This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2572683, IEEE Transactions on Pattern Analysis and Machine Intelligence

Fully Convolutional Networks for Semantic Segmentation

Evan Shelhamer*, Jonathan Long*, and Trevor Darrell, *Member, IEEE*

Abstract—Convolutional networks are powerful visual models that yield hierarchies of features. We show that convolutional networks by themselves, trained end-to-end, pixels-to-pixels, improve on the previous best result in semantic segmentation. Our key insight is to build "fully convolutional" networks that take input of arbitrary size and produce correspondingly-sized output with efficient inference and learning. We define and detail the space of fully convolutional networks, explain their application to spatially dense prediction tasks, and draw connections to prior models. We adapt contemporary classification networks (AlexNet, the VGG net, and GoogLeNet) into fully convolutional networks and transfer their learned representations by fine-tuning to the segmentation task. We then define a skip architecture that combines semantic information from a deep, coarse layer with appearance information from a shallow, fine layer to produce accurate and detailed segmentations. Our fully convolutional network achieves improved segmentation of PASCAL VOC (30% relative improvement to 67.2% mean IU on 2012), NYUDv2, SIFT Flow, and PASCAL-Context, while inference takes one tenth of a second for a typical image.

Index Terms—Semantic Segmentation, Convolutional Networks, Deep Learning, Transfer Learning

1 Introduction

Convolutional networks are driving advances in recognition. Convnets are not only improving for whole-image classification [1], [2], [3], but also making progress on local tasks with structured output. These include advances in bounding box object detection [4], [5], [6], part and keypoint prediction [7], [8], and local correspondence [8], [9].

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 [10], [11], [12], [13], [14], [15], [16], 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 [10], [11], [12], [13], [16], but lacks the efficiency of fully convolutional training. Our approach does not make use of pre- and post-processing complications, including superpixels [12], [14], proposals [14], [15], or post-hoc refinement by random fields or local classifiers [12], [14]. Our model transfers recent success in classification [1], [2], [3] 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 [10], [12], [13].

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

In the conference version of this paper [17], 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 4.3 and Figure 3). In this light, deep feature hierarchies encode location and semantics in a nonlinear local-to-global pyramid.

This journal paper extends our earlier work [17] through further tuning, analysis, and more results. 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 fol-

*Authors contributed equally

- *E. Shelhamer, J. Long, and T. Darrell are with the Department of Electrical Engineering and Computer Science (CS Division), UC Berkeley. E-mail: {shelhamer,jonlong,trevor}@cs.berkeley.edu.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2572683, IEEE Transactions on Pattern Analysis and Machine Intelligence

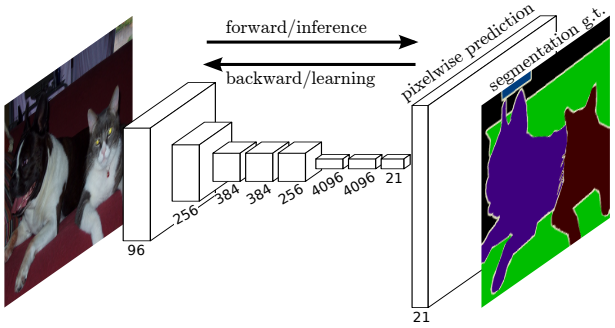

Fig. 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

lowing 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 Related Work

Our approach draws on recent successes of deep nets for image classification [1], [2], [3] and transfer learning [18], [19]. Transfer was first demonstrated on various visual recognition tasks [18], [19], then on detection, and on both instance and semantic segmentation in hybrid proposal-classifier models [5], [14], [15]. 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*. [20], which extended the classic LeNet [21] 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 [22] 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*. [10] 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*. [4], semantic segmentation by Pinheiro and Collobert [13], and image restoration by Eigen *et al*. [23] do fully convolutional inference. Fully convolutional training is rare, but used effectively by Tompson *et al*. [24] 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*. [10], Farabet *et al*. [12], and Pinheiro and Collobert [13]; boundary prediction for electron microscopy by Ciresan *et al*. [11] and for natural images by a hybrid convnet/nearest neighbor model by Ganin and Lempitsky [16]; and image restoration and depth estimation by Eigen *et al*. [23], [25]. Common elements of these approaches include

- small models restricting capacity and receptive fields;
- patchwise training [10], [11], [12], [13], [16];
- refinement by superpixel projection, random field regularization, filtering, or local classification [11], [12], [16];
- "interlacing" to obtain dense output [4], [13], [16];
- multi-scale pyramid processing [12], [13], [16];
- saturating $\tanh$ nonlinearities [12], [13], [23]; and
- ensembles [11], [16],

whereas our method does without this machinery. However, we do study patchwise training (Section 3.4) and "shift-and-stitch" dense output (Section 3.2) from the perspective of FCNs. We also discuss in-network upsampling (Section 3.3), of which the fully connected prediction by Eigen *et al*. [25] 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*. [14] and Gupta *et al*. [15] likewise adapt deep classification nets to semantic segmentation, but do so in hybrid proposal-classifier models. These approaches fine-tune an R-CNN system [5] 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 5.

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 [26] is a classic multi-scale representation made of fixed smoothing and differencing. The jet of Koenderink and van Doorn [27] is a rich, local feature defined by compositions of partial derivatives. In the context of deep networks, Sermanet *et al*. [28] fuse intermediate layers but discard resolution in doing so. In contemporary work Hariharan *et al*. [29] and Mostajabi *et al*. [30] also fuse multiple layers but do not learn end-to-end and rely on fixed bottom-up grouping.

FCN extensions Following the conference version of this paper [17], FCNs have been extended to new tasks and data. Tasks include region proposals [31], contour detection [32], depth regression [33], optical flow [34], and weakly-supervised semantic segmentation [35], [36], [37], [38].

In addition, new works have improved the FCNs presented here to further advance the state-of-the-art in semantic segmentation. The DeepLab models [39] raise output resolution by dilated convolution and dense CRF inference. The joint CRFasRNN [40] model is an end-to-end integration of the CRF for further improvement. ParseNet [41] normalizes features for fusion and captures context with global pooling. The "deconvolutional network" approach of [42] restores resolution by proposals, stacks of learned deconvolution, and unpooling. U-Net [43] combines skip layers and learned deconvolution for pixel labeling of microscopy images. The dilation architecture of [44] makes

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2572683, IEEE Transactions on Pattern Analysis and Machine Intelligence

thorough use of dilated convolution for pixel-precise output without a random field or skip layers.

3 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 3.2 describes a trick used for this purpose (e.g., by "fast scanning" [45]). We explain this trick in terms of network modification. As an efficient, effective alternative, we upsample in Section 3.3, reusing our implementation of convolution. In Section 3.4 we consider training by patchwise sampling, and give evidence in Section 4.4 that our whole image training is faster and equally effective.

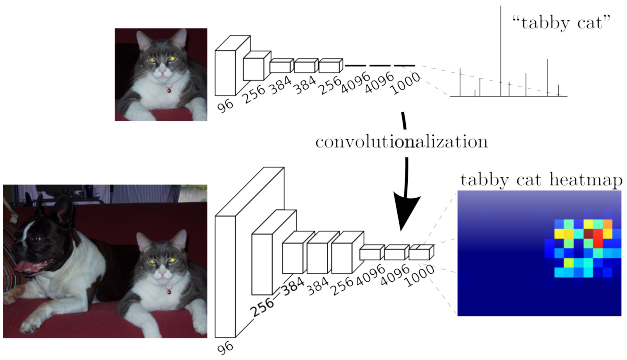

Fig. 2. Transforming fully connected layers into convolution layers enables a classification net to output a spatial map. Adding differentiable interpolation layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end pixelwise learning.

3.1 Adapting classifiers for dense prediction

Typical recognition nets, including LeNet [21], AlexNet [1], and its deeper successors [2], [3], 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.

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 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.

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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2572683, IEEE Transactions on Pattern Analysis and Machine Intelligence

3.2 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 [4], [45]. (This trick is also used in the algorithme à trous [46], [47] for wavelet transforms and related to the Noble identities [48] 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 [44].

3.3 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. [19]), these (convolution) layers are sometimes refered 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.

3.4 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 minibatch 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 [49] 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 [10], [11], [12] and mitigate the spatial correlation of dense patches [13], [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 4.4, and do not find that it yields faster or better convergence for dense prediction. Whole image training is effective and efficient.

4 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 [50]. 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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2572683, IEEE Transactions on Pattern Analysis and Machine Intelligence

TABLE 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

TABLE 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**

4.1 From classifier to dense FCN

We begin by convolutionalizing proven classification architectures as in Section 3. We consider the AlexNet[2] architecture [1] that won ILSVRC12, as well as the VGG nets [2] and the GoogLeNet[3] [3] 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 3.3. Table 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 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 [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. 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.

4.2 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 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 for the comparison of accumulation, online, and high momentum or "heavy" learning (discussed further in Section 6.2).

4.3 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 3.

While fully convolutionalized classifiers fine-tuned to semantic segmentation both recognize and localize, as shown in Section 4.1, 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 4). The stride of the network prediction limits the scale of detail in the upsampled output.

We address this by adding skips [51] 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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2572683, IEEE Transactions on Pattern Analysis and Machine Intelligence

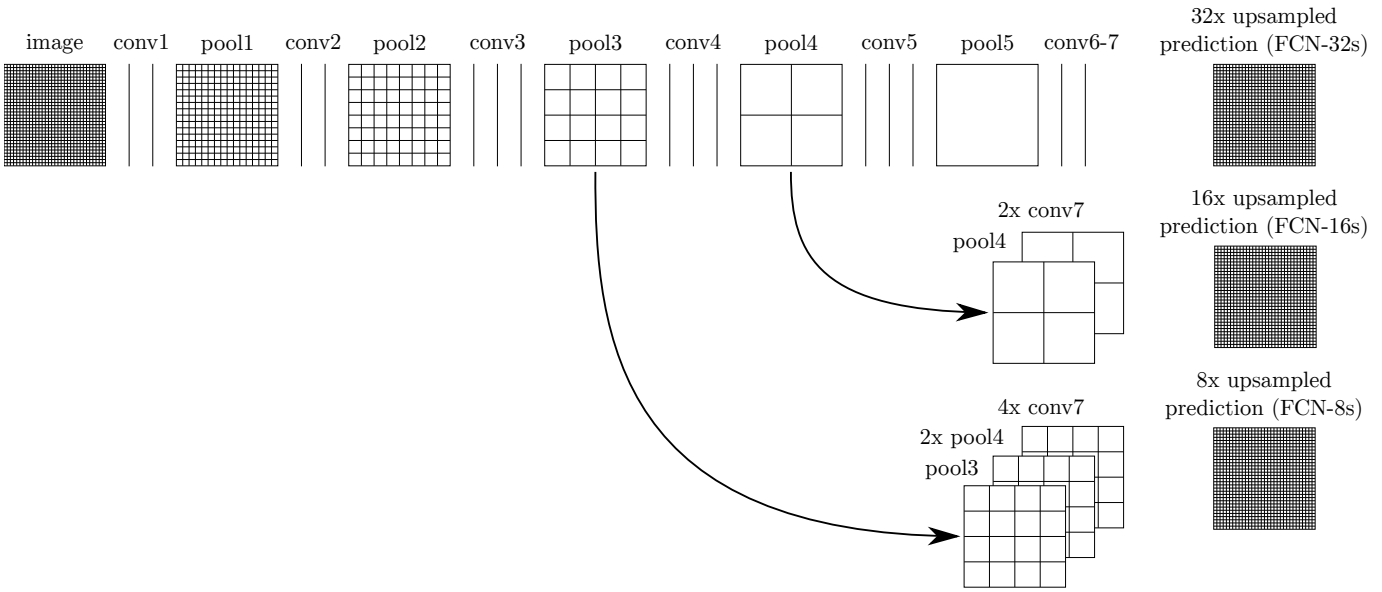

Fig. 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines. First row (FCN-32s): Our single-stream net, described in Section 4.1, 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.

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 [26]. By analogy to the jet of Koenderick and van Doorn [27], 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 3.3. 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 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 3.3. 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 6.3.)

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, so we do not continue fusing even shallower layers.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2572683, IEEE Transactions on Pattern Analysis and Machine Intelligence

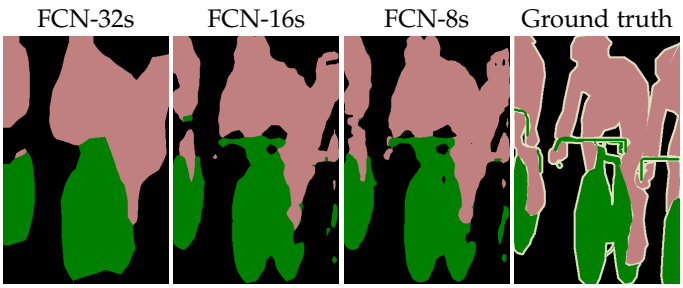

Fig. 4. Refining fully convolutional networks by fusing information from layers with different strides improves spatial detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

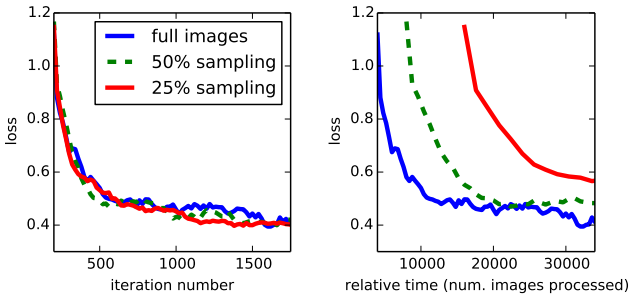

Fig. 5. Training on whole images is just as effective as sampling patches, but results in faster (wall clock time) convergence by making more efficient use of data. Left shows the effect of sampling on convergence rate for a fixed expected batch size, while right plots the same by relative wall clock time.

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 3. Figure 4 shows the progressively finer structure of the output.

TABLE 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

4.4 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 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*. [52] collected labels for a larger set of 8,498 PASCAL training images, which was used to train the previous best system, SDS [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 3.4, 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 [10], [11], [12], [13], [16], potentially resulting in higher variance batches that may accelerate convergence [53]. 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 3.1). Figure 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.

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 [54] on a single NVIDIA Titan X. Our models and code are publicly available at http://fcn.berkeleyvision.org.

5. There are training images from [52] included in the PASCAL VOC 2011 val set, so we validate on the non-intersecting set of 736 images.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2572683, IEEE Transactions on Pattern Analysis and Machine Intelligence

5 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 4.

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 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 [14], and the well-known R-CNN [5]. We achieve the best results on mean IU by 30% relative. Inference time is reduced $114\times$ (convnet only, ignoring proposals and refinement) or $286\times$ (overall).

NYUDv2 [55] 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*. [56]. We report results on the standard split of 795 training images and 654 testing images. Table 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*. [15], 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 6, computed on the standard split into 2,488 training and 200 test images,[6] show better performance on both tasks.

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 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 [5]	47.9	-	-
SDS [14]	52.6	51.6	$\sim$ 50 s
FCN-8s	**67.5**	**67.2**	$\sim$ **100 ms**

TABLE 5
Results on NYUDv2. *RGB-D* is early-fusion of the RGB and depth channels at the input. *HHA* is the depth embedding of [15] 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*. [15]	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 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$_3$ (o^3). The metric for geometry is pixel accuracy.

	pixel acc.	mean acc.	mean IU	f.w. IU	geom. acc.
Liu *et al*. [57]	76.7	-	-	-	-
Tighe *et al*. [58] transfer	-	-	-	-	90.8
Tighe *et al*. [59] SVM	75.6	41.1	-	-	-
Tighe *et al*. [59] SVM+MRF	78.6	39.2	-	-	-
Farabet *et al*. [12] natural	72.3	50.8	-	-	-
Farabet *et al*. [12] balanced	78.5	29.6	-	-	-
Pinheiro *et al*. [13]	77.7	29.8	-	-	-
FCN-8s	**85.9**	**53.9**	41.2	77.2	**94.6**

TABLE 7
Results on PASCAL-Context for the 59 class task. *CFM* is convolutional feature masking [60] and segment pursuit with the VGG net. O_2P is the second order pooling method [61] as reported in the *errata* of [62].

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**

PASCAL-Context [62] provides whole scene annotations of PASCAL VOC 2010. While there are 400+ classes, we follow the 59 class task defined by [62] that picks the most frequent classes. We train and evaluate on the training and val sets respectively. In Table 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%.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2572683, IEEE Transactions on Pattern Analysis and Machine Intelligence

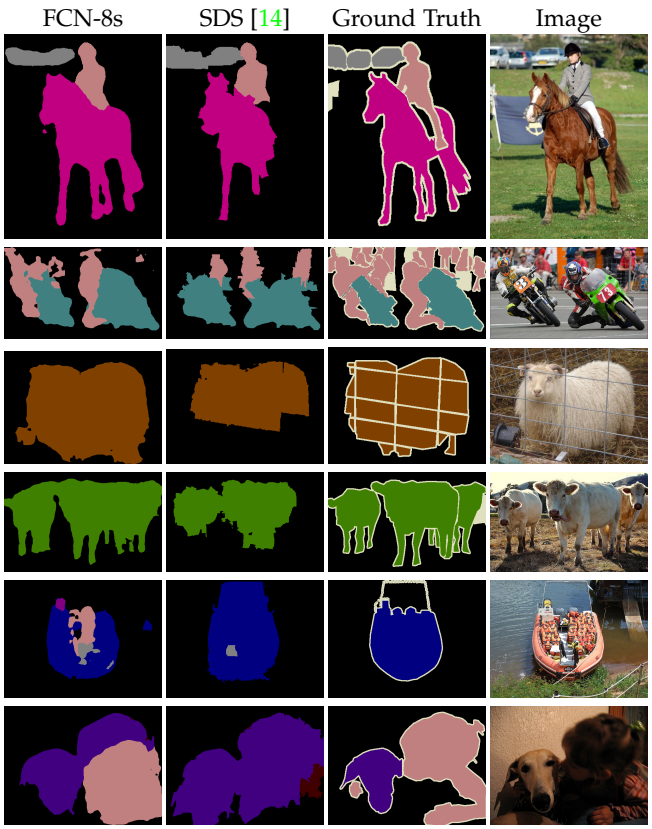

Fig. 6. Fully convolutional networks improve performance on PASCAL. The left column shows the output of our most accurate net, FCN-8s. The second shows the output of the previous best method by Hariharan *et al.* [14]. Notice the fine structures recovered (first row), ability to separate closely interacting 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.

6 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.

6.1 Cues

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?

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 8 for masked results.

TABLE 8
The role of foreground, background, and shape cues. All scores are the mean intersection over union metric *excluding background*. The architecture and optimization are fixed to those of FCN-32s (*Reference*) 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

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 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.

6.2 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 4.2). Here we detail a relationship between momentum and batch size that motivates heavy learning.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPAMI.2016.2572683, IEEE Transactions on Pattern Analysis and Machine Intelligence

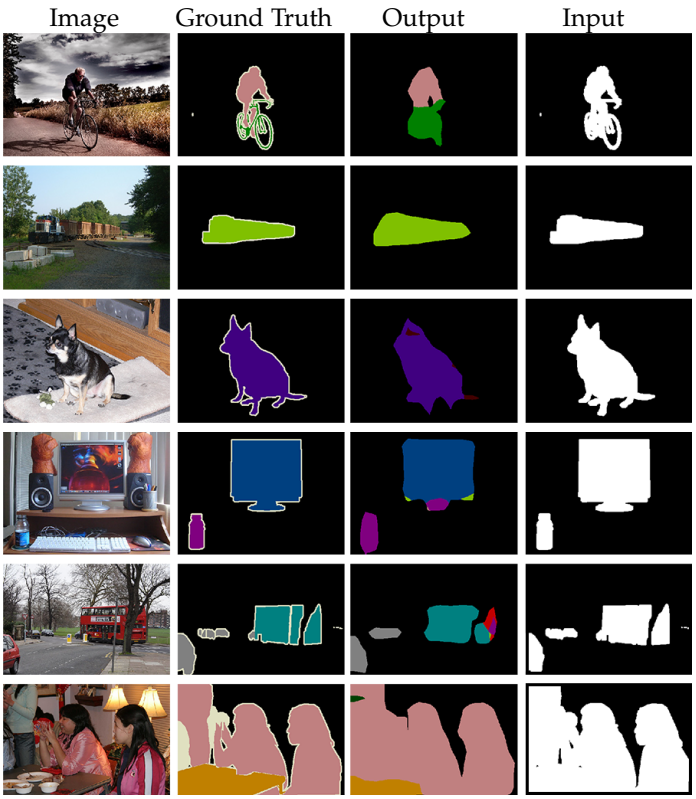

Fig. 7. FCNs learn to recognize by shape when deprived of other input detail. From left to right: regular image (not seen by network), ground truth, output, mask input.

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 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.

6.3 Upper bounds on IU

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-scale 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.

7 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.

Acknowledgements

This work was supported in part by DARPA's MSEE and SMISC programs, NSF awards IIS-1427425, IIS-1212798, IIS-1116411, and the NSF GRFP, Toyota, and the Berkeley Vision and Learning Center. We gratefully acknowledge NVIDIA for GPU donation. We thank Bharath Hariharan and Saurabh Gupta for their advice and dataset tools. We thank Sergio Guadarrama for reproducing GoogLeNet in Caffe. We thank Jitendra Malik for his helpful comments. Thanks to Wei Liu for pointing out an issue wth our SIFT Flow mean IU computation and an error in our frequency weighted mean IU formula.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks." in *NIPS*, 2012. 1, 2, 3, 5

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015. 1, 2, 3, 5

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015. 1, 2, 3, 5

[4] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," in *ICLR*, 2014. 1, 2, 4

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *PAMI*, 2015. 1, 2, 8

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *ECCV*, 2014. 1

[7] N. Zhang, J. Donahue, R. Girshick, and T. Darrell, "Part-based R-CNNs for fine-grained category detection," in *ECCV*, 2014, pp. 834–849. 1

[8] J. Long, N. Zhang, and T. Darrell, "Do convnets learn correspondence?" in *NIPS*, 2014. 1

[9] P. Fischer, A. Dosovitskiy, and T. Brox, "Descriptor matching with convolutional neural networks: a comparison to SIFT," *arXiv preprint arXiv:1405.5769*, 2014. 1

[10] F. Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, and P. E. Barbano, "Toward automatic phenotyping of developing embryos from videos," *Image Processing, IEEE Transactions on*, vol. 14, no. 9, pp. 1360–1371, 2005. 1, 2, 4, 7

[11] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images." in *NIPS*, 2012, pp. 2852–2860. 1, 2, 4, 7

[12] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *PAMI*, 2013. 1, 2, 4, 7, 8

[13] P. H. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling," in *ICML*, 2014. 1, 2, 4, 7, 8

[14] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *ECCV*, 2014. 1, 2, 4, 5, 7, 8, 9

[15] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik, "Learning rich features from RGB-D images for object detection and segmentation," in *ECCV*, 2014. 1, 2, 8

[16] Y. Ganin and V. Lempitsky, "N^4-fields: Neural network nearest neighbor fields for image transforms," in *ACCV*, 2014. 1, 2, 7

[17] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CVPR*, 2015. 1, 2

[18] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *ICML*, 2014. 2

[19] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *ECCV*, 2014, pp. 818–833. 2, 4

[20] O. Matan, C. J. Burges, Y. LeCun, and J. S. Denker, "Multi-digit recognition using a space displacement neural network," in *NIPS*, 1991, pp. 488–495. 2

[21] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," in *Neural Computation*, 1989. 2, 3

[22] R. Wolf and J. C. Platt, "Postal address block location using a convolutional locator network," in *NIPS*, 1994, pp. 745–745. 2

[23] D. Eigen, D. Krishnan, and R. Fergus, "Restoring an image taken through a window covered with dirt or rain," in *ICCV*, 2013, pp. 633–640. 2

[24] J. Tompson, A. Jain, Y. LeCun, and C. Bregler, "Joint training of a convolutional network and a graphical model for human pose estimation," in *NIPS*, 2014. 2

[25] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *NIPS*, 2014. 2

[26] P. Burt and E. Adelson, "The laplacian pyramid as a compact image code," *Communications, IEEE Transactions on*, vol. 31, no. 4, pp. 532–540, 1983. 2, 6

[27] J. J. Koenderink and A. J. van Doorn, "Representation of local geometry in the visual system," *Biological cybernetics*, vol. 55, no. 6, pp. 367–375, 1987. 2, 6

[28] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *CVPR*, 2013. 2

[29] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *CVPR*, 2015. 2

[30] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich, "Feedforward semantic segmentation with zoom-out features," in *CVPR*, 2015. 2

[31] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *NIPS*, 2015. 2

[32] S. Xie and Z. Tu, "Holistically-nested edge detection," in *ICCV*, 2015. 2

[33] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *PAMI*, 2015. 2

[34] P. Fischer, A. Dosovitskiy, E. Ilg, P. Husser, C. Hazırba, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Learning optical flow with convolutional networks," in *ICCV*, 2015. 2

[35] D. Pathak, P. Krähenbühl, and T. Darrell, "Constrained convolutional neural networks for weakly supervised segmentation," in *ICCV*, 2015. 2

[36] G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille, "Weakly- and semi-supervised learning of a DCNN for semantic image segmentation," in *ICCV*, 2015. 2

[37] J. Dai, K. He, and J. Sun, "Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation," in *ICCV*, 2015. 2

[38] S. Hong, H. Noh, and B. Han, "Decoupled deep neural network for semi-supervised semantic segmentation," in *NIPS*, 2015, pp. 1495–1503. 2

[39] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected CRFs," in *ICLR*, 2015. 2

[40] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr, "Conditional random fields as recurrent neural networks," in *ICCV*, 2015. 2

[41] W. Liu, A. Rabinovich, and A. C. Berg, "ParseNet: Looking wider to see better," *arXiv preprint arXiv:1506.04579*, 2015. 2

[42] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *ICCV*, 2015. 2

[43] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *MICCAI*, 2015. 2

[44] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *ICLR*, 2016. 2, 4

[45] A. Giusti, D. C. Cireşan, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," in *ICIP*, 2013. 3, 4

[46] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian, "A real-time algorithm for signal analysis with the help of the wavelet transform," pp. 286–297, 1989. 4

[47] S. Mallat, *A wavelet tour of signal processing*, 2nd ed. Academic press, 1999. 4

[48] P. P. Vaidyanathan, "Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial," *Proceedings of the IEEE*, vol. 78, no. 1, pp. 56–93, 1990. 4

[49] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *ICML*, 2013, pp. 1058–1066. 4

[50] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results," http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html. 4

[51] C. M. Bishop, *Pattern recognition and machine learning*. Springer-Verlag New York, 2006, p. 229. 5

[52] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *ICCV*, 2011. 7

[53] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, 1998, pp. 9–48. 7

[54] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014. 7

[55] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *ECCV*, 2012. 8
[56] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from RGB-D images," in *CVPR*, 2013. 8
[57] C. Liu, J. Yuen, and A. Torralba, "Sift flow: Dense correspondence across scenes and its applications," *PAMI*, vol. 33, no. 5, pp. 978–994, 2011. 8
[58] J. Tighe and S. Lazebnik, "Superparsing: scalable nonparametric image parsing with superpixels," in *ECCV*, 2010, pp. 352–365. 8
[59] ——, "Finding things: Image parsing with regions and per-exemplar detectors," in *CVPR*, 2013. 8
[60] J. Dai, K. He, and J. Sun, "Convolutional feature masking for joint object and stuff segmentation," in *CVPR*, 2015. 8
[61] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu, "Semantic segmentation with second-order pooling," in *ECCV*, 2012. 8
[62] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *CVPR*, 2014, pp. 891–898. 8

Evan Shelhamer is a PhD student at UC Berkeley advised by Trevor Darrell as a member of the Berkeley Vision and Learning Center. He graduated from UMass Amherst in 2012 with dual degrees in computer science and psychology and completed an honors thesis with Erik Learned-Miller. Evan's research interests are in visual recognition and machine learning with a focus on deep learning and end-to-end optimization. He is the lead developer of the Caffe framework.

Jonathan Long is a PhD candidate at UC Berkeley advised by Trevor Darrell as a member of the Berkeley Vision and Learning Center. He graduated from Carnegie Mellon University in 2010 with degrees in computer science, physics, and mathematics. Jon likes finding robust and rich solutions to recognition problems. His recent projects focus on segmentation and detection with deep learning. He is a core developer of the Caffe framework.

Trevor Darrell is on the faculty of the CS Division of the EECS Department at UC Berkeley and is also appointed at the UCB-affiliated International Computer Science Institute (ICSI). He is the director of the Berkeley Vision and Learning Center (BVLC) and is the faculty director of the PATH center in the UCB Institute of Transportation Studies PATH. His interests include computer vision, machine learning, computer graphics, and perception-based human computer interfaces. Prof. Darrell received the SM and PhD degrees from MIT in 1992 and 1996, respectively. He was previously on the faculty of the MIT EECS department from 1999-2008, where he directed the Vision Interface Group. He was a member of the research staff at Interval Research Corporation from 1996-1999. He obtained the BSE degree from the University of Pennsylvania in 1988, having started his career in computer vision as an undergraduate researcher in Ruzena Bajcsy's GRASP lab.