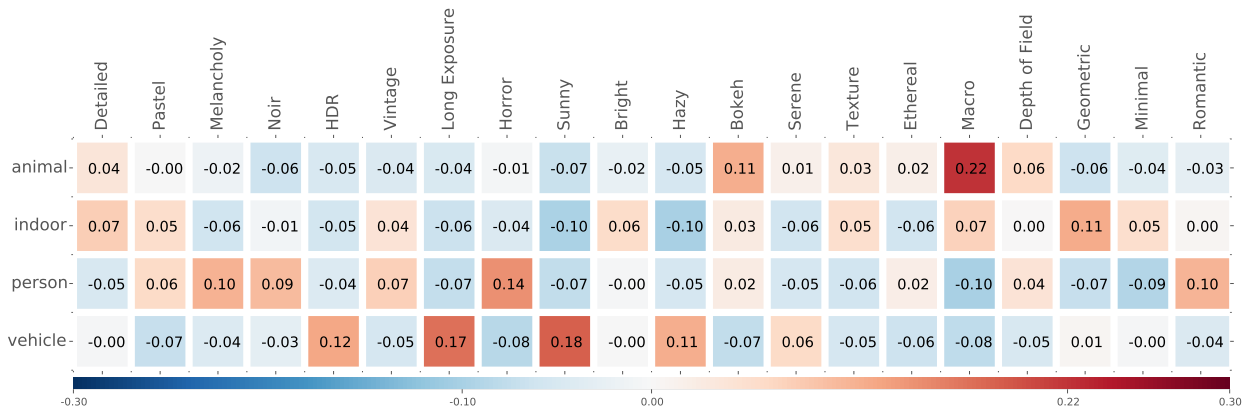


Figure 5.2: Correlation of PASCAL content classifier predictions (rows) against ground truth Flickr Style labels (columns). We see, for instance, that the Macro style is highly correlated with presence of animals, and that Long Exposure and Sunny style photographs often feature vehicles.

Details

Test images were grouped into 10 images per Human Interface Task (HIT). Each task asks the Turker to evaluate the style (e.g., "Is this image VINTAGE?") for each image. For each style, we provided a short blurb describing the style in words, and provided 12-15 hand-chosen positive and negative examples for each Flickr Group. To defend against spammers, each HIT included 2 sentinels: images which were very clearly positives and similar to the examples. HITs were rejected when Turkers got both sentinels wrong. Turkers were paid 0.10 per HIT, and were allowed to perform multiple hits. Manual inspection of the results indicate that the Turkers understood the task and were performing effectively. A few Turkers sent unsolicited feedback indicating that they were really enjoying the HITs ("some of the photos are beautiful") and wanted to perform them as effectively as possible.

Results

Results are presented in Table B.3 and Table B.4. In total, Turkers achieved 75% mean accuracy (ranging from 61% [*Romantic*] to 92% [*Macro*]) across styles, in comparison to 78% mean accuracy (ranging from 68% [*Depth of Field*] to 87% [*Macro*]) of our best method. Our algorithm did significantly worse than Turkers on *Macro* and *Horror*, and significantly better on *Vintage*, *Romantic*, *Pastel*, *Detailed*, *HDR*, and *Long Exposure* styles. We additionally used the Turker opinion as ground truth for our method's predictions. In switching from the default Flickr to the MTurk ground truth, our method's accuracy hardly changed from 78% to 77%. However, we saw that the accuracy of our *Vintage*, *Detailed*, *Long Exposure*, *Minimal*, *HDR*, and *Sunny* style classifiers significantly decreased, indicating machine-human disagreement on those styles.

Analysis

Some of this variance may be due to subtle difference from the Turk tasks that we provided, as compared to the definitions of the Flickr groups, but may also due to the Flickr groups' incorporating images that do not quite fit the common definition of the given style. For example, there may be a mismatch between different notions of *Romantic* and *vintage*, and how inclusively these terms are defined. Regardless, the high agreement seen in study validates our choice of data source.

5.2.2 Experiment: Wikipaintings

Setup and Results

With the same setup and features as in the Flickr experiments, we evaluate 85,000 images labeled with 25 different art styles. Detailed results are provided in Table B.5 and Table B.6. The best single-channel feature is MC-bit with 0.441 mean AP; feature fusion obtains 0.473 mean AP. Per-class accuracies range from 72% [*Symbolism*, *Expressionism*, *Art Nouveau*] to 94% [*Ukiyo-e*, *Minimalism*, *Color Field Painting*]. We did not perform a Mechanical Turk analysis of this dataset, as the Wikipainting community-of-experts labels were deemed inherently less noisy than Flickr Groups.

5.2.3 Experiment: AVA Style

Setup and Results

AVA (Murray, Marchesotti, and Perronnin 2012) is a dataset of 250K images from `dpchallenge.net`. We evaluate classification of aesthetic rating and of 14 different photographic style labels on the 14,000 images of the AVA dataset that have such labels. For the style labels, the publishers of the dataset provide a train/test split, where training images have only one label, but test images may have more than one label (ibid.). Our results are presented in Table B.1. For style classification, the best single feature is the DeCAF_6 convolution network feature, obtaining 0.579 mean AP. Feature fusion improves the result to 0.581 mean AP; both results beat the previous state-of-the-art of 0.538 mean AP (ibid.). Our results beat 0.54 mAP using both the AVA-provided class-imbalanced test split, and the class-balanced subsample that we consider to be more correct evaluation, and for which we provide numbers.

Low-level features

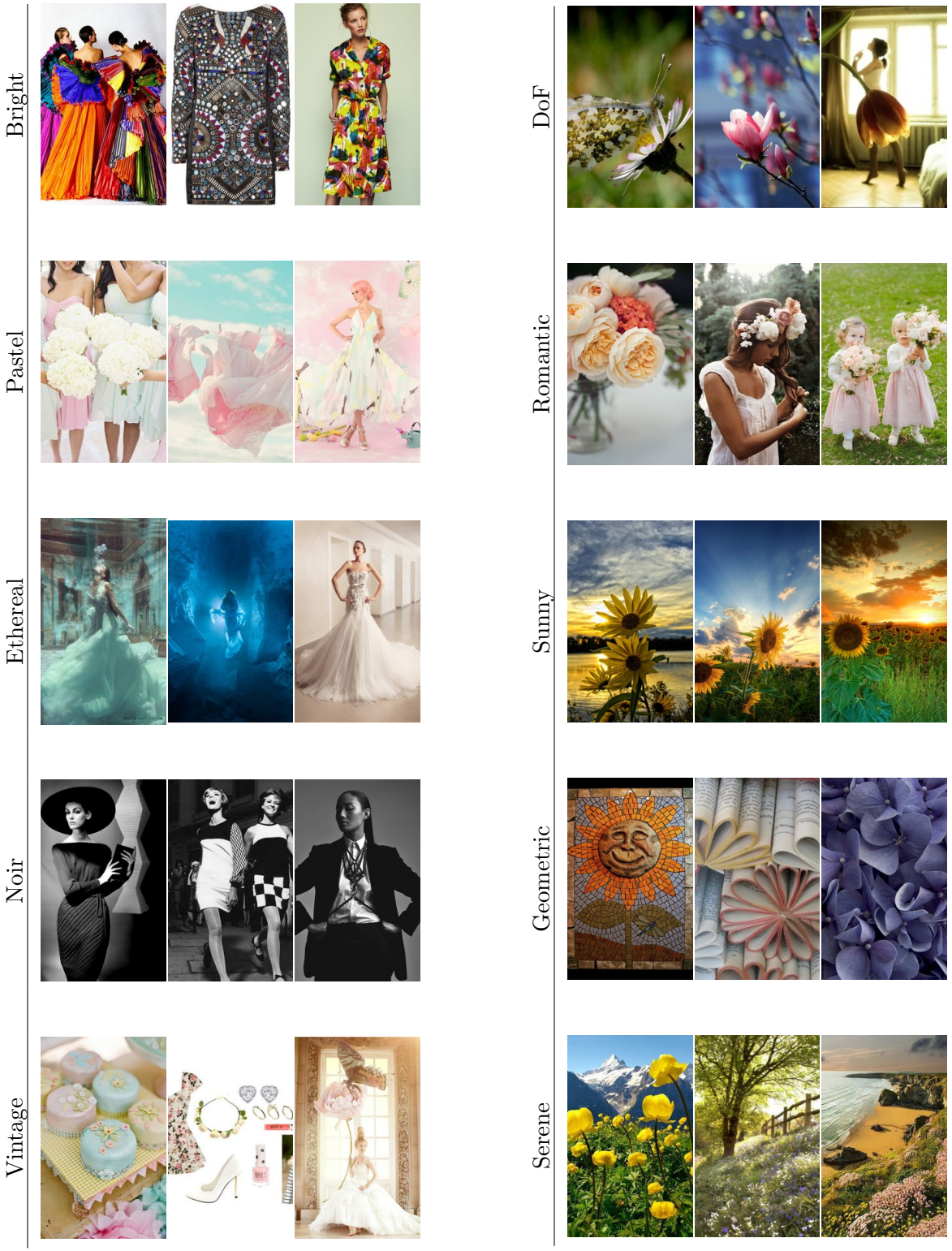
In all metrics, the DeCAF and MC-bit features significantly outperformed more low-level features on this dataset. For this reason, we did not evaluate the low-level features on the larger Flickr and Wikipaintings datasets (the AVA experiment was chronologically first).

5.2.4 Application: Style-Based Image Search

Style classifiers learned on our datasets can be used toward novel goals. For example, sources of stock photography or design inspiration may be better navigated with a vocabulary of style. Currently, companies expend labor to manually annotate stock photography with such labels. With our approach, any image collection can be searchable and rankable by style. To demonstrate, we apply our Flickr-learned style classifiers to a new dataset of 80K images gathered on Pinterest (also available with our code release); some results are shown in Figure 5.4. Interestingly, styles learned from photographs can be used to order paintings, and styles learned from paintings can be used to order photographs, as illustrated in Figure 5.3.

Cross-dataset style

Figure 5.3: Cross-dataset style. On the left are shown top scorers from the Wikipaintings set, for styles learned on the Flickr set. On the right, Flickr photographs are accordingly sorted by Painting style. (Figure best viewed in color.)



(a) Query: "dress". (b) Query: "flower".

Figure 5.4: Example of filtering image search results by style. Our Flickr Style classifiers are applied to images found on Pinterest. The images are searched by the text contents of their captions, then filtered by the response of the style classifiers. Here we show three out of top five results for different query/style combinations.

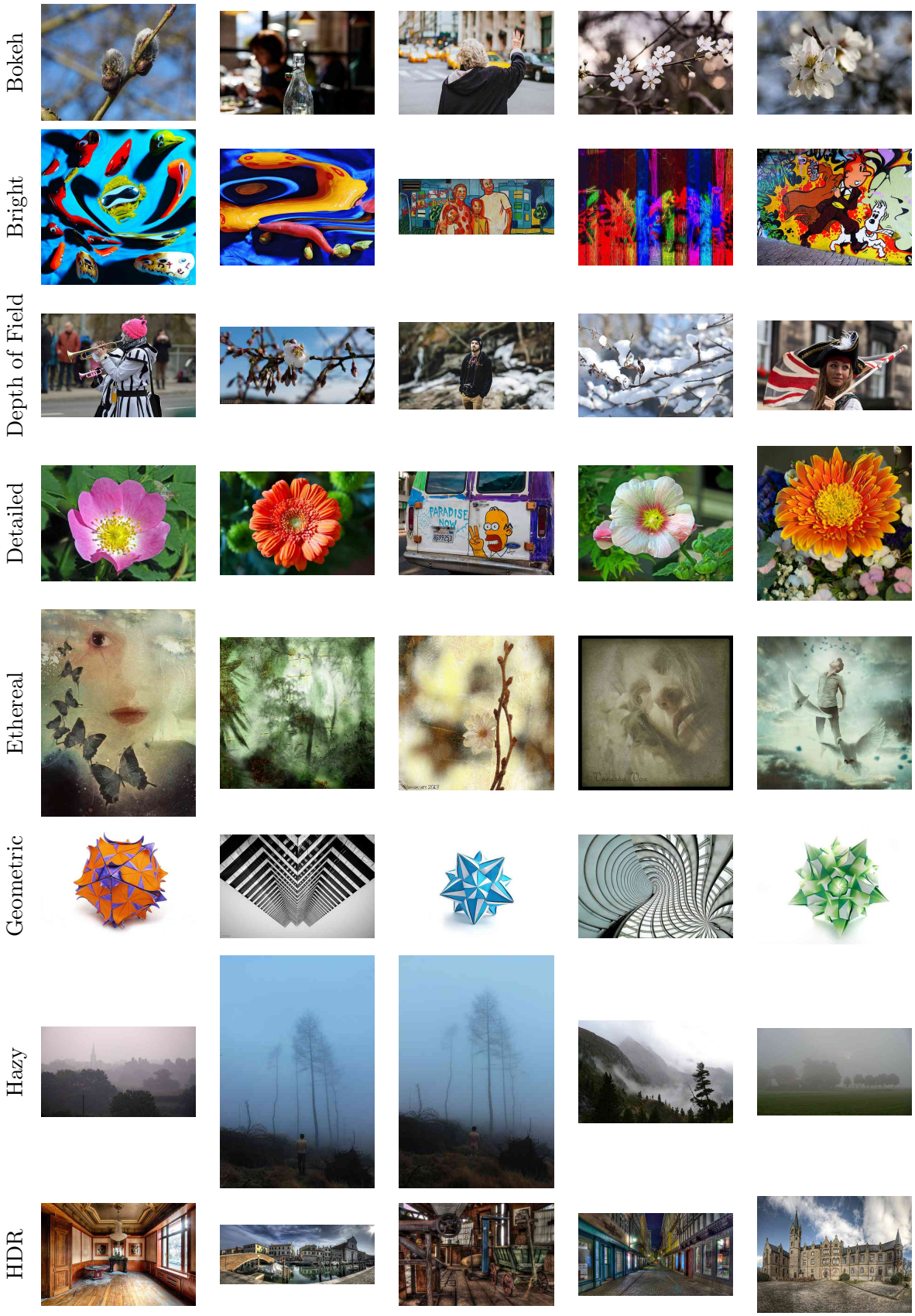


Figure 5.5: Top five most confident predictions on the Flickr Style test set: styles 1-8.

Chapter 6

Conclusion

We note a significant problem that has received little research attention: Anytime visual recognition. The problem is motivated by the properties of human visual perception and by the need to effectively schedule computationally expensive state-of-the-art computer vision methods for different computational budgets. We approach the problem from the perspective of reinforcement learning, and successfully learn fully general policies for selecting detector and classifier actions. To evaluate our approaches, we introduce a new metric of Anytime performance, based on the area under the performance vs. cost curve. In all experiments, we show that having a dynamic state (and thus allowing "closed-loop" policies) and planning ahead increases performance.

Main Contribution

We present a method for learning closed-loop policies for multi-class object detection, given existing object detectors and classifiers and a metric to optimize. The method learns the optimal policy using reinforcement learning, by observing execution traces in training. As with most reinforcement learning problems, the reward function is defined manually, with domain knowledge. Here, we derive it for the novel detection AP vs. Time evaluation that we suggest is useful for evaluating efficiency in recognition. If detection on an image is cut off after only half the detectors have been run, our method does 66% better than a random ordering, and 14% better than an intelligent baseline. In particular, our method learns to take action with no intermediate reward in order to improve the overall performance of the system.

Detection

For the classification task, we need to use a different inference mechanism and additionally need to train classifiers for partially-observed sets of features. We investigate methods such as different forms of imputation and classifier clustering for this task, and adjust the reward function and the featurization of the state. Using our method, we show improved Anytime performance on a synthetic classification task and two benchmark visual recognition tasks. Furthermore, on a hierarchically-structured dataset, we show that accuracy of predictions can be held constant for all budgets, while the specificty of predictions increases.

Classification

Anytime CNN Detection

Even the most recent state-of-the-art CNN-based detection methods are computationally expensive. We first consider approaches which can effectively reorder the sequence of regions to maximize the chance that correct detections will be found early, based on inference from relatively lightweight features. We show that basic strategies, such as simply reordering the boxes such that they do not have a degenerate spatial layout, provides a surprising boost, and that very simple features such as region and gradient statistics can effectively prioritize regions. Our main contribution is the Cascade CNN model, which adds a novel Reject layer between convolutional layers in the architecture. C-CNN obtains an 8x speedup of the R-CNN detection method with only a 10% degradation of state-of-the-art performance on PASCAL VOC detection.

Recognizing Style

We have also made significant progress in defining the problem of recognizing image style. We provide a novel dataset of several types of visual style – including for visual art – not previously considered in the literature. In preparation for an Anytime approach, we evaluate several types of visual features on the dataset, and demonstrate state-of-the-art results in prediction of both style and aesthetic quality (on an existing dataset). We confirm that our results are comparable to human performance, show that style is highly content-dependent, and demonstrate an image search application where results are queried by content and filtered by style.

6.1 Future Work

6.1.1 Detection and Classification

Decision Cost

Computation devoted to scheduling actions is far less significant than the computation due to running the actions in all of our work, and our framework does not explicitly consider this decision-making cost. However, a welcome extension would explicitly model the decision cost by drawing on existing theoretical work on meta-reasoning (such as Hay, Russell, and Sheva 2012). Interesting extensions could try to balance the cost of inference vs. the expected gain in efficiency.

NN Methods

Nearest-neighbor methods are well suited to settings with partially observed sets of features, and so could be a good addition to our work on Anytime classification. However, naive NN methods are too slow for our purposes, and we did not evaluate them. Locality-sensitive hashing methods such as Kulis 2009 may be an effective solution. In particular, the original method of Gao and Koller 2011 could potentially be extended with hashing to maintain its model-free advantages over a rigidly parametrized model at an acceptable speed.

Perception

Beyond the aspects of practical deployment of vision systems that our work is motivated by, we are curious to further investigate our model as a tool to study human cognition and the time course of visual perception. Only a few attempts have been made to explain this: for example, via sequential decision processes in Hegde (2008). While we have not made any claims about the biological mechanism of perception, our work in reinforcement learning-based feature selection as well as convolutional neural networks has explanatory potential if more tightly integrated in future work.

Adaptive Submodularity

The most intriguing future direction is in theoretical analysis of our Anytime method. Our MDP-based formulation is empirically successful, but fundamentally heuristic. Adaptive submodularity (Golovin and Krause 2011), a recently developed framework for obtaining famous near-optimality results of Nemhauser, Wolsey, and Fisher 1978 in the context of learning policies. Just as that work proved that greedy selection can be near-optimal if some conditions of the set function are satisfied, Golovin and Krause 2011 prove that greedy policies can also be near-optimal under certain conditions. Unfortunately, designing an appropriate objective function for our task of visual recognition is not straightforward.

Submodular Ideas

Information gain, a component of our reward function, can be shown to not be submodular Krause and Guestrin 2005, so the easy solution has a roadblock. The most promising way forward is pointed by recent work on active learning and robotic grasping. In Golovin, Krause, and Ray 2010, an adaptively submodular objective based on hypothesis space pruning is developed for an active learning task. In Javdani et al. 2012, robotic grasping is linked to a submodular set cover problem — and another set cover analogy is developed in Chen et al. 2014 for the problem of picking computer vision detections to evaluate with an oracle. An adaptively submodular objective for the general classification problem seems close. Alternatively, we could show that our reward function is empirically submodular – but that is not as interesting.

6.1.2 CNN-based recognition

Cascade CNN

The general structure of the Cascade CNN general structure of simply "thinning" input batches as they travel through the network is agnostic to the underlying mechanism. Although in our work we evaluate on the R-CNN method on the PASCAL dataset, the Cascade CNN can be applied to the SPP-net method of He et al. 2014, or the part-based method of Zhang et al. 2014. In fact, the Cascade CNN can be applied to any existing CNN that predicts in a class-imbalanced domain where speed is important — speech recognition, for example.

End-to-end

Although, the Cascade CNN is shown to be a strong method for speeding up CNN-based detection approaches, its layers were trained in a stage following the initial network training, not in an *end-to-end* fashion that is the hallmark of deep learning models. Furthermore, the thresholds were set in a separate process, using a special validation set. Future work should train and set thresholds of the Reject layers at the same time as the other layers are trained, not after the fact.

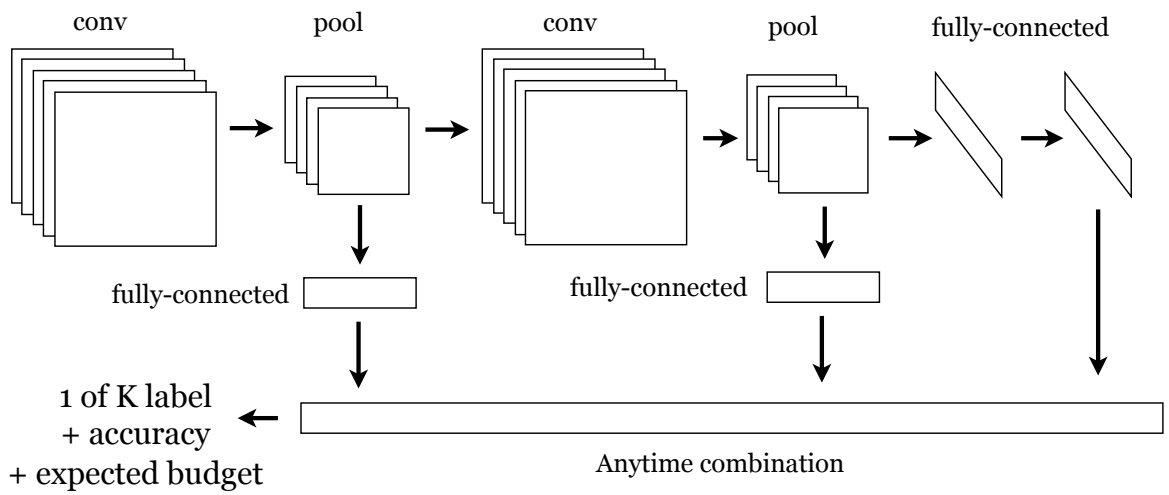


Figure 6.1: The proposed Anytime CNN augments traditional networks with fully-connected prediction layers after every computationally-expensive layer. All prediction layers feed into an Anytime combination layer that computes accuracy and back-propagates from cost-sensitive loss. Compare this architecture to Figure 4.5.

Anytime loss

More interestingly, networks could be augmented with an "Anytime loss" layer that combines classification output from multiple levels of the network in a cost-sensitive way. This would allow optimizing classification networks for arbitrary distributions of cost budgets. Figure 6.1 shows the idea: the new layer combines the outputs of all fully-connected layers, which are regularly placed throughout the network. The Anytime loss computes the expected accuracy of each prediction, takes into account the computational cost up to that layer of the network, and back-propagates according to the budget (for example, if no answers are allowed after 50 ms, and it takes 60 ms to get through the network fully, then the final fully connected layer predictions are never counted). This setup allows for modeling a distribution over budgets.

6.1.3 Image Style

Anytime

While we collected two datasets and showed first results on the challenging new task of recognizing image style, we did not evaluate Anytime performance of our features. Part of the reason was that one of the most interesting outcomes of this work was the success of features trained on object categorization datasets, and in particular the CNN-based feature. Although we make a separate Anytime contribution to the CNN in Chapter 4, it would still be interesting to evaluate Anytime performance of other visual features on the style datasets.

We propose several possible hypotheses to explain the success of general multi-layer features on the style dataset, despite not having been trained on the style task. Perhaps the network layers that we use as features are extremely good as general visual features for image representation in general. Another explanation is that object recognition depends on object appearance, e.g., distinguishing red from white wine, or different kinds of terriers, and that the model learns to repurpose these features for image style. Understanding and improving on these results is fertile ground for future work.

Appendix A

Unobserved Value Imputation: Detailed Results

We evaluate reconstruction and classification error on two datasets: Digits and Scenes-15. Two feature selection policies are considered: Random and Clustered. For each policy, we consider Independent and Block-wise selection patterns.

We find that mean imputation with classifier retraining is a reasonably well-performing approach. Nearest Neighbor methods perform best but are the most expensive. Gaussian imputation method performs very well, but is also expensive. Training additional classifiers for clusters of observed feature subsets did not improve performance.

Feature selection had four experimental conditions.

- In **Random, Independent** selection, each feature was selected independently, and the total number of possible masks for a given budget was not bounded, such that each test instance could have a unique mask (if the number of test instances N was less than the total number of possible feature subsets 2^F.
- In **Random, Block-wise** selection, there was no bound on the number of possible masks for a given budget, but features were selected by blocks.
- In **Clustered, Independent** selection, each feature was selected independently, but there were at most K possible masks for a given budget.
- In **Clustered, Block-wise** selection, there were at most K possible masks for a budget, and features were selected by blocks.

All datasets were first standardized by subtracting the row mean and dividing by the row standard deviation.

A.1 Digits

The Digits dataset contains 8x8 grayscale images of hand-written digits 1-10. Each of the 10 classes has 200 samples, for a total of 2000 images and 64 features.

The dataset was split 60-40 into training and testing sets. The number of clusters for clustered selection was 10. For block-wise feature selection, the number of blocks was set to 8.

Figure A.1 shows the results; here we summarize the conclusions:

- Mean imputation has highest reconstruction and classification error.
- Dot product-based kNN imputation performs worse than Gaussian imputation for both reconstruction and classification, and is slower.
- Euclidean distance-based kNN imputation performs best, but is slowest.
- Training additional classifiers for clusters of observed subsets slightly decreased classification error, but only if the number of clusters was high.
- These results hold for all policy experimental conditions.

A.2 Scenes-15

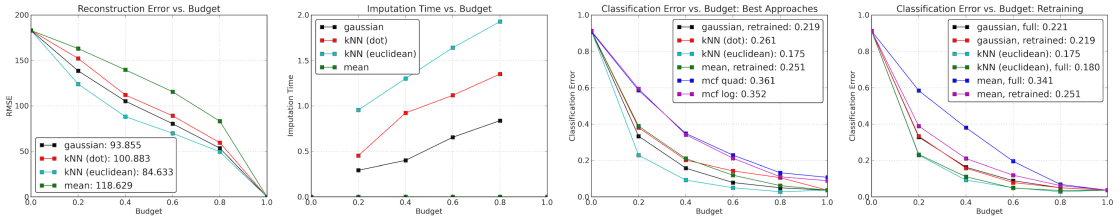
The Scene-15 dataset (**Lazebnik-CVPR-2006**) contains 4485 images from 15 visual scene classes. The task is to identify classify images according to scene.

Following (Xiao et al. 2010), we extracted 14 different visual features (GIST, HOG, TinyImages, LBP, SIFT, Line Histograms, Self-Similarity, Textons, Color Histograms, and variations). Separate multi-class linear SVMs were trained on each feature channel, using a random 100 positive example images per class for training. We used the liblinear implementation, and cross-validated the penalty parameter C.

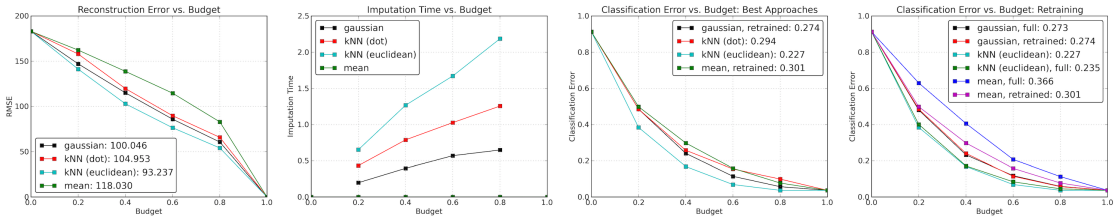
The trained SVMs were evaluated on the images not used for training, resulting in a dataset of 2238 vectors of 210 confidence values: 15 classes for each of the 14 feature channels. This dataset was split 60-40 into training and test sets for our experiments. The number of clusters for clustered selection was 10. For block-wise feature selection, the number of blocks was set to 5.

Figure A.2 shows the results. The conclusions are much the same as for the **Digits** dataset, with the following additional observations:

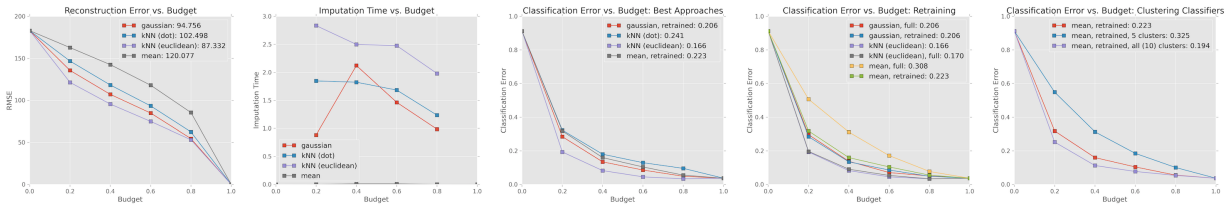
- Gaussian imputation is more costly than kNN on this data; the feature dimensions is more than twice that of the Digits data.
- For block-wise feature selection, mean imputation with retraining is as good as any other approach, and of course by far the fastest.



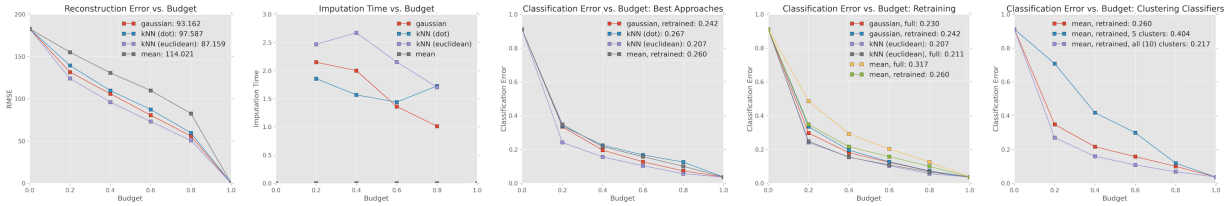
(a) Random, independent feature selection.



(b) Random, block-wise (8 blocks) feature selection.



(c) Clustered (10 clusters), independent feature selection.

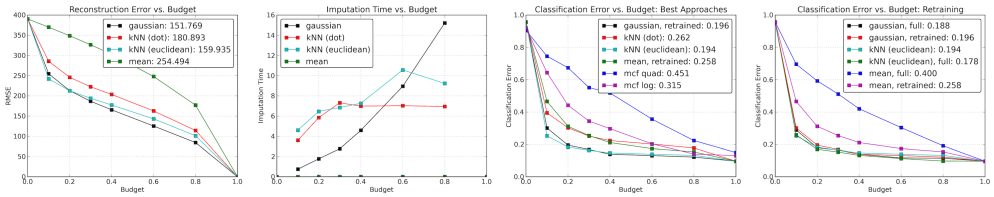


(d) Clustered (10 clusters), block-wise (8 blocks) feature selection.

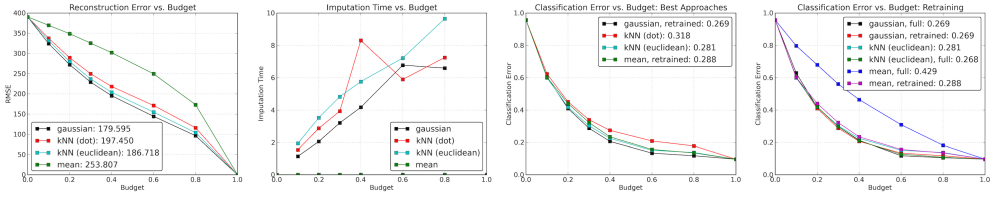
Figure A.1: All missing value imputation results on the **Digits** dataset.

A.3 Conclusion

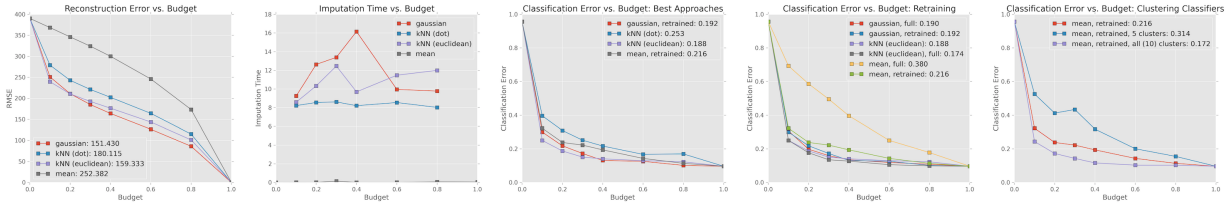
On both datasets, and for all feature selection approaches, we find that mean imputation (with a classifier trained on imputed data) is a well-performing approach. Nearest Neighbor and Gaussian methods perform best but are the most expensive; Gaussian scales poorly with number of features, while NN scales poorly with size of training set. Training additional classifiers for clusters of observed feature subsets also did not signficantly improve performance on these datasets.



(a) Random, independent feature selection.



(b) Random, block-wise (8 blocks) feature selection.



(c) Clustered (10 clusters), independent feature selection.

(d) Clustered (10 clusters), block-wise (8 blocks) feature selection.

Figure A.2: All missing value imputation results on the **Scenes-15** dataset.

Appendix B

Recognizing Image Style: Detailed Results

Table B.1: All per-class APs on all evaluated features on the AVA Style dataset.

	Fusion	$DeCAF_6$	MC-bit	Murray	L*a*b*	GIST	Saliency
Complementary_Colors	0.469	0.548	0.329	0.440	0.294	0.223	0.111
Duotones	0.676	0.737	0.612	0.510	0.582	0.255	0.233
HDR	0.669	0.594	0.624	0.640	0.194	0.124	0.101
Image_Grain	0.647	0.545	0.744	0.740	0.213	0.104	0.104
Light_On_White	0.908	0.915	0.802	0.730	0.867	0.704	0.172
Long_Exposure	0.453	0.431	0.420	0.430	0.232	0.159	0.147
Macro	0.478	0.427	0.413	0.500	0.230	0.269	0.161
Motion_Blur	0.478	0.467	0.458	0.400	0.117	0.114	0.122
Negative_Image	0.595	0.619	0.499	0.690	0.268	0.189	0.123
Rule_of_Thirds	0.352	0.353	0.236	0.300	0.188	0.167	0.228
Shallow_DOF	0.624	0.659	0.637	0.480	0.332	0.276	0.223
Silhouettes	0.791	0.801	0.801	0.720	0.261	0.263	0.130
Soft_Focus	0.312	0.354	0.290	0.390	0.127	0.126	0.114
Vanishing_Point	0.684	0.658	0.685	0.570	0.123	0.107	0.161
mean	0.581	0.579	0.539	0.539	0.288	0.220	0.152

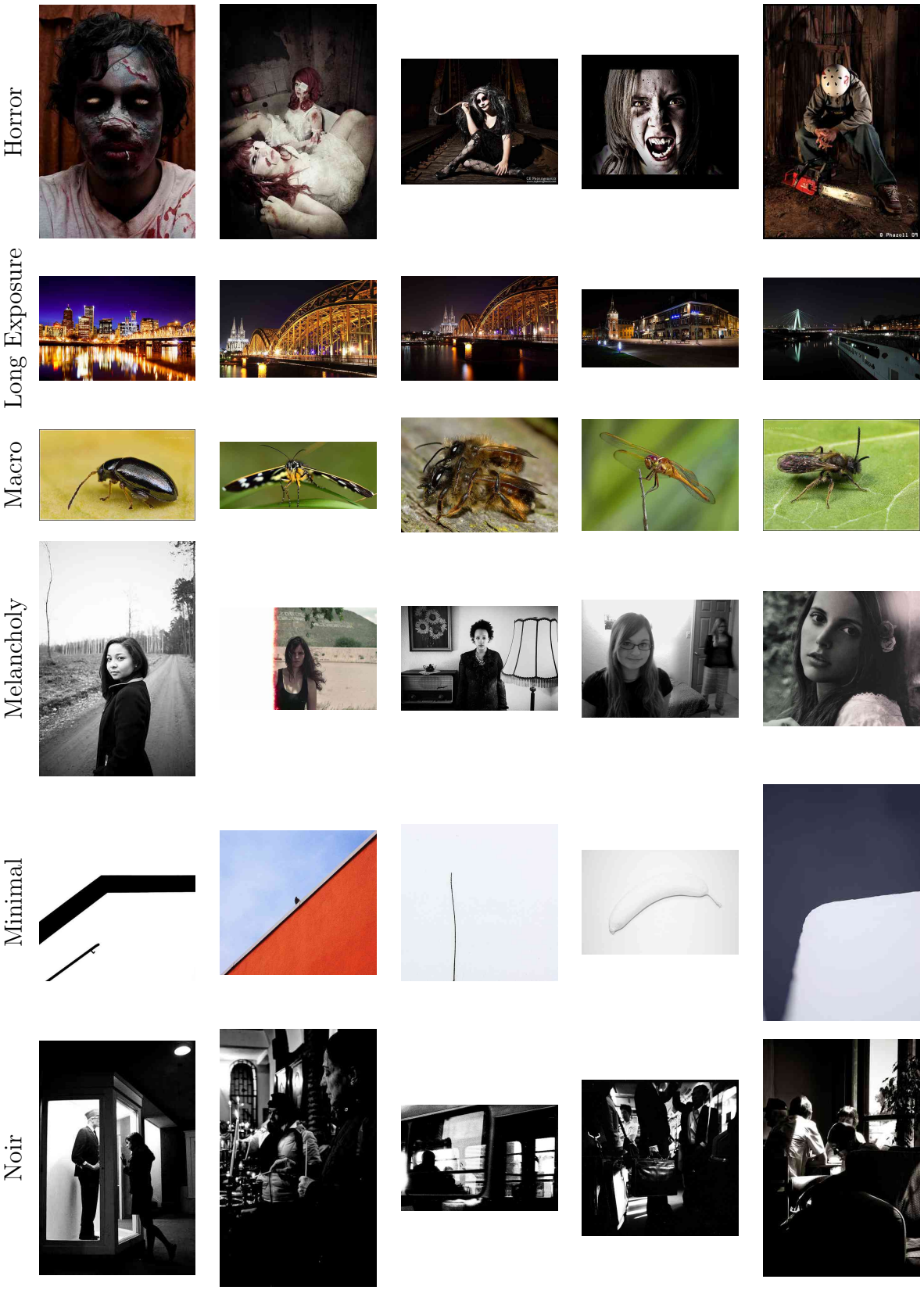


Figure B.1: Top five most confident predictions on the Flickr Style test set: styles 9-14.

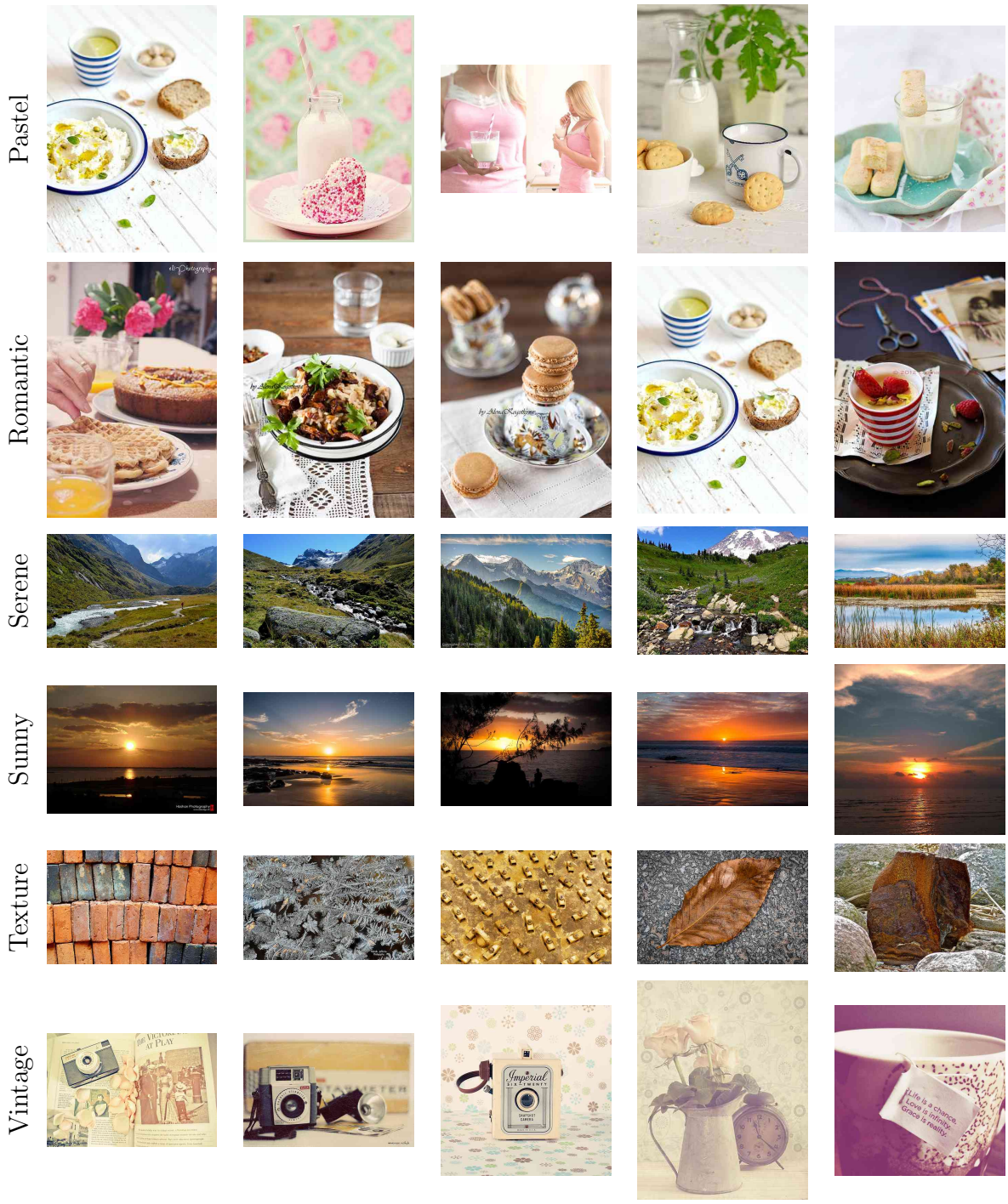


Figure B.2: Top five most confident predictions on the Flickr Style test set: styles 15-20.

Table B.2: All per-class APs on all evaluated features on the Flickr dataset.

	Fusion x Content	$DeCAF_6$	MC-bit
Bokeh	0.288	0.253	0.248
Bright	0.251	0.236	0.183
Depth_of_Field	0.169	0.152	0.148
Detailed	0.337	0.277	0.278
Ethereal	0.408	0.393	0.335
Geometric_Composition	0.411	0.355	0.360
HDR	0.487	0.406	0.475
Hazy	0.493	0.451	0.447
Horror	0.400	0.396	0.295
Long_Exposure	0.515	0.457	0.463
Macro	0.617	0.582	0.530
Melancholy	0.168	0.147	0.136
Minimal	0.512	0.444	0.481
Noir	0.494	0.481	0.408
Pastel	0.258	0.245	0.211
Romantic	0.227	0.204	0.185
Serene	0.281	0.257	0.239
Sunny	0.500	0.481	0.453
Texture	0.265	0.227	0.229
Vintage	0.282	0.273	0.222
mean	0.368	0.336	0.316

Table B.3: Comparison of Flickr Style per-class accuracies for our method and Mech Turkers.

	MTurk acc., Flickr g.t.	Our acc., Flickr g.t.	Our acc., MTurk g.t.
Bright	69.10	73.38	73.63
Depth of Field	68.92	68.50	81.05
Detailed	65.47	75.25	68.44
Ethereal	76.92	80.62	77.95
Geometric Composition	81.52	77.75	80.31
HDR	71.84	82.00	76.96
Hazy	83.49	80.75	81.64
Horror	89.85	84.25	81.64
Long Exposure	73.12	84.19	76.79
Macro	92.25	86.56	88.39
Melancholy	67.77	70.88	71.25
Minimal	79.71	83.75	78.57
Noir	81.35	85.25	85.88
Pastel	66.94	74.56	75.47
Romantic	60.91	68.00	66.25
Serene	69.49	70.44	76.80
Sunny	84.48	84.56	79.94
Vintage	68.77	75.50	67.80
Mean	75.11	78.12	77.15

Table B.4: Signficant deviations between human and machine accuracies on Flickr Style.

	Our acc., Flickr g.t.	Our acc., MTurk g.t.	% change from Flickr to MTurk g.t.
Vintage	75.50	67.80	-10.19
Detailed	75.25	68.44	-9.05
Long Exposure	84.19	76.79	-8.79
Minimal	83.75	78.57	-6.18
HDR	82.00	76.96	-6.15
Sunny	84.56	79.94	-5.46
Serene	70.44	76.80	9.03
Depth of Field	68.50	81.05	18.32
	Our acc., Flickr g.t.	MTurk acc., Flickr g.t.	Acc. difference
Horror	84.25	90.42	-6.17
Macro	86.56	91.71	-5.15
Romantic	68.00	61.04	6.96
Pastel	74.56	66.87	7.69
HDR	82.00	72.79	9.21
Long Exposure	84.19	73.83	10.35
Detailed	75.25	63.30	11.95

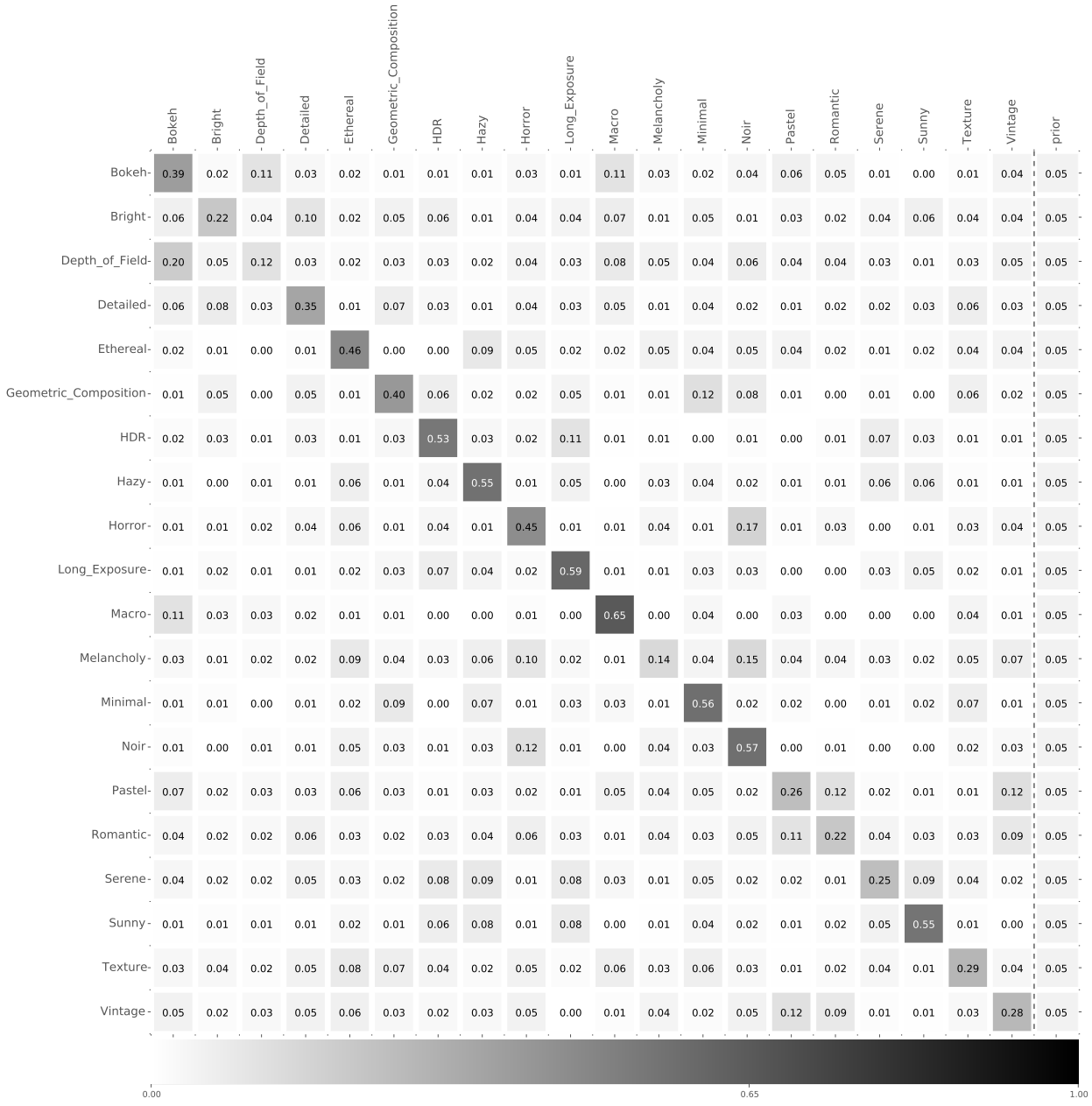


Figure B.3: Confusion matrix of our best classifier (Late-fusion × Content) on the Flickr dataset.

Table B.5: All per-class APs on all evaluated features on the Wikipaintings dataset.

	Fusion x Content	MC-bit	$DeCAF_6$
Abstract_Art	0.341	0.314	0.258
Abstract_Expressionism	0.351	0.340	0.243
Art_Informel	0.221	0.217	0.187
Art_Nouveau_(Modern)	0.421	0.402	0.197
Baroque	0.436	0.386	0.313
Color_Field_Painting	0.773	0.739	0.689
Cubism	0.495	0.488	0.400
Early_Renaissance	0.578	0.559	0.453
Expressionism	0.235	0.230	0.186
High_Renaissance	0.401	0.345	0.288
Impressionism	0.586	0.528	0.411
Magic_Realism	0.521	0.465	0.428
Mannerism_(Late_Renaissance)	0.505	0.439	0.356
Minimalism	0.660	0.614	0.604
Nave_Art_(Primitivism)	0.395	0.425	0.225
Neoclassicism	0.601	0.537	0.399
Northern_Renaissance	0.560	0.478	0.433
Pop_Art	0.441	0.398	0.281
Post-Impressionism	0.348	0.348	0.292
Realism	0.408	0.309	0.266
Rococo	0.616	0.548	0.467
Romanticism	0.392	0.389	0.343
Surrealism	0.262	0.247	0.134
Symbolism	0.390	0.390	0.260
Ukiyo-e	0.895	0.894	0.788
mean	0.473	0.441	0.356

Table B.6: Per-class accuracies on the Wikipaintings dataset, using the MC-bit feature.

Style	**Accuracy**	**Style**	**Accuracy**
Symbolism	71.24	Impressionism	82.15
Expressionism	72.03	Northern Renaissance	82.32
Art Nouveau (Modern)	72.77	High Renaissance	82.90
Nave Art (Primitivism)	72.95	Mannerism (Late Renaissance)	83.04
Surrealism	74.44	Pop Art	83.33
Post-Impressionism	74.51	Early Renaissance	84.69
Romanticism	75.86	Abstract Art	85.10
Realism	75.88	Cubism	86.85
Magic Realism	78.54	Rococo	87.33
Neoclassicism	80.18	Ukiyo-e	93.18
Abstract Expressionism	81.25	Minimalism	94.21
Baroque	81.45	Color Field Painting	95.58
Art Informel	82.09		

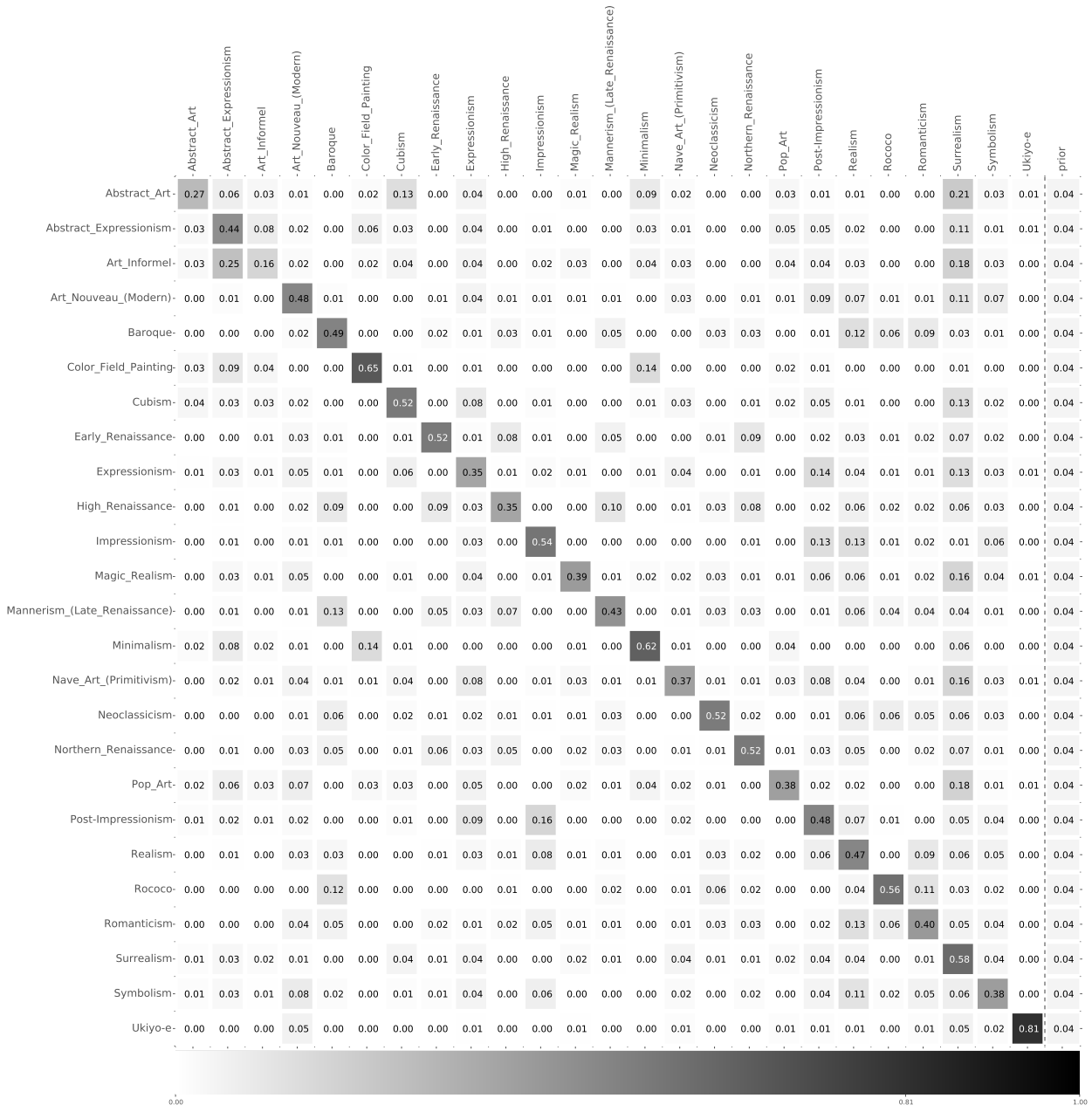


Figure B.4: Confusion matrix of our best classifier (Late-fusion × Content) on the Wikipaintings dataset.

Bibliography

Agarwal, Alekh, Olivier Chapelle, Miroslav Dudik, and John Langford (2012). "A Reliable Effective Terascale Linear Learning System". In: *Journal of Machine Learning Research*. arXiv: arXiv:1110.4198v3.

Alexe, Bogdan, Nicolas Heess, and Vittorio Ferrari (2012). "Searching for objects driven by context". In: *NIPS*.

Benbouzid, Djalel, Robert Busa-Fekete, and Balazs Kegl (2012). "Fast classification using sparse decision DAGs". In: *ICML*.

Bergamo, A. and L. Torresani (2012). "Meta-class features for large-scale object categorization on a budget". In: *CVPR*. ISBN: 978-1-4673-1228-8. DOI: 10.1109/CVPR.2012.6248040.

Borth, Damian, Rongrong Ji, Tao Chen, and Thomas M Breuel (2013). "Large-scale Visual Sentiment Ontology and Detectors Using Adjective Noun Pairs". In: *ACM MM*. ISBN: 9781450324045.

Bourdev, Lubomir and J Brandt (2005). "Robust Object Detection via Soft Cascade". In: *CVPR*. ISBN: 0-7695-2372-2. DOI: 10.1109/CVPR.2005.310.

Butko, N J and J R Movellan (2009). "Optimal scanning for faster object detection". In: *CVPR*, pp. 2751–2758. DOI: 10.1109/CVPR.2009.5206540.

Chen, Minmin, Zhixiang Xu, Kilian Q. Weinberger, Olivier Chapelle, and Dor Kedem (2012). "Classifier Cascade for Minimizing Feature Evaluation Cost". In: *AISTATS*.

Chen, Yuxin, Hiroaki Shioi, Cesar Antonio Fuentes Montesinos, Lian Pin Koh, Serge Wich, and Andreas Krause (2014). "Active Detection via Adaptive Submodularity". In: *ICML*.

Dalal, N and B Triggs (2005). "Histograms of Oriented Gradients for Human Detection". In: *CVPR*. Ieee, pp. 886–893. ISBN: 0-7695-2372-2. DOI: 10.1109/CVPR.2005.177.

Datta, Ritendra, Dhiraj Joshi, Jia Li, and James Z Wang (2006). "Studying Aesthetics in Photographic Images Using a Computational Approach". In: *ECCV*.

Deng, Jia, W. Dong, R. Socher, L.-J. Li, K. Li, and Li Fei-Fei (2009). "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR*.

Deng, Jia, Alexander C Berg, Kai Li, and Li Fei-fei (2010). "What Does Classifying More Than 10,000 Image Categories Tell Us ?" In: *ECCV*, pp. 71–84.

Deng, Jia, Sanjeev Satheesh, Alexander C Berg, and Li Fei-fei (2011). "Fast and Balanced: Efficient Label Tree Learning for Large Scale Object Recognition". In: *NIPS*. 1, pp. 1–9.

Deng, Jia, Jonathan Krause, Alexander C Berg, and Li Fei-fei (2012). "Hedging Your Bets: Optimizing Accuracy-Specificity Trade-offs in Large Scale Visual Recognition". In: *CVPR*.

Desai, Chaitanya, Deva Ramanan, and Charless Fowlkes (2011). "Discriminative models for multi-class object layout". In: *IJCV*. Ieee, pp. 229–236. ISBN: 978-1-4244-4420-5. DOI: 10.1109/ICCV.2009.5459256.

Dhar, Sagnik, Tamara L Berg, and Stony Brook (2011). "High Level Describable Attributes for Predicting Aesthetics and Interestingness". In: *CVPR*.

Divvala, S K, D Hoiem, J H Hays, A.a. Efros, and M Hebert (2009). "An empirical study of context in object detection". In: *CVPR*. Ieee, pp. 1271–1278. ISBN: 978-1-4244-3992-8. DOI: 10.1109/CVPR.2009.5206532.

Donahue, Jeff, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell (2013a). *DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition*. Tech. rep. arXiv: arXiv:1310.1531v1.

— (2013b). *DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition*. Tech. rep. arXiv: arXiv:1310.1531v1.

Duchi, John, Elad Hazan, and Yoram Singer (2011). "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research*.

Dulac-arnold, Gabriel, Nicolas Thome, and Matthieu Cord (2014). "Sequentially Generated Instance-Dependent Image Representations for Classification". In: *ICLR*. arXiv: arXiv:1312.6594v3.

Dulac-Arnold, Gabriel, Ludovic Denoyer, Philippe Preux, and Patrick Gallinari (2012). "Sequential approaches for learning datum-wise sparse representations". In: *Machine Learning* 89.1-2, pp. 87–122. ISSN: 0885-6125. DOI: 10.1007/s10994-012-5306-7.

Erhan, Dumitru, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov (2014). "Scalable Object Detection using Deep Neural Networks". In: *CVPR*.

Ernst, Damien, Pierre Geurts, and Louis Wehenkel (2005). "Tree-Based Batch Mode Reinforcement Learning". In: *Journal of Machine Learning Research* 6, pp. 503–556.

Everingham, M, L Van Gool, C K I Williams, J Winn, and A Zisserman (2010). *The PASCAL VOC Challenge 2010 Results*. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/ind

Fan, Xiaodong (2005). "Efficient Multiclass Object Detection by a Hierarchy of Classifiers". In: *CVPR*. Ieee, pp. 716–723. ISBN: 0-7695-2372-2. DOI: 10.1109/CVPR.2005.140.

Farrell, Ryan, Om Oza, Vlad I. Morariu, Trevor Darrell, and Larry S. Davis (2011). "Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance". In: *ICCV*. ISBN: 978-1-4577-1102-2. DOI: 10.1109/ICCV.2011.6126238.

Fei-Fei, Li, Asha Iyer, Christof Koch, and Pietro Perona (2007). "What do we perceive in a glance of a real-world scene?" In: *Journal of vision* 7.1, p. 10. ISSN: 1534-7362. DOI: 10.1167/7.1.10.

Felzenszwalb, Pedro F, Ross B Girshick, and David McAllester (2010). "Cascade object detection with deformable part models". In: *CVPR*. IEEE, pp. 2241–2248. ISBN: 978-1-4244-6984-0. DOI: 10.1109/CVPR.2010.5539906.

Felzenszwalb, Pedro F, Ross B Girshick, David McAllester, and Deva Ramanan (2010). "Object detection with discriminatively trained part-based models." In: *PAMI* 32.9, pp. 1627–1645. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2009.167.

Galleguillos, Carolina and Serge Belongie (2010). "Context based object categorization: A critical survey". In: *Computer Vision and Image Understanding* 114.6, pp. 712–722. ISSN: 10773142. DOI: 10.1016/j.cviu.2010.02.004.

Gao, Tianshi and Daphne Koller (2011). "Active Classification based on Value of Classifier". In: *NIPS*.

Gehler, Peter and Sebastian Nowozin (2009). "On Feature Combination for Multiclass Object Classification". In: *ICCV*.

Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik (2014). "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *CVPR*. arXiv: arXiv:1311.2524v3.

Golovin, Daniel and Andreas Krause (2011). "Adaptive Submodularity: A New Approach to Active Learning and Stochastic Optimization". In: *Journal of Artificial Intelligence Research*.

Golovin, Daniel, Andreas Krause, and Debajyoti Ray (2010). "Near-Optimal Bayesian Active Learning with Noisy Observations". In: *CoRR* abs/1010.3.

Graves, Alex (2013). "Generating Sequences With Recurrent Neural Networks". In: *CoRR* abs/1308.0850. URL: http://arxiv.org/abs/1308.0850.

Grubb, Alexander and J Andrew Bagnell (2012). "SpeedBoost: Anytime Prediction with Uniform Near-Optimality". In: *AISTATS*.

Gygli, Michael, Fabian Nater, and Luc Van Gool (2013). "The Interestingness of Images". In: *ICCV*.

Harel, Jonathan, Christof Koch, and Pietro Perona (2006). "Graph-Based Visual Saliency". In: *NIPS*.

Hastie, Trevor, Robert Tibshirani, Gavin Sherlock, Patrick Brown, David Botstein, and Michael Eisen (1999). "Imputing Missing Data for Gene Expression Arrays Imputation using the SVD".

Hay, Nicholas, Stuart Russell, and Beer Sheva (2012). "Selecting Computations: Theory and Applications". In: *UAI*. arXiv: arXiv:1207.5879v1.

He, He, Daume Hal III, and Jason Eisner (2012). "Cost-sensitive Dynamic Feature Selection". In: *ICML-W*.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2014). "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *ECCV*. arXiv: arXiv:1406.4729v1.

Hegde, Jay (2008). "Time course of visual perception: Coarse-to-fine processing and beyond". In: *Progress in Neurobiology*.

Isola, Phillip, Jianxiong Xiao, Antonio Torralba, and Aude Oliva (2011). "What makes an image memorable?" In: *CVPR*. Ieee. ISBN: 978-1-4577-0394-2. DOI: 10.1109/CVPR.2011.5995721.

Jaimovich, Ariel and Ian Mcgraw (2010). "FastInf : An Efficient Approximate Inference Library". In: *Journal of Machine Learning Research* 11, pp. 1733–1736.

Javdani, Shervin, Matthew Klingensmith, J Andrew Bagnell, Nancy S Pollard, and Siddhartha S Srinivasa (2012). *Efficient Touch Based Localization through Submodularity*. Tech. rep. arXiv: arXiv:1208.6067v2.

Ji, Shihao and Lawrence Carin (2007). "Cost-Sensitive Feature Acquisition and Classification". In: *Pattern Recognition*.

Jia, Yangqing (2013). *Caffe: an Open Source Convolutional Architecture for Fast Feature Embedding*. http://caffe.berkeleyvision.org/.

Joo, Jungseock, Weixin Li, Francis Steen, and Song-Chun Zhu (2014). "Visual Persuasion: Inferring Communicative Intents of Images". In: *CVPR*.

Karayev, Sergey, Mario Fritz, and Trevor Darrell (2014). "Anytime Recognition of Objects and Scenes". In: *CVPR*.

Karayev, Sergey, Tobias Baumgartner, Mario Fritz, and Trevor Darrell (2012). "Timely Object Recognition". In: *NIPS*.

Karayev, Sergey, Matthew Trentacoste, Helen Han, Aseem Agarwala, Trevor Darrell, Aaron Hertzmann, and Holger Winnemoeller (2014). "Recognizing Image Style". In: BMVC.

Keren, Daniel (2002). "Painter Identification Using Local Features and Naive Bayes". In: *ICPR*.

Khosla, A., A. Das Sarma, and R. Hamid (2014). "What Makes an Image Popular?" In: *WWW*.

Koren, Yehuda, Robert Bell, and Chris Volinsky (2009). "Matrix Factorization Techniques for Recommender Systems". In: pp. 42–49.

Krause, Andreas and Carlos Guestrin (2005). "Near-optimal Nonmyopic Value of Information in Graphical Models". In: *UAI*.

Krizhevsky, Alex, Ilya Sutskever, and Geoff E. Hinton (2012a). "ImageNet Classification with Deep Convolutional Neural Networks". In: *NIPS*.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012b). "ImageNet Classification with Deep Convolutional Neural Networks". In: *NIPS*.

Kulis, Brian (2009). "Learning to hash with binary reconstructive embeddings". In: *Advances in Neural Information Processing Systems*, pp. 1–9.

Kwok, Cody and Dieter Fox (2004). "Reinforcement Learning for Sensing Strategies". In: *IROS*.

Lampert, Christoph H, Matthew B Blaschko, and Thomas Hofmann (2008). "Beyond sliding windows: Object localization by efficient subwindow search". In: *CVPR*. DOI: 10.1109/CVPR.2008.4587586.

Lanckriet, Gert R G, Nellow Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan (2004). "Learning the Kernel Matrix with Semidefinite Programming". In: *JMLR* 5, pp. 27–72.

Lee, Su-In, Varun Ganapathi, and Daphne Koller (2006). "Efficient Structure Learning of Markov Networks using L 1 -Regularization". In: *NIPS*.

Li, Congcong and Tsuhan Chen (2009). "Aesthetic Visual Quality Assessment of Paintings". In: *IEEE Journal of Selected Topics in Signal Processing* 3.2, pp. 236–252. ISSN: 1932-4553. DOI: 10.1109/JSTSP.2009.2015077.

Lowe, David G (2004). "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2, pp. 91–110. ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94.

Lyu, Siwei, Daniel Rockmore, and Hany Farid (2004). "A Digital Technique for Art Authentication". In: *PNAS* 101.49.

Macé, Marc J-M, Olivier R Joubert, Jean-Luc Nespoulous, and Michèle Fabre-Thorpe (2009). "The time-course of visual categorizations: you spot the animal faster than the bird." In: *PloS one* 4.6, e5927. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0005927.

Marchesotti, Luca and Florent Perronnin (2013). "Learning beautiful (and ugly) attributes". In: *BMVC*.

Mensink, Thomas and Jan van Gemert (2014). "The Rijksmuseum Challenge: Museum-Centered Visual Recognition". In: *ICMR*.

Murray, Naila, Luca Marchesotti, and Florent Perronnin (2012). "AVA: A Large-Scale Database for Aesthetic Visual Analysis". In: *CVPR*.

Nemhauser, G L, L A Wolsey, and M L Fisher (1978). "An analysis of approximations for maximizing submodular set functions". In: *Mathematical Programming* 14, pp. 265–294.

Oliva, Aude and Antonio Torralba (2001). "Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope". In: *IJCV* 42.3, pp. 145–175.

Olshausen, B A and Others (1996). "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". In: *Nature* 381.6583, pp. 607–609. ISSN: 0028-0836.

Palermo, Frank, James Hays, and Alexei A Efros (2012). "Dating Historical Color Images". In: *ECCV*.

Pedregosa, Fabian et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Roweis, Sam (1999). "Gaussian Identities".

Roy, Nicholas and Geoffrey Gordon (2002). "Exponential Family PCA for Belief Compression in POMDPs". In: *NIPS*.

Sermanet, Pierre and David Eigen (2014). "OverFeat: Integrated Recognition , Localization and Detection using Convolutional Networks". In: *ICLR*. arXiv: arXiv:1312.6229v4.

Shamir, Lior, Tomasz Macura, Nikita Orlov, D. Mark Eckley, and Ilya G. Goldberg (2010). "Impressionism, Expressionism, Surrealism: Automated Recognition of Painters and Schools of Art". In: *ACM Trans. Applied Perc.* 7.2.

Shechtman, Eli and Michal Irani (2007). "Matching Local Self-Similarities across Images and Videos". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. DOI: 10.1109/CVPR.2007.383198.

Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman (2014). "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: *ICLR*. arXiv: arXiv:1312.6034v2.

Sutton, Richard S and Andrew G Barto (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Taylor, Graham W and Geoffrey E Hinton (2009). "Factored Conditional Restricted Boltzmann Machines for Modeling Motion Style". In: *ICML*.

Tenenbaum, J B and W T Freeman (2000). "Separating style and content with bilinear models." In: *Neural computation* 12.6, pp. 1247–83. ISSN: 0899-7667.

Torralba, Antonio, Kevin P Murphy, and William T Freeman (2004). "Contextual Models for Object Detection Using Boosted Random Fields". In: *MIT Computer Science and Artificial Intelligence Laboratory Technical Report*.

Torralba, Antonio, Kevin P Murphy, and William T Freeman (2007). "Sharing visual features for multiclass and multiview object detection." In: *PAMI* 29.5, pp. 854–869. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2007.1055.

Trapeznikov, Kirill, Venkatesh Saligrama, and David Castanon (2013). "Multi-Stage Classifier Design". In: *Machine Learning* 92.2-3, pp. 479–502. arXiv: arXiv:1205.4377v2.

Uijlings, J R R, K E A Van De Sande, T Gevers, and A W M Smeulders (2013). "Selective Search for Object Recognition". In: *IJCV*.

Vanrullen, Rufin and Simon J Thorpe (2001). "The Time Course of Visual Processing: From Early Perception to Decision-Making". In: *Journal of Cognitive Neuroscience* 13.4, pp. 454–461.

Vedaldi, Andrea, Varun Gulshan, Manik Varma, and Andrew Zisserman (2009). "Multiple kernels for object detection". In: *ICCV*, pp. 606–613. DOI: 10.1109/ICCV.2009.5459183.

Vijayanarasimhan, Sudheendra and Kristen Grauman (2011). "Large-Scale Live Active Learning: Training Object Detectors with Crawled Data and Crowds". In: *CVPR*.

Vijayanarasimhan, Sudheendra and Ashish Kapoor (2010). "Visual Recognition and Detection Under Bounded Computational Resources". In: *CVPR*, pp. 1006–1013.

Viola, Paul and Michael J Jones (2004). "Robust Real-Time Face Detection". In: *IJCV* 57.2, pp. 137–154.

Vogel, Julia and Nando de Freitas (2008). "Target-directed attention: Sequential decision-making for gaze planning". In: *ICRA*, pp. 2372–2379. DOI: 10.1109/ROBOT.2008.4543568.

Weiss, David, Benjamin Sapp, and Ben Taskar (2013). "Dynamic structured model selection". In: ICCV.

Xiao, Jianxiong, James Hays, K A Ehinger, A Oliva, and Antonio Torralba (2010). "SUN database: Large-scale scene recognition from abbey to zoo". In: *CVPR*.

Xu, Zhixiang, Kilian Q Weinberger, and Olivier Chapelle (2012). "The Greedy Miser: Learning under Test-time Budgets". In: *ICML*.

Xu, Zhixiang, Matt J Kusner, Kilian Q Weinberger, and Minmin Chen (2013). "Cost-Sensitive Tree of Classifiers". In: *ICML*. arXiv: arXiv:1210.2771v2.

Zhang, Ning, Jeff Donahue, Ross Girshick, and Trevor Darrell (2014). "Part-based R-CNNs for Fine-grained Category Detection". In: ECCV.