Local and Adaptive Image-to-Image Learning and Inference

by

Evan Gerard Shelhamer

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 Alexei A. Efros
Professor Bruno Olshausen

Spring 2019

Local and Adaptive Image-to-Image Learning and Inference

Abstract

Local and Adaptive Image-to-Image Learning and Inference

by

Evan Gerard Shelhamer

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Trevor Darrell, Chair

Much of the recent progress on visual processing has been driven by deep learning and its bicameral heart of composition and end-to-end optimization. The machinery of convolutional networks is now ubiquitous. Its diffusion however was neither instantaneous nor effortless. To advance across the frontiers of vision, deep learning had to be equipped with the right structures: the true, intrinsic structures of the visual world.

This thesis incorporates locality and scale structure into end-to-end learning for visual recognition. Locality structure is key for addressing image-to-image tasks that take image inputs and return image outputs. Scale structure is ubiquitous, and optimizing over it learns the degree of locality for the task and data. Alongside structure, this thesis examines adaptive computation to help cope with the variability of rich image-to-image prediction problems. These directions are studied through the lens of local recognition tasks that require inference of what and where.

Fully convolutional networks decompose image-to-image learning and inference into local scopes. Factorizing these scopes into structured and free-form parts, and learning both, optimizes their size and shape to control the degree of locality. Adaptive computation across time, computing layers according to their rate of change, exploits temporal locality to improve the efficiency of video processing. Adaptive computation across tasks, extracting a latent representation of local supervision, transcends locality to non-locally guide and correct inference. Locality is the defining principle of our fully convolutional networks. Adaptivity equips our networks to more fully engage with the vastness and variety of vision.

To my family.

Contents

Acknowledgments

I cannot imagine a finer constellation to have joined for my PhD.

Thanks to my advisor, Trevor Darrell, for making this whole pursuit possible at all. Trevor has given me so much advice, at many times, and in many time zones. Following ideas backward through the history or forward across the frontier, we have never lacked for schemes. I appreciate the chances to have tried on the hats and travelled in the directions that I did. Thanks to my committee: Alyosha Efros and Bruno Olshausen. From Alyosha I learned to keep my data near, chocolate nearer, and of course neighbors nearest. From Bruno I learned to admire the existence proofs of visual intelligence that surround us (especially jumping spiders). Thanks too to Jitendra Malik and Pieter Abbeel, whose enthusiasm and dedication have been inspirational ever since my very first visit to campus.

Thanks to my collaborators, in particular Jon Long, Jeff Donahue, Kate Rakelly, and Dequan Wang. Thanks to the Caffe crew, especially Yangqing Jia, Jon Long, and Jeff Donahue, but all in all everybody who took part. It has been an incredible opportunity to not only experiment and engineer but to cohere and to share.

Thanks to my fellow students in BAIR. Thanks to Jon Long for the words, Jeff Donahue for the visions, Sergey Karayev for the poems, Georgia for the weekend wisdom, Jon Barron for the expertly filtered views, Kate Rakelly for tea and travel, Eric Tzeng for not abandoning φιλοσοφία nor the painting of bike sheds, Lisa Anne Hendricks for the wine and therapy, Deepak Pathak for the night shifts and wild ideas, Shiry Ginosar for the sharing and caring, and Dequan Wang for the push.

Thanks to my hosts away from home: John Schulman at OpenAI, Max Jaderberg at DeepMind, Ross Girshick at FAIR, and Josh Tenenbaum at MIT. Thanks to the postdocs on campus, especially Stefanie Jegelka, who mentored me on the first project of my PhD.

Thanks to Tselil, Tynan, David, Coline, and Andrea for our home together. Thanks to the Marin house, for giving me a second home whenever I needed one, and in particular Cory and Ana for feeding me so many feasts. Thanks to Tselil too for embarking on a PhD with me here, and celebrating our own annual holiday ever since: stay correlated.

Thanks to my life back on the right coast that readied me for the left. Thanks to Erik Learned-Miller, Lisa Sanders, and Lee Spector for welcoming me into research in the Pioneer Valley. Thanks to Erik Learned-Miller too for supervising my undergraduate thesis at UMass Amherst, and connecting me to Berkeley from the Berkshires. Thanks to Katie Wynkoop, for the napping pillow and kaleidoscopic portrait that have lasted me throughout my PhD, and most of all for the perspective. Thanks to Sara Carter for all of the calls across the country to sustain the hearts and minds. Thanks to Laura for our adventures whenever I go through my MA/NY orbits. Thanks to Matt Tannenbaum for the bookstore and its books.

Thanks to Kelsey, for the science and romance, and the endless encouragement at the end of my PhD.

Thanks to my parents, for their love and their encouragement to seek whatever makes the most sense to me, to be happy, and to use my powers for good. This thesis is dedicated to them and all of my family.

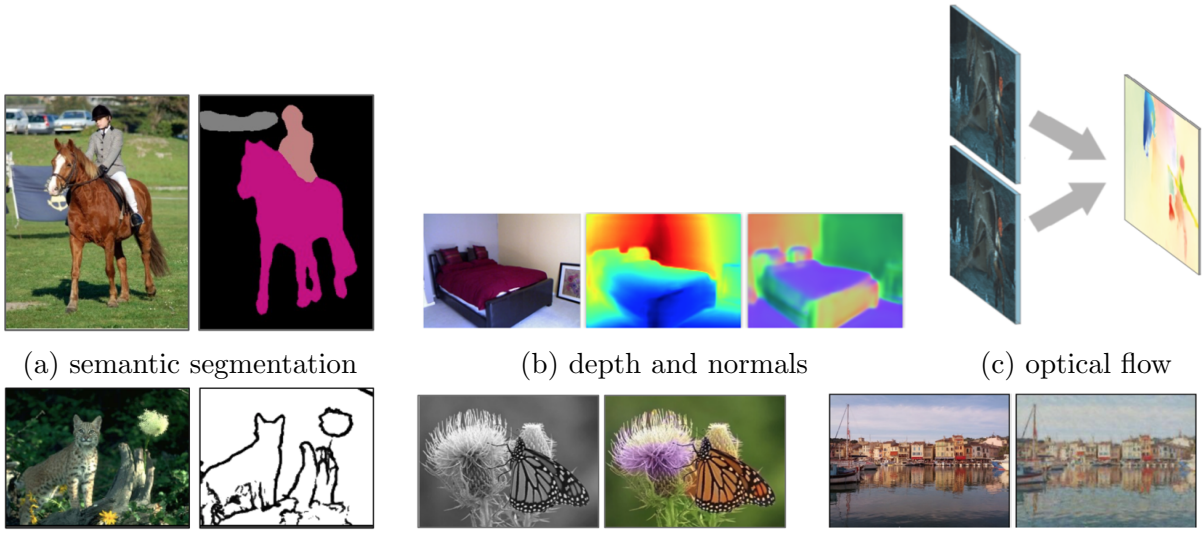
Chapter 1

Introduction

There are many and various kinds of image, each with its own particulars. A wealth of visual problems can be framed as *image-to-image* tasks, which take image inputs and return image outputs, to map between different kinds of images. For instance, consider semantic segmentation for categories, contours for grouping, intrinsic image decomposition for shape/paint/light, optical flow for motion, stylization for appearance, and many more. Figure 1.1 illustrates a selection of such tasks. While there are plenty of specifics, there is nevertheless common structure.

This thesis pursues a unified approach for learning image-to-image tasks end-to-end and pixels-to-pixels by harnessing locality structure. We define, design, optimize, and analyze purely local *fully convolutional* networks. Throughout we focus on semantic segmentation, the assignment of a category to every pixel, as emblematic of the general image-to-image problem. Learning engages with the blooming, buzzing confusion of the visual world by data while locality defines the scope of this engagement by design. Decomposing the representations and computations for vision into local modeling and processing is essential for practicality (and respects the nature of reality). This thesis argues for the reconciliation of learning and structure as a route to effective and efficient visual processing. We blur the line between learning and structure to yield models that respect locality and decide its degree by optimization. Incorporating structure in this fashion by composition, not substitution, points out a path to learning and inferring more, not less.

First, in Chapter 2 we explain fully convolutional networks: machines for image-to-image learning and inference that harness locality to decompose tasks into local scopes. Clockwork convolutional networks (Chapter 3) adaptively compute layers according to their rate of change to improve the efficiency of video processing. Guided networks (Chapter 4) extract a latent task representation from local supervision for non-local inference within and across images with quick updates for real-time interaction. Composing free-form filters with structured Gaussian filters, and learning both, optimizes over receptive size to decide the degree of locality (Chapter 5). Finally, in Chapter 6 we consider directions for further progress and the fuller understanding of fully convolutional networks. (In the appendices we attend to two key prerequisites for this thesis: the brewing of Caffe and coffee.)



(a) semantic segmentation (b) depth and normals (c) optical flow

(d) contours (e) colorization (f) stylization and translation

Figure 1.1: Image-to-image tasks are many and varied. While each kind of image and task has its respective details, they all share certain structures, like locality and scale. The general machinery of fully convolutional networks, the core subject of this thesis, serves as a unifying framework that has been adopted and extended for the tasks highlighted here and more. (Figures from [LSD15; EF15; Fis+15; XT15; ZIE16; Zhu+17]).

Summary of Open-Source Contributions

Code for the reproduction and extension of research in this thesis is free and open-source:

- `https://caffe.berkeleyvision.org`: Caffe is a deep learning framework made with expression, speed, and modularity in mind [Jia+14]. The swell in deep learning was propelled in part by a wave of open science and toolkits, including Caffe and its model zoo. Leading the development of Caffe from version 0.1 to 1.0 was a significant effort and core part of the technical and social contributions of this thesis.
- `https://fcn.berkeleyvision.org`: reference implementation of the fully convolutional networks in Chapter 2.
- `https://github.com/shelhamer/clockwork-fcn`: code and notebooks for the clockwork convolutional networks in Chapter 3.
- `https://github.com/shelhamer/revolver`: code and notebooks for the guided networks in Chapter 4.
- `https://github.com/shelhamer/fog`: code and notebooks for the free-form and structured filtering in Chapter 5.

Further open-source contributions can be found at `https://github.com/shelhamer`.

Summary of Publications

This thesis incorporates the following publications:

Material in Chapter 2 is published as

[SLD17; LSD15] Evan Shelhamer*, Jon Long*, Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation." *PAMI, 2017* and *CVPR, 2015*.

Material in Chapter 3 is published as

[She+16] Evan Shelhamer*, Kate Rakelly*, Judy Hoffman*, Trevor Darrell. "Clockwork Convnets for Video Semantic Segmentation". *ECCVW, 2016*.

Material in Chapter 4 is self-published as

[Rak+18] Kate Rakelly*, Evan Shelhamer*, Trevor Darrell, Alexei A. Efros, Sergey Levine. "Few-shot Segmentation Propagation with Guided Networks." *arXiv, 2018*.

Material in Chapter 5 is self-published as

[SWD19] Evan Shelhamer, Dequan Wang, Trevor Darrell. "Blurring the Line between Structure and Learning to Optimize and Adapt Receptive Fields." *arXiv, 2019*.

Chapter 2

Local Learning & Inference of Image-to-Image Tasks

Convolutional networks are driving advances in recognition. Convnets are not only improving for whole-image classification [KSH12; SZ15; Sze+15], but also making progress on local tasks with structured output. These include advances in bounding box object detection [Ser+14; Gir+15; He+14], part and keypoint prediction [Zha+14; LZD14], and local correspondence [LZD14; FDB14].

The natural next step in the progression from coarse to fine inference is to make a prediction at every pixel. Prior approaches have used convnets for semantic segmentation [Nin+05; Cir+12; Far+13; PC14; Har+14; Gup+14; GL14], in which each pixel is labeled with the class of its enclosing object or region, but with shortcomings that this work addresses.

We show that fully convolutional networks (FCNs) trained end-to-end, pixels-to-pixels on semantic segmentation exceed the previous best results without further machinery. To our knowledge, this is the first work to train FCNs end-to-end (1) for pixelwise prediction and (2) from supervised pre-training. Fully convolutional versions of existing networks predict dense outputs from arbitrary-sized inputs. Both learning and inference are performed whole-image-at-a-time by dense feedforward computation and backpropagation. In-network upsampling layers enable pixelwise prediction and learning in nets with subsampling.

This method is efficient, both asymptotically and absolutely, and precludes the need for the complications in other works. Patchwise training is common [Nin+05; Cir+12; Far+13; PC14; GL14], but lacks the efficiency of fully convolutional training. Our approach does not make use of pre- and post-processing complications, including superpixels [Far+13; Har+14], proposals [Har+14; Gup+14], or post-hoc refinement by random fields or local classifiers [Far+13; Har+14]. Our model transfers recent success in classification [KSH12; SZ15; Sze+15] to dense prediction by reinterpreting classification nets as fully convolutional and fine-tuning from their learned representations. In contrast, previous works have applied small convnets without supervised pre-training [Far+13; PC14; Nin+05].

Semantic segmentation faces an inherent tension between semantics and location: global information resolves *what* while local information resolves *where*. What can be done to

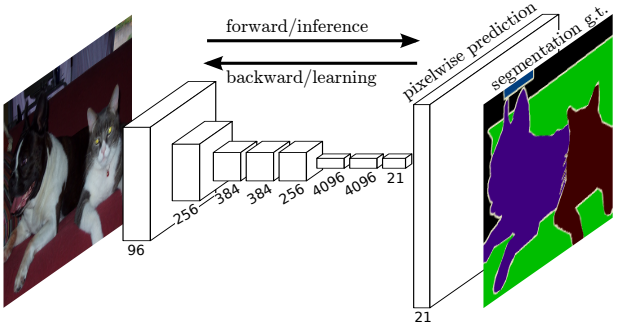


Figure 2.1: Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

navigate this spectrum from location to semantics? How can local decisions respect global structure? It is not immediately clear that deep networks for image classification yield representations sufficient for accurate, pixelwise recognition.

We cast pre-trained networks into fully convolutional form, and augment them with a skip architecture that takes advantage of the full feature spectrum. The skip architecture fuses the feature hierarchy to combine deep, coarse, semantic information and shallow, fine, appearance information (see Section 2.3.3 and Figure 2.3). In this light, deep feature hierarchies encode location and semantics in a nonlinear local-to-global pyramid.

For a fuller understanding of fully convolutional networks, we then carry out further tuning, analysis, and benchmark experiments. Alternative choices, ablations, and implementation details better cover the space of FCNs. Tuning optimization leads to more accurate networks and a means to learn skip architectures all-at-once instead of in stages. Experiments that mask foreground and background investigate the role of context and shape. Results on the object and scene labeling of PASCAL-Context reinforce merging object segmentation and scene parsing as unified pixelwise prediction.

In the next section, we review related work on deep classification nets, FCNs, recent approaches to semantic segmentation using convnets, and extensions to FCNs. The following sections explain FCN design, introduce our architecture with in-network upsampling and skip layers, and describe our experimental framework. Next, we demonstrate improved accuracy on PASCAL VOC 2011-2, NYUDv2, SIFT Flow, and PASCAL-Context. Finally, we analyze design choices, examine what cues can be learned by an FCN, and calculate recognition bounds for semantic segmentation.

2.1 Related Work

Our approach draws on recent successes of deep nets for image classification [KSH12; SZ15; Sze+15] and transfer learning [Don+14; ZF14]. Transfer was first demonstrated on various visual recognition tasks [Don+14; ZF14], then on detection, and on both instance and semantic segmentation in hybrid proposal-classifier models [Gir+15; Har+14; Gup+14]. We

now re-architect and fine-tune classification nets to direct, dense prediction of semantic segmentation. We chart the space of FCNs and relate prior models both historical and recent.

Fully convolutional networks To our knowledge, the idea of extending a convnet to arbitrary-sized inputs first appeared in Matan *et al.*[Mat+91], which extended the classic LeNet [LeC+98a] to recognize strings of digits. Because their net was limited to one-dimensional input strings, Matan *et al.*used Viterbi decoding to obtain their outputs. Wolf and Platt [WP94] expand convnet outputs to 2-dimensional maps of detection scores for the four corners of postal address blocks. Both of these historical works do inference and learning fully convolutionally for detection. Ning *et al.*[Nin+05] define a convnet for coarse multiclass segmentation of *C. elegans* tissues with fully convolutional inference.

Fully convolutional computation has also been exploited in the present era of many-layered nets. Sliding window detection by Sermanet *et al.*[Ser+14], semantic segmentation by Pinheiro and Collobert [PC14], and image restoration by Eigen *et al.*[EKF13] do fully convolutional inference. Fully convolutional training is rare, but used effectively by Tompson *et al.*[Tom+14] to learn an end-to-end part detector and spatial model for pose estimation, although they do not exposit on or analyze this method.

Dense prediction with convnets Several recent works have applied convnets to dense prediction problems, including semantic segmentation by Ning *et al.*[Nin+05], Farabet *et al.*[Far+13], and Pinheiro and Collobert [PC14]; boundary prediction for electron microscopy by Ciresan *et al.*[Cir+12] and for natural images by a hybrid convnet/nearest neighbor model by Ganin and Lempitsky [GL14]; and image restoration and depth estimation by Eigen *et al.*[EKF13; EPF14]. Common elements of these approaches include

- small models restricting capacity and receptive fields;
- patchwise training [Nin+05; Cir+12; Far+13; PC14; GL14];
- refinement by superpixel projection, random field regularization, filtering, or local classification [Far+13; Cir+12; GL14];
- "interlacing" to obtain dense output [Ser+14; PC14; GL14];
- multi-scale pyramid processing [Far+13; PC14; GL14];
- saturating tanh nonlinearities [Far+13; EKF13; PC14]; and
- ensembles [Cir+12; GL14],

whereas our method does without this machinery. However, we do study patchwise training (Section 2.2.4) and "shift-and-stitch" dense output (Section 2.2.2) from the perspective of FCNs. We also discuss in-network upsampling (Section 2.2.3), of which the fully connected prediction by Eigen *et al.*[EPF14] is a special case.

Unlike these existing methods, we adapt and extend deep classification architectures, using image classification as supervised pre-training, and fine-tune fully convolutionally to learn simply and efficiently from whole image inputs and whole image ground thruths.

Hariharan *et al.*[Har+14] and Gupta *et al.*[Gup+14] likewise adapt deep classification nets to semantic segmentation, but do so in hybrid proposal-classifier models. These approaches fine-tune an R-CNN system [Gir+15] by sampling bounding boxes and/or region proposals for detection, semantic segmentation, and instance segmentation. Neither method is learned end-to-end. They achieve the previous best segmentation results on PASCAL VOC and NYUDv2 respectively, so we directly compare our standalone, end-to-end FCN to their semantic segmentation results in Section 3.4.

Combining feature hierarchies We fuse features across layers to define a nonlinear local-to-global representation that we tune end-to-end. The Laplacian pyramid [BA83] is a classic multi-scale representation made of fixed smoothing and differencing. The jet of Koenderink and van Doorn [KD87] is a rich, local feature defined by compositions of partial derivatives. In the context of deep networks, Sermanet *et al.*[Ser+13] fuse intermediate layers but discard resolution in doing so. In contemporary work Hariharan *et al.*[Har+15] and Mostajabi *et al.*[MYS15] also fuse multiple layers but do not learn end-to-end and rely on fixed bottom-up grouping.

FCN extensions Following the publication of fully convolutional networks [LSD15; SLD17], FCNs have been extended to new tasks and data. Tasks include region proposals [Ren+15], contour detection [XT15], depth regression [Liu+15], optical flow [Fis+15], and weakly-supervised semantic segmentation [PKD15; Pap+15; DHS15a; HNH15].

In addition, new works have improved the FCNs presented here to further advance the state-of-the-art in semantic segmentation. The DeepLab models [Che+15; Che+18a] raise output resolution by dilated convolution and dense CRF inference. The joint CRFasRNN [Zhe+15] model is an end-to-end integration of the CRF for further improvement. ParseNet [LRB15] normalizes features for fusion and captures context with global pooling. The "deconvolutional network" approach of [NHH15] restores resolution by proposals, stacks of learned deconvolution, and unpooling. U-Net [RFB15] combines skip layers and learned deconvolution for pixel labeling of microscopy images. The dilation architecture of [YK16] makes thorough use of dilated convolution for pixel-precise output without a random field or skip layers.

2.2 Fully Convolutional Networks

Each layer output in a convnet is a three-dimensional array of size $h \times w \times d$, where h and w are spatial dimensions, and d is the feature or channel dimension. The first layer is the image, with pixel size $h \times w$, and d channels. Locations in higher layers correspond to the locations in the image they are path-connected to, which are called their *receptive fields*.

Convnets are inherently translation invariant. Their basic components (convolution, pooling, and activation functions) operate on local input regions, and depend only on *relative* spatial coordinates. Writing $\mathbf{x}_{ij}$ for the data vector at location (i, j) in a particular layer, and $\mathbf{y}_{ij}$ for the following layer, these functions compute outputs $\mathbf{y}_{ij}$ by

$$\mathbf{y}_{ij} = f_{ks}\left(\{\mathbf{x}_{si+\delta i, sj+\delta j}\}_{0 \le \delta i, \delta j < k}\right)$$

where k is called the kernel size, s is the stride or subsampling factor, and f_{ks} determines the layer type: a matrix multiplication for convolution or average pooling, a spatial max for max pooling, or an elementwise nonlinearity for an activation function, and so on for other types of layers.

This functional form is maintained under composition, with kernel size and stride obeying the transformation rule

$$f_{ks} \circ g_{k's'} = (f \circ g)_{k'+(k-1)s',ss'}.$$

While a general net computes a general nonlinear function, a net with only layers of this form computes a nonlinear *filter*, which we call a *deep filter* or *fully convolutional network*. An FCN naturally operates on an input of any size, and produces an output of corresponding (possibly resampled) spatial dimensions.

A real-valued loss function composed with an FCN defines a task. If the loss function is a sum over the spatial dimensions of the final layer, $\ell(\mathbf{x};\theta) = \sum_{ij} \ell'(\mathbf{x}_{ij};\theta)$, its parameter gradient will be a sum over the parameter gradients of each of its spatial components. Thus stochastic gradient descent on ℓ computed on whole images will be the same as stochastic gradient descent on ℓ', taking all of the final layer receptive fields as a minibatch.

When these receptive fields overlap significantly, both feedforward computation *and* backpropagation are much more efficient when computed layer-by-layer over an entire image instead of independently patch-by-patch.

We next explain how to convert classification nets into fully convolutional nets that produce coarse output maps. For pixelwise prediction, we need to connect these coarse outputs back to the pixels. Section 2.2.2 describes a trick used for this purpose (e.g., by "fast scanning" [Giu+13]). We explain this trick in terms of network modification. As an efficient, effective alternative, we upsample in Section 2.2.3, reusing our implementation of convolution. In Section 2.2.4 we consider training by patchwise sampling, and give evidence in Section 2.3.4 that our whole image training is faster and equally effective.

2.2.1 From Fully Connected to Fully Convolutional

Typical recognition nets, including LeNet [LeC+98a], AlexNet [KSH12], and its deeper successors [SZ15; Sze+15], ostensibly take fixed-sized inputs and produce non-spatial outputs. The fully connected layers of these nets have fixed dimensions and throw away spatial coordinates. However, fully connected layers can also be viewed as convolutions with kernels that cover their entire input regions. Doing so casts these nets into fully convolutional networks that take input of any size and make spatial output maps. This transformation is illustrated in Figure 2.2.

Furthermore, while the resulting maps are equivalent to the evaluation of the original net on particular input patches, the computation is highly amortized over the overlapping regions of those patches. For example, while AlexNet takes 1.2 ms (on a typical GPU) to infer the classification scores of a 227×227 image, the fully convolutional net takes 22 ms

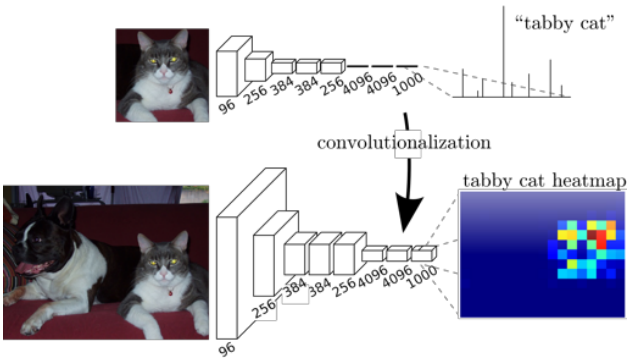


Figure 2.2: Transforming fully connected layers into convolution layers enables a classification net to output a spatial map. Adding differentiable interpolation layers and a spatial loss (as in Figure 4.1) produces an efficient machine for end-to-end pixelwise learning.

to produce a 10×10 grid of outputs from a 500×500 image, which is more than 5 times faster than the naïve approach[1].

The spatial output maps of these convolutionalized models make them a natural choice for dense problems like semantic segmentation. With ground truth available at every output cell, both the forward and backward passes are straightforward, and both take advantage of the inherent computational efficiency (and aggressive optimization) of convolution. The corresponding backward times for the AlexNet example are 2.4 ms for a single image and 37 ms for a fully convolutional 10×10 output map, resulting in a speedup similar to that of the forward pass.

While our reinterpretation of classification nets as fully convolutional yields output maps for inputs of any size, the output dimensions are typically reduced by subsampling. The classification nets subsample to keep filters small and computational requirements reasonable. This coarsens the output of a fully convolutional version of these nets, reducing it from the size of the input by a factor equal to the pixel stride of the receptive fields of the output units.

2.2.2 Shift-and-Stitch is Filter Dilation

Dense predictions can be obtained from coarse outputs by stitching together outputs from shifted versions of the input. If the output is downsampled by a factor of f, shift the input x pixels to the right and y pixels down, once for every (x, y) such that $0 \leq x, y < f$. Process each of these f^2 inputs, and interlace the outputs so that the predictions correspond to the pixels at the *centers* of their receptive fields.

Although this transformation naïvely increases the cost by a factor of f^2, there is a well-known trick for efficiently producing identical results [Giu+13; Ser+14]. (This trick is also

[1] Assuming efficient batching of single image inputs. The classification scores for a single image by itself take 5.4 ms to produce, which is nearly 25 times slower than the fully convolutional version.

used in the algorithme à trous [Hol+89; Mal99] for wavelet transforms and related to the Noble identities [Vai90] from signal processing.)

Consider a layer (convolution or pooling) with input stride s, and a subsequent convolution layer with filter weights f_{ij} (eliding the irrelevant feature dimensions). Setting the earlier layer's input stride to one upsamples its output by a factor of s. However, convolving the original filter with the upsampled output does not produce the same result as shift-and-stitch, because the original filter only sees a reduced portion of its (now upsampled) input. To produce the same result, dilate (or "rarefy") the filter by forming

$$f'_{ij} = \begin{cases} f_{i/s,j/s} & \text{if } s \text{ divides both } i \text{ and } j; \\ 0 & \text{otherwise,} \end{cases}$$

(with i and j zero-based). Reproducing the full net output of shift-and-stitch involves repeating this filter enlargement layer-by-layer until all subsampling is removed. (In practice, this can be done efficiently by processing subsampled versions of the upsampled input.)

Simply decreasing subsampling within a net is a tradeoff: the filters see finer information, but have smaller receptive fields and take longer to compute. This dilation trick is another kind of tradeoff: the output is denser without decreasing the receptive field sizes of the filters, but the filters are prohibited from accessing information at a finer scale than their original design.

Although we have done preliminary experiments with dilation, we do not use it in our model. We find learning through upsampling, as described in the next section, to be effective and efficient, especially when combined with the skip layer fusion described later on. For further detail regarding dilation, refer to the dilated FCN of [YK16].

2.2.3 Upsampling is (Fractionally Strided) Convolution

Another way to connect coarse outputs to dense pixels is interpolation. For instance, simple bilinear interpolation computes each output y_{ij} from the nearest four inputs by a linear map that depends only on the relative positions of the input and output cells:

$$y_{ij} = \sum_{\alpha,\beta=0}^{1} |1 - \alpha - \{i/f\}| \; |1 - \beta - \{i/j\}| \; x_{\lfloor i/f \rfloor + \alpha, \lfloor j/f \rfloor + \beta},$$

where f is the upsampling factor, and $\{\cdot\}$ denotes the fractional part.

In a sense, upsampling with factor f is convolution with a *fractional* input stride of $1/f$. So long as f is integral, it's natural to implement upsampling through "backward convolution" by reversing the forward and backward passes of more typical input-strided convolution. Thus upsampling is performed in-network for end-to-end learning by backpropagation from the pixelwise loss.

Per their use in deconvolution networks (esp. [ZF14]), these (convolution) layers are sometimes referred to as *deconvolution* layers. Note that the convolution filter in such a layer

need not be fixed (e.g., to bilinear upsampling), but can be learned. A stack of deconvolution layers and activation functions can even learn a nonlinear upsampling.

In our experiments, we find that in-network upsampling is fast and effective for learning dense prediction.

2.2.4 Patchwise Training is Loss Sampling

In stochastic optimization, gradient computation is driven by the training distribution. Both patchwise training and fully convolutional training can be made to produce any distribution of the inputs, although their relative computational efficiency depends on overlap and mini-batch size. Whole image fully convolutional training is identical to patchwise training where each batch consists of all the receptive fields of the output units for an image (or collection of images). While this is more efficient than uniform sampling of patches, it reduces the number of possible batches. However, random sampling of patches within an image may be easily recovered. Restricting the loss to a randomly sampled subset of its spatial terms (or, equivalently applying a DropConnect mask [Wan+13] between the output and the loss) excludes patches from the gradient.

If the kept patches still have significant overlap, fully convolutional computation will still speed up training. If gradients are accumulated over multiple backward passes, batches can include patches from several images. If inputs are shifted by values up to the output stride, random selection of all possible patches is possible even though the output units lie on a fixed, strided grid.

Sampling in patchwise training can correct class imbalance [Nin+05; Far+13; Cir+12] and mitigate the spatial correlation of dense patches [PC14; Har+14]. In fully convolutional training, class balance can also be achieved by weighting the loss, and loss sampling can be used to address spatial correlation.

We explore training with sampling in Section 2.3.4, and do not find that it yields faster or better convergence for dense prediction. Whole image training is effective and efficient.

2.3 Segmentation Architecture

We cast ILSVRC classifiers into FCNs and augment them for dense prediction with in-network upsampling and a pixelwise loss. We train for segmentation by fine-tuning. Next, we add skips between layers to fuse coarse, semantic and local, appearance information. This skip architecture is learned end-to-end to refine the semantics and spatial precision of the output.

For this investigation, we train and validate on the PASCAL VOC 2011 segmentation challenge [Eve+10]. We train with a per-pixel softmax loss and validate with the standard metric of mean pixel intersection over union, with the mean taken over all classes, including background. The training ignores pixels that are masked out (as ambiguous or difficult) in the ground truth.

Table 2.1: We adapt and extend three classification convnets. We compare performance by mean intersection over union on the validation set of PASCAL VOC 2011 and by inference time (averaged over 20 trials for a 500×500 input on an NVIDIA Titan X). We detail the architecture of the adapted nets with regard to dense prediction: number of parameter layers, receptive field size of output units, and the coarsest stride within the net. (These numbers give the best performance obtained at a fixed learning rate, not best performance possible.)

	FCN-AlexNet	FCN-VGG16	FCN-GoogLeNet[3]
mean IU	39.8	**56.0**	42.5
forward time	16 ms	100 ms	20 ms
conv. layers	8	16	22
parameters	57M	134M	6M
rf size	355	404	907
max stride	32	32	32

2.3.1 Pixel-to-Pixel Segmentation from Image Classification

We begin by convolutionalizing proven classification architectures from ILSVRC [Rus+15] as in Section 2.2. We consider the AlexNet[2] architecture [KSH12] that won ILSVRC12, as well as the VGG nets [SZ15] and the GoogLeNet[3] [Sze+15] which did exceptionally well in ILSVRC14. We pick the VGG 16-layer net[4], which we found to be equivalent to the 19-layer net on this task. For GoogLeNet, we use only the final loss layer, and improve performance by discarding the final average pooling layer. We decapitate each net by discarding the final classifier layer, and convert all fully connected layers to convolutions. We append a 1×1 convolution with channel dimension 21 to predict scores for each of the PASCAL classes (including background) at each of the coarse output locations, followed by a (backward) convolution layer to bilinearly upsample the coarse outputs to pixelwise outputs as described in Section 2.2.3. Table 2.1 compares the preliminary validation results along with the basic characteristics of each net. We report the best results achieved after convergence at a fixed learning rate (at least 175 epochs).

Our training for this comparison follows the practices for classification networks. We train by SGD with momentum. Gradients are accumulated over 20 images. We set fixed learning rates of 10^{-3}, 10^{-4}, and 5^{-5} for FCN-AlexNet, FCN-VGG16, and FCN-GoogLeNet, respectively, chosen by line search. We use momentum 0.9, weight decay of 5^{-4} or 2^{-4}, and doubled learning rate for biases. We zero-initialize the class scoring layer, as random initialization yielded neither better performance nor faster convergence. Dropout is included

[2]Using the publicly available `CaffeNet` reference model.

[3]We use our own reimplementation of GoogLeNet. Ours is trained with less extensive data augmentation, and gets 68.5% top-1 and 88.4% top-5 ILSVRC accuracy.

[4]Using the publicly available version from the Caffe model zoo.

Table 2.2: Comparison of image-to-image optimization by gradient accumulation, online learning, and "heavy" learning with high momentum. All methods are trained on a fixed sequence of 100,000 images (sampled from a dataset of 8,498) to control for stochasticity and equalize the number of gradient computations. The loss is not normalized so that every pixel has the same weight no matter the batch and image dimensions. Scores are the best achieved during training on a subset[5] of PASCAL VOC 2011 segval. Learning is end-to-end with FCN-VGG16.

	batch size	mom.	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-accum	20	0.9	86.0	66.5	51.9	76.5
FCN-online	1	0.9	89.3	76.2	60.7	81.8
FCN-heavy	1	0.99	**90.5**	**76.5**	**63.6**	**83.5**

where used in the original classifier nets (however, training without it made little to no difference).

Fine-tuning from classification to segmentation gives reasonable predictions from each net. Even the worst model achieved $\sim 75\%$ of the previous best performance. FCN-VGG16 already appears to be better than previous methods at 56.0 mean IU on val, compared to 52.6 on test [Har+14]. Although VGG and GoogLeNet are similarly accurate as classifiers, our FCN-GoogLeNet did not match FCN-VGG16. We select FCN-VGG16 as our base network.

2.3.2 Image-to-Image Learning

The image-to-image learning setting includes high effective batch size and correlated inputs. This optimization requires some attention to properly tune FCNs.

We begin with the loss. We do not normalize the loss, so that every pixel has the same weight regardless of the batch and image dimensions. Thus we use a small learning rate since the loss is summed spatially over all pixels.

We consider two regimes for batch size. In the first, gradients are accumulated over 20 images. Accumulation reduces the memory required and respects the different dimensions of each input by reshaping the network. We picked this batch size empirically to result in reasonable convergence. Learning in this way is similar to standard classification training: each minibatch contains several images and has a varied distribution of class labels. The nets compared in Table 2.1 are optimized in this fashion.

However, batching is not the only way to do image-wise learning. In the second regime, batch size *one* is used for online learning. Properly tuned, online learning achieves higher accuracy and faster convergence in both number of iterations and wall clock time. Additionally, we try a higher momentum of 0.99, which increases the weight on recent gradients in a similar way to batching. See Table 2.2 for the comparison of accumulation, online, and high momentum or "heavy" learning (discussed further in Section 2.5.2).

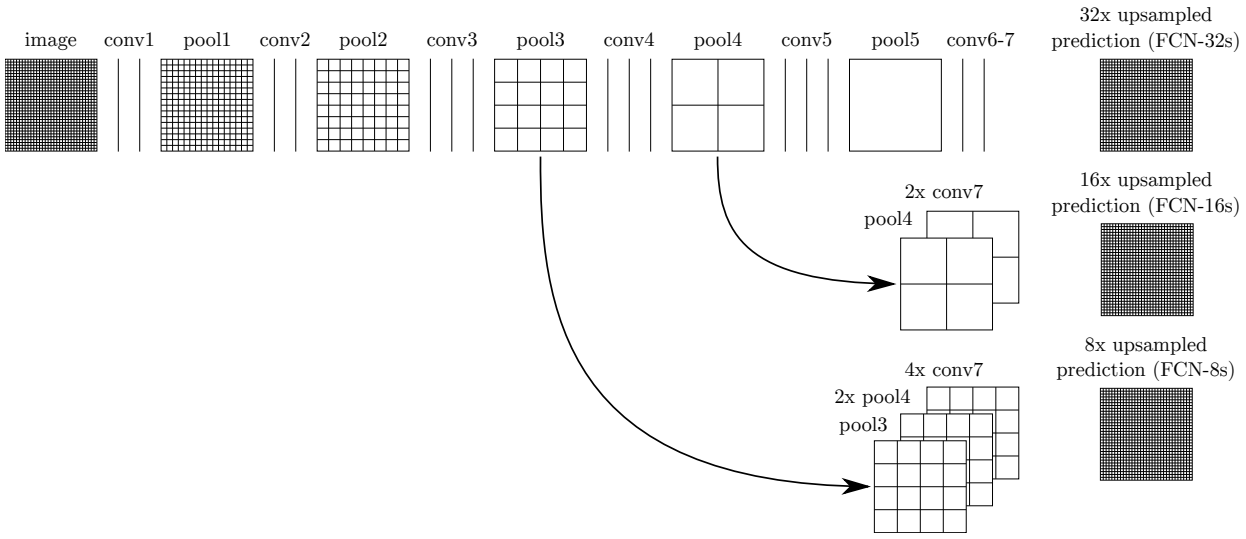


Figure 2.3: Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Our single-stream net, described in Section 2.3.1, upsamples stride 32 predictions back to pixels in a single step. Second row (FCN-16s): Combining predictions from both the final layer and the `pool4` layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Third row (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.

2.3.3 Combining *What* and *Where*

We define a new fully convolutional net for segmentation that combines layers of the feature hierarchy and refines the spatial precision of the output. See Figure 2.3.

While fully convolutionalized classifiers fine-tuned to semantic segmentation both recognize and localize, as shown in Section 2.3.1, these networks can be improved to make direct use of shallower, more local features. Even though these base networks score highly on the standard metrics, their output is dissatisfyingly coarse (see Figure 2.4). The stride of the network prediction limits the scale of detail in the upsampled output.

We address this by adding skips [Bis06] that fuse layer outputs, in particular to include shallower layers with finer strides in prediction. This turns a line topology into a DAG: edges skip ahead from shallower to deeper layers. It is natural to make more local predictions from shallower layers since their receptive fields are smaller and see fewer pixels. Once augmented with skips, the network makes and fuses predictions from several streams that are learned jointly and end-to-end.

Combining fine layers and coarse layers lets the model make local predictions that respect global structure. This crossing of layers and resolutions is a learned, nonlinear counterpart to the multi-scale representation of the Laplacian pyramid [BA83]. By analogy to the jet of

Koenderick and van Doorn [KD87], we call our feature hierarchy the *deep jet*.

Layer fusion is essentially an elementwise operation. However, the correspondence of elements across layers is complicated by resampling and padding. Thus, in general, layers to be fused must be aligned by scaling and cropping. We bring two layers into scale agreement by upsampling the lower-resolution layer, doing so in-network as explained in Section 2.2.3. Cropping removes any portion of the upsampled layer which extends beyond the other layer due to padding. This results in layers of equal dimensions in exact alignment. The offset of the cropped region depends on the resampling and padding parameters of all intermediate layers. Determining the crop that results in exact correspondence can be intricate, but it follows automatically from the network definition (and we include code for it in Caffe).

Having spatially aligned the layers, we next pick a fusion operation. We fuse features by concatenation, and immediately follow with classification by a "score layer" consisting of a 1×1 convolution. Rather than storing concatenated features in memory, we commute the concatenation and subsequent classification (as both are linear). Thus, our skips are implemented by first scoring each layer to be fused by 1×1 convolution, carrying out any necessary interpolation and alignment, and then *summing* the scores. We also considered max fusion, but found learning to be difficult due to gradient switching. The score layer parameters are zero-initialized when a skip is added, so that they do not interfere with existing predictions of other streams. Once all layers have been fused, the final prediction is then upsampled back to image resolution.

Skip Architectures for Segmentation We define a skip architecture to extend FCN-VGG16 to a three-stream net with eight pixel stride shown in Figure 2.3. Adding a skip from `pool4` halves the stride by scoring from this stride sixteen layer. The $2\times$ interpolation layer of the skip is initialized to bilinear interpolation, but is not fixed so that it can be learned as described in Section 2.2.3. We call this two-stream net FCN-16s, and likewise define FCN-8s by adding a further skip from `pool3` to make stride eight predictions. (Note that predicting at stride eight does not significantly limit the maximum achievable mean IU; see Section 2.5.3.)

We experiment with both *staged training* and *all-at-once training*. In the staged version, we learn the single-stream FCN-32s, then upgrade to the two-stream FCN-16s and continue learning, and finally upgrade to the three-stream FCN-8s and finish learning. At each stage the net is learned end-to-end, initialized with the parameters of the earlier net. The learning rate is dropped $100\times$ from FCN-32s to FCN-16s and $100\times$ more from FCN-16s to FCN-8s, which we found to be necessary for continued improvements.

Learning all-at-once rather than in stages gives nearly equivalent results, while training is faster and less tedious. However, disparate feature scales make naïve training prone to divergence. To remedy this we scale each stream by a fixed constant, for a similar in-network effect to the staged learning rate adjustments. These constants are picked to approximately equalize average feature norms across streams. (Other normalization schemes should have similar effect.)

With FCN-16s validation score improves to 65.0 mean IU, and FCN-8s brings a minor improvement to 65.5. At this point our fusion improvements have met diminishing returns,

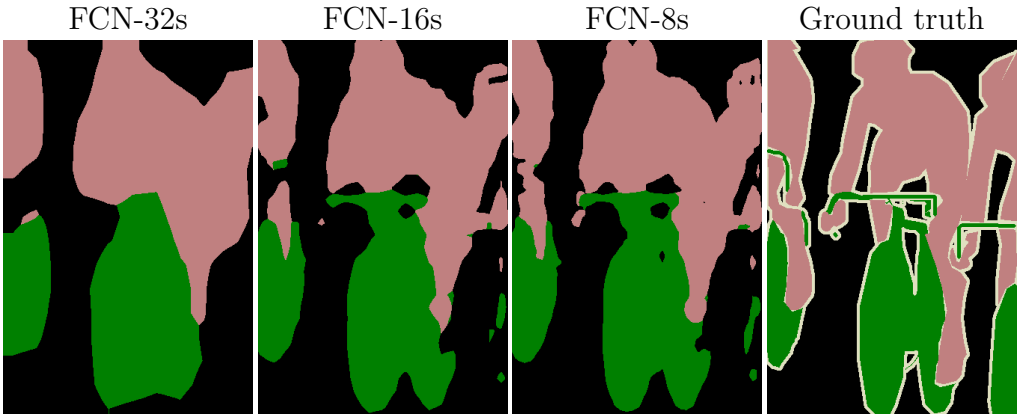


Figure 2.4: Refining fully convolutional networks by fusing information from layers with different strides improves spatial detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 2.3).

so we do not continue fusing even shallower layers.

To identify the contribution of the skips we compare scoring from the intermediate layers in isolation, which results in poor performance, or dropping the learning rate without adding skips, which gives negligible improvement in score without refining the visual quality of output. All skip comparisons are reported in Table 2.3. Figure 2.4 shows the progressively finer structure of the output.

Table 2.3: Comparison of FCNs on a subset[5] of PASCAL VOC 2011 segval. Learning is end-to-end with batch size one and high momentum, with the exception of the fixed variant that fixes all features. Note that FCN-32s is FCN-VGG16, renamed to highlight stride, and the FCN-poolX are truncated nets with the same strides as FCN-32/16/8s.

	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-32s	90.5	76.5	63.6	83.5
FCN-16s	91.0	78.1	65.0	84.3
FCN-8s at-once	91.1	**78.5**	65.4	84.4
FCN-8s staged	**91.2**	77.6	**65.5**	**84.5**
FCN-32s fixed	82.9	64.6	46.6	72.3
FCN-pool5	87.4	60.5	50.0	78.5
FCN-pool4	78.7	31.7	22.4	67.0
FCN-pool3	70.9	13.7	9.2	57.6

2.3.4 Experimental Framework

Fine-tuning We fine-tune all layers by backpropagation through the whole net. Fine-tuning the output classifier alone yields only 73% of the full fine-tuning performance as compared in Table 2.3. Fine-tuning in stages takes 36 hours on a single GPU. Learning FCN-8s all-at-once takes half the time to reach comparable accuracy. Training from scratch gives substantially lower accuracy.

More training data The PASCAL VOC 2011 segmentation training set labels 1,112 images. Hariharan *et al.*[Har+11] collected labels for a larger set of 8,498 PASCAL training images, which was used to train the previous best system, SDS [Har+14]. This training data improves the FCN-32s validation score[5] from 57.7 to 63.6 mean IU and improves the FCN-AlexNet score from 39.8 to 48.0 mean IU.

Loss The per-pixel, unnormalized softmax loss is a natural choice for segmenting images of any size into disjoint classes, so we train our nets with it. The softmax operation induces competition between classes and promotes the most confident prediction, but it is not clear that this is necessary or helpful. For comparison, we train with the sigmoid cross-entropy loss and find that it gives similar results, even though it normalizes each class prediction independently.

Patch sampling As explained in Section 2.2.4, our whole image training effectively batches each image into a regular grid of large, overlapping patches. By contrast, prior work randomly samples patches over a full dataset [Nin+05; Cir+12; Far+13; PC14; GL14], potentially resulting in higher variance batches that may accelerate convergence [LeC+98c]. We study this tradeoff by spatially sampling the loss in the manner described earlier, making an independent choice to ignore each final layer cell with some probability $1 - p$. To avoid changing the effective batch size, we simultaneously increase the number of images per batch by a factor $1/p$. Note that due to the efficiency of convolution, this form of rejection sampling is still faster than patchwise training for large enough values of p (e.g., at least for $p > 0.2$ according to the numbers in Section 2.2.1). Figure 2.5 shows the effect of this form of sampling on convergence. We find that sampling does not have a significant effect on convergence rate compared to whole image training, but takes significantly more time due to the larger number of images that need to be considered per batch. We therefore choose unsampled, whole image training in our other experiments.

Class balancing Fully convolutional training can balance classes by weighting or sampling the loss. Although our labels are mildly unbalanced (about 3/4 are background), we find class balancing unnecessary.

Dense Prediction The scores are upsampled to the input dimensions by backward convolution layers within the net. Final layer backward convolution weights are fixed to bilinear interpolation, while intermediate upsampling layers are initialized to bilinear interpolation, and then learned. This simple, end-to-end method is accurate and fast.

[5]There are training images from [Har+11] included in the PASCAL VOC 2011 val set, so we validate on the non-intersecting set of 736 images.

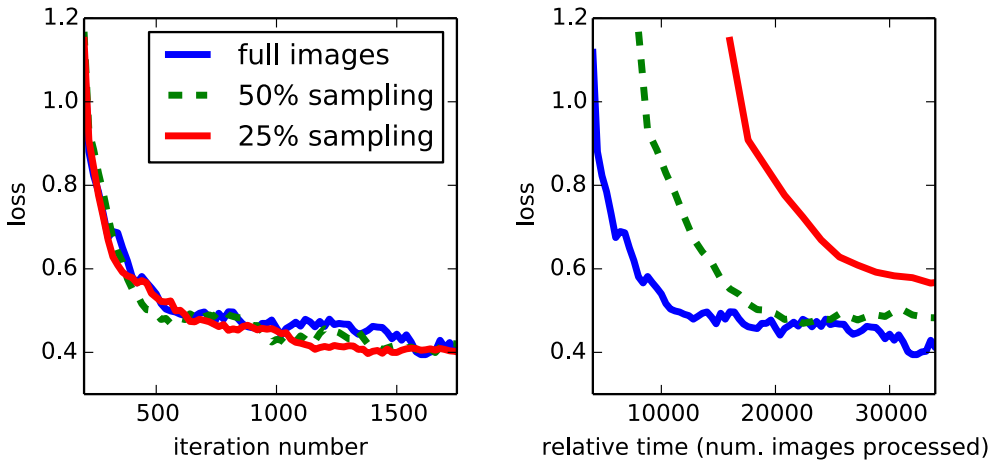


Figure 2.5: Training on whole images is just as effective as sampling patches, but results in faster (wall clock time) convergence by making more efficient use of data. Left shows the effect of sampling on convergence rate for a fixed expected batch size, while right plots the same by relative wall clock time.

Augmentation We tried augmenting the training data by randomly mirroring and "jittering" the images by translating them up to 32 pixels (the coarsest scale of prediction) in each direction. This yielded no noticeable improvement.

Implementation All models are trained and tested with Caffe [Jia+14] on a single NVIDIA Titan X. Our models and code are publicly available at `http://fcn.berkeleyvision.org`.

2.4 Results

We test our FCN on semantic segmentation and scene parsing, exploring PASCAL VOC, NYUDv2, SIFT Flow, and PASCAL-Context. Although these tasks have historically distinguished between objects and regions, we treat both uniformly as pixel prediction. We evaluate our FCN skip architecture on each of these datasets, and then extend it to multi-modal input for NYUDv2 and multi-task prediction for the semantic and geometric labels of SIFT Flow. All experiments follow the same network architecture and optimization settings decided on in Section 2.3.

Metrics We report metrics from common semantic segmentation and scene parsing evaluations that are variations on pixel accuracy and region intersection over union (IU):

- pixel accuracy: $\sum_i n_{ii} / \sum_i t_i$
- mean accuraccy: $(1/n_{\text{cl}}) \sum_i n_{ii}/t_i$

- mean IU: $(1/n_{\text{cl}}) \sum_i n_{ii} / \left(t_i + \sum_j n_{ji} - n_{ii}\right)$
- frequency weighted IU: $(\sum_k t_k)^{-1} \sum_i t_i n_{ii} / \left(t_i + \sum_j n_{ji} - n_{ii}\right)$

where n_{ij} is the number of pixels of class i predicted to belong to class j, there are n_{cl} different classes, and $t_i = \sum_j n_{ij}$ is the total number of pixels of class i.

PASCAL VOC Table 2.4 gives the performance of our FCN-8s on the test sets of PASCAL VOC 2011 and 2012, and compares it to the previous best, SDS [Har+14], and the well-known R-CNN [Gir+15]. We achieve the best results on mean IU by 30% relative. Inference time is reduced 114× (convnet only, ignoring proposals and refinement) or 286× (overall).

NYUDv2 [Sil+12] is an RGB-D dataset collected using the Microsoft Kinect. It has 1,449 RGB-D images, with pixelwise labels that have been coalesced into a 40 class semantic segmentation task by Gupta *et al.*[GAM13]. We report results on the standard split of 795 training images and 654 testing images. Table 2.5 gives the performance of several net variations. First we train our unmodified coarse model (FCN-32s) on RGB images. To add depth information, we train on a model upgraded to take four-channel RGB-D input (early fusion). This provides little benefit, perhaps due to similar number of parameters or the difficulty of propagating meaningful gradients all the way through the net. Following the success of Gupta *et al.*[Gup+14], we try the three-dimensional HHA encoding of depth and train a net on just this information. To effectively combine color and depth, we define a "late fusion" of RGB and HHA that averages the final layer scores from both nets and learn the resulting two-stream net end-to-end. This late fusion RGB-HHA net is the most accurate.

SIFT Flow is a dataset of 2,688 images with pixel labels for 33 semantic classes ("bridge", "mountain", "sun"), as well as three geometric classes ("horizontal", "vertical", and "sky"). An FCN can naturally learn a joint representation that simultaneously predicts both types of labels. We learn a two-headed version of FCN-32/16/8s with semantic and geometric prediction layers and losses. This net performs as well on both tasks as two independently trained nets, while learning and inference are essentially as fast as each independent net by itself. The results in Table 2.6, computed on the standard split into 2,488 training and 200 test images,[6] show better performance on both tasks.

PASCAL-Context [Mot+14] provides whole scene annotations of PASCAL VOC 2010. While there are 400+ classes, we follow the 59 class task defined by [Mot+14] that picks the most frequent classes. We train and evaluate on the training and val sets respectively. In Table 2.7 we compare to the previous best result on this task. FCN-8s scores 39.1 mean IU for a relative improvement of more than 10%.

[6] Three of the SIFT Flow classes are not present in the test set. We made predictions across all 33 classes, but only included classes actually present in the test set in our evaluation.

Table 2.4: Our FCN gives a 30% relative improvement on the previous best PASCAL VOC 11/12 test results with faster inference and learning.

	mean IU VOC2011 test	mean IU VOC2012 test	inference time
R-CNN [Gir+15]	47.9	-	-
SDS [Har+14]	52.6	51.6	$\sim$ 50 s
FCN-8s	**67.5**	**67.2**	$\sim$ **100 ms**

Table 2.5: Results on NYUDv2. *RGB-D* is early-fusion of the RGB and depth channels at the input. *HHA* is the depth embedding of [Gup+14] as horizontal disparity, height above ground, and the angle of the local surface normal with the inferred gravity direction. *RGB-HHA* is the jointly trained late fusion model that sums RGB and HHA predictions.

	pixel acc.	mean acc.	mean IU	f.w. IU
Gupta *et al.*[Gup+14]	60.3	-	28.6	47.0
FCN-32s RGB	61.8	44.7	31.6	46.0
FCN-32s RGB-D	62.1	44.8	31.7	46.3
FCN-32s HHA	58.3	35.7	25.2	41.7
FCN-32s RGB-HHA	**65.3**	**44.0**	**33.3**	**48.6**

Table 2.6: Results on SIFT Flow[6] with semantics (center) and geometry (right). Farabet is a multi-scale convnet trained on class-balanced or natural frequency samples. Pinheiro is the multi-scale, recurrent convnet RCNN$_3$ ($\circ^3$). The metric for geometry is pixel accuracy.

	pixel acc.	mean acc.	mean IU	f.w. IU	geom. acc.
Liu *et al.*[LYT11]	76.7	-	-	-	-
Tighe *et al.*[TL10] transfer	-	-	-	-	90.8
Tighe *et al.*[TL13] SVM	75.6	41.1	-	-	-
Tighe *et al.*[TL13] SVM+MRF	78.6	39.2	-	-	-
Farabet *et al.*[Far+13] natural	72.3	50.8	-	-	-
Farabet *et al.*[Far+13] balanced	78.5	29.6	-	-	-
Pinheiro *et al.*[PC14]	77.7	29.8	-	-	-
FCN-8s	**85.9**	**53.9**	41.2	77.2	**94.6**

Table 2.7: Results on PASCAL-Context for the 59 class task. *dai2015convolutional* is convolutional feature masking [DHS15b] and segment pursuit with the VGG net. O_2P is the second order pooling method [Car+12] as reported in the *errata* of [Mot+14].

59 class	pixel acc.	mean acc.	mean IU	f.w. IU
O_2P	-	-	18.1	-
CFM	-	-	34.4	-
FCN-32s	65.5	49.1	36.7	50.9
FCN-16s	66.9	51.3	38.4	52.3
FCN-8s	**67.5**	**52.3**	**39.1**	**53.0**

Table 2.8: The role of foreground, background, and shape cues. All scores are the mean intersection over union metric *e*xcluding background. The architecture and optimization are fixed to those of FCN-32s (*R*eference) and only input masking differs.

	train		test		
	FG	BG	FG	BG	mean IU
Reference	keep	keep	keep	keep	84.8
Reference-FG	keep	keep	keep	mask	81.0
Reference-BG	keep	keep	mask	keep	19.8
FG-only	keep	mask	keep	mask	76.1
BG-only	mask	keep	mask	keep	37.8
Shape	mask	mask	mask	mask	29.1

2.5 Analysis

We examine the learning and inference of fully convolutional networks. Masking experiments investigate the role of context and shape by reducing the input to only foreground, only background, or shape alone. Defining a "null" background model checks the necessity of learning a background classifier for semantic segmentation. We detail an approximation between momentum and batch size to further tune whole image learning. Finally, we measure bounds on task accuracy for given output resolutions to show there is still much to improve.

2.5.1 Cues and Context

Given the large receptive field size of an FCN, it is natural to wonder about the relative importance of foreground and background pixels in the prediction. Is foreground appearance sufficient for inference, or does the context influence the output? Conversely, can a network learn to recognize a class by its shape and context alone?

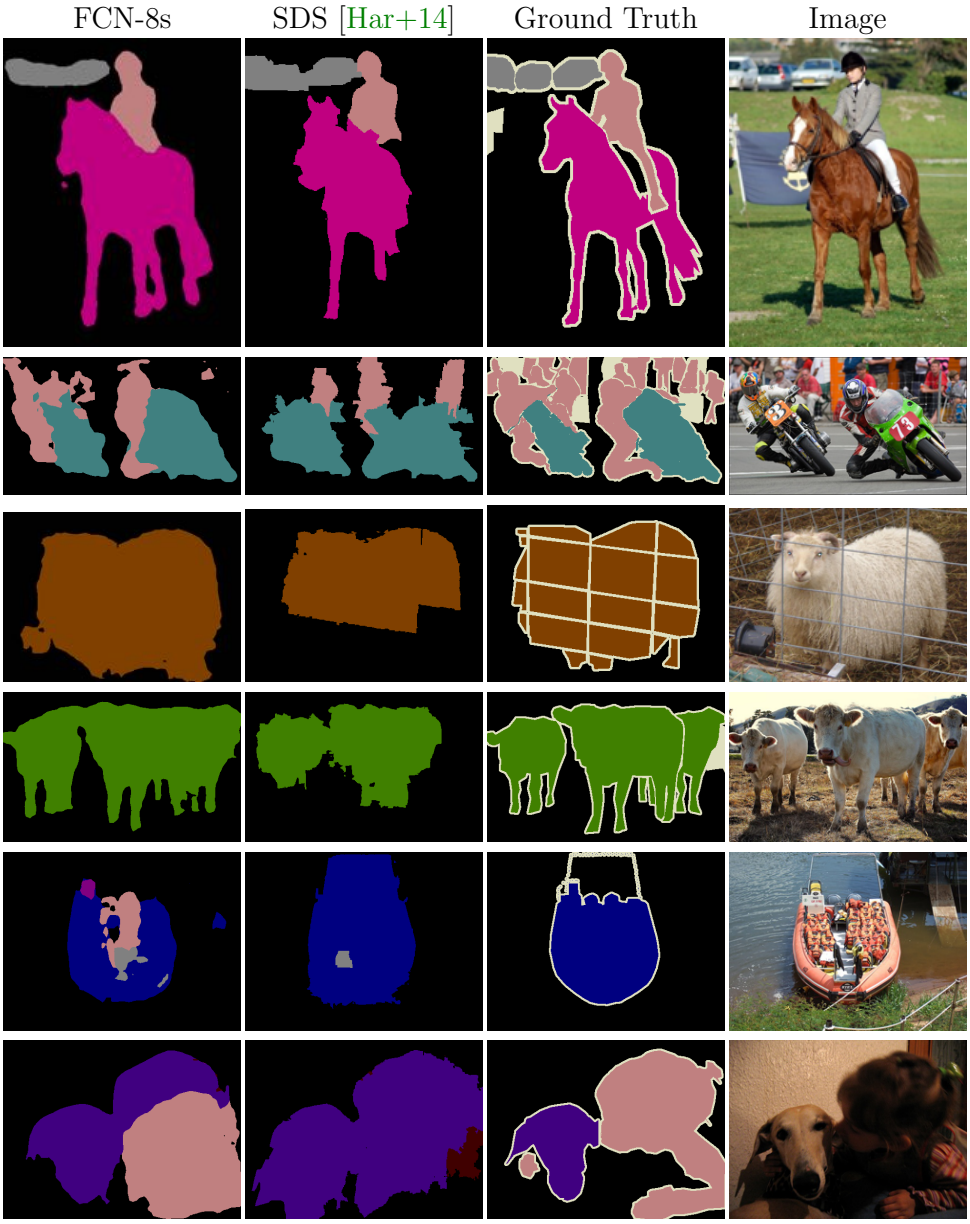


Figure 2.6: Fully convolutional networks improve performance on PASCAL. The left column shows the output of our most accurate net, FCN-8s. The second shows the output of the previous best method by Hariharan *et al.*[Har+14]. Notice the fine structures recovered (first row), ability to separate closely interacting objects (second row), and robustness to occluders (third row). The fifth and sixth rows show failure cases: the net sees lifejackets in a boat as people and confuses human hair with a dog.

Masking To explore these issues we experiment with masked versions of the standard PASCAL VOC segmentation challenge. We both mask input to networks trained on normal

PASCAL, and learn new networks on the masked PASCAL. See Table 2.8 for masked results.

Masking the foreground at inference time is catastrophic. However, masking the foreground during learning yields a network capable of recognizing object segments without observing a single pixel of the labeled class. Masking the background has little effect overall but does lead to class confusion in certain cases. When the background is masked during both learning and inference, the network unsurprisingly achieves nearly perfect background accuracy; however certain classes are more confused. All-in-all this suggests that FCNs do incorporate context even though decisions are driven by foreground pixels.

To separate the contribution of shape, we learn a net restricted to the simple input of foreground/background masks. The accuracy in this shape-only condition is lower than when only the foreground is masked, suggesting that the net is capable of learning context to boost recognition. Nonetheless, it is surprisingly accurate. See Figure 2.7.

Background modeling It is standard in detection and semantic segmentation to have a background model. This model usually takes the same form as the models for the classes of interest, but is supervised by negative instances. In our experiments we have followed the same approach, learning parameters to score all classes including background. Is this actually necessary, or do class models suffice?

To investigate, we define a net with a "null" background model that gives a constant score of zero. Instead of training with the softmax loss, which induces competition by normalizing across classes, we train with the sigmoid cross-entropy loss, which independently normalizes each score. For inference each pixel is assigned the highest scoring class. In all other respects the experiment is identical to our FCN-32s on PASCAL VOC. The null background net scores 1 point lower than the reference FCN-32s and a control FCN-32s trained on all classes including background with the sigmoid cross-entropy loss. To put this drop in perspective, note that discarding the background model in this way reduces the total number of parameters by less than 0.1%. Nonetheless, this result suggests that learning a dedicated background model for semantic segmentation is not vital.

2.5.2 Tuning Optimization Momentum and Batch Size

In comparing optimization schemes for FCNs, we find that "heavy" online learning with high momentum trains more accurate models in less wall clock time (see Section 2.3.2). Here we detail a relationship between momentum and batch size that motivates heavy learning.

By writing the updates computed by gradient accumulation as a non-recursive sum, we will see that momentum and batch size can be approximately traded off, which suggests alternative training parameters. Let g_t be the step taken by minibatch SGD with momentum at time t,

$$g_t = -\eta \sum_{i=0}^{k-1} \nabla_\theta \ell(x_{kt+i}; \theta_{t-1}) + p g_{t-1},$$

where $\ell(x; \theta)$ is the loss for example x and parameters θ, $p < 1$ is the momentum, k is the batch size, and η is the learning rate. Expanding this recurrence as an infinite sum with

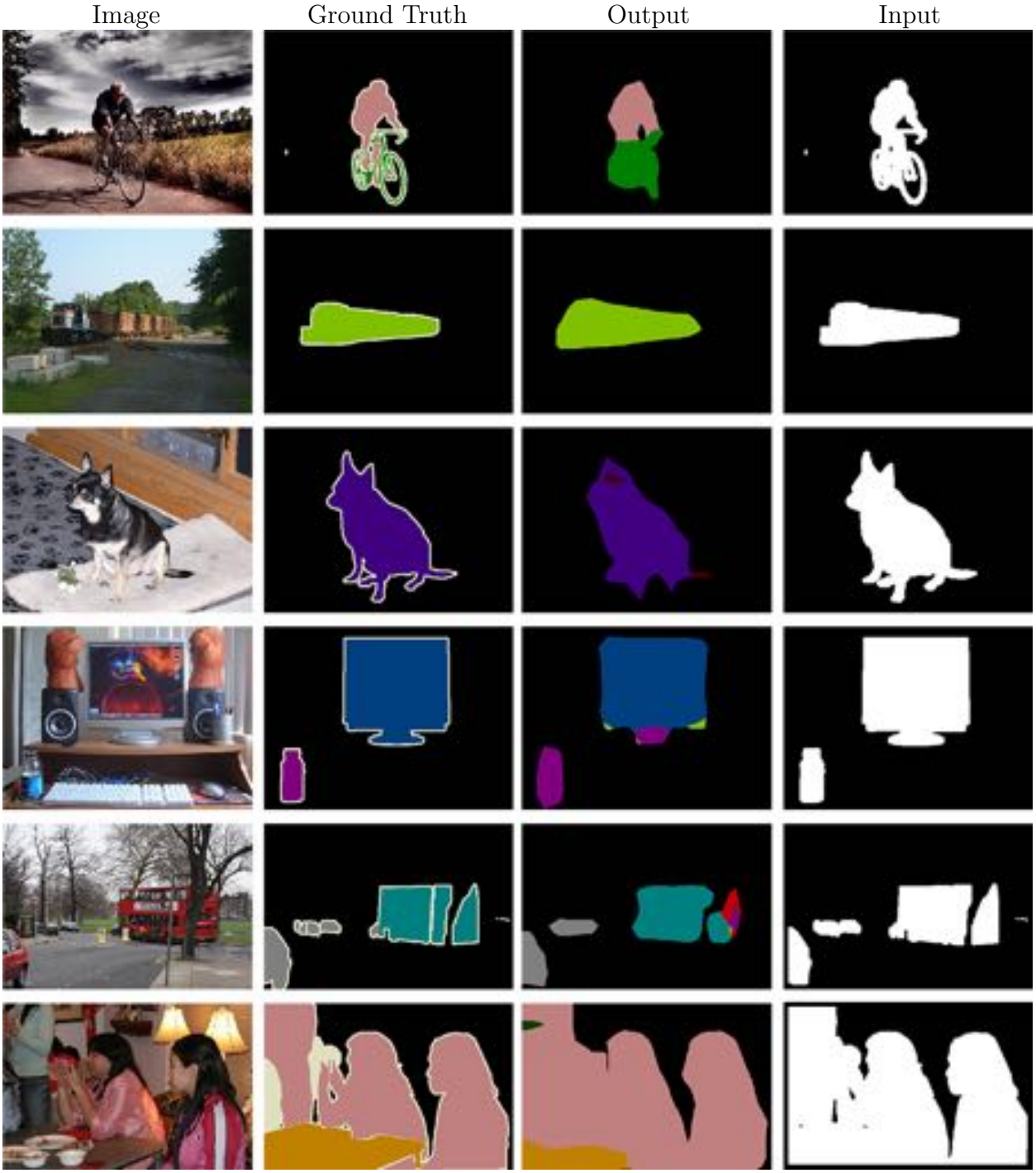


Figure 2.7: FCNs learn to recognize by shape when deprived of other input detail. From left to right: regular image (not seen by network), ground truth, output, mask input.

geometric coefficients, we have

$$g_t = -\eta \sum_{s=0}^{\infty} \sum_{i=0}^{k-1} p^s \nabla_\theta \ell(x_{k(t-s)+i}; \theta_{t-s}).$$

In other words, each example is included in the sum with coefficient $p^{\lfloor j/k \rfloor}$, where the index j orders the examples from most recently considered to least recently considered. Approximating this expression by dropping the floor, we see that learning with momentum p and batch size k appears to be similar to learning with momentum p' and batch size k' if $p^{(1/k)} = p'^{(1/k')}$. Note that this is not an exact equivalence: a smaller batch size results in more frequent weight updates, and may make more learning progress for the same number of gradient computations. For typical FCN values of momentum 0.9 and a batch size of 20 images, an approximately equivalent training regime uses momentum $0.9^{(1/20)} \approx 0.99$ and a batch size of one, resulting in online learning. In practice, we find that online learning works well and yields better FCN models in less wall clock time.

2.5.3 Oracle Accuracy for Intersection-over-Union

FCNs achieve good performance on the mean IU segmentation metric even with spatially coarse semantic prediction. To better understand this metric and the limits of this approach with respect to it, we compute approximate upper bounds on performance with prediction at various resolutions. We do this by downsampling ground truth images and then upsampling back to simulate the best results obtainable with a particular downsampling factor. The following table gives the mean IU on a subset[5] of PASCAL 2011 val for various downsampling factors.

factor	mean IU
128	50.9
64	73.3
32	86.1
16	92.8
8	96.4
4	98.5

Pixel-perfect prediction is clearly not necessary to achieve mean IU well above state-of-the-art, and, conversely, mean IU is a not a good measure of fine localization accuracy. The gaps between oracle and state-of-the-art accuracy at every stride suggest that recognition and not resolution is the bottleneck for this metric.

2.6 Conclusion

Fully convolutional networks are a rich class of models that address many pixelwise tasks. FCNs for semantic segmentation dramatically improve accuracy by transferring pre-trained classifier weights, fusing different layer representations, and learning end-to-end on whole images. End-to-end, pixel-to-pixel operation simultaneously simplifies and speeds up learning and inference. All code for this paper is open source in Caffe, and all models are freely available in the Caffe Model Zoo. Further works have demonstrated the generality of fully convolutional networks for a variety of image-to-image tasks.

Chapter 3

Adaptive Inference for Efficient Video Processing

Semantic segmentation is a central visual recognition task. End-to-end convolutional network approaches have made progress on the accuracy and execution time of still-image semantic segmentation, but video semantic segmentation has received less attention. Potential applications include UAV navigation, autonomous driving, archival footage recognition, and wearable computing. The computational demands of video processing are a challenge to the simple application of image methods on every frame, while the temporal continuity of video offers an opportunity to reduce this computation.

Fully convolutional networks (FCNs) [SLD17; EF15; Fis+15] have been shown to obtain remarkable results, but the execution time of repeated per-frame processing limits application to video. Adapting these networks to make use of the temporal continuity of video reduces inference computation while suffering minimal loss in recognition accuracy. The temporal rate of change of features, or feature "velocity", across frames varies from layer to layer. In particular, deeper layers in the feature hierarchy change more slowly than shallower layers over video sequences. We propose that network execution can be viewed as an aspect of architecture and define the "clockwork" FCN (c.f. clockwork recurrent networks [Kou+14]). Combining these two insights, we group the layers of the network into stages, and set separate update rates for these levels of representation. The execution of a stage on a given frame is determined by either a fixed clock rate ("fixed-rate") or data-driven ("adaptive"). The prediction for the current frame is then the fusion (via the skip layer architecture of the FCN) of these computations on multiple frames, thus exploiting the lower resolution and slower rate-of-change of deeper layers to share information across frames.

We demonstrate the efficacy of the architecture for both fixed and adaptive schedules. We show results on multiple datasets for a pipelining schedule designed to reduce latency for real-time recognition as well as a fixed-rate schedule designed to reduce computation and hence time and power. Next we learn the clock-rate adaptively from the data, and demonstrate computational savings when little motion occurs in the video without sacrificing recognition accuracy during dynamic scenes. We verify our approach on synthetic frame sequences made

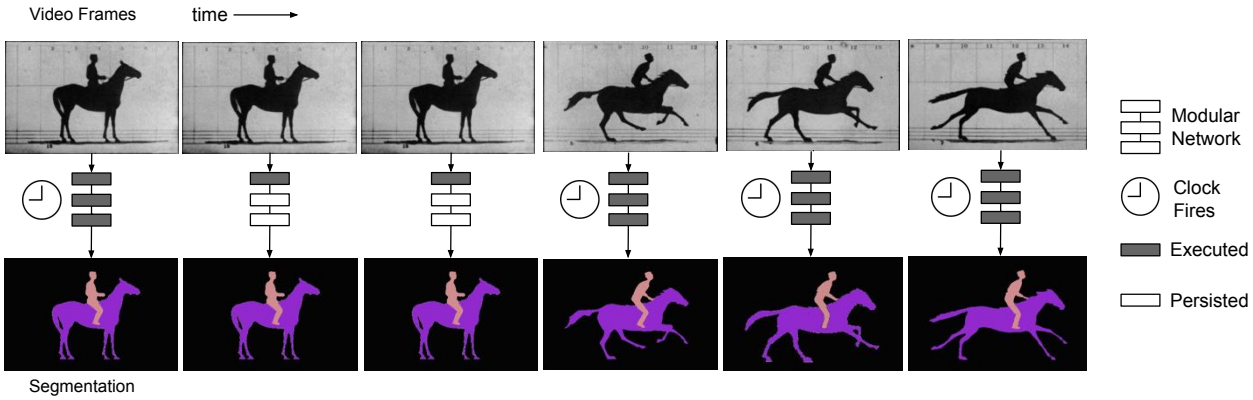


Figure 3.1: Our adaptive clockwork method illustrated with the famous *The Horse in Motion* [Muy82], captured by Eadweard Muybridge in 1878 at the Palo Alto racetrack. The clock controls network execution: past the first stage, computation is scheduled only at the time points indicated by the clock symbol. During static scenes cached representations persist, while during dynamic scenes new computations are scheduled and output is combined with cached representations.

from PASCAL VOC [Eve+10] and evaluate on videos from the NYUDv2 [Sil+12], YouTube-Objects [Pre+12], and Cityscapes [Cor+16] datasets.

3.1 Related Work

We extend fully convolutional networks for image semantic segmentation to video semantic segmentation. Convnets have been applied to video to learn spatiotemporal representations for classification and detection but rarely for dense pixelwise, frame-by-frame inference. Practicality requires network acceleration, but generic techniques do not exploit the structure of video. There is a large body of work on video segmentation, but the focus has not been on *semantic* segmentation, nor are methods computationally feasible beyond short video shots.

Fully Convolutional Networks A fully convolutional network (FCN) is a model designed for pixelwise prediction [SLD17]. Every layer in an FCN computes a local operation, such as convolution or pooling, on relative spatial coordinates. This locality makes the network capable of handling inputs of any size while producing output of corresponding dimensions. Efficiency is preserved by computing single, dense forward inference and backward learning passes. Current classification architectures – AlexNet [KSH12], GoogLeNet [Sze+15], and VGG [SZ15] – can be cast into corresponding fully convolutional forms. These networks are learned end-to-end, are fast at inference and learning time, and can be generalized with respect to different image-to-image tasks. FCNs yield state-of-the-art results for semantic segmentation [SLD17], boundary prediction [XT15], and monocular depth estimation [EF15]. While these tasks process each image in isolation, FCNs extend to video. As

more and more visual data is captured as video, the baseline efficiency of fully convolutional computation will not suffice.

Video Networks and Frame Selection Time can be incorporated into a network by spatiotemporal filtering or recurrence. Spatiotemporal filtering, i.e. 3D convolution, can capture motion for activity recognition [Ji+13; Kar+14]. For video classification, networks can integrate over time by early, late, or slow fusion of frame features [Kar+14]. Recurrence can capture long-term dynamics and propagate state across time, as in the popular long short-term memory (LSTM) [HS97]. Joint convolutional-recurrent networks filter within frames and recur across frames: the long-term recurrent convolutional network [Don+15] fuses frame features by LSTM for activity recognition and captioning. Frame selection reduces computation by focusing computational resources on important frames identified by the model: space-time interest points [Lap05] are video keypoints engineered for sparsity, and a whole frame selection and recognition policy can be learned end-to-end for activity detection [Yeu+16]. These video recognition approaches do not address frame-by-frame, pixelwise output. For optical flow, an intrinsically temporal task, a cross-frame FCN is state-of-the-art among fast methods [Fis+15].

Network Acceleration Although FCNs are fast, video demands computation that is faster still, particularly for real-time inference. The spatially dense operation of the FCN amortizes the computation of overlapping receptive fields common to contemporary architectures. However, the standard FCN does nothing to temporally amortize the computation of sequential inputs. Computational concerns can drive architectural choices. For instance, GoogLeNet requires less computation and memory than VGG, although its segmentation accuracy is worse [SLD17]. Careful but time-consuming model search can improve networks within a fixed computational budget [HS15]. Methods to reduce computation and memory include reduced precision by weight quantization [VSM11], low-rank approximations with clustering, [Den+14], low-rank approximations with end-to-end tuning [JVZ14], and kernel approximation methods like the fast food transformation [Yan+15]. None of these generic acceleration techniques harness the frame-to-frame structure of video. The proposed clockwork speed-up is orthogonal and compounds any reductions in absolute inference time. Our clockwork insight holds for all layered architectures whatever the speed/quality operating point chosen.

Semantic Segmentation Much work has been done to address the problem of segmentation in video. However, the focus has not been on semantic segmentation. Instead research has addressed spatio-temporal "supervoxels" [Gru+10; XC12], unsupervised and motion-driven object segmentation [SM98; PF13; Fra+15], or weakly supervising the segmentation of tagged videos [Har+12; Tan+13; Liu+14]. These methods are not suitable for real-time or the complex multi-class, multi-object scenes encountered in semantic segmentation settings. Fast Object Segmentation in Unconstrained Videos [PF13] infers only figure-ground segmentation at 0.5s/frame with offline computed optical flow and superpixels. Although its proposals have high recall, even when perfectly parallelized [Fra+15] this method takes $>$ 15s/frame and a separate recognition step is needed for semantic segmentation. In contrast the standard FCN computes a full semantic segmentation in 0.1s/frame.

3.2 Fast Frames and Slow Semantics

Our approach is inspired by observing the time course of learned, hierarchical features over video sequences. Drawing on the local-to-global idea of skip connections for fusing global, deep layers with local, shallow layers, we reason that the semantic representation of deep layers is relevant across frames whereas the shallow layers vary with more local, volatile details. Persisting these features over time can be seen as a temporal skip connection.

Measuring the relative difference of features across frames confirms the temporal coherence of deeper layers. Consider a given score layer (a linear predictor of pixel class from features), ℓ, with outputs $S_\ell \in [K \times H \times W]$, where K is the number of categories and H, W is the output dimensions for layer ℓ. We can compute the difference at time t with a score map distance function d_{sm}, chosen to be the hamming distance of one hot encodings.

$$d_{\text{sm}}(S_\ell^t, S_\ell^{t-1}) = d_{\text{hamming}}(\phi(S_\ell^t), \phi(S_\ell^{t-1}))$$

Table 3.1 reports the average of these temporal differences for the score layers, as computed over all videos in the YouTube-Objects dataset [Pre+12]. It is perhaps unsurprising that the deepest score layer changes an order of magnitude less than the shallower layers on average. We therefore hypothesize that caching deeper layer scores from past frames can inform the inference of the current frame with relatively little reduction in accuracy.

The slower rate of change of deep layers can be attributed to architectural and learned invariances. More pooling affords more robustness to translation and noise, and learned features may be tuned to the supervised classes instead of general appearance.

score layer	temporal difference	depth	semantic accuracy
`pixels`	.26 ± .18	0	-
`pool3`	.11 ± .06	9	9.6%
`pool4`	.11 ± .06	13	20.7%
`fc7`	.02 ± .02	19	65.5%

Table 3.1: The average temporal difference over all YouTube-Objects videos of the respective pixelwise class score outputs from a spectrum of network layers. The deeper layers are more stable across frames – that is, we observe supervised convnet features to be "slow" features [WS02]. The temporal difference is measured as the proportion of label changes in the output. The layer depth counts the distance from the input in the number of parametric and non-linear layers. Semantic accuracy is the intersection-over-union metric on PASCAL VOC of our frame processing network fine-tuned for separate output predictions (Section 3.4).

While deeper layers are more stable than shallower layers, for videos with enough motion the score maps throughout the network may change substantially. For example, in Figure 3.2

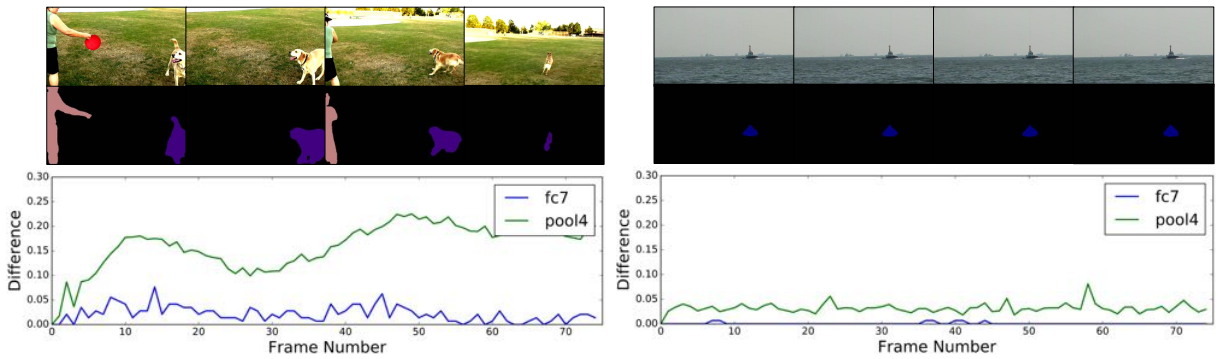


Figure 3.2: The proportional difference between adjacent frames of semantic predictions from a mid-level layer (`pool4`, green) and the deepest layer (`fc7`, blue) are shown for the first 75 frames of two videos. We see that for a video with lots of motion (left) the difference values are large while for a relatively static video (right) the difference values are small. In both cases, the differences of the deeper `fc7` are smaller than the differences of the shallower `pool4`. The "velocity" of deep features is slow relative to shallow features and most of all the input. At the same time, the differences between shallow and deep layers are dependent since the features are compositional; this motivates our adaptive clock updates in Section 3.3.3

we show the differences for the first 75 frames of a video with large motion (left) and with small motion (right). We would like our network to adaptively update only when the deepest, most semantic layer (`fc7`) score map is likely to change. We notice that though the intermediate layer (`pool4`) difference is always larger than the deepest layer difference for any given frame, the `pool4` differences are much larger for the video with large motion than for the video with relatively small motion. This observation forms the motivation for using the intermediate differences as an indicator to determine the firing of an adaptive clock.

3.3 A Clockwork Network

We adapt the fully convolutional network (FCN) approach for image-to-image mapping [SLD17] to video frame processing. While it is straightforward to perform inference with a still-image segmentation network on every video frame, this naïve computation is inefficient. Furthermore, disregarding the sequential nature of the input not only sacrifices efficiency but discards potential temporal recognition cues. The temporal coherence of video suggests the persistence of visual features from prior frames to inform inference on the current frame. To this end we define the clockwork FCN, inspired by the clockwork recurrent network [Kou+14], to carry temporal information across frames. A generalized notion of clockwork relates both of these networks.

We consider both throughput and latency in the execution of deep networks across video sequences. The inference time of the regular FCN-8s at $\sim$ 100ms per frame of size 500×500

on a standard GPU can be too slow for video. We first define fixed clocks then extend to adaptive and potentially learned clockwork to drive network processing. Whatever the

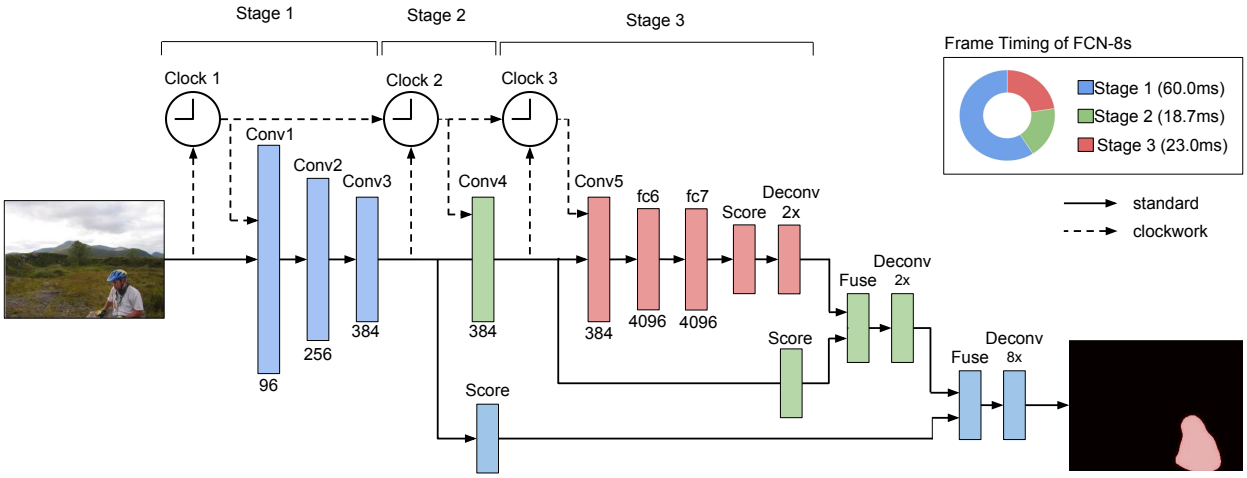


Figure 3.3: The clockwork FCN with its stages and corresponding clocks.

task, any video network can be accelerated by our clockwork technique. A schematic of our clockwork FCN is shown in Figure 3.3.

There are several choice points in defining a clockwork architecture. We define a novel, generalized clockwork framework, which can purposely schedule deeper layers more slowly than shallower layers. We form our modules by grouping the layers of a convnet to span the feature hierarchy. Our networks persists both state and output across time steps. The clockwork recurrent network of [Kou+14], designed for long-term dependency modeling of time series, is an instance of our more general scheme for clockwork computation. The differences in architecture and outputs over time between clockwork recurrence and our clockwork are shown in Figure 3.4.

While different, these nets can be expressed by generalized clockwork equations

$$y_H^{(t)} = f_T\left(C_H^{(t)} \odot f_H(y_H^{(t-1)}) + C_I^{(t)} \odot f_I(x^{(t)})\right) \tag{3.1}$$

$$y_O^{(t)} = f_O\left(C_O^{(t)} \odot f_H(y_H^{(t)})\right) \tag{3.2}$$

with the state update defined by Equation 3.1 and the output defined by Equation 3.2. The data $x^{(t)}$, hidden state $y_H^{(t)}$ output $y_O^{(t)}$ vary with time t. The functions f_I, f_H, f_O, f_T define input, hidden state, output, and transition operations respectively and are fixed across time. The input, hidden, and output clocks $C_I^{(t)}, C_H^{(t)}, C_O^{(t)}$ modulate network operations by the elementwise product $\odot$ with the corresponding function evaluations. We recover the standard recurrent network (SRN), clockwork recurrent network (clock RN), and our network (clock FCN) in this family of equations. The settings of functions and clocks are collected in Table 3.2.

network	f_I	f_H	f_O	f_T	C_I	C_H	C_O
SRN	W_I	W_H	TanH	TanH	$\mathbb{1}$	$\mathbb{1}$	$\mathbb{1}$
clock RN	W_I	W_H	TanH	TanH	C	C	C
clock FCN	$\circ$	I	ReLU	I	C	$\overline{C}$	$\mathbb{1}$

Table 3.2: The standard recurrent network (SRN), clockwork recurrent network (clock RN), and our network (clock FCN) in generalized clockwork form. The recurrent networks have learned hidden weights W_H and non-linear transition functions f_T, while clock FCN persists state by the identity I. Both recurrent modules are flat with linear input weights W_I, while clock FCN modules have hierarchical features by layer composition $\circ$. The SRN has trivial constant, all-ones $\mathbb{1}$ clocks. The clock RN has a shared input, hidden, and output clock with exponential rates. Our clock FCN has alternating input and hidden clocks $C, \overline{C}$ to compute or cache and has a constant, all-ones $\mathbb{1}$ output clock to fuse output on every frame.

Inspired by the clockwork RN, we investigate persisting features and scheduling layers to process video with a semantic segmentation convnet. Recalling the lessened semantic rate of deeper layers observed in Section 3.2, the skip layers in FCNs originally included to preserve resolution by fusing outputs are repurposed for this staged computation. We cache features and outputs over time at each step to harness the continuity of video. In contrast, the clockwork RN persists state but output is only made according to the clock, and each clockwork RN module is connected to itself and all slower modules across time whereas a module in our network is only connected to itself across time.

3.3.1 Architecture as Execution Schedule

Clockwork architectures partition a network into modules or stages that are executed according to different schedules. In the standard view, the execution of an architecture is an all-or-nothing operation that follows from the definition of the network. Relaxing the strict identification of architecture and execution instead opens up a range of potential schedules. These schedules can be encompassed by the introduction of one first-class architectural element: the clock.

A clock defines a dynamic cut in the computation graph of a network. As clocks mask state in the representation, as detailed in Equations 3.1 and 3.2, clocks likewise mask execution in the computation. When a clock is on, its edges are intact and execution traverses to the next nodes/modules. When a clock is off, its edges are cut and execution is blocked. Alternatives such as computing the next stage or caching a past stage can be scheduled by a paired clock C and counter-clock $\overline{C}$ with complementary sets of edges. Any layer (or composition of layers) with binary output can serve as a clock. As a layer, a clock can be fixed or learned. For instance, the following are simple clocks of the form $f(x, t)$ for features x and time t:

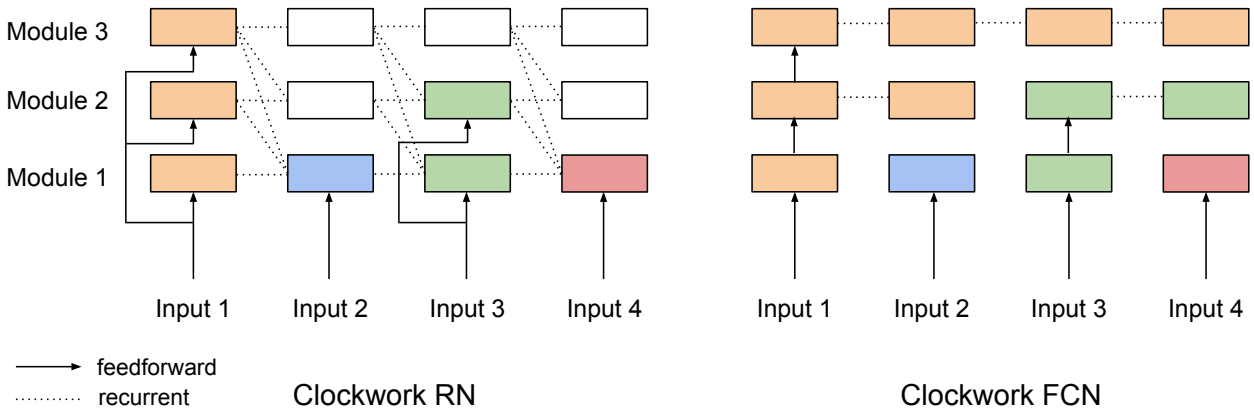


Figure 3.4: A comparison of the layer connectivity and time course of outputs in the clockwork recurrent network [Kou+14] and in our clockwork FCN. Module color marks the time step of evaluation, and blank modules are disconnected from the network output. The clock RN is flat with respect to the input while our network has a hierarchical feature representation. Each clock RN module is temporally connected to itself and slower modules while in our network each module is only temporally connected to itself. Features persist over time in both architectures, but in our architecture they contribute to the network output at each step.

- 1 to always execute
- $t \equiv 0 \pmod 2$ to execute every other time
- $\|x_t - x_{t-1}\| > \theta$ to execute for a difference threshold

3.3.2 Pipelining and Fixed-Rate Clockwork

Having incorporated scheduling into the network with clocks, we can optimize the schedule for various tasks by altering the clocks.

Pipelining To reduce latency for real-time recognition we pipeline the computation of sequential frames analogously to instruction pipelining in processors. We instantiate a three-stage pipeline, in which stage 1 reflects frame i, stage 2 frame $i-1$, and stage 3 frame $i-2$. The total time to process the frame is the time of the longest stage, stage 1 in our pipeline, plus the time for interpolating and fusing outputs. Our 3-stage pipeline FCN reduces latency by 59%. A 2-stage variation further balances latency and accuracy.

Fixed-Rate To reduce overall computation we limit the execution rates of stages and persist features across frames for skipped stages. Given the learned invariance and slow semantics of deep layers observed in Section 3.2, the deeper layers can be executed at a lower rate to save computation while other stages update. These clock rates are free parameters in

the schedule for exchanging inference speed and accuracy. We again divide the network into three stages, and compare rates for the stages. The exponential clockwork schedule is the natural choice of halving the rate at each stage for more efficiency. The alternating clockwork schedule consolidates the earlier stages to execute these on every frame and executes the last stage on every other frame for more accuracy. These different sets of rates cover part of the accuracy/efficiency spectrum.

The current stages are divided into the original score paths of the FCN-8s architecture, but they need not be. One could prioritize latency, spatial refinement, or certain output classes by rebalancing the computation. It is possible to partially compute a span of layers and defer their full execution to a following stage; this can be accomplished by sparse evaluation through dynamic striding and dilation [YK16]. In principle the stage progression can be decided online in lieu of fixing a schedule for all inference. We turn to adaptive clockwork for deciding execution.

3.3.3 Adaptive Clockwork

All of the clocks considered thus far have been fixed functions of time but not the data. Setting these clocks gives rise to many schedules that can be tuned to a given task or video, but this introduces a tedious dimension of model search. Much of the video captured in the wild is static and dynamic in turn with a variable amount of motion and semantic progression at any given time. Choosing many stages or a slow clock rate may reduce computation, but will likewise result in a steep decline in accuracy for dynamic scenes. Conversely, faster update rates or fewer stages may capture transitory details but will needlessly compute and re-compute stable scenes. Adaptive clocks fire based on the input and network state, resulting in a responsive schedule that varies with the dynamism of the scene. The clock can fire according to any function of the input and network state. A difference clock can fire on the temporal difference of a feature across frames. A confidence clock can fire on peaks in the score map for a single frame. This approach extends inference from a pre-determined architecture to a set of architectures to choose from for each frame, relying on the full FCN for high accuracy in dynamic scenes while taking advantage of cached representations in more static scenes.

$$\text{threshold clock } \|x_t - x_{t-1}\| > \theta \qquad\qquad \text{learned clock } f_\theta(x_t, x_{t-1})$$

The simplest adaptive clock is a threshold, but adaptive clocks could likewise be learned (for example as a temporal convolution across frames). The threshold can be optimized for a specific tradeoff along the accuracy/efficiency curve. Given the hierarchical dependencies of layers and the relative stability of deep features observed in Section 3.2, we threshold differences at a shallower stage for adaptive scheduling of deeper stages. The sensitivity of the adaptive clock can even be set on unannotated video by thresholding the proportional temporal difference of output labels as in Table 3.1. Refer to Section 3.4.3 for the results of threshold-adaptive clockwork with regard to clock rate and accuracy.

3.4 Results

Our base network is FCN-8s, the fully convolutional network of [SLD17]. The architecture is adapted from the VGG16 architecture [SZ15] and fine-tuned from ILSVRC pre-training. The net is trained with batch size one, high momentum, and all skip layers at once.

In our experiments we report two common metrics for semantic segmentation that measure the region intersection over union (IU):

- mean IU: $(1/n_{\text{cl}}) \sum_i n_{ii} / \left(t_i + \sum_j n_{ji} - n_{ii}\right)$
- frequency weighted IU: $\left(\sum_k t_k\right)^{-1} \sum_i t_i n_{ii} / \left(t_i + \sum_j n_{ji} - n_{ii}\right)$

for n_{ij} the number of pixels of class i predicted to belong to class j, where there are n_{cl} different classes, and for $t_i = \sum_j n_{ij}$ the total number of pixels of class i.
We evaluate our clockwork FCN on four video semantic segmentation datasets.

Synthetic sequences of translated scenes We first validate our method by evaluating on synthetic videos of moving crops of PASCAL VOC images [Eve+10] in order to score on a ground truth annotation at every frame. For source data, we select the 736 image subset of the PASCAL VOC 2011 segmentation validation set used for FCN-8s validation in [SLD17]. Video frames are generated by sliding a crop window across the image by a predetermined number of pixels, and generated translations are vertical or horizontal according to the portrait or landscape aspect of the chosen image. Each synthetic video is six frames long. For each seed image, a "fast" and "slow" video is made with 32 pixel and 16 pixel frame-to-frame displacements respectively.

NYU-RGB clips The NYUDv2 dataset [Sil+12] collects short RGB-D clips and includes a segmentation benchmark with high-quality but temporally sparse pixel annotations (every tenth video frame is labeled). We run on video from the "raw" clips subsampled 10X and evaluate on every labeled frame. We consider RGB input alone as the depth frames of the full clips are noisy and uncurated. Our pipelined and fixed-rate clockwork FCNs are run on the entire clips and accuracy is reported for those frames included in the segmentation test set.

Youtube-Objects The Youtube-Objects dataset [Pre+12] provides videos collected from Youtube that contain objects from ten PASCAL classes. We restrict our attention to a subset of the videos that have pixelwise annotations by [JG14] as the original annotations include only initial frame bounding boxes. This subset was drawn from all object classes, and contains 10,167 frames from 126 shots, for which every 10th frame is human-annotated. We run on only annotated frames, effectively 10X subsampling the video. We directly apply our networks derived from PASCAL VOC supervision and do not fine-tune to the video annotations.

Cityscapes The Cityscapes dataset [Cor+16] collects frames from video recorded at 17hz by a car-mounted camera while driving through German cities. While annotations are

temporally sparse, the preceding and following input frames are provided. Our network is learned on the `train` split and then all schedules are evaluated on `val`.

3.4.1 Pipelining Reduces Latency

16 pixel shift	Time (% of full)	Mean IU	fwIU	Mean IU-bdry	fwIU-bdry
3-Stage Baseline	59%	9.2	52.6	6.1	9.4
3-Stage Pipeline	59%	56.0	76.5	44.6	42.9
2-Stage Baseline	77%	22.5	64.7	16.6	21.9
2-Stage Pipeline	77%	**6**3.3	**8**1.7	**5**2.3	**5**1.0
Frame Oracle	100%	65.9	83.6	57.0	56.3
32 pixel shift	Time (% of full)	Mean IU	fwIU	Mean IU-bdry	fwIU-bdry
3-Stage Baseline	59%	9.2	52.6	6.0	9.4
3-Stage Pipeline	59%	45.5	67.4	37.7	36.0
2-Stage Baseline	77%	22.4	62.8	16.2	21.7
2-Stage Pipeline	77%	**5**7.8	**7**6.6	**4**6.6	**4**5.1
Frame Oracle	100%	65.6	82.6	55.8	55.3

Table 3.3: Pipelined segmentation of translated PASCAL sequences. Synthesized video of translating PASCAL scenes allows for assessment of the pipeline at every frame. The pipelined FCN segments with higher accuracy in the same time envelope as the every-other-frame evaluation of the full FCN. Metrics are computed on the standard masks and a 10-pixel band at boundaries.

Pipelined execution schedules reduce latency by producing an output each time the first stage is computed. Later stages are persisted from previous frames and their outputs are fused with the output of the first stage computed on the current frame. The number of stages is determined by the number of clocks. We consider a full **3-stage pipeline** and a condensed **2-stage pipeline** where the stages are defined by the modules in Figure 3.3. In the pipelined schedule, all clock rates are set to 1, but clocks fire simultaneously to update every stage in parallel. This is made possible by asynchrony in stage state, so that a later stage is independent of the current frame but not past frames.

To assess our pipelined accuracy and speed, we compare to reference methods that bound both recognition and time. A frame oracle evaluates the full FCN on every frame to give the best achievable accuracy for the network independent of timing. As latency baselines for our pipelines, we truncate the FCN to end at the given stage. Both of our staged, pipelined schedules execute at lower latency than the oracle with better accuracy for fixed latency than the baselines. We verify these results on synthetic PASCAL sequences as reported in Table 3.3. Results on PASCAL, NYUD, and YouTube are reported in Table 3.4.

Our pipeline scheduled networks reduce latency with minimal accuracy loss relative to the standard FCN run on each frame without time restriction. These quantitative results

demonstrate that the deeper layer representations from previous frames contain useful information that can be effectively combined with low-level predictions for the current frame.

		NYUD		Youtube		Pascal Shift 16	
Schedule	Time (% of full)	Mean IU	fwIU	Mean IU	fwIU	Mean IU	fwIU
3-Stage Baseline	59%	8.1	22.2	12.2	74.2	9.2	54.7
3-Stage Pipeline	59%	25.1	38.0	58.1	87.0	56.0	76.5
2-Stage Baseline	77%	16.5	32.1	21.5	7.8	22.5	64.7
2-Stage Pipeline	77%	**2**6.4	**3**9.5	**6**4.0	**8**9.2	**6**3.3	**8**1.7
Frame Oracle	100%	31.1	45.5	70.0	91.5	65.9	83.6

Table 3.4: Pipelined execution of semantic segmentation on three different datasets. Inference approaches include pipelines of different lengths and a full FCN frame oracle. We also show baselines with comparable latency to the pipeline architectures. Our pipelined network offers the best accuracy of computationally comparable approaches running near frame rate. The loss in accuracy relative to the frame oracle is less than the relative speed-up.

We show a qualitative result for our pipelined FCN on a sequence from the YouTube-Objects dataset [Pre+12]. Figure 3.5 shows one example where our pipeline FCN is particularly useful. Our network quickly detects the occlusion of the car while the baseline lags and does not immediately recognize the occlusion or reappearance.

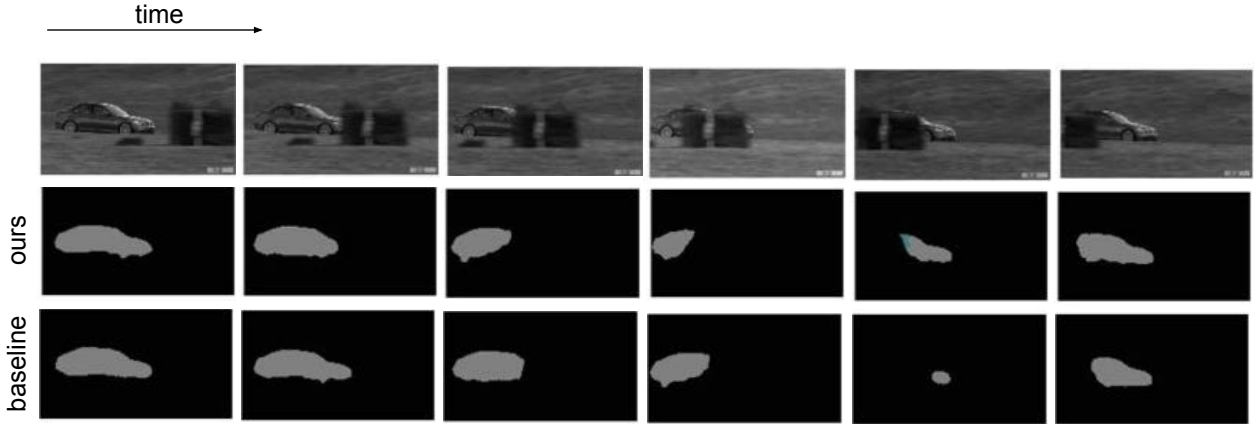


Figure 3.5: Pipelined vs. standard FCN on YouTube video. Our method is able to detect the occlusion of the car as it is happening unlike the lagging baseline computed on every other frame.

3.4.2 Fixed-Rate Clockwork Raises Throughput

Fixed-rate clock schedules reduce overall computation relative to full, every frame evaluation by assigning different update rates to each stage such that later stages are executed less often. Rates can be set aggressively low for extreme efficiency or conservatively high to maintain accuracy while sparing computation. The **exponential clockwork** schedule executes the first stage on every frame then updates following stages exponentially less often by halving with each stage. The **alternating clockwork** schedule combines stages 2 and 3, executes the first stage on every frame, then schedules the following combined stage every other frame.

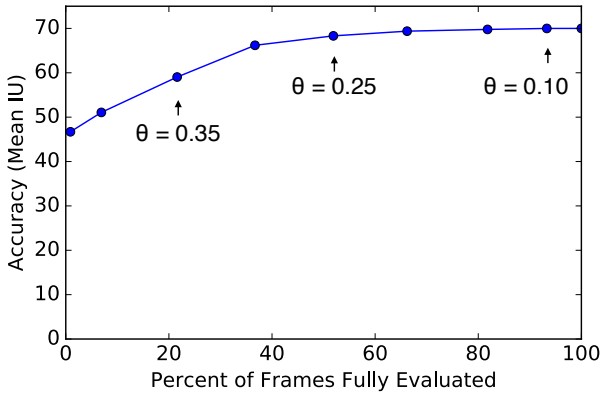
A frame oracle that evaluates the full FCN on *every* frame is the reference model for accuracy. Evaluating the full FCN on *every other* frame is the reference model for computation. Due to the distribution of execution time over stages, this is faster than either clockwork schedule, though clockwork offers higher accuracy. Alternating clockwork achieves higher accuracy than the every other frame reference. See Table 3.5.

16 pixel shift	Clock Rates	Mean IU	fwIU	Mean IU-bdry	fwIU-bdry
Skip Frame Baseline	(2,2,2)	63.0	81.5	60.2	52.2
Exponential	(1,2,4)	61.4	80.4	50.5	49.1
Alternating	(1,1,2)	**6**4.7	**8**2.6	**5**4.8	**5**3.7
Frame Oracle	(1,1,1)	65.9	83.6	57.0	56.3
32 pixel shift	**Clock Rates**	**Mean IU**	**fwIU**	**Mean IU-bdry**	**fwIU-bdry**
Skip Frame Baseline	(2,2,2)	59.5	77.9	49.4	48.2
Exponential	(1,2,4)	55.5	74.7	46.3	44.8
Alternating	(1,1,2)	**6**1.9	**7**9.6	**5**1.7	**5**0.6
Frame Oracle	(1,1,1)	65.6	82.6	55.8	55.3

Table 3.5: Fixed-rate segmentation of translated PASCAL sequences. We evaluate the network on synthesized video of translating PASCAL scenes to assess the effect of persisting layer features across frames. Metrics are computed on the standard masks and a 10-pixel band at boundaries.

	NYUD		Youtube		Cityscapes	
Schedule	Mean IU	fwIU	Mean IU	fwIU	Mean IU	fwIU
Skip Frame Baseline	27.7	41.3	65.6	89.7	62.1	87.4
Alternating	28.5	42.4	67.0	90.3	**6**4.4	**8**8.6
Adaptive	**2**8.9	**4**3.3	**6**8.5	**9**1.0	61.8	87.6
Frame Oracle	31.1	45.5	70.0	91.4	65.9	83.6

Table 3.6: Fixed-rate and adaptive clockwork FCN evaluation. We score our network on three datasets with an alternating schedule that executes the later stage every other frame and an adaptive schedule that executes according to a frame-by-frame threshold on the difference in output. The adaptative threshold is tuned to execute the full network on 50% of frames to equalize computation between the alternating and adaptive schedules.



Method	% Full Frames	Mean IU
Adaptive [$\theta = 0.10$]	93%	70.0
Adaptive [$\theta = 0.25$]	52%	68.3
Adaptive [$\theta = 0.35$]	21%	59.0
Frame Oracle	100%	70.0

Figure 3.6: Adaptive Clockwork performance across the Youtube-Objects dataset. We examine various adaptive difference thresholds θ and plot accuracy (mean IU) against the percentage of frames that the adaptive clock chooses to fully compute. A few corresponding thresholds are indicated.

Exponential clockwork shows degraded accuracy yet takes 1.5× the computation of evaluation on every other frame, so we discard this fixed schedule in favor of adaptive clockwork. Although exponential rates suffice for the time series modeled by the clockwork recurrent network [Kou+14], these rates deliver unsatisfactory results for the task of video semantic segmentation. See Table 3.6 for alternating clockwork results on NYUD, YouTube-Objects, and Cityscapes.

3.4.3 Adaptive Clockwork is Efficient and Accurate

The best clock schedule can be data-dependent and unknown before segmenting a video. Therefore, we next evaluate our adaptive clock rate as described in Section 3.3.3. In this case the adaptive clock only fully processes a frame if the relative difference in `pool4` score is larger than some threshold θ. This threshold may be interpreted as the the fraction of the score map that must switch labels before the clock updates the upper layers of the network. See Table 3.6 for adaptive clockwork results on NYUD, YouTube-Objects, and Cityscapes.

We experiment with varying thresholds on the Youtube-Objects dataset to measure accuracy and efficiency. We pick thresholds in $\theta = [0.1, 0.5]$ as well as $\theta = 0.0$ for unconditionally updating on every frame.

In Figure 3.6 (left) we report mean IU accuracy as a function of our adaptive clock firing rate; that is, the percentage of frames the clock decides to fully process in the network. The thresholds which correspond to a few points on this curve are indicated with mean IU (right). Notice that our adaptive clockwork is able to fully process only 52% of the frames while suffering a minimal loss in mean IU ($\theta = 0.25$). This indicates that our adaptive clockwork is capable of discovering semantically stationary scenes and saves significant computation by only updating when the output score map is predicted to change.

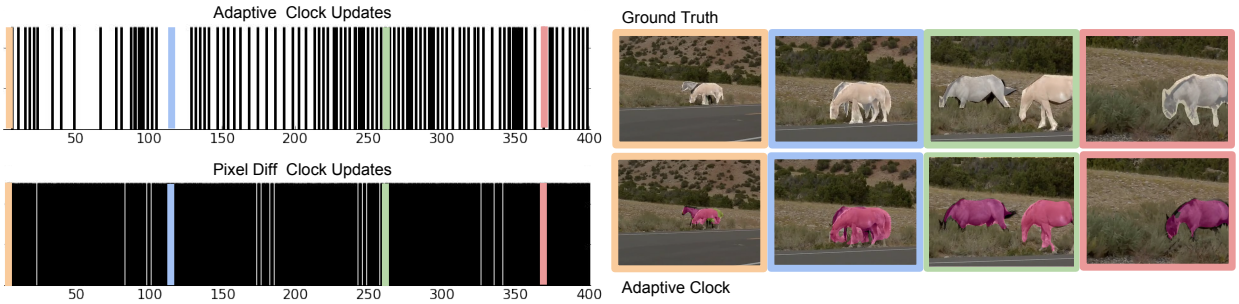


Figure 3.7: An illustrative example of our adaptive clockwork method on a video from Youtube-Objects. On the left, we compare clock updates over time (shown in black) of our adaptive clock as well as a clock based on pixel differences. Our adaptive clock updates the full network on only 26% of the frames, determined by the threshold $\theta = 0.25$ on the proportional output label change across frames, while scheduling updates based on pixel difference alone results in updating 90% of the frames. On the right we show output segmentations from the adaptive clockwork network as well as ground truth segments for select frames from dynamic parts of the scene (second and third frames shown) and relatively static periods (first and second frames shown).

For a closer inspection, we study one Youtube video in more depth in Figure 3.7. We first visualize the clock updates for our adaptive method (top left) and for a simple pixel difference baseline (bottom left), where black indicates the clock is on and the corresponding frame is fully computed. This video has significant change in certain sections (ex: at frame ~ 100 there is zoom and at ~ 350 there is motion) with long periods of relatively little motion (ex: frames $110 - 130$). While the pixel difference metric is susceptible to the changes in minor image statistics from frame to frame, resulting in very frequent updates, our method only updates during periods of semantic change and can cache deep features with minimal loss in segmentation accuracy: compare adaptive clock segmentations to ground truth (right).

3.5 Conclusion

Generalized clockwork architectures encompass many kinds of temporal networks, and incorporating execution into the architecture opens up many strategies for scheduling computation. We define a clockwork fully convolutional network for video semantic segmentation in this framework. Motivated by the stability of deep features across sequential frames, our network persists features across time in a temporal skip architecture. By exploring fixed and adaptive schedules, we are able to tune processing for latency, overall computation time, and recognition performance. With adaptive, data-driven clock rates the network is scheduled online to segment dynamic and static scenes alike while maintaining accuracy. In this way our adaptive clockwork network is a bridge between convnets and event-driven vision architectures. The clockwork perspective on temporal networks suggests further architectural

variations for spatiotemporal video processing.

Chapter 4

Non-local Inference from Local Supervision

Many tasks of scientific and practical interest require grouping pixels, such as cellular microscopy, medical imaging, and graphic design. Furthermore, a single image might need to be segmented in several ways, for instance to first segment all people, then focus on a single person, and finally pick out their face. Learning a particular type of segmentation, or even extending an existing model to a new task like a new semantic class, generally requires collecting and annotating a large amount of data and (re-)training a large model for many iterations. Interactive segmentation with a supervisor in-the-loop can cope with less supervision, but requires at least a little annotation for each image, entailing significant effort over image collections or videos. Faced with endless varieties of segmentation and countless images, yet only so much expertise and time, a segmentor should be able to learn from varying amounts of supervision and propagate that supervision to unlabeled pixels and images.

We frame these needs as the problem of *guided* segmentation: given supervision from few or many images and pixels, collect and propagate this supervision to segment any given images, and do so quickly and with generality across tasks. The amount of supervision may vary widely, from a lone annotated pixel, millions of pixels in a fully annotated image, or even more across a collection of images as in conventional supervised learning for segmentation. The number of classes to be segmented may also vary depending on the task, such as when segmenting categories like cats vs. dogs, or when segmenting instances to group individual people. Guided segmentation extends few-shot learning to the structured output setting, and the non-episodic accumulation of supervision as data is progressively annotated. Guided segmentation broadens the scope of interactive segmentation by integrating supervision across images and segmenting unannotated images.

As a first step towards solving this novel problem, we propose guided networks to extract *guidance*, a latent task representation, from variable amounts of supervision (see Figure 4.1). To do so we meta-learn how to extract and follow guidance by training episodically on tasks synthesized from a large, fully annotated dataset. Once trained, our model can quickly and cumulatively incorporate annotations to perform new tasks not seen during training. Guided

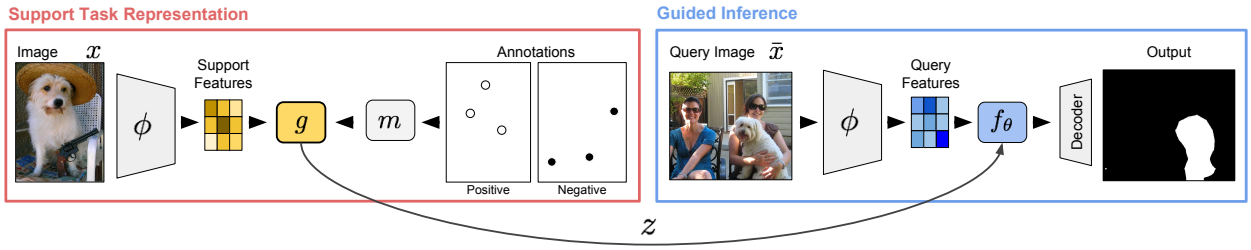


Figure 4.1: A guide g extracts a latent task representation z from an annotated image (red) for inference by $f_\theta(\bar{x}, z)$ on a different, unannotated image (blue).

networks reconcile static and interactive modes of inference: a guided model is both able to make predictions on its own, like a fully supervised model, and to incorporate expert supervision for defining new tasks or correcting errors, like an interactive model. Guidance, unlike static model parameters, does not require optimization to update: it can be quickly extended or corrected during inference. Unlike annotations, guidance is latent and low-dimensional: it can be collected and propagated across images and episodes for inference without the supervisor in-the-loop as needed by interactive models.

We evaluate our method on a variety of challenging segmentation problems in Section 5.4: interactive image segmentation, semantic segmentation, video object segmentation, and real-time interactive video segmentation, as shown in 4.2. We further perform novel exploratory experiments aimed at understanding the characteristics and limits of guidance. We compare guidance with standard supervised learning across the few-shot and many-shot extremes of support size to identify the boundary between few-shot and many-shot learning for segmentation. We demonstrate that in some cases, our model can generalize to guide tasks at a different level of granularity, such as meta-learning from instance supervision and then guiding semantic segmentation of categories.

4.1 Related Work

Guided segmentation extends few-shot learning to structured output models, statistically dependent data, and variable supervision in amount of annotation (shot) and numbers of classes (way). Guided segmentation spans different kinds of segmentation as special cases determined by the supervision that constitutes a task, such as a collection of category masks for semantic segmentation, sparse positive and negative pixels in an image for interactive segmentation, or a partial annotation of an object on the first frame of a clip for video object segmentation.

Few-shot learning Few-shot learning [FFFP06; LST15] holds the promise of data efficiency: in the extreme case, one-shot learning requires only a single annotation of a new concept. The present wave of methods [KZS15; San+16; Vin+16; WH16; Ber+16; FAL17; RL17; SSZ17] frame it as direct optimization for the few-shot setting: they synthesize episodes by sampling supports and queries, define a task loss, and learn a task model for

Figure 4.2: Guided segmentation groups different kinds of segmentation in one problem statement.

inference of the queries given the support supervision. While these works address a setting with a fixed, small number of examples and classes at meta-test time, we explore settings where the number of annotations and classes is flexible.

Our approach is most closely related to episodically optimized metric learning approaches. We design a novel, efficient segmentation architecture for metric learning, inspired by Siamese networks [CHL05; HCL06] and few-shot metric methods [KZS15; Vin+16; SSZ17] that learn a distance to retrieve support annotations for the query. In contrast to existing meta-learning schemes, we examine how a meta-learned model generalizes across task families with a nested structure, such as performing semantic segmentation after meta-learning on instance segmentation tasks.

Segmentation There are many kinds of segmentation, and many current directions for deep learning techniques [GG+17]. We take up semantic [Eve+10; LYT11], interactive [KWT88; BJ01], and semi-supervised video object segmentation [PT+17] as challenge problems for our unified view with guidance. See Fig. 4.2 for summaries of these tasks.

For semantic segmentation [Sha+17] proposes a one-shot segmentor (OSLSM), which requires few but densely annotated images, and must independently infer one annotation and class at a time. Our guided segmentor can segment from sparsely annotated pixels and perform multi-way inference. For video object segmentation one-shot video object segmentation (OSVOS) by [Cae+17] achieve high accuracy by fine-tuning during inference, but this online optimization is too costly in time and fails with sparse annotations. Our guided segmentor is feed-forward, hence quick, and segments more accurately from extremely sparse annotations. [Che+18b] impressively achieve state-of-the-art accuracy and real-time, interactive video object segmentation by replacing online optimization with offline metric learning and nearest neighbor inference on a deep, spatiotemporal embedding; however, they focus exclusively on video segmentation. We consider a variety of segmentation tasks, and investigate how guidance transfers across semantic and instance tasks and how it scales with more annotation. For interactive segmentation, [Xu+16; Man+18] learn state-of-the-art interactive object segmentation, and [Man+18] only needs four annotations per object. However, these purely interactive methods infer each task in isolation and cannot pool supervision across tasks and images without optimization, while our guided segmentor quickly propagates supervision non-locally between images.

4.2 Guided Segmentation

Akin to few-shot learning, we divide the input into an annotated support, which supervises the task to be done, and an unannotated query on which to do the task. The common setting in which the support contains K distinct classes and S examples of each is referred to as K-way, S-shot learning [LST15; FFFP06; Vin+16]. For guided segmentation tasks we add a further pixel dimension to this setting, as we must now consider the number of support pixel annotations for each image, as well as the number of annotated support images. We denote the number of annotated pixels per image as P, and consider the settings of (S, P)-shot learning for various S and P. In particular, we focus on sparse annotations where P is small, as these are more practical to collect, and merely require the annotator to point to the segment(s) of interest. This type of data collection is more efficient than collecting dense masks by at least an order of magnitude [Bea+16]. Since segmentation commonly has imbalanced classes and sparse annotations, we consider mixed-shot and semi-supervised supports where the shot varies by class and some points are unlabeled. This is in contrast to the standard few-shot assumption of class-balanced supports.

We define a guided segmentation task as the set of input-output pairs $(\mathcal{T}_i, \mathcal{Y}_i)$ sampled from a task distribution $\mathcal{P}$, adopting and extending the notation of [GB18]. The task inputs are

$$\mathcal{T} = \left\{ \{(x_1, L_1), \dots (x_S, L_S)\} \cup \{\bar{x}_1, \dots, \bar{x}_Q\} \; ; \; x_s, \bar{x}_q \sim \mathcal{P}_l(\mathbb{R}^N) \right\}$$
$$L_s = \{(p_j, l_j) : j \in \{1 \,..\, P\}, \; l \in \{1 \,..\, K\} \cup \{\varnothing\}\}$$

where S is the number of annotated support images x_s, Q is the number of unannotated query images $\bar{x}_q$, and L_s are the support annotations. The annotations are sets of point-label pairs (p, l) with $|L_s| = P$ per image, where every label l is one of the K classes or unknown $(\varnothing)$. The task outputs, that is the targets for the support-defined segmentation task on the queries, are

$$\mathcal{Y} = (y_1, \dots, y_Q), \quad y_q = \{(p_j, l_j) : p_j \in \bar{x}_q\}$$

Our model handles general way K, but for exposition we focus on binary tasks with $K = 2$, or $L = (+, -)$. We let $Q = 1$ throughout as inference of each query is independent in our model.

4.3 Guided Networks

Our approach to guided segmentation has two parts: (1) extracting a task representation from the semi-supervised, structured support and (2) segmenting the query given the task representation. We define the task representation as $z = g(x, +, -)$, and the query segmentation guided by that representation as $\hat{y} = f(\bar{x}, z)$. The design of the task representation

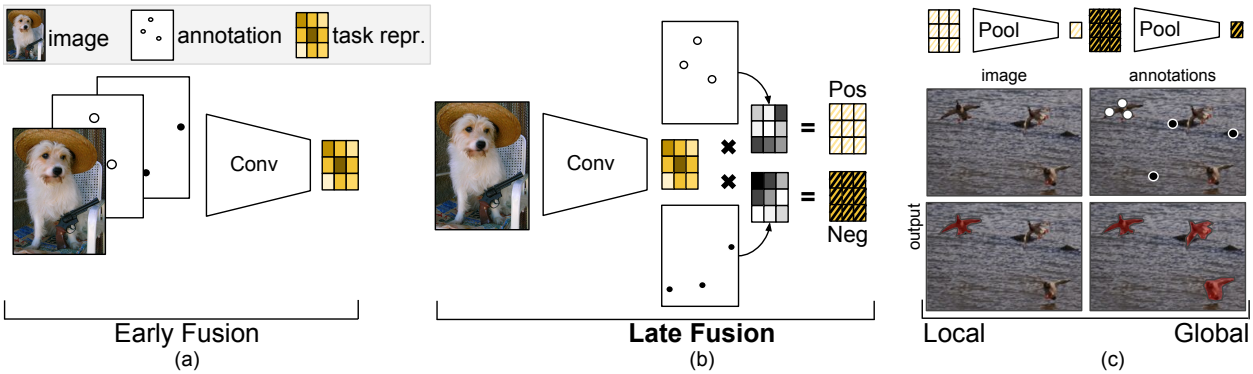


Figure 4.3: Extracting a task representation or "guidance" from the support. (a) Early fusion simply concatenates the image and annotations. (b) Our late fusion factorizes into image and annotation streams, improves accuracy, and updates quickly given new annotations. (c) Globalizing the task representation propagates appearance non-locally: a single bird is annotated in this example, but global guidance causes all the similar-looking birds to be segmented (red) regardless of location.

z and its encoder g is crucial for guided segmentation to handle the hierarchical structure of images and pixels, the high and variable dimensions of images and their pixelwise annotations, the semi-supervised nature of support with many unannotated pixels, and skewed support distributions.

We examine how to best design the guide g and inference f as deep networks. Our method is one part architecture and one part optimization. For architecture, we define branched fully convolutional networks, with a guide branch for extracting the task representation from the support with a novel late fusion technique (Section 4.3.1), and an inference branch for segmenting queries given the guidance (Section 4.3.2). For optimization, we adapt episodic meta-learning to image-to-image learning for structured output (Section 4.3.3), and increase the diversity of episodes past existing practice by sampling within and *across* segmentation task families like categories and instances.

4.3.1 Guidance: from Supervision to Latent Task Representation

The task representation z must fuse the visual information from the image with the annotations in order to determine what should be segmented in the query. As images with (partial) segmentations, our support is statistically dependent because pixels are spatially correlated, semi-supervised because the full supervision is arduous to annotate, and high dimensional and class-skewed because scenes are sizable and complicated. For simplicity, we first consider a binary task with $(1, P)$-shot support consisting of one image with an arbitrary number of annotated pixels P, and then extend to multi-way tasks and general (S, P)-shot support. To begin we decompose the support encoder $g(x_s, +_s, -_s)$ across receptive fields indexed by i for local task representations $z_i = g(x_{si}, +_{si}, -_{si})$; this is the same independence assumption

made by existing fully convolutional approaches to structured output. See Figure 4.3 for an overview and our novel late global fusion technique.

Early Fusion (prior work) Stacking the image and annotations channel-wise at the input makes $z_{si} = g_{\text{early}}(x, +, -) = \phi_S(x \oplus + \oplus -)$ with a support feature extractor ϕ_S. This early fusion strategy, employed by [Xu+16], gives end-to-end learning full control of how to fuse. Masking the image by the positive pixels [Sha+17; Yoo+17] instead forces invariance to context, potentially speeding up learning, but precludes learning from the background and disturbs input statistics. All early fusion techniques suffer from an inherent modeling issue: incompatibility of the support and query representations. Stacking requires distinct ϕ_S, ϕ_Q while masking disturbs the input distribution. Early fusion is slow, since changes in annotations trigger a full pass through the network, and only one task can be inferred at a time, limiting existing interactive and few-shot segmentors alike [Xu+16; Man+18; Sha+17].

Late Fusion (ours) We resolve the learning and inference issues of early fusion by factorizing features and annotations in the guide architecture as $z_{si} = g_{\text{late}}(x, +, -) = \psi(\phi(\bar{x}), m(+), m(-))$. We first extract visual features from the image alone by $\phi(x)$, map the annotations into masks in the feature layer coordinates $m(+), m(-)$, and then fuse both by ψ chosen to be element-wise product. This factorization into visual and annotation branches defines the spatial relationship between image and annotations, improving learning sample efficiency and inference computation time. Fixing m to interpolation and ψ to multiplication, the task representation can be updated quickly by only recomputing the masking and not features ϕ. See Figure 4.3 (center). We do not model a distribution over z, although this is a possible extension of our work for regularization or sampling diverse segmentations.

Our late fusion architecture can now share the feature extractor ϕ for joint optimization through the support and query. Sharing improves learning efficiency with convergence in fewer iterations and task accuracy with 60% relative improvement for video object segmentation. Late fusion reduces inference time, as only the masking needs to be recomputed to incorporate new annotations, making it capable of real-time interactive video segmentation. Optimization-based methods [Cae+17] need seconds or minutes to update.

Locality We are generally interested in segmentation tasks that are determined by visual characteristics and not absolute location in space or time, i.e. the task is to group pixels of an object and not pixels in the bottom-left of an image. When the support and query images differ, there is no known spatial correspondence, and the only mapping between support and query should be through features. To fit the architecture to this task structure, we merge the local task representations by $m_P(\{z_{si} : \forall i\})$ for all positions i. Choosing global pooling for m_P globalizes the task by discarding the spatial dimensions. The pooling step can be done by averaging, our choice, or other reductions. The effect of pooling in an image with multiple visually similar objects is shown in Figure 4.3 (right).

Multi-Shot and Multi-Way The full (S, P)-shot setting requires summarizing the entire support with a variable number of images with varying amounts of pixelwise annotations. Note in this case that the annotations might be divided across the support, for instance one frame of a video may only have positives while a different frame has only negatives, so S-shot cannot always be reduced to 1-shot, as done in prior work [Sha+17]. We form the full task

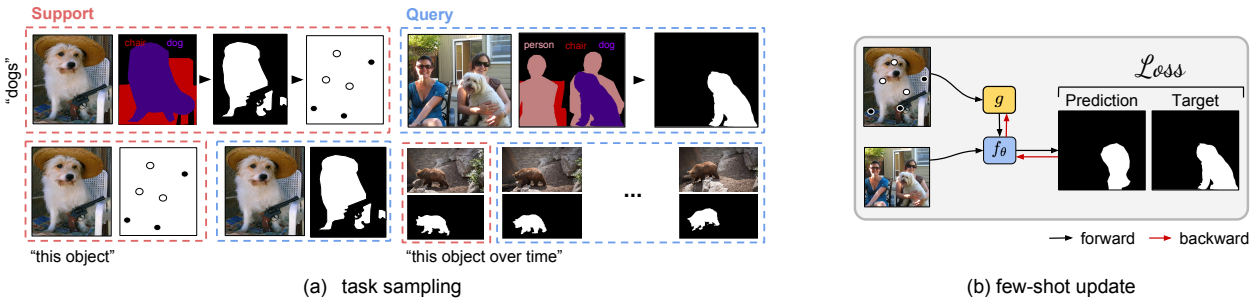


Figure 4.4: Optimization for guided segmentation. (a) Synthesizing tasks from densely annotated segmentation data. (b) One task update: episodic training reduces to supervised learning.

representation $z_S = m_S(\{z_1, \ldots, z_S\})$ simply and differentiably by averaging the shot-wise representations z_s. While we have considered binary tasks thus far, we extend guidance to multi-way inference do in our experiments. We construct a separate guide for each class, averaging across all shots containing annotations for that class. Note that all the guides share ϕ for efficiency and differ only in the masking.

4.3.2 Guiding Inference by Metric Learning

Inference in a static segmentation model is simply $\hat{y} = f_\theta(\bar{x})$ for output y, parameters θ, and input $\bar{x}$. Guided inference is a function $\hat{y} = f(\bar{x}, z)$ of the query given the guidance extracted from the support. We further structure inference by $f(\phi(\bar{x}), z)$, where ϕ is a fully convolutional encoder from input pixels to visual features.

Multiple forms of conditioning are possible and have been explored for low-dimensional classification and regression problems by the few-shot learning literature. In preliminary experiments we consider parameter regression, nearest neighbor and prototype retrieval, and metric learning on fused features. We select metric learning with feature fusion because it is simple and robust to optimize. Note that feature fusion is similar to siamese architectures, but we directly optimize the classification loss rather than a contrastive loss.

In particular we fuse features by $m_f = \phi(x) \oplus \text{tile}(z)$ which concatenates the guide with the query features, while tiling z to the spatial dimensions of the query. The fused query-support feature is then scored by a small convolutional network f_θ that can be interpreted as a learned distance metric for retrieval from support to query. For multi-way guidance, the fusions of the query and each guide are batched for parallel inference.

4.3.3 Episodic Optimization and Task Distributions

We distinguish between optimizing the parameters of the model during training (learning) and adapting the model during inference (guidance). Thus during training, we wish to "learn to guide." In standard supervised learning, the model parameters θ are optimized according to the loss between prediction $\hat{y} = f_\theta(x)$ and target y. We reduce the problem of learning

to guide to supervised learning by jointly optimizing the parameters of the guidance branch g and the segmentation branch f according to the loss between $f_\theta(\bar{x}, z)$ and query target y, see Figure 4.4.

For clarity, we distinguish between tasks, a given support and query for segmentation, and task distributions that define a kind of segmentation problem. For example, semantic segmentation is a task distribution while segmenting birds (a semantic class) is a task. We train a guided network for each task distribution by optimizing episodically on sampled tasks. The supports and queries that comprise an episode are synthesized from a fully labeled dataset. We first sample a task, then a subset of images containing that task which we divide into support and query. During training, the target for the query image is available, while for testing it is not. We binarize support and query annotations to encode the task, and spatially sample support annotations for sparsity.

Given inputs and targets, we train the network with the pixelwise cross-entropy loss between the predicted and target segmentation of the query. See Sections **??** and **??** for more details on data processing and network optimization respectively.

After learning, the model parameters are fixed, and task inference is determined by guidance. While we evaluate for varying support size S, as described in 4.3.2, we train with $S = 1$ for efficiency while sampling $P \sim \text{Uniform}(1, 100)$. Once learned, our guided networks can operate at different (S, P) shots to address sparse and dense pixelwise annotations with the same model, unlike existing methods that train for particular shot and way. In our experiments, we train with tasks sampled from a single task distribution, but co- or cross-supervision of distributions is possible. Intriguingly, we see some transfer between distributions when evaluating a guided network on a different distribution than it was trained on in Section 4.4.3.

4.4 Results

We evaluate our guided segmentor on a variety of problems that are representative of segmentation as a whole: interactive segmentation, semantic segmentation, and video object segmentation. These are conventionally regarded as separate problems, but we demonstrate that each can be viewed as an instantiation of guided segmentation. As a further demonstration of our method, we present results for real-time, interactive video segmentation from dot annotations. To better understand the characteristics of guidance, we experiment with cross-task supervision in Section 4.4.2 and guiding with large-scale supports in Section 4.4.3.

To standardize evaluation we select one metric for all tasks: the intersection-over-union (IU) of the positives averaged across all tasks and masks. This choice allows us to compare scores across the different kinds of segmentation we consider without skew from varying numbers of classes or images. Note that this metric is not equivalent to the mean IU across classes that is commonly reported for semantic segmentation. Please refer to Section **??** for more detail.

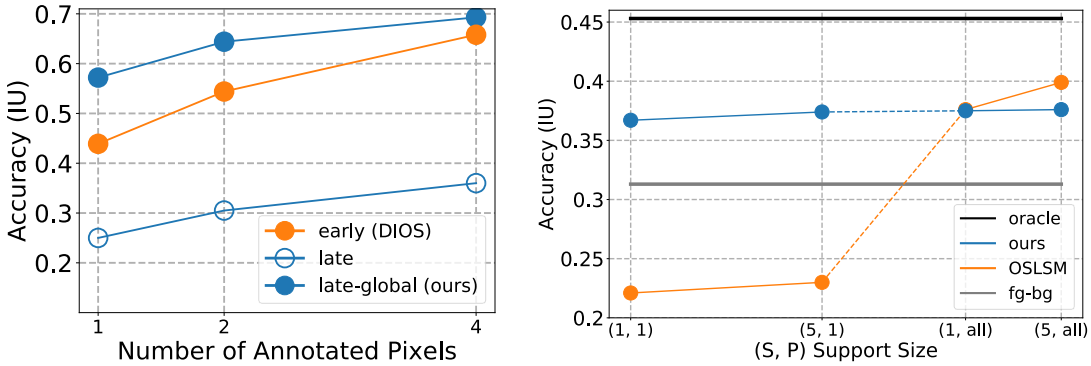


Figure 4.5: (left) Interactive segmentation of objects in images. (right) Guided semantic segmentation of held-out classes: we are state-of-the-art with only two points and competitive with full annotations.

We include fine-tuning and foreground-background segmentation as baselines for all problems. Fine-tuning simply attempts to optimize the model on the support. Foreground-background verifies that methods are learning to co-vary their output with the support supervision and sets an accuracy floor.

The backbone of our networks is VGG-16 [SZ15], pre-trained on ILSVRC [Rus+15], and cast into fully convolutional form [SLD17]. This choice is made for fair comparison with existing works across our challenge tasks of semantic, interactive, and video object segmentation without confounds of architecture, pre-training data, and so forth.

4.4.1 Guided Interactive/Video Object/Semantic Segmentation

Interactive Image Segmentation We recover this problem as a special case of guided segmentation when the support and query images are identical. We evaluate on PASCAL VOC [**pascal**] and compare to deep interactive object selection (DIOS) [Xu+16], because it is state-of-the-art and shares our focus on learning for label efficiency and generality. Our approach differs in support encoding: DIOS fuses early while we fuse late and globally. Our guided segmentor is more accurate with extreme sparsity and intrinsically faster to update, as DIOS must do a full forward pass. See Figure 4.5 (left). From this result we decide on late-global guidance throughout.

Video Object Segmentation We evaluate our guided segmentor on the DAVIS 2017 benchmark [PT+17] of 2–3 second videos. For this problem, the object indicated by the fully annotated first frame must be segmented across the video. We then extend the benchmark to sparse annotations to gauge how methods degrade. We compare to OSVOS [Cae+17], a state-of-the-art online optimization method that fine-tunes on the annotated frame and then segments the video frame-by-frame. While [Che+18b] presents impressive results on this task and on real-time interactive video segmentation without optimization, their scope is limited to video, and they employ orthogonal improvements that make comparison difficult.

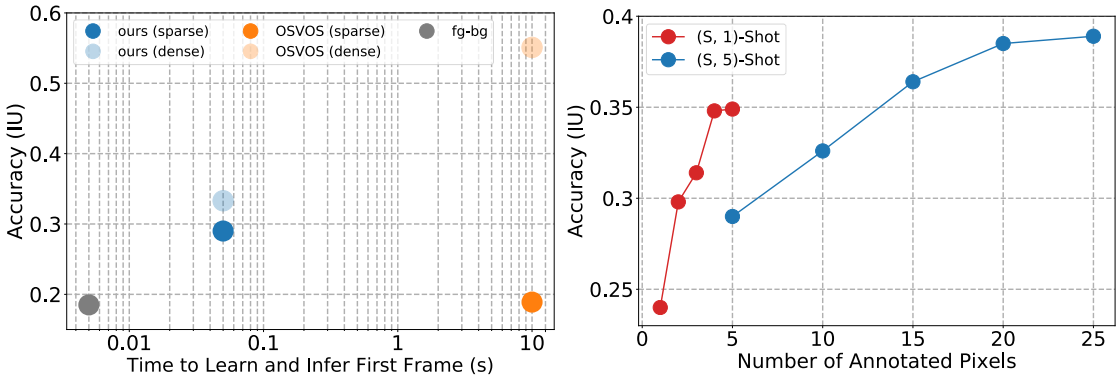


Figure 4.6: (left) Accuracy-time evaluation for sparse and dense video object segmentation on DAVIS'17 val. (right) Real-time interactive video segmentation on simulated dot interactions.

We were unable to reproduce their results in our own experimental framework. See Figure 4.6 (left) for a comparison of accuracy, speed, and annotation sparsity.

In the dense regime our method achieves 33.3% accuracy for 80% relative improvement over OSVOS in the same time envelope. Given (much) more time fine-tuning significantly improves in accuracy, but takes 10+min/video. Guidance is $\sim 200\times$ faster at 3sec/video. Our method handles extreme sparsity with little degradation, maintaining 87% of the dense accuracy with only 5 points for positive and negative. Fine-tuning struggles to optimize over so few annotations.

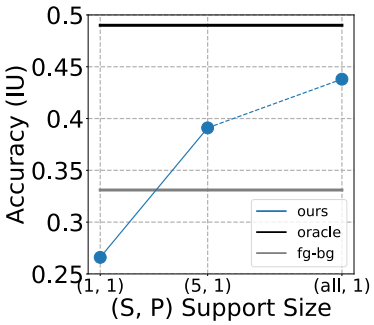
Interactive Video Segmentation By dividing guidance and inference, our guided segmentor can interactively segment video in real time. As an initial evaluation, we simulate interactions with randomly-sampled dot annotations. We define a benchmark by fixing the amount of annotation and measuring accuracy as the annotations are given. The accuracy-annotation tradeoff curve is plotted in Figure 4.6 (right). Our guided segmentor improves with both dimensions of shot, whether images (S) or pixels (P). Our guided architecture is feedforward and fast, and faster still to update for changes to the annotations.

Semantic Segmentation Semantic segmentation is a challenge for learning from little data due to the high intra-class variance of appearance. For this problem it is crucial to evaluate on not only held-out inputs, but held-out classes, to be certain the guided learner has not covertly learned to be an unguided semantic segmentor. To do so we follow the experimental protocol of [Sha+17] and score by averaging across four class-wise splits of PASCAL VOC [Eve+10], with has 21 classes (including background), and compare to OSLSM.

Our approach achieves state-of-the-art sparse results that rival the most accurate dense results with just two labeled pixels: see Figure 4.5 (right). OSLSM is incompatible with missing annotations, as it does early fusion by masking, and so is only defined for $\{0, 1\}$ annotations. To evaluate it we map all missing annotations to negative. Foreground-background is a strong baseline, and we were unable to improve on it with fine-tuning. The oracle is trained on all classes (nothing is held-out).

4.4.2 Switching between Class and Instance Tasks

We carry out a novel examination of meta-learning with cross-task supervision. In the language of task distributions, the distribution of instance tasks for a given semantic category are nested in the distribution of tasks for that category. We investigate whether meta-training on the sub-tasks (instances) can address the super-tasks (classes). This tests whether guidance can capture an enumerative definition of a semantic class as the union of instances in that category.



To do so, we meta-train our guided segmentor on interactive *instance* segmentation tasks draw from all classes of PASCAL VOC [Eve+10], and then evaluate the model on *semantic* segmentation tasks from all categories. We experiment with $(S, 1)$ support from semantic annotations, where S varies from one image to all the images in the training set, shown in the plot to the right. We compare to foreground-background as a class-agnostic accuracy floor, and a standard semantic segmentation net trained with semantic labels as an oracle. Increasing the amount of semantic annotations for guidance steadily increases accuracy.

4.4.3 Scaling between Few and Many Annotations

Thus far we have considered guidance in a variable but constrained scale of annotations, ranging from a single pixel in a single image to a few fully annotated images. We meta-learned our guided networks over episodes with such support sizes, and they perform accordingly well in this regime. Here we consider a much wider spectrum of support sizes, with the goal of understanding how guidance compares to standard supervised learning at both ends of the spectrum. To the best of our knowledge, this is the first evaluation of how few-shot learning scales to many-shot usage for structured output.

For this experiment we compare guidance and supervised learning on a transfer task between disjoint semantic categories. We take the classes of PASCAL VOC [Eve+10] as source classes, and take the non-intersecting classes of COCO [Lin+14] as the target classes. We divide COCO 2017 validation into class-balanced train/test halves to look at transfer from a practical amount of annotation (thousands instead of more than a hundred thousand images). Our guided segmentor is meta-trained with semantic tasks sampled from the source classes, then guided with 5,989 densely annotated semantic masks from the target classes. For fair comparison, the supervised learner is first trained on the source classes, and then fine-tuned on the same annotated target data. Both methods share the same ILSVRC pre-training, backbone architecture, and (approximate) number of parameters. In this many-shot regime, guidance achieves 95% of supervised learning performance. A key point of this result is to shed light on the spectrum of supervision that spans few-shot and many-shot settings, and encourage future work to explore bridging the two.

4.5 Conclusion

Guided segmentation unifies annotation-bound segmentation problems. Guided networks reconcile task-driven and interactive inference by extracting guidance, a latent task representation, from any amount of supervision given. With guidance our segmentor revolver can learn and infer tasks without optimization, improve its accuracy near-instantly with more supervision, and once-guided can segment new images without the supervisor in the loop.

Chapter 5

Learning & Adapting the Degree of Locality

Although the visual world is varied, it nevertheless has ubiquitous structure. Structured factors, such as scale, admit clear theories and efficient representation design. Unstructured factors, such as what makes a cat look like a cat, are too complicated to model analytically, requiring free-form representation learning. How can recognition harness structure without restraining the representation?

Free-form representations are structure-agnostic, making them general, but not exploiting structure is computationally and statistically inefficient. Structured representations like steerable filtering [FA91; SF95; Jac+16], scattering [BM13; SM13], and steerable networks [CW17] are efficient but constrained to the chosen structures. We propose a new, semi-structured compositional filtering approach to blur the line between free-form and structured representations and learn both. Doing so learns local features and the degree of locality.

Free-form filters, directly defined by the parameters, are general and able to cope with unknown variations, but are parameter inefficient. Structured factors, such as scale and orientation, are enumerated like any other variation, and require duplicated learning across different layers and channels. Nonetheless, end-to-end learning of free-form parameters is commonly the most accurate approach to complex visual recognition tasks when there is sufficient data.

Structured filters, indirectly defined as a function of the parameters, are theoretically clear and parameter efficient, but constrained. Their effectiveness hinges on whether or not they encompass the true structure of the data. If not, the representation is limiting, and subject to error. At least, this is a danger when *substituting* structure to replace learning.

We *compose* free-form and structured filters, as shown in Figure 5.1, and learn both end-to-end. Free-form filters are not constrained by our composition. This makes our approach more expressive, not less, while still able to efficiently learn the chosen structured factors. In this way our semi-structured networks can reduce to existing networks as a special case. At the same time, our composition can learn different receptive fields that cannot be realized in the standard parameterization of free-form filters. Adding more free-form parameters

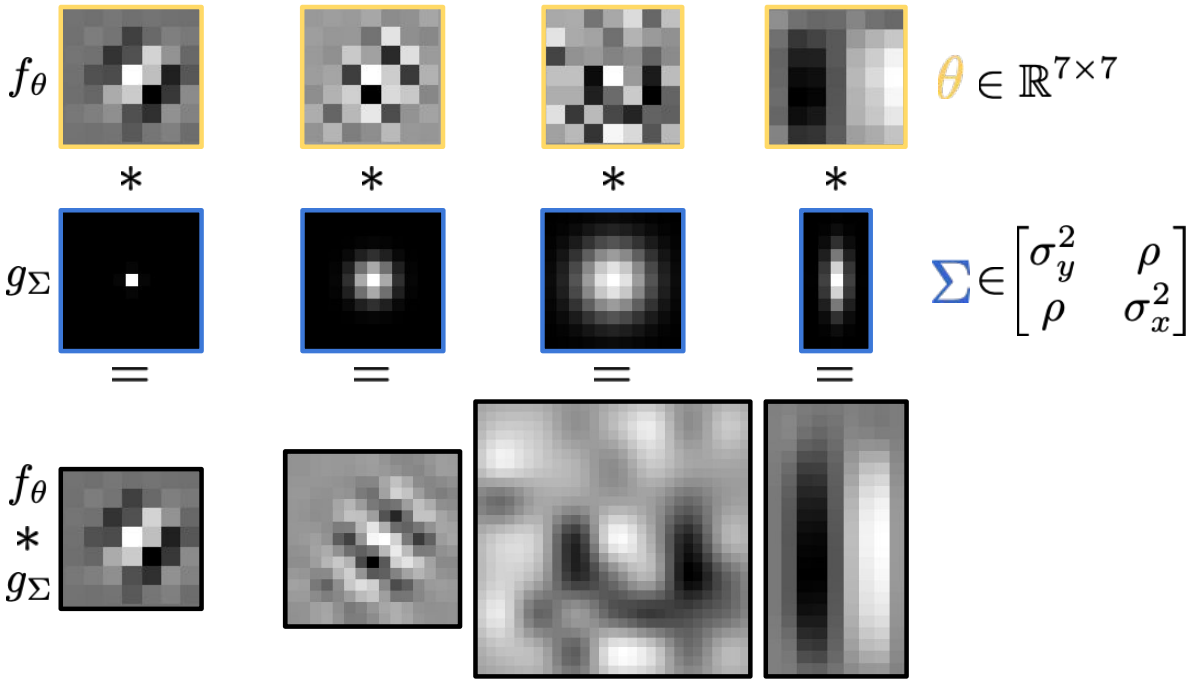


Figure 5.1: We compose free-form filters f_θ and structured Gaussian filters g_Σ by convolution $*$ to define a more general family of semi-structured filters than can be learned by either alone. Our composition makes receptive field scale, aspect, and orientation differentiable in a low-dimensional parameterization for efficient end-to-end learning.

or dilating cannot learn the same family of filters. Figure 5.2 offers one example of the impracticality of architectural alternatives.

Gaussian structure represents scale, aspect, and orientation through covariance [Lin94]. Optimizing these factors carries out a form of differentiable architecture search over receptive fields, reducing the need for onerous hand-design or expensive discrete search. Any 2D Gaussian has the same, low number of covariance parameters no matter its spatial extent, so receptive field optimization is low-dimensional and efficient. Because the Gaussian is smooth, our filtering is guaranteed to be proper from a signal processing perspective and avoid aliasing.

Our contributions include: (1) defining semi-structured compositional filtering to bridge classic ideas for scale-space representation design and current practices for representation learning, (2) exploring a variety of receptive fields that our approach can learn, and (3) adapting receptive fields with accurate and efficient dynamic Gaussian structure.

5.1 Related Work

Composing structured Gaussian filters with free-form learned filters draws on structured filter design and representation learning. Our work is inspired by the transformation invariance

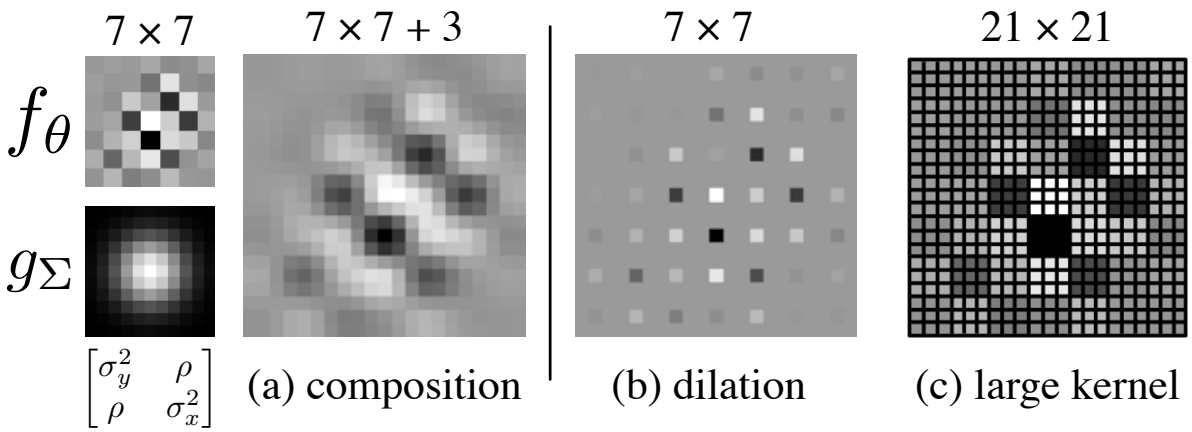


Figure 5.2: Our composition (a) cannot be practically reduced to dilation (b) or more free-form parameters (c). Adding more parameters, here about 10× by 21^2 vs. 7^2, is inefficient and limited: more parameters take more data and optimization to learn and the maximum scale is fixed. Dilating the filter has side effects and is constrained: sparse sampling causes artifacts and the sparsity and scale are fixed.

of scale-space [Lin94], the parsimony of steerable filtering [FA91; Per95; BM13; CW17], and the adaptivity of dynamic inference [OAVE93; Jad+15; DB+16; Dai+17]. Analysis that the effective receptive field size of deep networks is limited [Luo+16], and only is a fraction of the theoretical size, motivates our goal of making unbounded receptive field size and varied receptive field shapes practically learnable.

Transformation Invariance Gaussian scale-space and its affine extension connect covariance to spatial structure for transformation invariance [Lin94]. We jointly learn structured transformations via Gaussian covariance and features via free-form filtering. Enumerative methods cover a set of transformations, rather than learning to select transformations: image pyramids [BA83] and feature pyramids [KSJ14; SLD17; Lin+17] cover scale, scattering [BM13] covers scales and rotations, and steerable networks [CW17] cover discrete groups. Our learning and inferring covariance relates to scale selection [Lin98], as exemplified by the scale invariant feature transform [Low04]. Scale-adaptive convolution [Zha+17a] likewise selects scales, but without our Gaussian structure and smoothness.

Steering Steering indexes a continuous family of filters by linearly weighting a structured basis, such as Gaussian derivatives. Steerable filters [FA91] index orientation and deformable kernels [Per95] index orientation and scale. Such filters can be stacked into a deep, structured network [Jac+16]. These methods have elegant structure, but are constrained to it. We make use of Gaussian structure, but keep generality by composing with free-form filters.

Dynamic Inference Dynamic inference adapts the model to each input. Dynamic routing [OAVE93], spatial transformers [Jad+15], dynamic filter nets [DB+16], and deformable convolution [Dai+17] are all dynamic, but lack local structure. We incorporate Gaussian structure to improve efficiency while preserving accuracy.

Proper signal processing, by blurring when downsampling, improves the shift-equivariance of learned filtering [Zha19]. We reinforce these results with our experiments on blurred dilation, to complement their focus on blurred stride. While we likewise blur, and confirm the need for smoothing to prevent aliasing, our focus is on how to jointly learn and compose structured and free-form filters.

5.2 A Clear Review of Blurring

We introduce the elements of our chosen structured filters first, and then compose free-form filters with this structure in the next section. While the Gaussian and scale-space ideas here are classic, our end-to-end optimized composition and its use for receptive field learning are novel.

5.2.1 Gaussian Structure

The choice of structure determines the filter characteristics that can be represented and learned.

We choose Gaussian structure. For modeling, it is differentiable for end-to-end learning, low-dimensional for efficient optimization, and still expressive enough to represent a variety of shapes. For signal processing, it is smooth and admits efficient filtering. In particular, the Gaussian has these attractive properties for our purposes:

- shift-invariance for convolutional filtering,
- normalization to preserve input and gradient norms for stable optimization,
- separability to reduce computation by replacing a 2D filter with two 1D filters,
- and cascade smoothing from semi-group structure to decompose filtering into smaller, cumulative steps.

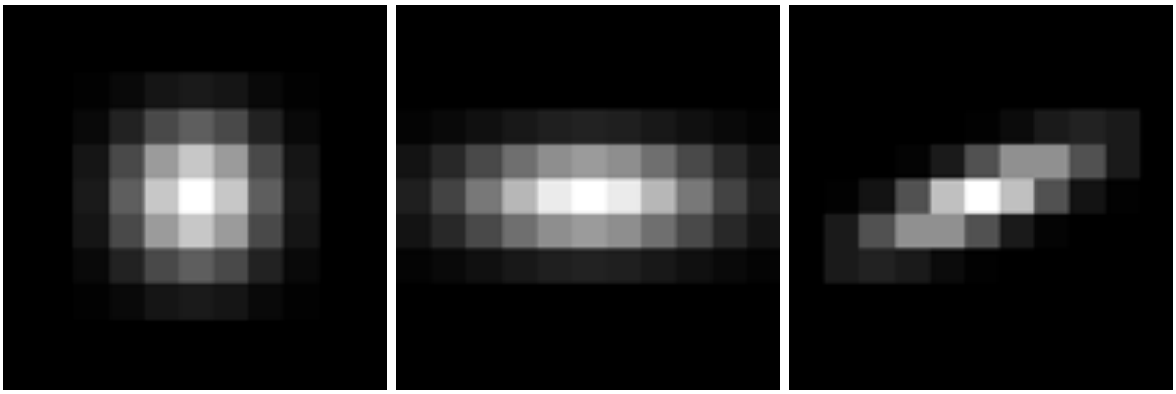
In fact, the Gaussian is the unique filter satisfying these and further scale-space axioms [Koe84; Bab+86; Lin94].

The Gaussian kernel in 2-D is

$$G(\mathbf{x}; \mathbf{\Sigma}) = \frac{1}{2\pi\sqrt{\det \mathbf{\Sigma}}} e^{-\mathbf{x}^T \mathbf{\Sigma}^{-1} \mathbf{x}/2} \tag{5.1}$$

for input coordinates x and covariance $\Sigma \in \mathbb{R}^{2\times 2}$, a symmetric positive-definite matrix.

The structure of the Gaussian is controlled by its covariance Σ. Note that we are concerned with the spatial covariance, where the coordinates are considered as random variables, and not the covariance of the feature dimensions. Therefore the elements of the covariance matrix are σ_y^2, σ_x^2 for the y, x coordinates and ρ for their correlation. The standard, isotropic Gaussian has identity covariance $\left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$. There is progressively richer structure in spherical,



(a) spherical $\begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$ (b) diagonal $\begin{bmatrix} \sigma_y^2 & 0 \\ 0 & \sigma_x^2 \end{bmatrix}$ (c) full $\begin{bmatrix} \sigma_y^2 & \rho \\ \rho & \sigma_x^2 \end{bmatrix}$

Figure 5.3: Gaussian covariances come in families with progressively richer structure: (a) spherical covariance has one parameter for scale; (b) diagonal covariance has two parameters for scale and aspect; and (c) full covariance has three parameters for scale, aspect, and orientation/slant.

diagonal, and full covariance: Figure 5.3 illustrates these kinds and the scale, aspect, and orientation structure they represent.

Selecting the right spatial covariance yields invariance to a given spatial transformation. The standard Gaussian indexes scale-space, while the full covariance Gaussian indexes its *affine* extension [Lin94]. We leverage this transformation property of Gaussians to learn receptive field shape in Section 5.3.1 and dynamically adapt their structure for local spatially invariant filtering in Section 5.3.2.

From the Gaussian kernel $G(x, \Sigma)$ we instantiate a Gaussian filter $g_\Sigma(\cdot)$ in the standard way: (1) evaluate the kernel at the coordinates of the filter coefficients and (2) renormalize by the sum to correct for this discretization. We decide the filter size according to the covariance by setting the half size $= \lceil 2\sigma \rceil$ in each dimension. This covers $\pm\ 2\sigma$ to include 95% of the true density no matter the covariance. (We found that higher coverage did not improve our results.) Our filters are always odd-sized to keep coordinates centered.

5.2.2 Covariance Parameterization & Optimization

The covariance Σ is symmetric positive definite, requiring proper parameterization for unconstrained optimization. We choose the log-Cholesky parameterization [PB96] for iterative optimization because it is simple and quick to compute: $\Sigma = U'U$ for upper-triangular U with positive diagonal. We keep the diagonal positive by storing its log, hence *log*-Cholesky, and exponentiating when forming Σ. (See [PB96] for a primer on covariance parameterization.)

Here is an example for full covariance Σ with elements σ_y^2, σ_x^2 for the y, x coordinates

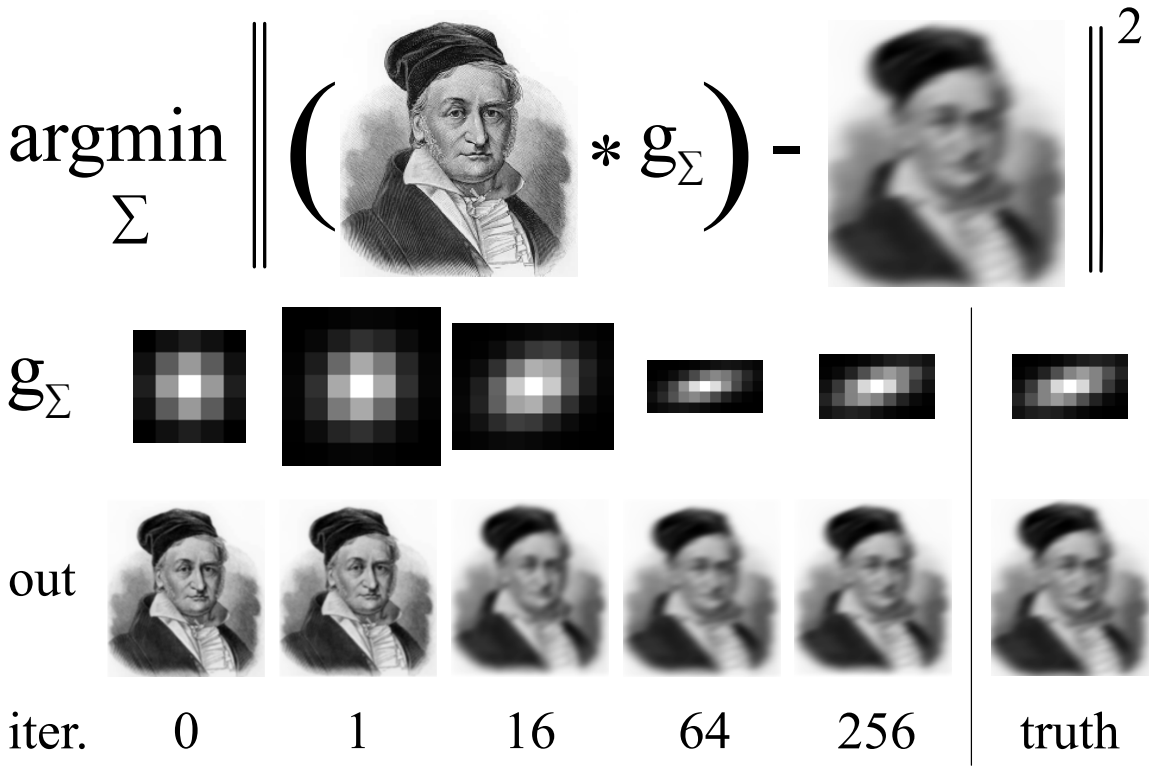


Figure 5.4: Recovering an unknown blur by optimizing over covariance. Gradient optimization of the structured parameters Σ quickly converges to the true Gaussian. Although this is a simple example, it shows the effectiveness of the Gaussian for representing scale, aspect, and orientation.

and ρ for their correlation:

$$\begin{bmatrix} \sigma_y^2 & \rho \\ \rho & \sigma_x^2 \end{bmatrix} \leftarrow \begin{bmatrix} +1 & -2 \\ -2 & +8 \end{bmatrix} = \begin{bmatrix} +1 & +0 \\ -2 & +2 \end{bmatrix} \begin{bmatrix} +1 & -2 \\ +0 & +2 \end{bmatrix}$$
$$= (\log(1), -2, \log(2)).$$

Spherical and diagonal covariance are parameterized by fixing $\rho = 0$ and tying/untying σ_y, σ_x. Note that we overload notation and use Σ interchangeably for the covariance matrix and its log-Cholesky parameters.

Our composition learns Σ by end-to-end optimization of structured parameters, not statistical estimation of empirical distributions. In this way the Gaussian is determined by the task loss, and not by input statistics, as is more common.

5.2.3 Learning to Blur

As a pedagogical example, consider the problem of optimizing covariance to reproduce an unknown blur. That is, given a reference image and a blurred version of it, which Gaus-

(a) limit as $\sigma \to 0$ (b) $\sigma < 1$ (c) limit as $\sigma \to \infty$

Figure 5.5: Special cases of the Gaussian are helpful for differentiable model search. (a) The identity is recovered by filtering with a delta as variance goes to zero. (b) A smoothed delta from small variance is a good initialization to make use of pre-training. (c) Global average pooling is recovered as variance goes to infinity. Each filter is normalized separately to highlight the relationship between points.

sian filter causes this blur? Figure 5.4 shows such an optimization: from an identity-like initialization the covariance parameters quickly converge to the true Gaussian.

Given the full covariance parameterization, optimization controls scale, aspect, and orientation. Each degree of freedom can be seen across the iterates of this example. Had the true blur been simpler, for instance spherical, it could still be swiftly recovered in the full parameterization.

Notice how the size and shape of the filter vary over the course of optimization: this is only possible through structure. For a Gaussian filter, its covariance is the intrinsic structure, and its coefficients follow from it. The filter size and shape change while the dimension of the covariance itself is constant. Lacking structure, free-form parameterization couples the number of parameters and filter size, and so cannot search over size and shape in this fashion.

5.3 Semi-Structured Compositional Filtering

Composition and backpropagation are the twin engines of deep learning [Fuk80; LeC+98b]: composing learned linear operations with non-linearities yields deep representations. Deep visual representations are made by composing convolutions to learn rich features and *receptive fields*, which characterize the spatial extent of the features. Although each filter might be small, and relatively simple, their composition can represent and learn large, complex receptive fields. For instance, a stack of two 3×3 filters is effectively 5×5 but with fewer degrees of freedom ($2 \cdot 3^2$ vs. 5^2). Composition therefore induces factorization of the representation, and this factorization determines the generality and efficiency of the representation.

Our semi-structured composition factorizes the representation into spatial Gaussian receptive fields and free-form features. This composition is a novel approach to making receptive field shape differentiable, low-dimensional, and decoupled from the number of parameters. Our approach jointly learns the structured and free-form parameters while guaranteeing proper sampling for smooth signal processing. Purely free-form filters cannot learn shape and size in this way: shape is entangled in all the parameters and size is bounded by the number of parameters. Purely structured filters, restricted to Gaussians and their derivatives for instance, lack the generality of free-form filters. Our factorization into structured and free-form filters is efficient for the representation, optimization, and inference of receptive fields without sacrificing the generality of features.

Receptive field size is a key design choice in the architecture of fully convolutional networks for local prediction tasks [SLD17]. The problem of receptive field design is commonly encountered with each new architecture, dataset, or task. Optimizing our semi-structured filters is equivalent to differentiable architecture search over receptive field size and shape. By making this choice differentiable, we show that learning can adjust to changes in the architecture and data in Section 5.4.2. Trying candidate receptive fields by enumeration is expensive, whether by manual search or automated search [ZL17; Kan+18; LSY19]. Semi-structured composition helps relieve the effort and computational burden of architecture design by relaxing the receptive field from a discrete decision into a continuous optimization.

5.3.1 Composing with Convolution and Covariance

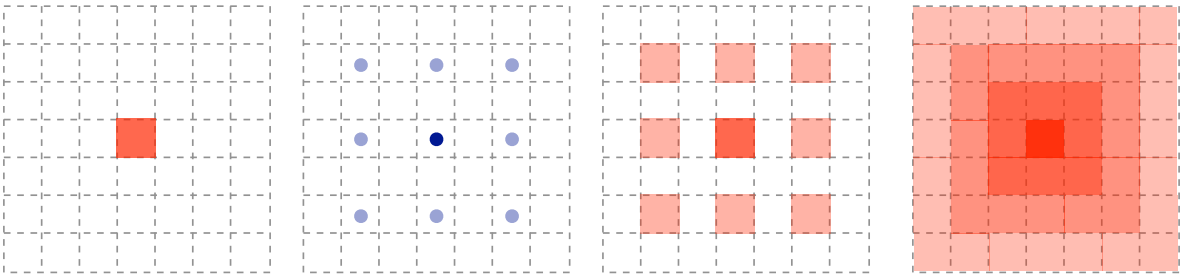
Our composition $f_\theta \circ g_\Sigma$ combines a free-form f_θ with a structured Gaussian g_Σ. The computation of our composition reduces to convolution, and so it inherits the efficiency of aggressively tuned convolution implementations. Convolution is associative, so compositional filtering of an input I can be decomposed into two steps of convolution by

$$I * (g_\Sigma * f_\theta) = I * g_\Sigma * f_\theta. \tag{5.2}$$

This decomposition has computational advantages. The Gaussian step can be done by specialized filtering that harnesses separability, cascade smoothing, and other Gaussian structure. Memory can be spared by only keeping the covariance parameters and recreating the Gaussian filters as needed (which is quick, although it is a space-time tradeoff). Each compositional filter can always be explicitly formed by $g_\Sigma * f_\theta$ for visualization (see Figure 5.1) or other analysis.

Both θ and Σ are differentiable for end-to-end learning.

How the composition is formed alters the effect of the Gaussian on the free-form filter. Composing by convolution with the Gaussian then the free-form filter has two effects: it shapes and blurs the filter. Composing by convolution with the Gaussian and resampling *according to the covariance* purely shapes the filter. That is, blurring and resampling first blurs with the Gaussian, and then warps the sampling points for the following filtering by the covariance. Either operation might have a role in representation learning, so we experiment



(a) input (b) dilated filter (c) output (d) with blur

Figure 5.6: Blurring prevents aliasing or "gridding" artifacts by smoothing dilation to respect the sampling theorem.

with each in Table 5.2. In both cases the composed filter is dense, unlike a sparse filter from dilation.

When considering covariance optimization as differentiable receptive field search, there are special cases of the Gaussian that are useful for particular purposes. See Figure 5.5 for how the Gaussian can be reduced to the identity, initialized near the identity, or reduced to average pooling. The Gaussian includes the identity in the limit, so our models can recover a standard networks without our composition of structure. By initializing near the identity, we are able to augment pre-trained networks without interference, and let learning decide whether or not to make use of structure.

Blurring for Smooth Signal Processing Blurring (and resampling) by the covariance guarantees proper sampling for correct signal processing. It synchronizes the degree of smoothing and the sampling rate to avoid aliasing. Their combination can be interpreted as a smooth, circular extension of dilated convolution [Che+15; YK16] or as a smooth, affine restriction of deformable convolution [Dai+17]. Figure 5.6 contrasts dilation with blurring & resampling. For a further perspective, note this combination is equivalent to downsampling/upsampling with a Gaussian before/after convolving with the free-form filter.

Even without learning the covariance, blurring can improve dilated architectures. Dilation is prone to gridding artifacts [YKF17; Wan+18]. We identify these artifacts as aliasing caused by the spatial sparsity of dilated filters. We fix this by smoothing with standard deviation proportional to the dilation rate. Smoothing when subsampling is a fundamental technique in signal processing to avoid aliasing [OS09], and the combination serves as a simple alternative to the careful re-engineering of dilated architectures. Improvements from blurring dilation are reported in Table 5.3.

Compound Gaussian Structure Gaussian filters have a special compositional structure we can exploit: cascade smoothing. Composing a Gaussian g_Σ with a Gaussian $g_{\Sigma'}$ is still Gaussian with covariance $\Sigma + \Sigma'$. This lets us efficiently assemble *compound* receptive fields made of multiple Gaussians. Center-surround [Kuf53] receptive fields, which boost contrast, can be realized by such a combination as Difference-of-Gaussian [RS65] (DoG) filters, which subtract a larger Gaussian from a smaller Gaussian. Our joint learning of their covariances tunes the contrastive context of the receptive field, extending [Din+18] which

learns contrastive filters with fixed receptive field sizes.

Design Choices Having defined our semi-structured composition, we cover the design choices involved in its application. As a convolutional composition, it can augment any convolution layer in the architecture. We focus on including our composition in late, deep layers to show the effect without much further processing. We add compositional filtering to the output and decoder layers of fully convolutional networks because the local tasks they address rely on the choice of receptive fields.

Having decided where to compose, we must decide how much structure to compose. There are degrees of structure, from minimal structure, where each layer or stage has only one shared Gaussian, to dynamic structure, where each receptive field has its own structure that varies with the input. In between there is channel structure, where each free-form filter has its own Gaussian shared across space, or multiple structure, where each layer or filter has multiple Gaussians to cover different shapes. We explore minimal structure and dynamic structure in order to examine the effect of composition for static and dynamic inference, and leave the other degrees of structure to future work.

5.3.2 Dynamic Gaussian Structure

Semi-structured composition learns a rich family of receptive fields, but visual structure is richer still, because structure locally varies while our filters are fixed. Even a single image contains variations in scale and orientation, so one-size-and-shape-fits-all structure is suboptimal. *Dynamic* inference replaces static, global parameters with dynamic, local parameters that are inferred from the input to adapt to these variations. Composing with structure by convolution cannot locally adapt, since the filters are constant across the image. We can nevertheless extend our composition to dynamic structure by representing local covariances and instantiating local Gaussians accordingly. Our composition makes dynamic inference efficient by decoupling low-dimensional, Gaussian structure from high-dimensional, free-form filters.

There are two routes to dynamic Gaussian structure: local filtering and deformable sampling. Local filtering has a different filter kernel for each position, as done by dynamic filter networks [DB+16]. This ensures exact filtering for dynamic Gaussians, but is too computationally demanding for large-scale recognition networks. Deformable sampling adjusts the position of filter taps by arbitrary offsets, as done by deformable convolution [Dai+17]. We exploit deformable sampling to dynamically form sparse approximations of Gaussians.

We constrain deformable sampling to Gaussian structure by setting the sampling points through covariance. Figure 5.7 illustrates these Gaussian deformations. We relate the default deformation to the standard Gaussian by placing one point at the origin and circling it with a ring of eight points on the unit circle at equal distances and angles. We consider the same progression of spherical, diagonal, and full covariance for dynamic structure. This low-dimensional structure differs from the high degrees of freedom in a dynamic filter network, which sets free-form filter parameters, and deformable convolution, which sets free-form offsets. In this way our semi-structured composition requires only a small, constant number

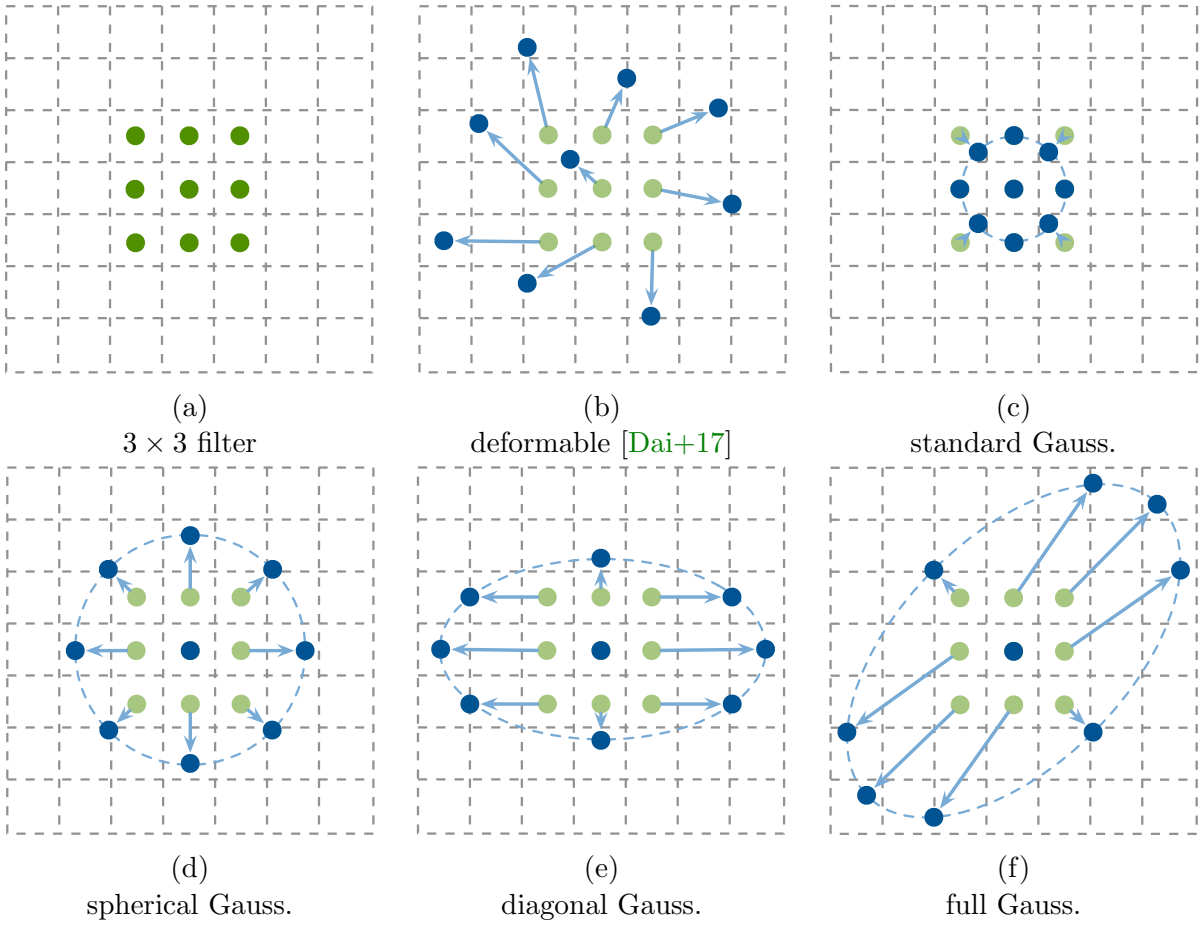


Figure 5.7: Gaussian deformation (c-f) structures dynamic receptive fields by controlling the sampling points (blue) through the covariance. The low-dimensionality of covariance is more efficient than free-form deformation (b) for learning and inference. Although it is less general, it still expresses a variety of shapes.

of covariance parameters independent of the sampling resolution and the kernel size k, while deformable convolution has constant resolution and requires $2k^2$ offset parameters for a $k \times k$ filter.

To infer the local covariances, we follow the deformable approach [Dai+17], and learn a convolutional regressor for each dynamic filtering step. The regressor, which is simply a convolution layer, first infers the covariances which then determine the dynamic filtering that follows. The low-dimensional structure of our dynamic parameters makes this regression more efficient than free-form deformation, as it only has three outputs for each full covariance, or even just one for each spherical covariance. Since the covariance is differentiable, the regression is learned end-to-end from the task loss without further supervision.

We experiment with dynamic structure in Section 5.4.3.

5.4 Results

We experiment with the local task of semantic segmentation, because our method learns the size and shape of local receptive fields. As a recognition task, semantic segmentation requires a balance between local scope, to infer where, and global scope, to infer what. Existing approaches must take care with receptive field design, and their experimental development takes significant model search.

Data CityScapes [Cor+16] is a challenging dataset of varied urban scenes from the perspective of a car-mounted camera. We follow the standard training and evaluation protocols and train/validation splits, with $2,975$ finely-annotated training images and 500 validation images. We score results by the common intersection-over-union metric—the intersection of predicted and true pixels divided by their union then averaged over classes—on the validation set. We evaluate the network itself without post-processing, test-time augmentation, or other accessories to isolate the effect of receptive field learning.

Architecture and Optimization For backbones we choose strong fully convolutional networks derived from residual networks [He+16]. The dilated residual net (DRN) [YKF17] has high resolution and receptive field size through dilation. Deep layer aggregation (DLA) [Yu+18] fuses layers by hierarchical and iterative skip connections. We also define a ResNet-34 backbone as a simple architecture of the kind used for ablations and exploratory experiments. Together they are representative of common architectural patterns in state-of-the-art fully convolutional networks.

We train our models by stochastic gradient descent for 240 epochs with momentum 0.9, batch size 16, and weight decay 10^{-4}. Training follows the "poly" learning rate schedule [Che+18a; Zha+17b] with initial rate 0.01. The input images are cropped to 800×800 and augmented by random scaling within $[0.5, 2]$, random rotation within 10 degrees, and random color distortions as in [How13]. We train with synchronized, in-place batch normalization [RBPK18]. For fair comparison, we reproduce the DRN and DLA baselines in our same setting, which improves on their reported results.

Baselines The chosen DRN and DLA architectures are strong methods on their own, but they can be further equipped for learning global spatial transformations and local deformations. Spatial transformer networks [Jad+15] and deformable convolution [Dai+17] learn dynamic global/local transformations respectively. Spatial transformers serve as a baseline for structure, because they are restricted to a parametric class of transformations. Deformable convolution serves as a baseline for local, dynamic inference without structure. For comparison in the static setting, we simplify both methods to instead learn static transformations.

Naturally, because our composition is carried out by convolution (for static inference), we compare to the baseline of including a free-form convolution layer on its own.

We will release code and reference models for our static and dynamic compositional filtering methods.

method	IU
DRN-A [YKF17]	72.4
+ 3×3 Conv.	72.9
+ STN (static) [Jad+15]	70.5
+ Deformable (static) [Dai+17]	72.2
+ Composition (ours)	**73.5**
+ CCL [Din+18]	73.1
+ DoG (ours)	**74.1**
DLA-34 [Yu+18]	76.1
+ Composition (ours)	**78.2**

Table 5.1: Our composition improves the accuracy of strong, carefully designed architectures.

5.4.1 Learning Receptive Fields

We first show that semi-structured compositional filtering improves the accuracy of strong fully convolutional networks. We then examine how to best implement our composition and confirm the value of smooth signal processing.

Augmenting Backbone Architectures Semi-structured filtering improves the accuracy of strong fully convolutional networks. We augment the last, output stage with a single instance of our composition and optimize end-to-end. See Table 5.1 for the accuracies of the backbones, baselines, and our filtering. Static composition by convolution improves on the backbone by 1-2 points, and dynamic composition boosts the improvement to 4 points (see Section 5.4.3).

Our simple composition improves on the accuracy of the static receptive fields learned by a spatial transformer and deformable convolution. Spatial transformers and our static composition each learn a global transformation, but our Gaussian parameterization is more effectively optimized. Deformable convolution learns local receptive fields, but its free-form parameterization takes more computation and memory. Our edition of DoG, which learns the surround size, improves the accuracy a further 0.5 point.

Note that the backbones are agressively-tuned architectures which required significant model search and engineering effort. Our composition is still able to deliver improvement through learning without further engineering. In the next subsection, we show that joint optimization of our composition does effective model search when the chosen architecture is suboptimal.

How to Compose As explained in Section 5.3.1, we can compose with a Gaussian structured filter by blurring alone or blurring and resampling. As either can be learned end-to-end, we experiment with both and report their accuracies in Table 5.2. From this comparison we choose blurring and resampling for the remainder of our experiments.

Blurred Dilation To isolate the effect of blurring without learning, we smooth dilation with a blur proportional to the dilation rate. CCL [Din+18] and ASPP [Che+18a] are care-

method	IU
ResNet-34	64.8
+ Blur	66.3
+ Blur-Resample	**68.1**
+ DoG Blur	70.3
+ DoG Blur-Resample	**71.4**
DRN-A [YKF17]	72.4
+ Blur	72.2
+ Blur-Resample	**73.5**

Table 5.2: For semi-structured composition, blurring + resampling improves on blurring alone. This holds for the simple composition of a Gaussian and free-form filter, and the compound composition of Difference of Gaussian filtering.

method	IU
DRN-A [YKF17]	72.4
w/ CCL [Din+18]	73.1
+ Blur	**74.0**
w/ ASPP [Che+18a]	74.1
+ Blur	**74.3**

Table 5.3: Blurred dilation respects the sampling theorem by smoothing in proportion to the dilation rate. Blurring in this way gives a small boost in accuracy.

fully designed dilation architectures for context modeling, but neither blurs before dilating. Improvements from blurred dilation are reported in Table 5.3. Although the gains are small, this establishes that smoothing can help. This effect should only increase with dilation rate.

The small marginal effect of blurring without learning shows that most of our improvement is from joint optimization of our composition and dynamic inference.

5.4.2 Differentiable Receptive Field Search

Our composition makes local receptive fields differentiable in a low-dimensional, structured parameterization. This turns choosing receptive fields into a task for learning, instead of designing or manual searching. We demonstrate that this differentiable receptive field search is able to adjust for changes in the architecture and data. Table 5.4 shows how receptive field optimization counteracts the reduction of the architectural receptive field size and the enlargement of the input. These controlled experiments, while simple, reflect a realistic lack of knowledge in practice: for a new architecture or dataset, the right design is unknown.

For these experiments we include our composition in the last stage of the network and

method	no. params	epoch	IU	Δ
DRN-A [YKF17]	many	240	72.4	0
Smaller Receptive Field				
ResNet-34	many	240	64.8	-7.6
+ 3 × 3 Conv.	some	+20	65.8	-6.6
+ Composition	some	+20	68.1	-4.6
+ DoG	some	+20	68.9	-3.5
. . . End-to-End	many	240	**71.4**	-1.0
and 2× Enlarged Input				
ResNet-34	many	240	56.2	-16.2
+ 3 × 3 Conv.	some	+20	56.7	-15.7
+ Composition	some	+20	57.8	-14.6
+ DoG	some	+20	62.7	-9.7
. . . End-to-End	many	240	**66.5**	-5.9

Table 5.4: Adjusting to architecture and data by differentiable receptive field search. When the architectural receptive field is reduced, the learned covariance compensates to enlarge it. When the input is additionally enlarged 2×, the learned covariance grows further still.

only optimize this stage. We do this to limit the scope of learning to the joint optimization of our composition, since then any effect is only attributable to the composition itself. We verify that end-to-end learning further improves results, but controlling for it in this way eliminates the possibility of confounding effects.

In the extreme, we can do *structural* fine-tuning by including our composition in a pre-trained network and only optimizing the covariance. When fine-tuning the structure alone, optimization either reduces the Gaussian to a delta, doing no harm, or slightly enlarges the receptive field, giving a one point boost. Therefore the special case of the identity, as explained in Figure 5.5, is learnable in practice. This shows that our composition helps or does no harm, and further supports the importance of jointly learning the composition as we do.

5.4.3 Adapting Receptive Fields

Learning the covariance optimizes receptive field size and shape. Dynamic inference of the covariance takes this a step further, and adaptively adjusts receptive fields to vary with the input. By locally regressing the covariance, our approach can better cope with factors of variation *within* an image, and do so efficiently through structure.

We compare our Gaussian deformation with free-form deformation in Table 5.5. Controlling deformable convolution by Gaussian structure improves efficiency while preserving accuracy to within one point. While free-form deformations are more general in principle, in

Cityscapes Validation			
method	dyn.?	no. dyn. params	IU
DRN-A [YKF17]		-	72.4
+ Static Composition (ours)		-	73.5
+ Gauss. Deformation (ours)	✓	1	**76.6**
+ Free-form Deformation [Dai+17]	✓	$2k^2$	**76.6**
ResNet-34		-	64.8
+ Static Composition (ours)		-	68.1
+ Gauss. Deformation (ours)	✓	1	74.2
+ Free-form Deformation [Dai+17]	✓	$2k^2$	**75.1**
Cityscapes Test			
DRN-A [YKF17]		-	71.2
+ Gauss. Deformation (ours)	✓	1	**74.3**
+ Free-form Deformation [Dai+17]	✓	$2k^2$	73.6

Table 5.5: Dynamic Gaussian deformation reduces parameters, improves computational efficiency, and rivals the accuracy of free-form deformation. Even restricting the deformation to scale by spherical covariance suffices to nearly equal the free-form accuracy.

practice there is a penalty in efficiency. Recall that the size of our structured parameterization is independent of the free-form filter size. On the other hand unstructured deformable convolution requires $2k^2$ parameters for a $k \times k$ filter.

Qualitative results for dynamic Gaussian structure are shown in Figure 5.8. The inferred local scales reflect scale structure in the input.

In these experiments we restrict the Gaussian to spherical covariance with a single degree of freedom for scale. Our results show that making scale dynamic through spherical covariance suffices to achieve essentially equal accuracy as general, free-form deformations. Including further degrees of freedom by diagonal and full covariance does not give further improvement on this task and data. As scale is perhaps the most ubiquitous transformation in the distribution of natural images, scale modeling might suffice to handle many variations.

5.5 Conclusion

Composing structured Gaussian and free-form filters makes receptive field size and shape differentiable for direct optimization. Through receptive field learning, our semi-structured models do by gradient optimization what current free-form models have done by discrete design. That is, in our parameterization changes in structured *weights* would require changes in free-form *architecture*.

Our method learns local receptive fields. While we have focused on locality in space, the principle is more general, and extends to locality in time and other dimensions.

Factorization of this sort points to a reconciliation of structure and learning, through which known structure is respected and unknown detail is learned freely.

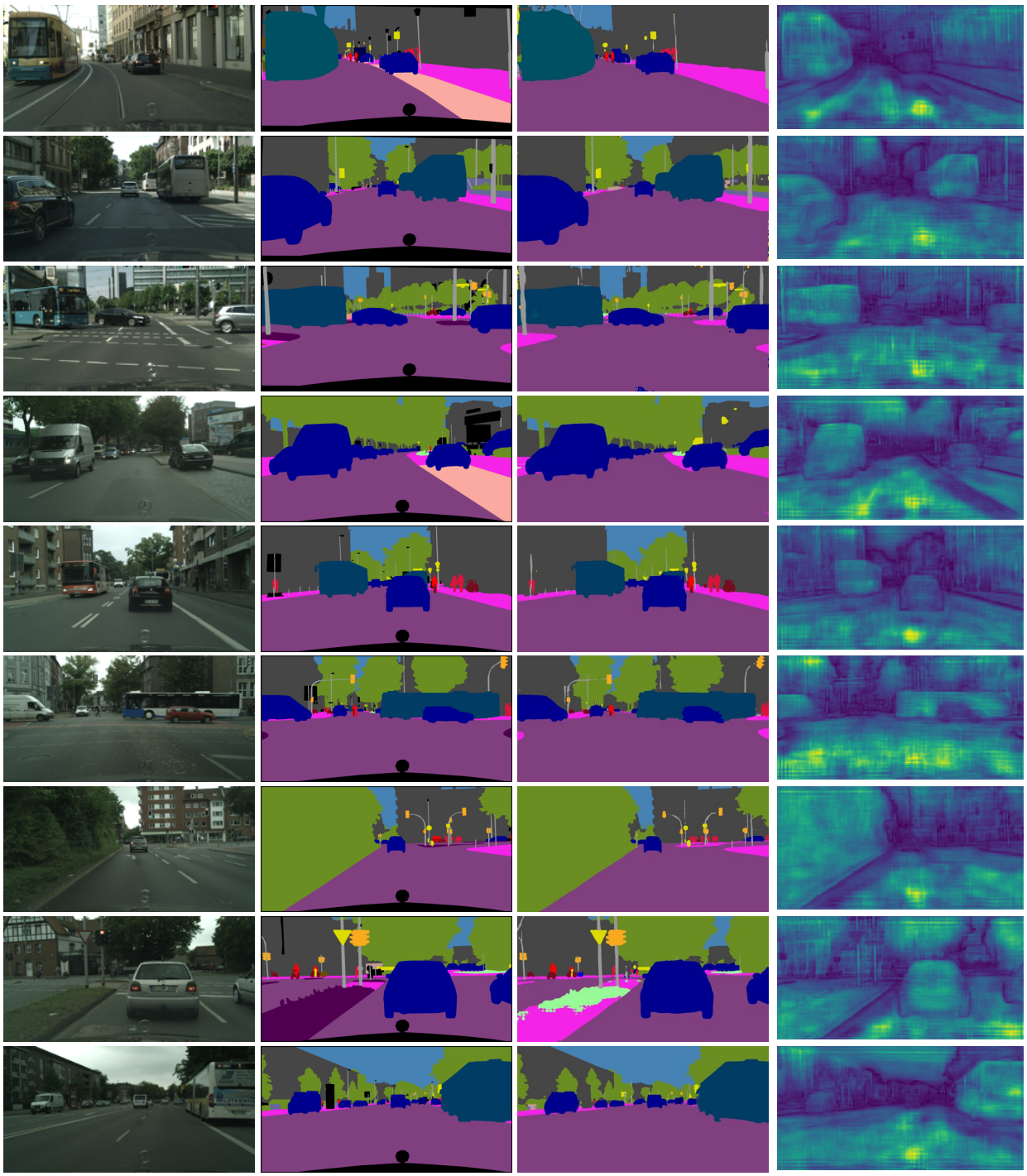


Figure 5.8: Qualitative results for dynamic inference of scale: (a) input images; (b) truths; (c) outputs; and (d) scale estimates with small visualized as blue/dark and large visualized as yellow/bright. The scale estimates have a certain amount of structure: coherent segments and boundaries between them can be seen.

Chapter 6

Conclusion

Fully convolutional networks are local machines for image-to-image learning and inference. By learning end-to-end and pixels-to-pixels they serve as a general framework for such tasks. Following the release of this framework, there has been a broad convergence in approach, not only for semantic segmentation [Com], but for a variety of tasks. By harnessing locality, fully convolutional networks make deep learning for image-to-image tasks not only tractable but attractive in their effectiveness and efficiency.

Blurring the line between structure and learning affords optimization of not just local models, but even the degree of locality. Composing structured Gaussian filters with free-form filters, and learning both, optimizes over receptive field size. Dynamic inference adapts receptive fields to cope with scale variation. By factorizing the representation into free-form and structured parts, we define strictly more general models, which learn more and not less.

Rather than choose learning or structure, we can hope to reconcile both by composition. Our best models do not perceive by parameters alone: convolution, residual connections, certain receptive field geometries, and myriad other elements cannot be practically learned or even represented without structure. Nor do our best models inspect by invariances alone: there is more in the visual world than is dreamt of in our designs, and closing the gap is a role for learning. Bridging old and new ideas in representation design and representation learning, we can see to it that progress takes the path of a spiral, not a circle.

Future Directions Of course, more research is needed. We point out three directions for locality and adaptivity to go from here.

- Locality in generality. We focused on spatial locality in the visual domain for supervised learning tasks. However, locality is much more general. Temporal locality could be learned and adapted for time series modeling. Spectral locality, in frequency, could be learned and adapted for audio, alongside temporal locality. Spatio-temporal locality for video and temporal-spectral locality for audio could be learned jointly but distinctly through diagonal covariance. Apart from type and domain, locality could be tuned to different families of learning task. The majority of architectures for unsupervised

and self-supervised learning are adopted from supervised learning, but it could be the case that they are suboptimal for these objectives. By optimizing over locality, the representations learned from these objectives might prove more useful downstream, while still being transferrable through fine-tuning.

- Learning more structure. Together, receptive field size and stride determine the locality of a network. While we explored learning receptive field size, we did not attempt to learn stride. It might be possible to learn stride by parameterizing the sampling by a periodic function and optimizing end-to-end. This would in effect decide the resolution of the model through data.

- Adaptivity by optimization during inference. The adaptivity in this work was either designed, for clockwork, or one-step and feedforward, for scale and guidance. While this improves over fixed, non-adaptive inference it is nevertheless limited. For scale, our convolutional scale predictor is limited as a local filter, and not itself immune to scale variation. For guidance, the latent task representation extracted is limited to averaging in a fixed embedding, and not further refined. Gradient optimization during inference offers an avenue to multi-step adaptivity. This directions requires choosing what unsupervised objective to optimize and which variables to optimize over. Minimizing output entropy with respect to scale could take advantage of our factorization to improve task confidence through structure. Such a model would tune itself, lowering the boundary between learning and inference.

Appendix A

Brewing Caffe

Caffe was both a project and a community.
Thanks to everyone who contributed to the code[1], to the conversation[2], and in a wealth of other ways. I would like to share a few thoughtful sips of the experience, the last from the bottom of the coffee cup.

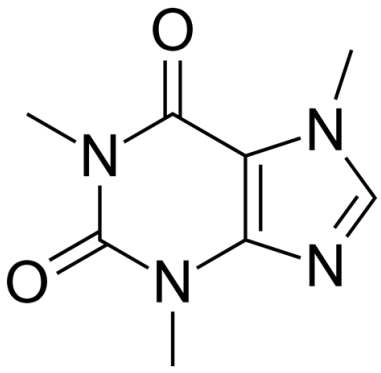


Figure A.1: Caffeine.

- *The ride never stops.* The (frame)work is never done. Decide what to do, and what not to do, deliberately. Be honest with yourself about the purpose of your work, and recognize progress from procrastination.
- *Do it yourself, for yourself.* Caffe was the engine of my own projects. Make your own progress, for satisfaction and sustainability, or risk exhaustion.
- *Selflessness is self interest.* No matter who you are, the world can do more research. If you can open up your projects to others, without losing your own science to service, you multiply your own progress.

Each of us can locally convolve, making sense of input for research output, but open science and community lets us globally integrate, making sure to have impact on the world.

Happy brewing!

[1] https://github.com/BVLC/caffe
[2] https://groups.google.com/forum/#!forum/caffe-users

Appendix B

Brewing Coffee

There is always more science to be done, and ever more research is needed. But first: **coffee**. Here I would like to share the recipe for the communal coffee that I have brewed many times during my PhD as our BAIRista.

What

- *Brewer.* Yama tower. 25 cup. Ideally rigged up at one's desk to appreciate the process.
- *Filters.* Multi-use ceramic filter & single-use paper filter.
- *Grinder.* Baratza Virtuoso. Excellent and reliable.
- *Coffee.* One pound. Light enough roast. Prefer floral, natural process coffees for this purpose, but remember that variety is nice.
- *Water.* 2.5 liters. Clear and cool.
- *Time.*

Figure B.1: The brewer.

How

1. Measure out the water and fill the reservoir, making sure to first close the valve.
2. Fit ceramic filter in the bottom of the middle chamber.
3. Grind the coffee at the eighth notch from finest. Pour into the chamber such that the top is level, but do not shake to distribute it evenly.

4. Set paper filter on the coffee, and gently press flat.

5. Wet the coffee by opening the valve and letting enough
   water flow to cover the the chamber with 1 inch of water.
   Close valve, then slowly tilt chamber in a circle to distribute.

6. Brew: adjust the valve to a drip rate of 40 drips/minute
   or one drip every 1.5 seconds.

7. Take time. Breathe. Do honest work. Sleep.
   Brew for 12 hours.

8. Bottle and keep cool. Old-timey, swing cap bottles are best.

9. Pour yourself a glass and share.

Adjustment to other quantities, equipment, and life choices is left as an exercise to the brewer. Experimentation is key, both in empirical science and in coffee.

Happy brewing!

Bibliography

[BA83] P. Burt and E. Adelson. "The Laplacian pyramid as a compact image code". In: *Communications, IEEE Transactions on* 31.4 (1983), pp. 532–540.

[Bab+86] J Babaud et al. "Uniqueness of the Gaussian Kernel for Scale-Space Filtering". In: *TPAMI* (1986).

[Bea+16] Amy Bearman et al. "What's the point: Semantic segmentation with point supervision". In: *ECCV*. 2016.

[Ber+16] Luca Bertinetto et al. "Learning feed-forward one-shot learners". In: *NIPS*. 2016.

[Bis06] Christopher M Bishop. "Pattern recognition and machine learning". In: Springer-Verlag New York, 2006, p. 229.

[BJ01] Yuri Y Boykov and M-P Jolly. "Interactive graph cuts for optimal boundary & region segmentation of objects in ND images". In: *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*. Vol. 1. IEEE. 2001, pp. 105–112.

[BM13] Joan Bruna and Stéphane Mallat. "Invariant scattering convolution networks". In: *TPAMI* (2013).

[Cae+17] Sergi Caelles et al. "One-Shot Video Object Segmentation". In: *CVPR*. 2017.

[Car+12] J. Carreira et al. "Semantic Segmentation with Second-Order Pooling". In: *ECCV*. 2012.

[Che+15] Liang-Chieh Chen et al. "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs". In: *ICLR*. 2015.

[Che+18a] Liang-Chieh Chen et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *PAMI* (2018).

[Che+18b] Yuhua Chen et al. "Blazingly Fast Video Object Segmentation with Pixel-Wise Metric Learning". In: *CVPR*. 2018, pp. 1189–1198.

[CHL05] Sumit Chopra, Raia Hadsell, and Yann LeCun. "Learning a similarity metric discriminatively, with application to face verification". In: *CVPR*. Vol. 1. IEEE. 2005, pp. 539–546.

[Cir+12] Dan C Ciresan et al. "Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images." In: *NIPS*. 2012, pp. 2852–2860.

[Com] *PASCAL VOC Segmentation Results: comp6 2012*. Accessed 01 May 2019. 2019. URL: http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?challengeid=11\&compid=6.

[Cor+16] Marius Cordts et al. "The cityscapes dataset for semantic urban scene understanding". In: *CVPR*. 2016.

[CW17] Taco S Cohen and Max Welling. "Steerable CNNs". In: *ICLR*. 2017.

[Dai+17] Jifeng Dai et al. "Deformable Convolutional Networks". In: *ICCV*. 2017.

[DB+16] Bert De Brabandere et al. "Dynamic Filter Networks". In: *NIPS*. 2016.

[Den+14] Emily L Denton et al. "Exploiting linear structure within convolutional networks for efficient evaluation". In: *NIPS*. 2014.

[DHS15a] J. Dai, K. He, and J. Sun. "BoxSup: Exploiting Bounding Boxes to Supervise Convolutional Networks for Semantic Segmentation". In: *ICCV*. 2015.

[DHS15b] Jifeng Dai, Kaiming He, and Jian Sun. "Convolutional Feature Masking for Joint Object and Stuff Segmentation". In: *CVPR*. 2015.

[Din+18] Henghui Ding et al. "Context contrasted feature and gated multi-scale aggregation for scene segmentation". In: *CVPR*. 2018.

[Don+14] Jeff Donahue et al. "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition". In: *ICML*. 2014.

[Don+15] Jeffrey Donahue et al. "Long-term recurrent convolutional networks for visual recognition and description". In: *CVPR*. 2015, pp. 2625–2634.

[EF15] David Eigen and Rob Fergus. "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture". In: *ICCV*. 2015.

[EKF13] David Eigen, Dilip Krishnan, and Rob Fergus. "Restoring an Image Taken through a Window Covered with Dirt or Rain". In: *ICCV*. 2013, pp. 633–640.

[EPF14] David Eigen, Christian Puhrsch, and Rob Fergus. "Depth Map Prediction from a Single Image using a Multi-Scale Deep Network". In: *NIPS*. 2014.

[Eve+10] Mark Everingham et al. "The pascal visual object classes (voc) challenge". In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[FA91] William T. Freeman and Edward H Adelson. "The design and use of steerable filters". In: *TPAMI* (1991).

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *arXiv preprint arXiv:1703.03400* (2017).

[Far+13] Clément Farabet et al. "Learning hierarchical features for scene labeling". In: *PAMI* (2013).

[FDB14] Philipp Fischer, Alexey Dosovitskiy, and Thomas Brox. "Descriptor Matching with Convolutional Neural Networks: a Comparison to SIFT". In: *arXiv preprint arXiv:1405.5769* (2014).

[FFFP06] Li Fei-Fei, Rob Fergus, and Pietro Perona. "One-shot learning of object categories". In: *PAMI* (2006).

[Fis+15] Philipp Fischer et al. "Learning Optical Flow with Convolutional Networks". In: *ICCV*. 2015.

[Fra+15] Katerina Fragkiadaki et al. "Learning to Segment Moving Objects in Videos". In: *CVPR*. 2015.

[Fuk80] Kunihiko Fukushima. "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". In: *Biological cybernetics* 36.4 (1980), pp. 193–202.

[GAM13] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. "Perceptual Organization and Recognition of Indoor Scenes from RGB-D Images". In: *CVPR*. 2013.

[GB18] Victor Garcia and Joan Bruna. "Few-Shot Learning with Graph Neural Networks". In: *ICLR*. 2018.

[GG+17] Alberto Garcia-Garcia et al. "A review on deep learning techniques applied to semantic segmentation". In: *arXiv preprint arXiv:1704.06857* (2017).

[Gir+15] Ross Girshick et al. "Region-based Convolutional Networks for Accurate Object Detection and Segmentation". In: *PAMI* (2015).

[Giu+13] Alessandro Giusti et al. "Fast image scanning with deep max-pooling convolutional neural networks". In: *ICIP*. 2013.

[GL14] Yaroslav Ganin and Victor Lempitsky. "N^4-Fields: Neural Network Nearest Neighbor Fields for Image Transforms". In: *ACCV*. 2014.

[Gru+10] Matthias Grundmann et al. "Efficient hierarchical graph-based video segmentation". In: *CVPR*. IEEE. 2010, pp. 2141–2148.

[Gup+14] Saurabh Gupta et al. "Learning Rich Features from RGB-D Images for Object Detection and Segmentation". In: *ECCV*. 2014.

[Har+11] Bharath Hariharan et al. "Semantic Contours from Inverse Detectors". In: *ICCV*. 2011.

[Har+12] Glenn Hartmann et al. "Weakly supervised learning of object segmentations from web-scale video". In: *ECCV-W*. Springer. 2012, pp. 198–208.

[Har+14] Bharath Hariharan et al. "Simultaneous Detection and Segmentation". In: *ECCV*. 2014.

[Har+15] Bharath Hariharan et al. "Hypercolumns for Object Segmentation and Fine-grained Localization". In: *CVPR*. 2015.

[HCL06] Raia Hadsell, Sumit Chopra, and Yann LeCun. "Dimensionality reduction by learning an invariant mapping". In: *CVPR*. Vol. 2. IEEE. 2006, pp. 1735–1742.

[He+14] Kaiming He et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *ECCV*. 2014.

[He+16] Kaiming He et al. "Deep residual learning for image recognition". In: *CVPR*. 2016.

[HNH15] Seunghoon Hong, Hyeonwoo Noh, and Bohyung Han. "Decoupled deep neural network for semi-supervised semantic segmentation". In: *NIPS*. 2015, pp. 1495–1503.

[Hol+89] Matthias Holschneider et al. "A real-time algorithm for signal analysis with the help of the wavelet transform". In: (1989), pp. 286–297.

[How13] Andrew G Howard. "Some improvements on deep convolutional neural network based image classification". In: *arXiv preprint arXiv:1312.5402* (2013).

[HS15] Kaiming He and Jian Sun. "Convolutional neural networks at constrained time cost". In: *CVPR*. 2015.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[Jac+16] Jorn-Henrik Jacobsen et al. "Structured Receptive Fields in CNNs". In: *CVPR*. 2016.

[Jad+15] Max Jaderberg et al. "Spatial Transformer Networks". In: *NIPS*. 2015.

[JG14] Suyog Dutt Jain and Kristen Grauman. "Supervoxel-Consistent Foreground Propagation in Video". In: *ECCV*. 2014.

[Ji+13] Shuiwang Ji et al. "3D convolutional neural networks for human action recognition". In: *PAMI* 35.1 (2013), pp. 221–231.

[Jia+14] Yangqing Jia et al. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *Proceedings of the ACM International Conference on Multimedia*. MM '14. New York, NY, USA: ACM, 2014, pp. 675–678.

[JVZ14] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. "Speeding up Convolutional Neural Networks with Low Rank Expansions". In: *BMVC*. 2014.

[Kan+18] Kirthevasan Kandasamy et al. "Neural architecture search with bayesian optimisation and optimal transport". In: *NIPS*. 2018.

[Kar+14] Andrej Karpathy et al. "Large-scale video classification with convolutional neural networks". In: *CVPR*. 2014, pp. 1725–1732.

[KD87] Jan J Koenderink and Andrea J van Doorn. "Representation of local geometry in the visual system". In: *Biological cybernetics* 55.6 (1987), pp. 367–375.

[Koe84] Jan J Koenderink. "The structure of images". In: *Biological cybernetics* 50.5 (1984), pp. 363–370.

[Kou+14] Jan Koutník et al. "A Clockwork RNN". In: *ICML*. 2014.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." In: *NIPS*. 2012.

[KSJ14] Angjoo Kanazawa, Abhishek Sharma, and David Jacobs. "Locally scale-invariant convolutional neural networks". In: *arXiv preprint arXiv:1412.5104* (2014).

[Kuf53] Stephen W Kuffler. "Discharge patterns and functional organization of mammalian retina". In: *Journal of neurophysiology* 16.1 (1953), pp. 37–68.

[KWT88] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. "Snakes: Active contour models". In: *International journal of computer vision* 1.4 (1988), pp. 321–331.

[KZS15] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition". In: *ICML Deep Learning Workshop*. Vol. 2. 2015.

[Lap05] Ivan Laptev. "On space-time interest points". In: *IJCV* 64.2-3 (2005), pp. 107–123.

[LeC+98a] Y. LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[LeC+98b] Y. LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[LeC+98c] Yann A LeCun et al. "Efficient backprop". In: *Neural networks: Tricks of the trade*. 1998, pp. 9–48.

[Lin+14] Tsung-Yi Lin et al. "Microsoft COCO: Common objects in context". In: *ECCV*. 2014, pp. 740–755.

[Lin+17] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection". In: *CVPR*. 2017.

[Lin94] Tony Lindeberg. *Scale-space theory in computer vision*. Vol. 256. Springer Science & Business Media, 1994.

[Lin98] Tony Lindeberg. "Feature detection with automatic scale selection". In: *International journal of computer vision* 30.2 (1998), pp. 79–116.

[Liu+14] Xiao Liu et al. "Weakly Supervised Multiclass Video Segmentation". In: *CVPR*. IEEE. 2014, pp. 57–64.

[Liu+15] Fayao Liu et al. "Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields". In: *PAMI* (2015).

[Low04] D.G. Lowe. "Distinctive image features from scale-invariant keypoints". In: *IJCV* (2004).

[LRB15] Wei Liu, Andrew Rabinovich, and Alexander C Berg. "ParseNet: Looking Wider to See Better". In: *arXiv preprint arXiv:1506.04579* (2015).

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *CVPR* (2015).

[LST15] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. "Human-level concept learning through probabilistic program induction". In: *Science* (2015).

[LSY19] Hanxiao Liu, Karen Simonyan, and Yiming Yang. "Darts: Differentiable architecture search". In: *ICLR*. 2019.

[Luo+16] Wenjie Luo et al. "Understanding the Effective Receptive Field in Deep Convolutional Neural Networks". In: *NIPS*. 2016.

[LYT11] Ce Liu, Jenny Yuen, and Antonio Torralba. "Sift flow: Dense correspondence across scenes and its applications". In: *PAMI* 33.5 (2011), pp. 978–994.

[LZD14] Jonathan Long, Ning Zhang, and Trevor Darrell. "Do Convnets Learn Correspondence?" In: *NIPS*. 2014.

[Mal99] Stéphane Mallat. *A wavelet tour of signal processing*. 2nd. Academic press, 1999.

[Man+18] Kevis-Kokitsi Maninis et al. "Deep extreme cut: From extreme points to object segmentation". In: *CVPR*. 2018.

[Mat+91] Ofer Matan et al. "Multi-digit recognition using a space displacement neural network". In: *NIPS*. 1991, pp. 488–495.

[Mot+14] Roozbeh Mottaghi et al. "The role of context for object detection and semantic segmentation in the wild". In: *CVPR*. 2014, pp. 891–898.

[Muy82] Eadweard Muybridge. *The Horse In Motion*. Library of Congress Prints and Photographs Division. 1882. URL: http://hdl.loc.gov/loc.pnp/cph.3a45870.

[MYS15] Mohammadreza Mostajabi, Payman Yadollahpour, and Gregory Shakhnarovich. "Feedforward semantic segmentation with zoom-out features". In: *CVPR*. 2015.

[NHH15] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. "Learning Deconvolution Network for Semantic Segmentation". In: *ICCV*. 2015.

[Nin+05] Feng Ning et al. "Toward automatic phenotyping of developing embryos from videos". In: *Image Processing, IEEE Transactions on* 14.9 (2005), pp. 1360–1371.

[OAVE93] Bruno A Olshausen, Charles H Anderson, and David C Van Essen. "A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information". In: *Journal of Neuroscience* 13.11 (1993), pp. 4700–4719.

[OS09] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. 3rd. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009.

[Pap+15] George Papandreou et al. "Weakly-and Semi-supervised learning of a DCNN for semantic image segmentation". In: *ICCV*. 2015.

[PB96] José C Pinheiro and Douglas M Bates. "Unconstrained parametrizations for variance-covariance matrices". In: *Statistics and computing* 6.3 (1996), pp. 289–296.

[PC14] Pedro HO Pinheiro and Ronan Collobert. "Recurrent Convolutional Neural Networks for Scene Labeling". In: *ICML*. 2014.

[Per95] Pietro Perona. "Deformable kernels for early vision". In: *TPAMI* (1995).

[PF13] Anestis Papazoglou and Vittorio Ferrari. "Fast Object Segmentation in Unconstrained Video". In: *ICCV*. 2013.

[PKD15] Deepak Pathak, Philipp Krähenbühl, and Trevor Darrell. "Constrained Convolutional Neural Networks for Weakly Supervised Segmentation". In: *ICCV*. 2015.

[Pre+12] Alessandro Prest et al. "Learning object class detectors from weakly annotated video". In: *CVPR*. IEEE. 2012, pp. 3282–3289.

[PT+17] Jordi Pont-Tuset et al. "The 2017 DAVIS Challenge on Video Object Segmentation". In: *arXiv:1704.00675* (2017).

[Rak+18] Kate Rakelly et al. "Few-Shot Segmentation Propagation with Guided Networks". In: *arXiv preprint arXiv:1806.07373* (2018).

[RBPK18] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. "In-place activated batchnorm for memory-optimized training of dnns". In: *CVPR*. 2018.

[Ren+15] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *NIPS*. 2015.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *MICCAI*. 2015.

[RL17] Sachin Ravi and Hugo Larochelle. "Optimization as a model for few-shot learning". In: *ICLR*. 2017.

[RS65] Robert W Rodieck and Jonathan Stone. "Analysis of receptive fields of cat retinal ganglion cells". In: *Journal of neurophysiology* 28.5 (1965), pp. 833–849.

[Rus+15] Olga Russakovsky et al. "Imagenet large scale visual recognition challenge". In: *IJCV* 115.3 (2015), pp. 211–252.

[San+16] Adam Santoro et al. "Meta-learning with memory-augmented neural networks". In: *ICML*. 2016.

[Ser+13] Pierre Sermanet et al. "Pedestrian Detection with Unsupervised Multi-Stage Feature Learning". In: *CVPR*. 2013.

[Ser+14] Pierre Sermanet et al. "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks". In: *ICLR*. 2014.

[SF95] Eero P Simoncelli and William T Freeman. "The steerable pyramid: A flexible architecture for multi-scale derivative computation". In: *ICIP*. 1995.

[Sha+17] Amirreza Shaban et al. "One-Shot Learning for Semantic Segmentation". In: *BMVC*. 2017.

[She+16] Evan Shelhamer et al. "Clockwork convnets for video semantic segmentation". In: *ECCV Workshops*. 2016, pp. 852–868.

[Sil+12] Nathan Silberman et al. "Indoor Segmentation and Support Inference from RGBD Images". In: *ECCV*. 2012.

[SLD17] Evan Shelhamer, Jonathan Long, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *PAMI* (2017).

[SM13] Laurent Sifre and Stéphane Mallat. "Rotation, scaling and deformation invariant scattering for texture discrimination". In: *CVPR*. 2013.

[SM98] Jianbo Shi and Jitendra Malik. "Motion segmentation and tracking using normalized cuts". In: *ICCV*. IEEE. 1998, pp. 1154–1160.

[SSZ17] Jake Snell, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4080–4090.

[SWD19] Evan Shelhamer, Dequan Wang, and Trevor Darrell. "Blurring the Line Between Structure and Learning to Optimize and Adapt Receptive Fields". In: *arXiv preprint arXiv:1904.11487* (2019).

[SZ15] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *ICLR*. 2015.

[Sze+15] Christian Szegedy et al. "Going Deeper with Convolutions". In: *CVPR*. 2015.

[Tan+13] Ke Tang et al. "Discriminative segment annotation in weakly labeled video". In: *CVPR*. IEEE. 2013, pp. 2483–2490.

[TL10] Joseph Tighe and Svetlana Lazebnik. "Superparsing: scalable nonparametric image parsing with superpixels". In: *ECCV*. 2010, pp. 352–365.

[TL13] Joseph Tighe and Svetlana Lazebnik. "Finding Things: Image Parsing with Regions and Per-Exemplar Detectors". In: *CVPR*. 2013.

[Tom+14] Jonathan Tompson et al. "Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation". In: *NIPS*. 2014.

[Vai90] Parishwad P Vaidyanathan. "Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial". In: *Proceedings of the IEEE* 78.1 (1990), pp. 56–93.

[Vin+16] Oriol Vinyals et al. "Matching networks for one shot learning". In: *NIPS*. 2016, pp. 3630–3638.

[VSM11] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao. "Improving the speed of neural networks on CPUs". In: *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*. Vol. 1. 2011.

[Wan+13] Li Wan et al. "Regularization of neural networks using dropconnect". In: *ICML*. 2013, pp. 1058–1066.

[Wan+18] Panqu Wang et al. "Understanding convolution for semantic segmentation". In: *WACV*. 2018.

[WH16] Yu-Xiong Wang and Martial Hebert. "Learning to learn: Model regression networks for easy small sample learning". In: *ECCV*. 2016.

[WP94] Ralph Wolf and John C Platt. "Postal address block location using a convolutional locator network". In: *NIPS*. 1994, pp. 745–745.

[WS02] Laurenz Wiskott and Terrence J Sejnowski. "Slow feature analysis: Unsupervised learning of invariances". In: *Neural computation* 14.4 (2002), pp. 715–770.

[XC12] Chenliang Xu and Jason J Corso. "Evaluation of super-voxel methods for early video processing". In: *CVPR*. IEEE. 2012, pp. 1202–1209.

[XT15] Saining Xie and Zhuowen Tu. "Holistically-Nested Edge Detection". In: *ICCV*. 2015.

[Xu+16] Ning Xu et al. "Deep interactive object selection". In: *CVPR*. 2016, pp. 373–381.

[Yan+15] Zichao Yang et al. "Deep Fried Convnets". In: *ICCV*. 2015.

[Yeu+16] Serena Yeung et al. "End-to-end Learning of Action Detection from Frame Glimpses in Videos". In: *CVPR*. 2016.

[YK16] Fisher Yu and Vladlen Koltun. "Multi-Scale Context Aggregation by Dilated Convolutions". In: *ICLR*. 2016.

[YKF17] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. "Dilated residual networks". In: *CVPR*. 2017.

[Yoo+17] Jae Shin Yoon et al. "Pixel-Level Matching for Video Object Segmentation using Convolutional Neural Networks". In: *ICCV*. 2017, pp. 2186–2195.

[Yu+18] Fisher Yu et al. "Deep layer aggregation". In: *CVPR*. 2018.

[ZF14] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *ECCV*. 2014, pp. 818–833.

[Zha+14] Ning Zhang et al. "Part-based R-CNNs for fine-grained category detection". In: *ECCV*. 2014, pp. 834–849.

[Zha+17a] Rui Zhang et al. "Scale-adaptive convolutions for scene parsing". In: *ICCV*. 2017, pp. 2031–2039.

[Zha+17b] Hengshuang Zhao et al. "Pyramid scene parsing network". In: *CVPR*. 2017.

[Zha19] Richard Zhang. "Making Convolutional Networks Shift-Invariant Again". In: (2019).

[Zhe+15] Shuai Zheng et al. "Conditional Random Fields as Recurrent Neural Networks". In: *ICCV*. 2015.

[Zhu+17] Jun-Yan Zhu et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *arXiv preprint arXiv:1703.10593* (2017).

[ZIE16] Richard Zhang, Phillip Isola, and Alexei A Efros. "Colorful Image Colorization". In: *ECCV*. 2016.

[ZL17] Barret Zoph and Quoc V Le. "Neural architecture search with reinforcement learning". In: *ICLR*. 2017.