PixelNet: Representation *of* the pixels, *by* the pixels, and *for* the pixels.

Aayush Bansal[1] Xinlei Chen[1] Bryan Russell[2] Abhinav Gupta[1] Deva Ramanan[1]
[1]Carnegie Mellon University [2]Adobe Research
http://www.cs.cmu.edu/~aayushb/pixelNet/

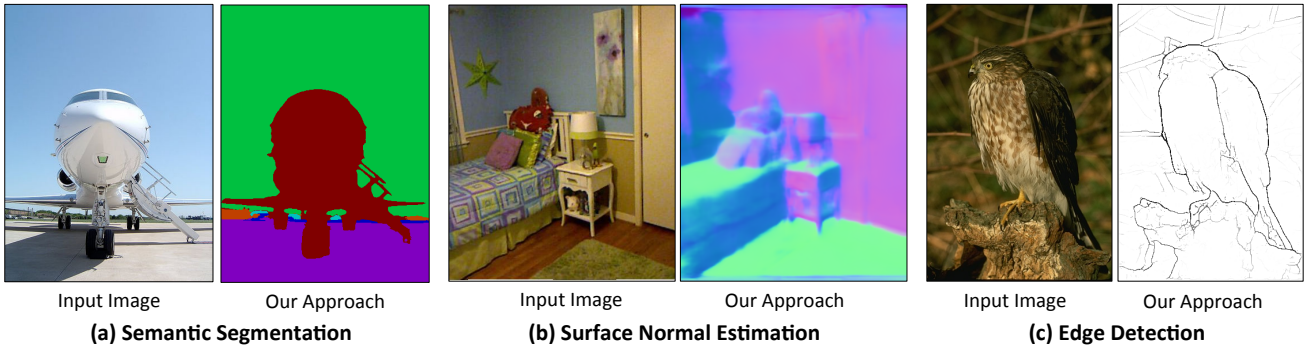


Figure 1. Our framework applied to three different pixel prediction problems with minor modification of the architecture (last layer) and training process (epochs). Note how our approach recovers the fine details for segmentation (left), surface normal (middle), and semantic boundaries for edge detection (right).

Abstract

We explore design principles for general pixel-level prediction problems, from low-level edge detection to mid-level surface normal estimation to high-level semantic segmentation. Convolutional predictors, such as the fully-convolutional network (FCN), have achieved remarkable success by exploiting the spatial redundancy of neighboring pixels through convolutional processing. Though computationally efficient, we point out that such approaches are not statistically efficient during learning precisely because spatial redundancy limits the information learned from neighboring pixels. We demonstrate that stratified sampling of pixels allows one to (1) add diversity during batch updates, speeding up learning; (2) explore complex nonlinear predictors, improving accuracy; and (3) efficiently train state-of-the-art models tabula rasa *(i.e., "from scratch") for diverse pixel-labeling tasks. Our single architecture produces state-of-the-art results for semantic segmentation on PASCAL-Context dataset, surface normal estimation on NYUDv2 depth dataset, and edge detection on BSDS.*

1. Introduction

A number of computer vision problems can be formulated as a dense pixel-wise prediction problem. These include **low-level** tasks such as edge detection [21, 64, 94] and optical flow [5, 30, 86], **mid-level** tasks such as depth/normal recovery [6, 24, 25, 81, 89], and **high-level** tasks such as keypoint prediction [13, 36, 78, 91], object detection [43], and semantic segmentation [16, 27, 40, 62, 67, 82]. Though such a formulation is attractive because of its generality, one obvious difficulty is the enormous associated output space. For example, a 100×100 image with 10 discrete class labels per pixel yields an output label space of size 10^5. One strategy is to treat this as a *spatially-invariant label prediction* problem, where one predicts a separate label per pixel using a convolutional architecture. Neural networks with convolutional output predictions, also called Fully Convolutional Networks (FCNs) [16, 62, 65, 77], appear to be a promising architecture in this direction.

But is this the ideal formulation of dense pixel-labeling? While *computationally efficient* for generating predictions at test time, we argue that it is *not statistically efficient* for gradient-based learning. Stochastic gradient descent (SGD) assumes that training data are sampled independently and from an identical distribution (*i.i.d.*) [11]. Indeed, a commonly-used heuristic to ensure approximately *i.i.d.* samples is random permutation of the training data, which can significantly improve learnability [56]. It is well known that pixels in a given image are highly correlated and not independent [45]. Following this observation, one might be tempted to randomly permute pixels during learning, but this destroys the spatial regularity that convolutional architectures so cleverly exploit! In this paper, we explore the

tradeoff between statistical and computational efficiency for convolutional learning, and investigate simply *sampling* a modest number of pixels across a small number of images for each SGD batch update, exploiting convolutional processing where possible.

Contributions: (1) We experimentally validate that, thanks to spatial correlations between pixels, just sampling a small number of pixels per image is sufficient for learning. More importantly, sampling allows us to train end-to-end particular non-linear models not earlier possible, and explore several avenues for improving both the efficiency and performance of FCN-based architectures. (2) In contrast to the vast majority of models that make use of pre-trained networks, we show that pixel-level optimization can be used to train models *tabula rasa*, or "from scratch" with simple random Gaussian initialization. Intuitively, pixel-level labels provide a large amount of supervision compared to image-level labels, given proper accounting of correlations. Without using any extra data, our model outperforms previous unsupervised/self-supervised approaches for semantic segmentation on PASCAL VOC-2012 [26], and is competitive to fine-tuning from pre-trained models for surface normal estimation. (3). Using a single architecture and without much modification in parameters, we show state-of-the-art performance for edge detection on BSDS [4], surface normal estimation on NYUDv2 depth dataset [83], and semantic segmentation on the PASCAL-Context dataset [68].

2. Background

In this section, we review related work by making use of a unified notation that will be used to describe our architecture. We address the pixel-wise prediction problem where, given an input image X, we seek to predict outputs Y. For pixel location p, the output can be binary $Y_p \in \{0, 1\}$ (e.g., edge detection), multi-class $Y_p \in \{1, \ldots, K\}$ (e.g., semantic segmentation), or real-valued $Y_p \in \mathbb{R}^N$ (e.g., surface normal prediction). There is rich prior art in modeling this prediction problem using hand-designed features (representative examples include [3, 14, 21, 38, 59, 69, 80, 82, 87, 88, 96]).

Convolutional prediction: We explore *spatially-invariant* predictors $f_{\theta,p}(X)$ that are end-to-end trainable over model parameters θ. The family of fully-convolutional and skip networks [65, 77] are illustrative examples that have been successfully applied to, e.g., edge detection [94] and semantic segmentation [12, 16, 27, 30, 62, 60, 67, 71, 75]. Because such architectures still produce separate predictions for each pixel, numerous approaches have explored post-processing steps that enforce spatial consistency across labels via e.g., bilateral smoothing with fully-connected Gaussian CRFs [16, 52, 101] or bilateral solvers [8], dilated spatial convolutions [97], LSTMs [12], and convolutional pseudo priors [93]. In contrast, our work does *not* make use of such contextual post-processing, in an effort to see how far a pure "pixel-level" architecture can be pushed.

Multiscale features: Higher convolutional layers are typically associated with larger receptive fields that capture high-level global context. Because such features may miss low-level details, numerous approaches have built predictors based on multiscale features extracted from multiple layers of a CNN [19, 24, 25, 27, 75, 89]. Hariharan et al.[40] use the evocative term "hypercolumns" to refer to features extracted from multiple layers that correspond to the same pixel. Let

$$h_p(X) = [c_1(p), c_2(p), \ldots, c_M(p)]$$

denote the multi-scale hypercolumn feature computed for pixel p, where $c_i(p)$ denotes the feature vector of convolutional responses from layer i centered at pixel p (and where we drop the explicit dependance on X to reduce clutter). Prior techniques for up-sampling include shift and stitch [62], converting convolutional filters to dilation operations [16] (inspired by the *algorithme à trous* [63]), and deconvolution/unpooling [30, 62, 71]. We similarly make use of multi-scale features, along with *sparse* on-demand upsampling of filter responses, with the goal of reducing the memory footprints during learning.

Pixel-prediction: One may cast the pixel-wise prediction problem as operating over the hypercolumn features where, for pixel p, the final prediction is given by

$$f_{\theta,p}(X) = g(h_p(X)).$$

We write θ to denote both parameters of the hypercolumn features h and the pixel-wise predictor g. Training involves back-propagating gradients via SGD to update θ. Prior work has explored different designs for h and g. A dominant trend is defining a linear predictor on hypercolumn features, e.g., $g = w \cdot h_p$. FCNs [62] point out that linear prediction can be efficiently implemented in a coarse-to-fine manner by upsampling coarse predictions (with deconvolution) rather than upsampling coarse features. DeepLab [16] incorporates filter dilation and applies similar deconvolution and linear-weighted fusion, in addition to reducing the dimensionality of the fully-connected layers to reduce memory footprint. ParseNet [60] added spatial context for a layer's responses by average pooling the feature responses, followed by normalization and concatenation. HED [94] output edge predictions from intermediate layers, which are deeply supervised, and fuses the predictions by linear weighting. Importantly, [67] and [27] are noteable exceptions to the linear trend in that *non-linear* predictors g are used. This does pose difficulties during learning - [67] precomputes and stores superpixel feature maps due to memory constraints, and so cannot be trained end-to-end.

Sampling: We demonstrate that sparse sampling of hypercolumn features allows for exploration of highly non-

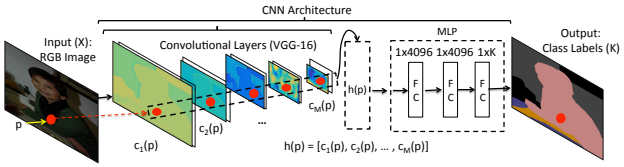


Figure 2. **PixelNet:** We input an image to a convolutional neural network, and extract hypercolumn descriptor for a sampled pixel from multiple convolutional layers. The hypercolumn descriptor is then fed to a multi-layer perceptron (MLP) for the non-linear optimization, and the last layer of MLP outputs the required response for the task. See text for more details about the use of network at training/test time.

linear g, which in turn significantly boosts performance. Our insight is inspired by past approaches that use sampling to training networks for surface normal estimation [6] and image colorization [55], though we focus on general design principles by analyzing the impact of sampling for efficiency, accuracy, and *tabula rasa* learning for diverse tasks.

Accelerating SGD: There exists a large literature on accelerating stochastic gradient descent. We refer the reader to [11] for an excellent introduction. Though naturally a sequential algorithm that processes one data example at a time, much recent work focuses on mini-batch methods that can exploit parallelism in GPU architectures [18] or clusters [18]. One general theme is efficient online approximation of second-order methods [10], which can model correlations between input features. Batch normalization [46] computes correlation statistics between samples in a batch, producing noticeable improvements in convergence speed. Our work builds similar insights directly into convolutional networks without explicit second-order statistics.

3. PixelNet

This section describes our approach for pixel-wise prediction, making use of the notation introduced in the previous section. We first formalize our pixelwise prediction architecture, and then discuss statistically efficient mini-batch training.

Architecture: As in past work, our architecture makes use of multiscale convolutional features, which we write as a hypercolumn descriptor:

$$h_p = [c_1(p), c_2(p), \ldots, c_M(p)]$$

We learn a nonlinear predictor $f_{\theta,p} = g(h_p)$ implemented as a multi-layer perceptron (MLP) [9] defined over hypercolumn features. We use a MLP, which can be implemented as a series of "fully-connected" layers followed by ReLU activation functions. Importantly, the last layer must be of size K, the number of class labels or real valued outputs being predicted. See Figure 2.

Sparse predictions: We now describe an efficient method for generating sparse pixel predictions, which will be used at train-time (for efficient mini-batch generation). Assume that we are given an image X and a sparse set of (sampled) pixel locations $P \subset \Omega$, where Ω is the set of all pixel positions.

1. Perform a forward pass to compute dense convolutional responses at all layers $\{c_i(p) : \forall i, p \in \Omega\}$
2. For each sampled pixel $p \in P$, compute its hypercolumn feature h_p *on demand* as follows:
   (a) For each layer i, compute the 4 discrete locations in the feature map c_i closest to p
   (b) Compute $c_i(p)$ via bilinear interpolation
3. Rearrange the sparse of hypercolumn features $\{h_p : p \in P\}$ into a matrix for downstream processing (e.g., MLP classification).

The above pipeline only computes $|P|$ hypercolumn features rather than full dense set of size $|\Omega|$. We experimentally demonstrate that this approach offers an excellent tradeoff between amortized computation (to compute $c_i(p)$) and reduced storage (to compute h_p). Note that our multiscale sampling layer simply acts as a selection operation, for which a (sub) gradient can easily be defined. This means that backprop can also take advantage of sparse computations for nonlinear MLP layers and convolutional processing for the lower layers.

Mini-batch sampling: At each iteration of SGD training, the true gradient over the model parameters θ is approximated by computing the gradient over a relatively small set of samples from the training set. Approaches based on FCN [62] include features for all pixels from an image in a mini-batch. As nearby pixels in an image are highly correlated [45], sampling them will not hurt learning. To ensure a diverse set of pixels (while still enjoying the amortized benefits of convolutional processing), we use a modest number of pixels ($\sim 2,000$) per image, but sample many images per batch. Naive computation of dense grid of hypercolumn descriptors takes almost all of the (GPU) memory, while $2,000$ samples takes a small amount using our sparse sampling layer. This allows us to explore more images per batch, significantly increasing sample diversity.

Dense predictions: We now describe an efficient method for generating dense pixel predictions with our network, which will be used at test-time. Dense prediction proceeds by following step (1) from above; and instead of sampling in (2) above, we take all the pixels now. This produces a dense grid of hypercolumn features, which are then (3) processed by pixel-wise MLPs implemented as 1x1 filters (representing each fully-connected layer). The memory intensive portion of this computation is the dense grid of

hypercolumn features. This memory footprint is reasonable at test time because a single image can be processed at a time, but at train-time, we would like to train on batches containing many images as possible (to ensure diversity).

4. Analysis

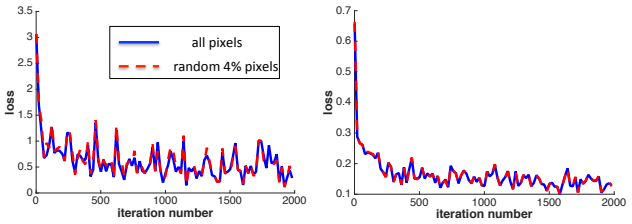
In this section, we analyze the properties of pixel-level optimization using semantic segmentation and surface normal estimation to understand the design choices for pixel-level architectures. We chose the two varied tasks (classification and regression) for analysis to verify the generalizability of these findings. We use a single-scale 224×224 image as input. We also show *sampling* augmented with careful *batch-normalization* can allow for a model to be trained from scratch (without pre-trained ImageNet model as an initialization) for semantic segmentation and surface normal estimation. We explicitly compare the performance of our approach with previous approaches in Section 5.

Default network: For most experiments we fine-tune a VGG-16 network [84]. VGG-16 has 13 convolutional layers and three fully-connected (*fc*) layers. The convolutional layers are denoted as $\{1_1, 1_2, 2_1, 2_2, 3_1, 3_2, 3_3, 4_1, 4_2, 4_3, 5_1, 5_2, 5_3\}$. Following [62], we transform the last two *fc* layers to convolutional filters[1], and add them to the set of convolutional features that can be aggregated into our multi-scale hypercolumn descriptor. To avoid confusion with the **fc** layers in our MLP, we will henceforth denote the *fc* layers of VGG-16 as conv-6 and conv-7. We use the following network architecture (unless otherwise specified): we extract hypercolumn features from conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ with on-demand interpolation. We define a MLP over hypercolumn features with 3 fully-connected (fc) layers of size $4,096$ followed by ReLU [53] activations, where the last layer outputs predictions for K classes (with a softmax/cross-entropy loss) or K outputs with a euclidean loss for regression.

Semantic Segmentation: We use training images from PASCAL VOC-2012 [26] for semantic segmentation, and additional labels collected on 8498 images by Hariharan et al. [41]. We used the held-out (non-overlapping) validation set to show most analysis. However, at some places we have used the test set where we wanted to show comparison with previous approaches. We report results using the standard metrics of region intersection over union (**IoU**) averaged over classes (higher is better). We mention it as IoU (V) when using the validation set for evaluation, and IoU (T) when showing on test set.

Surface Normal Estimation: The NYU Depth v2 dataset [83] is used to evaluate the surface normal maps.

[1]For alignment purposes, we made a small change by adding a spatial padding of 3 cells for the convolutional counterpart of *fc6* since the kernel size is 7×7.



a. Semantic Segmentation b. Surface Normal Estimation

Figure 3. Given a fixed number of (5) images per SGD batch, we analyse convergence properties using all pixels vs. randomly sampling 4% , or $2,000$ pixels for semantic segmentation and surface normal estimation. This experiment suggest that sampling does not hurt convergence.

There are 1449 images, of which 795 are trainval and remaining 654 are used for evaluation. Additionally, there are $220,000$ frames extracted from raw Kinect data. We use the normals of Ladicky et al.[54] and Wang et al.[89], computed from depth data of Kinect, as ground truth for 1449 images and 220K images respectively. We compute six statistics, previously used by [6, 24, 31, 32, 33, 89], over the angular error between the predicted normals and depth-based normals to evaluate the performance – **Mean**, **Median**, **RMSE**, **11.25°**, **22.5°**, and **30°** – The first three criteria capture the mean, median, and RMSE of angular error, where lower is better. The last three criteria capture the percentage of pixels within a given angular error, where higher is better.

4.1. Sampling

We examine how sampling a few pixels from a fixed set of images does not harm convergence. Given a fixed number of (5) images per batch, we find that sampling a small fraction (4%) of the pixels per image does not affect learnability (Figure 3 and Table 1). This validates our hypothesis that much of the training data for a pixel-level task is correlated within an image, implying that randomly sampling a few pixels is sufficient. Our results are consistent with those reported in Long *et al.* [62], who similarly examine the effect of sampling a fraction (25-50%) of patches per training image.

Long *et al.* [62] also perform an additional experiment where the total number of pixels in a batch is kept constant when comparing different sampling strategies. While this ensures that each batch will contain more diverse pixels, each batch will also process a larger number of images. If there are no significant computational savings due to sampling, additional images will increase wall-clock time and slow convergence. In the next section, we show that adding additional computation after sampling (by replacing a linear classifier with a multi-layer perceptron) fundamentally changes this tradeoff (Table 4).

Method	IoU (V)	Mean	Median	RMSE	11.25°	22.5°	30°
All Pixels	44.4	25.6	19.9	32.5	29.1	54.9	66.8
Random 4% Pixels	44.6	25.7	20.1	32.5	28.9	54.7	66.7

Table 1. **Sampling:** We demonstrate that sampling few pixels for each mini-batch yields similar accuracy as using all pixels. The results are computed on models trained for 10 epochs and $10,000$ iterations for semantic segmentation and surface normal estimation, respectively.

Method	IoU (T)	Mean	Median	RMSE	11.25°	22.5°	30°
Linear (no bn)	3.6	24.8	19.4	31.2	28.7	56.4	68.8
Linear (bn)	62.4	22.5	16.1	29.7	37.0	62.8	73.3
MLP	**67.4**	**19.8**	**12.0**	**28.0**	**47.9**	**70.0**	**77.8**

Table 2. **Linear vs. MLP:** A multi-layer perceptron over hypercolumn features gives better performance over linear model without requiring normalization/scaling. Note: bn stands for batch-normalization.

4.2. Linear vs. MLP

Most previous approaches have focussed on linear predictors combining the information from different convolutional layers (also called 'skip-connections'). Here we contrast the performance of non-linear models via MLP with corresponding linear models. For this analysis, we use a VGG-16 (pre-trained on ImageNet) as initialization and use skip-connections from conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ layers to show the benefits of a non-linear model over a linear model. We randomly sample $2,000$ pixels per image from a set of five 224×224 images per SGD iteration for the optimization.

A major challenge in using skip-connections is how to combine the information as the dynamic range varies across the different layers. The top-row in Table 2 shows how the model leads to degenerate outputs for semantic segmentation when 'naively' concatenating features from different convolutional layers in a linear model. Similar observation was made by [60]. To counter this issue, previous work has explored normalization [60], scaling [40], etc. We use batch-normalization [46] for the convolutional layers before concatenating them to properly train a model with a linear predictor. The middle-row in Table 2 shows how adding batch-normalization allows us to train a linear model for semantic segmentation, and improve the performance for surface normal estimation. While we have to take care of normalization for linear models, we do not need them while using a MLP and can naively concatenate features from different layers. The last row in Table 2 shows the performance on different tasks when using a MLP. Note that performance of linear model (with batch-normalization) is similar to one obtained by Hypercolum [40] (62.7%), and FCN [62] (62%).

***Deconvolution* vs. *on-demand* compute:** A naive implementation of our approach is to use deconvolution layers to upsample conv-layers, followed by feature concatenation, and mask out the pixel-level outputs. This is similar to the sampling experiment of Long *et al.* [62]. While reasonable for a linear model, naively computing a dense grid of hypercolumn descriptors and processing them with a MLP is impossible if *conv-7* is included in the hypercolumn descriptor (the array dimensions exceed *INT_MAX*). For practical purposes, if we consider skip-connections only from conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ layers at cost of some performance, naive deconvolution would still take more than **12X** memory compared to our approach. Slightly better would be masking the dense grid of hypercolumn descriptors before MLP processing, which is still **8X** more expensive. Most computational savings come from not being required to keep an extra copy of the data required by deconvolution and concatenation operators. Table 3 highlights the differences in computational requirements between deconvolution vs. *on-demand* compute (for the more forgiving setting of $\{1_2, 3_3, 5_3\}$-layered hypercolumn features). Clearly, *on-demand* compute requires less resources.

Model	#features	sample (#)	Memory (MB)	Disk Space (MB)	BPS
FCN-32s [62]	4,096	50,176	2,010	518	20.0
FCN-8s [62]	4,864	50,176	2,056	570	19.5
FCN/Deconvolution					
Linear	1,056	50,176	2,267	1,150	6.5
MLP	1,056	50,176	3,914	1,232	1.4
FCN/Sampling					
Linear	1,056	2,000	2,092	1,150	5.5
MLP	1,056	2,000	2,234	1,232	5.1
PixelNet/On-demand					
Linear	1,056	2,000	322	60	43.3
MLP	1,056	2,000	465	144	24.5
MLP (+*conv-7*)	5,152	2,000	1,024	686	8.8

Table 3. **Computational Requirements:** We record the number of dimensions for hypercolumn features from conv-$\{1_2, 3_3, 5_3\}$, number of samples (for our model), memory usage, model size on disk, number of mini-batch updates per second (BPS measured by forward/backward passes). We use a single 224×224 image as the input. We compared our network with FCN [62] where a deconvolution layer is used to upsample the result in various settings. Besides FCN-8s and FCN-32s here we first compute the upsampled feature map, and then apply the classifiers for FCN [62]. Clearly from the table, our approach require less computational resources as compared to other settings.

Does statistical diversity matter? We now analyze the influence of statistical diversity on optimization given a fixed computational budget (7GB memory on a NVIDIA TITAN-X). We train a non-linear model using 1 image $\times$ 40,000 pixels per image vs. 5 images $\times$ 2,000 pixels per image. Table 4 shows that sampling fewer pixels from more images outperforms more pixels extracted from fewer images. This demonstrates that statistical diversity outweighs the computational savings in convolutional processing when a MLP

Method	IoU_1 (V)	IoU_2 (V)	Mean	Median	RMSE	11.25°	22.5°	30°
$1 \times 40,000$	7.9	15.5	24.8	19.5	31.6	29.7	56.1	68.5
$5 \times 2,000$	**38.4**	**47.9**	**23.4**	**17.2**	**30.5**	**33.9**	**60.6**	**71.8**

Table 4. **Statistical Diversity Matters:** For a given computational budget, using diverse set of pixels from more images shows better performance over more pixels from a few images. IoU_1 and IoU_2 show performance for 10K and 20K iterations of SGD for semantic segmentation. For surface normal estimation, we show performance for 10K iterations of SGD. This suggest that sampling leads to faster convergence.

classifier is used.

4.3. Training from scratch

Prevailing methods for training deep models make use of a pre-trained (e.g., ImageNet [79]) model as initialization for fine-tuning for the task at hand. Most network architectures (including ours) improve in performance with pre-trained models. A major concern is the availability of sufficient data to train deep models for pixel-level prediction problems. However, because our optimization is based on randomly-sampled pixels instead of images, there is potentially more unique data available for SGD to learn a model from a random initialization. We show how *sampling* and *batch-normalization* enables models to be trained from scratch. This enables our networks to be used in problems with limited training data, or where natural image data does not apply (e.g., molecular biology, tissue segmentation etc. [70, 95]). We will show that our results also have implications for unsupervised representation learning [1, 20, 23, 34, 37, 48, 57, 55, 66, 72, 73, 74, 76, 90, 99, 100].

Random Initialization: We randomly initialize the parameters of a VGG-16 network from a Gaussian distribution. Training a VGG-16 network architecture is not straight forward, and required stage-wise training for the image classification task [84]. It seems daunting to train such a model from scratch for a pixel-level task where we want to learn both coarse and fine information. In our experiments, we found batch normalization to be an effective tool for converging a model trained from scratch.

We train the models for semantic segmentation and surface normal estimation. The middle-row in Table 5 shows the performance for semantic segmentation and surface normal estimation trained from scratch. The model trained from scratch for surface normal estimation is within 2-3% of current state-of-the-art performing method. The model for semantic segmentation achieves 48.7% on PASCAL VOC-2012 test set when trained from scratch. To the best of our knowledge, these are the best numbers reported on these two tasks when trained from scratch, and exceeds the performance of other unsupervised/self-supervised approaches [20, 55, 74, 90, 99] that required extra ImageNet data [79].

Initialization	IoU (T)	Mean	Median	RMSE	11.25°	22.5°	30°
ImageNet	67.4	19.8	12.0	28.2	47.9	70.0	77.8
Random	48.7	21.2	13.4	29.6	44.2	66.6	75.1
Geometry	52.4	-	-	-	-	-	-

Table 5. **Initialization:** We study the influence of initialization on accuracy. PixelNet can even be trained reasonably well with random Gaussian initialization or starting from a model trained for surface normal prediction (Geometry).

Self-Supervision via Geometry: We briefly present the performance of models trained from our pixel-level optimization in context of self-supervision. The task of surface normal estimation does not require any human-labels, and is primarily about capturing geometric information. In this section, we explore the applicability of fine-tuning a geometry model (trained from scratch) for more semantic tasks (such as semantic segmentation and object detection). Table 5 (last row) and Table 6 shows the performance of our approach on semantic segmentation and object detection respectively. Note that the NYU depth dataset is a small indoor scene dataset and does not contain most of the categories present in PASCAL VOC dataset. Despite this, it shows 4% (segmentation) and 9% (detection) improvement over naive scratch models. It is best known result for semantic segmentation in an unsupervised/self-supervised manner, and is competitive with the previous unsupervised work [20] on object detection[2] that uses ImageNet (without labels), particularly on indoor scene furniture categories (e.g., chairs, sofa, table, tv, bottle). We posit that geometry is a good cue for unsupervised representation learning as it can learn from a few examples and can even generalize to previously unseen categories. Future work may utilize depth information from videos (c.f. [98]) and use them to train models for surface normal estimation. This can potentially provide knowledge about more general categories. Finally, we add a minor supervision by taking the geometry-based model fine-tuned for segmentation, and further fine-tuning it for object detection. We get an extra 5% boost over the performance.

5. Generalizability

In this section we demonstrate the generalizability of PixelNet, and apply (with minor modifications) it to the high-level task of semantic segmentation, mid-level surface normal estimation, and the low-level task of edge detection. The in-depth analysis for each of these tasks are in appendices.

[2]We used a single scale for object detection and use the same parameters as Fast-RCNN except a step-size of 70K, and fine-tuned it for 200K iterations. Doersch et al. [20] reports better results in a recent version by the use of multi-scale detection, and smarter initialization and rescaling.

VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**mAP**
DPM-v5 [28]	33.2	60.3	10.2	16.1	27.3	54.3	58.2	23.0	20.0	24.1	26.7	12.7	58.1	48.2	43.2	12.0	21.1	36.1	46.0	43.5	33.7
HOG+MID [7]	51.7	61.5	17.9	27.0	24.0	57.5	60.2	47.9	21.1	42.2	48.9	29.8	58.3	51.9	34.3	22.2	36.8	40.2	54.3	50.9	41.9
RCNN-Scratch [2]	49.9	60.6	24.7	23.7	20.3	52.5	64.8	32.9	20.4	43.5	34.2	29.9	49.0	60.4	47.5	28.0	42.3	28.6	51.2	50.0	40.7
VGG-16-Scratch [20]	56.1	58.6	23.3	25.7	12.8	57.8	61.2	45.2	21.4	47.1	39.5	35.6	60.1	61.4	44.9	17.3	37.7	33.2	57.9	51.2	42.4
VGG-16-Context-v2 [20]	63.6	64.4	42.0	42.9	18.9	67.9	69.5	65.9	28.2	48.1	58.4	58.5	66.2	64.9	54.1	26.1	43.9	55.9	69.8	50.9	53.0
VGG-16-Geometry	55.9	61.6	29.5	31.1	24.0	66.3	70.6	56.7	32.4	53.2	58.5	49.4	72.1	66.4	53.6	21.8	38.6	55.1	65.4	58.2	51.0
VGG-16-Geom+Seg	62.8	68.7	39.9	37.5	27.4	75.9	73.8	70.3	33.8	57.2	62.7	60.1	72.8	69.5	60.7	22.5	40.8	62.0	70.5	59.2	56.4
VGG-16-ImageNet [35]	73.6	77.9	68.8	56.2	35.0	76.8	78.1	83.1	39.7	73.4	65.6	79.6	81.3	73.3	66.1	30.4	67.2	67.9	77.5	66.5	66.9

Table 6. **Evaluation on VOC-2007:** Our model (VGG-16-Geometry) trained on a few indoor scene examples of NYU-depth dataset performs 9% better than scratch, and is competitive with [20] that used images from ImageNet (without labels) to train. Note that we used 110K iterations of SGD to train our model, and it took less than 3 days. [20] required training for around 8 weeks. Finally, we added a minor supervision (VGG-16-Geom+Seg) using the non-overlapping segmentation dataset and improve the performance further by 5%.

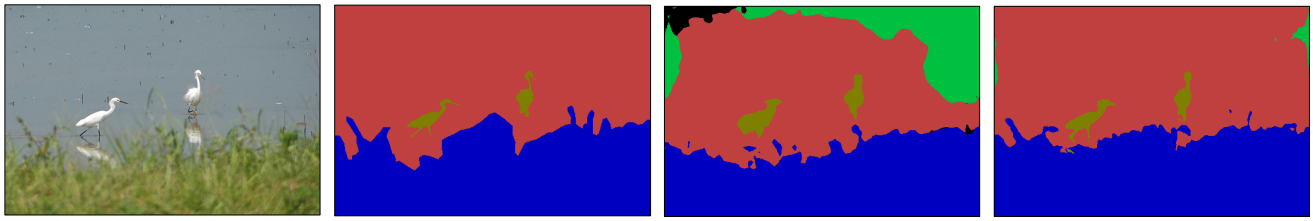


Figure 4. Segmentation results on PASCAL-Context 59-class. Our approach uses an MLP to integrate information from both lower (*e.g.* 1_2) and higher (*e.g. conv-7*) layers, which allows us to better capture both global structure (object/scene layout) and fine details (small objects) compared to *FCN-8s*.

5.1. Semantic Segmentation

Training: For all the experiments we used the publicly available *Caffe* library [49]. All trained models and code will be released. We make use of ImageNet-pretrained values for all convolutional layers, but train our MLP layers "from scratch" with Gaussian initialization ($\sigma = 10^{-3}$) and dropout [85] ($r = 0.5$). We fix momentum 0.9 and weight decay 0.0005 throughout the fine-tuning process. We use the following update schedule (unless otherwise specified): we tune the network for 80 epochs with a fixed learning rate (10^{-3}), reducing the rate by $10\times$ twice every 8 epochs until we reach 10^{-5}.

Dataset: The PASCAL-Context dataset [4] augments the original sparse set of PASCAL VOC 2010 segmentation annotations [26] (defined for 20 categories) to pixel labels for the whole scene. While this requires more than 400 categories, we followed standard protocol and evaluate on the 59-class and 33-class subsets. The results for PASCAL VOC-2012 dataset [26] are in Appendix A.

Evaluation Metrics: We report results on the standard metrics of pixel accuracy (AC) and region intersection over union (IU) averaged over classes (higher is better). Both are calculated with DeepLab evaluation tools[3].

Results: Table 12 shows performance of our approach compared to previous work. Our approach without CRF does better than previous approaches based on it. Due to space constraints, we show only one example output in Figure 6 and compare against FCN-8s [62]. Notice that we capture fine-scale details, such as the leg of birds. More analysis and details are in Appendix A.

[3] https://bitbucket.org/deeplab/deeplab-public/

Model	**59-class**		**33-class**	
	AC (%)	IU (%)	AC (%)	IU (%)
FCN-8s [61]	46.5	35.1	67.6	53.5
FCN-8s [62]	50.7	37.8	-	-
DeepLab (v2 [15])	-	37.6	-	-
DeepLab (v2) + CRF [15]	-	39.6	-	-
CRF-RNN [101]	-	39.3	-	-
ConvPP-8 [93]	-	**41.0**	-	-
PixelNet	**51.5**	**41.4**	**69.5**	**56.9**

Table 7. **Evaluation on PASCAL-Context [4]:** Most recent approaches [15, 93, 101] except FCN-8s, use spatial context post-processing. We achieve results better than previous approaches without any CRF. CRF post-processing could be applied to any local unary classifier (including our method).

5.2. Surface Normal Estimation

We use NYU-v2 depth dataset, and same evaluation criteria as defined earlier in Section 4. We improve the state-of-the-art for surface normal estimation [6] using the analysis for general pixel-level optimization. While Bansal et al. [6] extracted hypercolumn features from $1 \times 1 \times 4096$ *conv-7* of VGG-16, we provided sufficient padding at *conv-6* to

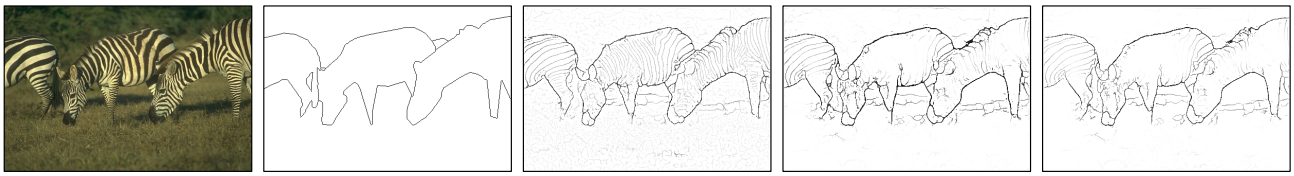


Figure 5. Qualitative results for edge detection. Notice that our approach generates more semantic edges for *zebra* compared to HED [94]. There are more similar examples of *eagle*, and *giraffe* in Appendix C. Best viewed in the electronic version.

NYUDv2 test	Mean	Median	RMSE	11.25°	22.5°	30°
Fouhey et al. [31]	35.3	31.2	41.4	16.4	36.6	48.2
E-F (VGG-16) [24]	20.9	13.2	-	44.4	67.2	75.9
UberNet (1-Task) [51]	21.4	15.6	-	35.3	65.9	76.9
MarrRevisited [6]	19.8	12.0	28.2	47.9	70.0	77.8
PixelNet	**19.2**	**11.3**	**27.2**	**49.4**	**70.7**	**78.5**

Table 8. **Evaluation on NYUv2 depth dataset [83]:** We improve the previous state-of-the-art [6] using the analysis from general pixel-level prediction problems, and multi-scale prediction.

	ODS	**OIS**	**AP**
Human [4]	.800	.800	-
SE-Var [22]	.746	.767	.803
OEF [39]	.749	.772	.817
DeepNets [50]	.738	.759	.758
CSCNN [44]	.756	.775	.798
HED [94] (Updated version)	.788	.808	.840
UberNet (1-Task) [51]	.791	.809	**.849**
PixelNet (Uniform)	.767	.786	.800
PixelNet (Biased)	**.795**	**.811**	.830

Table 9. **Evaluation on BSDS [4]:** Our approach performs better than previous approaches *specifically* trained for edge detection. Here, we show two results using our architecture: a. Uniform; and b. Biased. For the former, we randomly sample positive and negative examples while we do a biased sampling of 50% positives (from a total 10% positives in edge dataset) and 50% negatives. As shown, biased sampling improves performance for edge detection. Finally, we achieve F-score of 0.8 which is same as humans.

have $4 \times 4 \times 4096$ *conv-7*. This provided diversity in *conv-7* features for different pixels in a image instead of same *conv-7* earlier. Further we use a multi-scale prediction to improve the results.

Training: We use the same network architecture as described earlier. The last *fc*-layer of MLP has ($\sigma = 5 * 10^{-3}$). We set the initial learning rate to 10^{-3}, reducing the rate by $10\times$ after 50K SGD iteration. The network is trained for 60K iterations.

Results: Table 13 compares our improved results with previous state-of-the-art approaches [6, 24]. More analysis and details are in Appendix B.

5.3. Edge Detection

Dataset: The standard dataset for edge detection is BSDS-500 [4], which consists of 200 training, 100 validation, and 200 testing images. Each image is annotated by $\sim$5 humans to mark out the contours. We use the same augmented data (rotation, flipping, totaling 9600 images without resizing) used to train the state-of-the-art Holistically-nested edge detector (HED) [94]. We report numbers on the testing images. During training, we follow HED and only use positive labels where a consensus (≥ 3 out of 5) of humans agreed.

Training: We use the same baseline network and training strategy defined earlier in Section 5.1. A sigmoid cross-entropy loss is used to determine the whether a pixel is belonging to an edge or not. Due to the highly skewed class distribution, we also normalized the gradients for positives and negatives in each batch (as in [94]). In case of skewed class label distribution, sampling offers the flexibility to let the model focus more on the rare classes.

Results: Table 16 shows the performance of PixelNet for edge detection. The last 2 rows in Table 16 contrast the performance between uniform and biased sampling. Clearly biased sampling toward positives can help the performance. Qualitatively, we find our network tends to have better results for semantic-contours (*e.g.* around an object), particularly after including *conv-7* features. Figure 9 shows some qualitative results comparing our network with the HED model. Interestingly, our model explicitly removed the edges inside the *zebra*, but when the model cannot recognize it (*e.g.* its head is out of the picture), it still marks the edges on the black-and-white stripes. Our model appears to be making use of more higher-level information than past work on edge detection. More analysis and details are in Appendix C.

6. Discussion

We have described a convolutional pixel-level architecture that, with minor modifications, produces state-of-the-art accuracy on diverse high-level, mid-level, and low-level tasks. We demonstrate results on highly-benchmarked semantic segmentation, surface normal estimation, and edge detection datasets. Our results are made possible by careful analysis of computational and statistical considerations associated with convolutional predictors. Convolution exploits spatial redundancy of pixel neighborhoods for efficient computation, but this redundancy also impedes learning. We propose a simple solution based on stratified sam-

pling that injects diversity while taking advantage of amortized convolutional processing. Finally, our efficient learning scheme allow us to explore nonlinear functions of multi-scale features that encode both high-level context and low-level spatial detail, which appears relevant for most pixel prediction tasks.

Appendices

In Section A we present extended analysis of PixelNet architecture for semantic segmentation on PASCAL Context [68] and PASCAL VOC-2012 [26], and ablative analysis for parameter selection. In Section B we show comparison of improved surface normal with previous state-of-the-art approaches on NYU-v2 depth dataset [83]. In Section C we compare our approach with prior work on edge detection on BSDS [4]. Note that we use the default network and training mentioned in the main draft.

A. Semantic Segmentation

Dataset. The PASCAL-Context dataset [4] augments the original sparse set of PASCAL VOC 2010 segmentation annotations [26] (defined for 20 categories) to pixel labels for the whole scene. While this requires more than 400 categories, we followed standard protocol and evaluate on the 59-class and 33-class subsets. We also evaluated our approach on the standard PASCAL VOC-2012 dataset [26] to compare with a wide variety of approaches.

Training. For all the experiments we used the publicly available *Caffe* library [49]. All trained models and code will be released. We make use of ImageNet-pretrained values for all convolutional layers, but train our MLP layers "from scratch" with Gaussian initialization ($\sigma = 10^{-3}$) and dropout [85]. We fix momentum 0.9 and weight decay 0.0005 throughout the fine-tuning process. We use the following update schedule (unless otherwise specified): we tune the network for 80 epochs with a fixed learning rate (10^{-3}), reducing the rate by $10\times$ twice every 8 epochs until we reach 10^{-5}.

Qualitative Results. We show qualitative outputs in Figure 6 and compare against FCN-8s [62]. Notice that we capture fine-scale details, such as the leg of birds (row 2) and plant leaves (row 3).

Evaluation Metrics. We report results on the standard metrics of pixel accuracy (AC) and region intersection over union (IU) averaged over classes (higher is better). Both are calculated with DeepLab evaluation tools[4].

Analysis-1: Dimension of MLP *fc* Layers. We analyze performance as a function of the size of the MLP *fc* layers. We experimented the following dimensions for our *fc* layers: $\{1024, 2048, 4096, 6144\}$. Table 10 lists the results. We use 5 images per SGD batch and sample 2000 pixels per image, and conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ for skip connections to do this analysis. We can see that with more dimensions the network tends to learn better, potentially because it can capture more information (and with drop-out alleviating overfitting [85]). In the main paper we fix the size to $4,096$ as a good trade-off between performance and speed.

Dimension	AC (%)	IU (%)
1024	41.6	33.2
2048	43.2	34.2
4096	44.0	34.9
6144	44.2	35.1

Table 10. **Dimension of MLP fc layers:** We vary the dimension of the MLP *fc* layers on the PASCAL Context 59-class segmentation task from $\{1024, 2048, 4096, 6144\}$. We observe that 4096 is a good trade-off between performance and speed.

Analysis-2: Number of Mini-batch Samples. Continuing from our analysis on statistical diversity (Section 4.2 in main paper), we plot performance as a function of the number of sampled pixels per image. In the first sampling experiment, we fix the batch size to 5 images and sample $\{500, 1000, 2000, 4000\}$ pixels from each image. We use conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ for skip connections for this analysis. The results are shown in Table 11. We observe that: 1) even sampling only 500 pixels per image (on average 2% of the $\sim$20,000 pixels in an image) produces reasonable performance after just 96 epochs. 2) performance is roughly constant as we increase the number of samples.

We also perform experiments where the samples are drawn from the same image. When sampling 2000 pixels from a single image (comparable in size to batch of 500 pixels sampled from 5 images), performance dramatically drops. This phenomena consistently holds for additional pixels (Table 11, bottom rows), verifying our central thesis that statistical diversity of samples can trump the computational savings of convolutional processing during learning.

Analysis-3: Adding conv-7. While our diagnostics reveal the importance of architecture design and sampling, our best results still do not quite reach the state-of-the-art. For example, a single-scale FCN-32s [62], without any low-level layers, can already achieve 35.1. This suggests that their penultimate *conv-7* layer does capture cues relevant for pixel-level prediction. In practice, we find that simply concatenating *conv-7* significantly improves performance.

Following the same training process, the results of our model with *conv-7* features are shown in Table 12. From this we can see that *conv-7* is greatly helping the performance of semantic segmentation. Even with reduced scale, we are able to obtain a similar IU achieved by FCN-8s [62], without any extra modeling of context [16, 60, 93, 101].

[4] https://bitbucket.org/deeplab/deeplab-public/

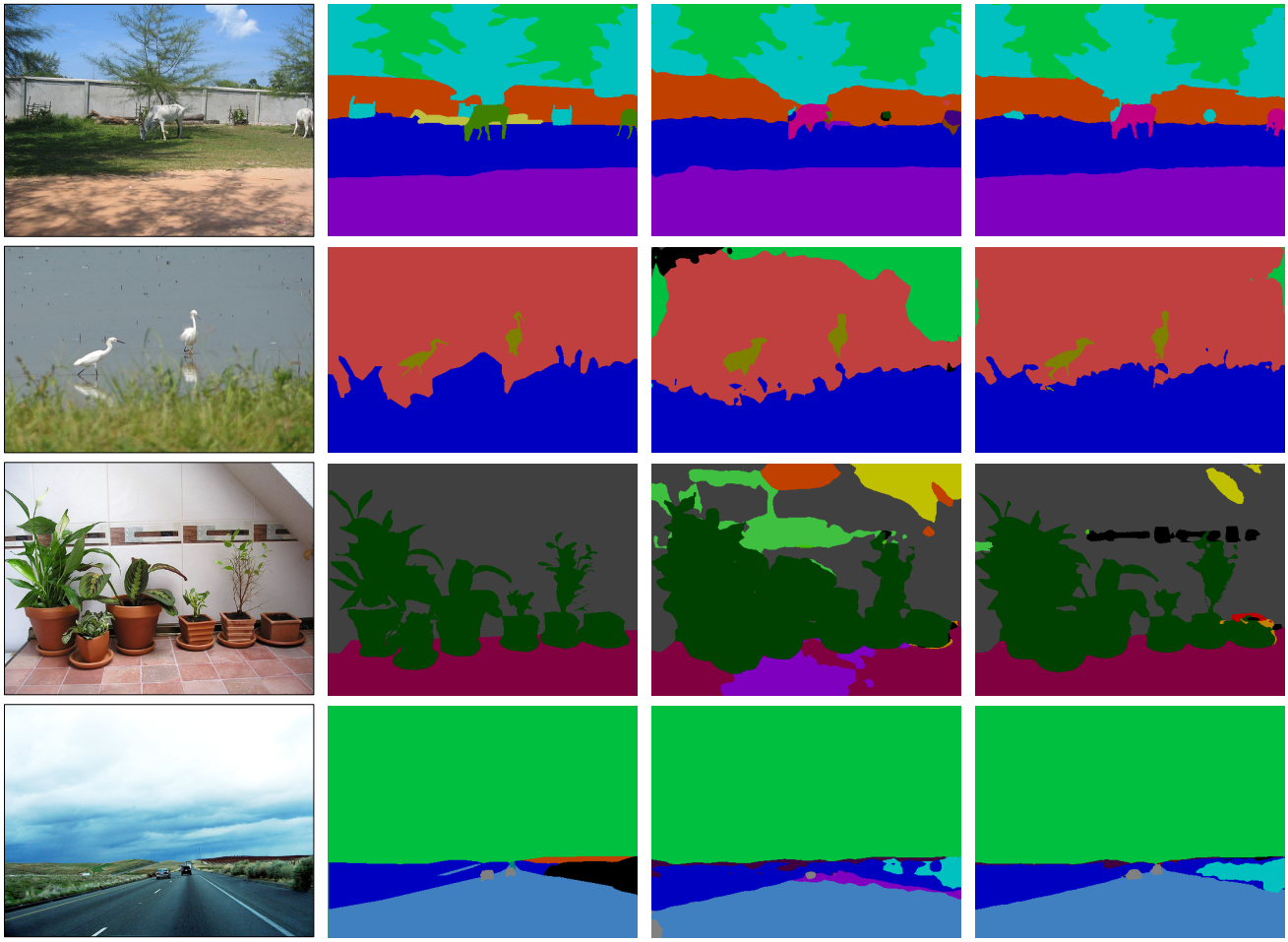


Figure 6. Segmentation results on PASCAL-Context 59-class. Our approach uses an MLP to integrate information from both lower (*e.g.* 1_2) and higher (*e.g. conv-7*) layers, which allows us to better capture both global structure (object/scene layout) and fine details (small objects) compared to *FCN-8s*.

For fair comparison, we also experimented with single scale training with 1) half scale $0.5\times$, and 2) full scale $1.0\times$ images. We use 5 images per SGD batch, and sample 2000 pixels per image. We find the results are better without $0.25\times$ training, reaching 37.4% and 37.6% IU, respectively, even closer to the FCN-8s performance (37.8% IU). For the 33-class setting, we are already doing better with the baseline model plus *conv-7*.

Analysis-4: Multi-scale. All previous experiments process test images at a single scale ($0.25\times$ or $0.5\times$ its original size), whereas most prior work [16, 60, 62, 101] use multiple scales from full-resolution images. A smaller scale allows the model to access more context when making a prediction, but this can hurt performance on small objects. Following past work, we explore test-time averaging of predictions across multiple scales. We tested combinations of $0.25\times$, $0.5\times$ and $1\times$. For efficiency, we just fine-tune the model trained on small scales (right before reducing the learning rate for the first time) with an initial learning rate of 10^{-3} and step size of 8 epochs, and end training after 24 epochs. The results are also reported in Table 12. Multi-scale prediction generalizes much better (41.0% IU). Note our pixel-wise predictions do not make use of contextual post-processing (even outperforming some methods that post-processes FCNs to do so [15, 101]).

Evaluation on PASCAL VOC-2012 [26]. We use the same settings, and evaluate our approach on PASCAL VOC-2012. Our approach, without any special consideration of parameters for this dataset, achieves mAP of **69.7%**[5]. This is much better than previous approaches, e.g. 62.7% for Hypercolumns [40], 62% for FCN [62], 67% for DeepLab (without CRF) [16] etc. Our performance on VOC-2012

[5]Per-class performance is available at http://host.robots.ox.ac.uk:8080/anonymous/PZH9WH.html.

$N \times M$	AC (%)	IU (%)
500×5	43.7	34.8
1000×5	43.8	34.7
2000×5	43.8	34.7
4000×5	43.9	34.9
2000×1	32.6	24.6
10000×1	33.3	25.2

Table 11. **Varying SGD mini-batch construction:** We vary the SGD mini-batch construction on the PASCAL Context 59-class segmentation task. $N \times M$ refers to a mini-batch constructed from N pixels sampled from each of M images (a total of N×M pixels sampled for optimization). We see that a small number of pixels per image (500, or 2%) are sufficient for learning. Put in another terms, given a fixed budget of N pixels per mini-batch, performance is maximized when spreading them across a large number of images M. This validates our central thesis that statistical diversity trumps the computational savings of convolutional processing during learning.

is similar to Mostajabi et al [67] despite the fact we use information from only 6 layers while they used information from all the layers. In addition, they use a rectangular region of 256×256 (called *sub-scene*) around the super-pixels. We posit that fine-tuning (or back-propagating gradients to conv-layers) enables efficient and better learning with even lesser layers, and without extra *sub-scene* information in an end-to-end framework. Finally, the use of super-pixels in [67] inhibit capturing detailed segmentation mask (and rather gives "blobby" output), and it is computationally less-tractable to use their approach for per-pixel optimization as information for each pixel would be required to be stored on disk.

B. Surface Normal Estimation

Dataset. The NYU Depth v2 dataset [83] is used to evaluate the surface normal maps. There are 1449 images, of which 795 are trainval and remaining 654 are used for evaluation. Additionally, there are $220,000$ frames extracted from raw Kinect data. We use the normals of Ladicky et al.[54] and Wang et al.[89], computed from depth data of Kinect, as ground truth for 1449 images and 220K images respectively.

Evaluation Criteria. We compute six statistics, previously used by [6, 24, 31, 32, 33, 89], over the angular error between the predicted normals and depth-based normals to evaluate the performance – **Mean**, **Median**, **RMSE**, **11.25°**, **22.5°**, and **30°** – The first three criteria capture the mean, median, and RMSE of angular error, where lower is better. The last three criteria capture the percentage of pixels within a given angular error, where higher is better.

Qualitative Results. We show two examples in Figure 7 demonstrating where the improvement comes for the surface normal estimation. Note how with multi-scale prediction, the room-layout including painting on the wall improved.

Analysis-1: Global Scene Layout. We follow Bansal et al. [6], and present our results both with and without Manhattan-world rectification to fairly compare against previous approaches, as [31, 32, 89] use it and [24] do not. We rectify our normals using the vanishing point estimates from Hedau et al. [42]. Table 13 compares our approach with existing work. Similar to Bansal et al. [6], our approach performs worse with Manhattan-world rectification (unlike Fouhey et al. [31]) though it improves slightly on this criteria as well.

Analysis-2: Local Object Layout. Bansal et al. [6] stressed the importance of fine details in the scene generally available around objects. We followed their [6] local object evaluation that considers only those pixels which belong to a particular object (such as chair, sofa and bed). Table 14 shows comparison of our approach with Wang et al. [89], Eigen and Fergus [24], and MarrRevisited (Bansal et al. [6]). We consistently improve the performance by **1-3%** on all statistics for all the objects.

C. Edge Detection

In this section, we demonstrate that our same architecture can produce state-of-the-art results for low-level edge detection. The standard dataset for edge detection is BSDS-500 [4], which consists of 200 training, 100 validation, and 200 testing images. Each image is annotated by ∼5 humans to mark out the contours. We use the same augmented data (rotation, flipping, totaling 9600 images without resizing) used to train the state-of-the-art Holistically-nested edge detector (HED) [94]. We report numbers on the testing images. During training, we follow HED and only use positive labels where a consensus (≥ 3 out of 5) of humans agreed.

Baseline. We use the same baseline network that was defined for semantic segmentation, only making use of pre-trained *conv* layers. A sigmoid cross-entropy loss is used to determine the whether a pixel is belonging to an edge or not. Due to the highly skewed class distribution, we also normalized the gradients for positives and negatives in each batch (as in [94]).

Training. We use our previous training strategy, consisting of batches of 5 images with a total sample size of $10,000$ pixels. Each image is randomly resized to half its scale (so 0.5 and 1.0 times) during learning. The initial learning rate is again set to 10^{-3}. However, since the training data is already augmented, we found the network converges much faster than when training for segmentation. To avoid over-training and over-fitting, we reduce the learning rate at 15 epochs and 20 epochs (by a factor of 10) and end training at 25 epochs.

Baseline Results. The results on BSDS, along with other

Model	59-class		33-class	
	AC (%)	*IU* (%)	*AC* (%)	*IU* (%)
FCN-8s [61]	46.5	35.1	67.6	53.5
FCN-8s [62]	50.7	37.8	-	-
DeepLab (v2 [15])	-	37.6	-	-
DeepLab (v2) + CRF [15]	-	39.6	-	-
CRF-RNN [101]	-	39.3	-	-
ParseNet [60]	-	40.4	-	-
ConvPP-8 [93]	-	**41.0**	-	-
baseline (conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$)	44.0	34.9	62.5	51.1
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (0.25,0.5)	46.7	37.1	66.6	54.8
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (0.5)	47.5	37.4	66.3	54.0
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (0.5-1.0)	48.1	37.6	67.3	54.5
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (0.5-0.25,0.5,1.0)	**51.5**	**41.4**	**69.5**	**56.9**

Table 12. Our final results and baseline comparison on PASCAL-Context. Note that while most recent approaches spatial context post-processing [15, 60, 93, 101], we focus on the FCN [62] per-pixel predictor as most approaches are its descendants. Also, note that we (without any CRF) achieve results better than previous approaches. CRF post-processing could be applied to any local unary classifier (including our method). Here we wanted to compare with other local models for a "pure" analysis.

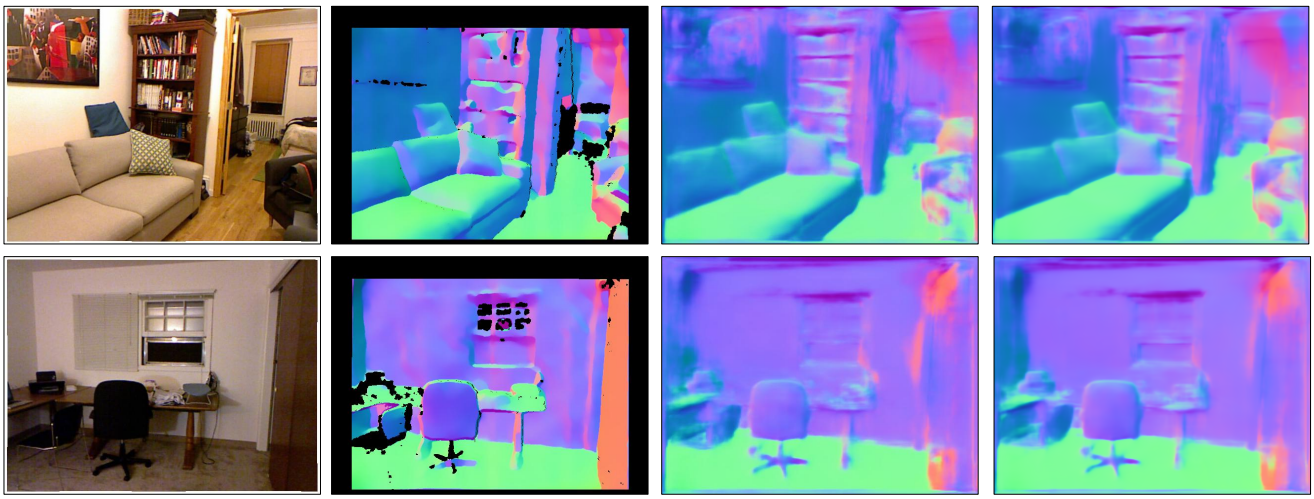


Figure 7. Surface normal map estimated on single 2D images from NYU-v2 depth dataset. We show two examples as how the analysis from pixel-level prediction improved the results from MarrRevisited (Bansal et al. [6]). The normals of painting on wall in first row, and room-layout (left side) of second row improved from previous state-of-the-art.

concurrent methods, are reported in Table 16. We apply standard non-maximal suppression and thinning technique using the code provided by [21]. We evaluate the detection performance using three standard measures: fixed contour threshold (ODS), per-image best threshold (OIS), and average precision (AP).

Analysis-1: Sampling. Whereas uniform sampling sufficed for semantic segmentation [62], we found the extreme rarity of positive pixels in edge detection required focused sampling of positives. We compare different strategies for sampling a fixed number (2000 pixels per image) training examples in Table 15. Two obvious approaches are uniform and balanced sampling with an equal ratio of positives and negatives (shown to be useful for object detection [17, 35]). We tried ratios of 0.25, 0.5 and 0.75, and found that balancing consistently improved performance[6].

Analysis-2: Adding *conv-7*. We previously found that adding features from higher layers is helpful for semantic segmentation. Are such high-level features also helpful for edge detection, generally regarded as a low-level task? To

[6]Note that simple class balancing [94] in each batch is already used, so the performance gain is *unlikely* from label re-balancing.

NYUDv2 test	Mean	Median	RMSE	11.25°	22.5°	30°
Fouhey et al. [31]	35.3	31.2	41.4	16.4	36.6	48.2
E-F (VGG-16) [24]	20.9	13.2	-	44.4	67.2	75.9
UberNet (1-Task) [51]	21.4	15.6	-	35.3	65.9	76.9
MarrRevisited [6]	19.8	12.0	28.2	47.9	70.0	77.8
PixelNet	**19.2**	**11.3**	**27.2**	**49.4**	**70.7**	**78.5**
Manhattan World						
Wang et al. [89]	26.9	14.8	-	42.0	61.2	68.2
Fouhey et al. [32]	35.2	17.9	49.6	40.5	54.1	58.9
Fouhey et al. [31]	36.3	19.2	50.4	39.2	52.9	57.8
MarrRevisited [6]	23.9	11.9	35.9	48.4	66.0	72.7
PixelNet	**23.6**	**11.8**	**35.5**	**48.6**	**66.3**	**73.0**

Table 13. **Evaluation on NYUv2 depth dataset [83]: Global Scene Layout.** We improve the previous state-of-the-art [6] using the analysis from general pixel-level prediction problems, and multi-scale prediction.

NYUDv2 test	Mean	Median	RMSE	11.25°	22.5°	30°
Chair						
Wang et al. [89]	44.7	35.8	54.9	14.2	34.3	44.3
E-F (AlexNet) [24]	38.2	32.5	46.3	14.4	34.9	46.6
E-F (VGG-16) [24]	33.4	26.6	41.5	18.3	43.0	55.1
MarrRevisited [6]	32.0	24.1	40.6	21.2	47.3	58.5
PixelNet	**31.5**	**23.9**	**39.9**	**21.6**	**47.4**	**59.1**
Sofa						
Wang et al. [89]	36.0	27.6	45.4	21.6	42.6	53.1
E-F (AlexNet) [24]	27.0	21.3	34.0	25.5	52.4	63.4
E-F (VGG-16) [24]	21.6	16.8	27.3	32.5	63.7	76.3
MarrRevisited [6]	20.9	15.9	27.0	34.8	66.1	77.7
PixelNet	**20.2**	**15.1**	**26.3**	**37.2**	**67.8**	**78.9**
Bed						
Wang et al. [89]	28.6	18.5	38.7	34.0	56.4	65.3
E-F (AlexNet) [24]	23.1	16.3	30.8	36.4	62.0	72.6
E-F (VGG-16) [24]	19.9	13.6	27.1	43.0	68.2	78.3
MarrRevisited [6]	19.6	13.4	26.9	43.5	69.3	79.3
PixelNet	**19.0**	**13.1**	**26.1**	**44.3**	**70.7**	**80.6**

Table 14. **Evaluation on NYUv2 depth dataset [83]: Local Object Layout.** We consistently improve on all metrics for all the objects. This suggest that our approach is able to better capture the fine details present in the scene.

	ODS	OIS	AP
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ *Uniform*	.767	.786	.800
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ (25%)	.792	.808	.826
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ (50%)	.791	.807	.823
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ (75%)	.790	.805	.818

Table 15. Comparison of different sampling strategies during training. *Top row:* Uniform pixel sampling. *Bottom rows:* Biased sampling of positive examples. We sample a fixed percentage of positive examples (25%,50% and 75%) for each image. Notice a significance difference in performance.

answer this question, we again concatenated *conv-7* features

	ODS	OIS	AP
Human [4]	.800	.800	-
Canny	.600	.640	.580
Felz-Hutt [29]	.610	.640	.560
gPb-owt-ucm [4]	.726	.757	.696
Sketch Tokens [58]	.727	.746	.780
SCG [92]	.739	.758	.773
PMI [47]	.740	.770	.780
SE-Var [22]	.746	.767	.803
OEF [39]	.749	.772	.817
DeepNets [50]	.738	.759	.758
CSCNN [44]	.756	.775	.798
HED [94]	.782	.804	.833
HED [94] (Updated version)	.790	.808	.811
HED merging [94] (Updated version)	.788	.808	**.840**
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ (50%)	.791	.807	.823
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (50%)	**.795**	**.811**	.830
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ (25%)-(0.5×,1.0×)	.792	.808	.826
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (25%)-(0.5×,1.0×)	**.795**	**.811**	.825
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ (50%)-(0.5×,1.0×)	.791	.807	.823
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (50%)-(0.5×,1.0×)	**.795**	**.811**	.830
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (25%)-(1.0×)	.792	.808	.837
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (50%)-(1.0×)	.791	.803	**.840**

Table 16. Evaluation on BSDS [4]. Our approach performs better than previous approaches *specifically* trained for edge detection.

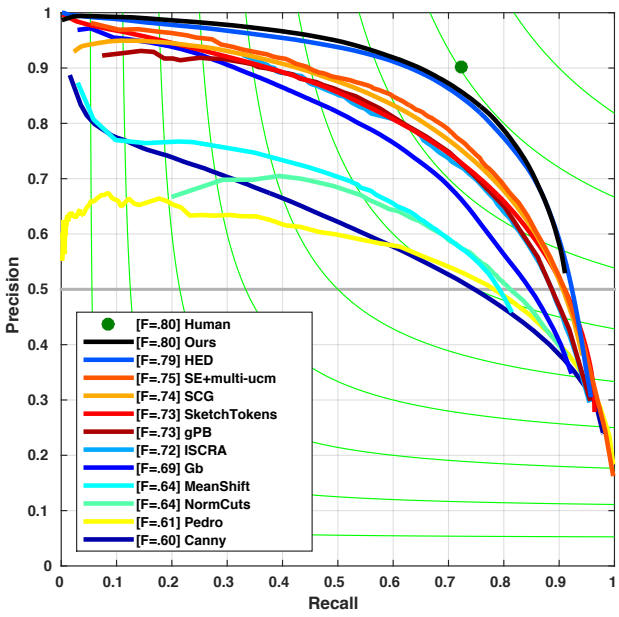


Figure 8. Results on BSDS [4]. While our curve is mostly overlapping with HED, our detector focuses on more high-level semantic edges. See qualitative results in Fig.9.

with other *conv* layers $\{ 1_2, 2_2, 3_3, 4_3, 5_3 \}$. Please refer to the results at Table 16, using the second sampling strategy. We find it still helps performance a bit, but not as significantly for semantic segmentation (clearly a high-level task). Our final results as a single output classifier are very com-

Figure 9. Qualitative results for edge detection. Notice that our approach generates more semantic edges for *zebra*, *eagle*, and *giraffe* compared to HED [94]. Best viewed in the electronic version.

petitive to the state-of-the-art.

Qualitatively, we find our network tends to have better results for semantic-contours (*e.g.* around an object), particularly after including *conv-7* features. Figure 9 shows some qualitative results comparing our network with the HED model. Interestingly, our model explicitly removed the edges inside the *zebra*, but when the model cannot recognize it (*e.g.* its head is out of the picture), it still marks the edges on the black-and-white stripes. Our model appears to be making use of much higher-level information than past work on edge detection.

Note to Readers: An earlier version of this work appeared on arXiv[7]. We have incorporated extensive analysis to un-

[7] https://arxiv.org/pdf/1609.06694.pdf

derstand the underlying design principles of general pixel-prediction tasks, and how they can be trained from scratch. We will release the source code and required models on our project page.

Acknowledgements: This work was in part supported by NSF Grants IIS 0954083, IIS 1618903, and support from Google and Facebook, and Uber Presidential Fellowship to AB.

References

[1] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *ICCV 2015*, pages 37–45, 2015. 6

[2] P. Agrawal, R. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*. 2014. 7

[3] P. Arbeláez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik. Semantic segmentation using regions and parts. In *CVPR*. IEEE, 2012. 2

[4] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *TPAMI*, 2011. 2, 7, 8, 9, 12, 13

[5] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 92(1), 2011. 1

[6] A. Bansal, B. Russell, and A. Gupta. Marr Revisited: 2D-3D model alignment via surface normal prediction. In *CVPR*, 2016. 1, 3, 4, 7, 8, 11, 12, 13

[7] A. Bansal, A. Shrivastava, C. Doersch, and A. Gupta. Mid-level elements for object detection. *CoRR*, abs/1504.07284, 2015. 7

[8] J. T. Barron and B. Poole. The fast bilateral solver. *CoRR*, abs/1511.03296, 2015. 2

[9] C. M. Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995. 3

[10] A. Bordes, L. Bottou, and P. Gallinari. Sgd-qn: Careful quasi-newton stochastic gradient descent. *JMLR*, 10, 2009. 3

[11] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010. 1, 3

[12] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki. Scene labeling with lstm recurrent neural networks. In *CVPR*, 2015. 2

[13] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1611.08050*, 2016. 1

[14] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*. 2012. 2

[15] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016. 7, 11

[16] 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. 1, 2, 10, 11

[17] X. Chen and A. Gupta. An implementation of faster rcnn with study for region sampling. *arXiv preprint arXiv:1702.02138*, 2017. 13

[18] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, et al. Large scale distributed deep networks. In *NIPS*, 2012. 3

[19] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015. 2

[20] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *International Conference on Computer Vision (ICCV)*, 2015. 6, 7

[21] P. Dollár and C. Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013. 1, 2, 13

[22] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *TPAMI*, 37(8), 2015. 8, 12

[23] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *CoRR*, abs/1605.09782, 2016. 6

[24] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 1, 2, 4, 8, 11, 12

[25] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014. 1, 2

[26] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *IJCV*, 2010. 2, 4, 7, 9, 11

[27] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *TPAMI*, 35(8), 2013. 1, 2

[28] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9), 2010. 7

[29] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2), 2004. 12

[30] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015. 1, 2

[31] D. F. Fouhey, A. Gupta, and M. Hebert. Data-driven 3D primitives for single image understanding. In *ICCV*, 2013. 4, 8, 11, 12

[32] D. F. Fouhey, A. Gupta, and M. Hebert. Unfolding an indoor origami world. In *ECCV*, 2014. 4, 11, 12

[33] D. F. Fouhey, A. Gupta, and M. Hebert. Single image 3D without a single 3D image. In *ICCV*, 2015. 4, 11

[34] R. Garg, B. G. V. Kumar, G. Carneiro, and I. D. Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *ECCV*, pages 740–756, 2016. 6

[35] R. Girshick. Fast r-cnn. In *ICCV*, 2015. 7, 13

[36] G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik. Using k-poselets for detecting people and localizing their keypoints. 2014. 1

[37] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun. Unsupervised learning of spatiotemporally coherent metrics. In *ICCV 2015*, pages 4086–4093, 2015. 6

[38] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *ICCV*. IEEE, 2009. 2

[39] S. Hallman and C. C. Fowlkes. Oriented edge forests for boundary detection. In *CVPR*, 2015. 8, 12

[40] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 1, 2, 5, 11

[41] B. Hariharan, P. Arbelez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011. 4

[42] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009. 11

[43] L. Huang, Y. Yang, Y. Deng, and Y. Yu. Densebox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*, 2015. 1

[44] J.-J. Hwang and T.-L. Liu. Pixel-wise deep learning for contour detection. *arXiv preprint arXiv:1504.01989*, 2015. 8, 12

[45] A. Hyvärinen, J. Hurri, and P. O. Hoyer. *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision.*, volume 39. Springer Science & Business Media, 2009. 1, 3

[46] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In D. Blei and F. Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 448–456. JMLR Workshop and Conference Proceedings, 2015. 3, 5

[47] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson. Crisp boundary detection using pointwise mutual information. In *Computer Vision–ECCV 2014*, pages 799–814. Springer, 2014. 12

[48] D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. In *ICCV 2015*, pages 1413–1421. 6

[49] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACMMM*, 2014. 6, 9

[50] J. J. Kivinen, C. K. Williams, N. Heess, and D. Technologies. Visual boundary prediction: A deep neural prediction network and quality dissection. In *AISTATS*, volume 1, page 9, 2014. 8, 12

[51] I. Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. *CoRR*, abs/1609.02132, 2016. 8, 12

[52] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011. 2

[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 4

[54] L. Ladicky, B. Zeisl, and M. Pollefeys. Discriminatively trained dense surface normal estimation. In *ECCV*, 2014. 4, 11

[55] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. In *European Conference on Computer Vision (ECCV)*, 2016. 3, 6

[56] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012. 1

[57] D. Li, W.-C. Hung, J.-B. Huang, S. Wang, N. Ahuja, and M.-H. Yang. Unsupervised visual representation learning by graph-based consistent constraints. In *ECCV*, 2016. 6

[58] J. Lim, C. Zitnick, and P. Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *CVPR*, 2013. 12

[59] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing via label transfer. *TPAMI*, 33(12), 2011. 2

[60] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015. 2, 5, 10, 11

[61] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. 7, 11

[62] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional models for semantic segmentation. In *CVPR*, 2015. 1, 2, 3, 4, 5, 7, 9, 10, 11, 13

[63] J. M. M. Holschneider, R. Kronland-Martinet and P. Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. *In Wavelets, Time-Frequency Methods and Phase Space*, pages 289–297, 1989. 2

[64] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *TPAMI*, 26(5), 2004. 1

[65] O. Matan, C. J. Burges, Y. LeCun, and J. S. Denker. Multi-digit recognition using a space displacement neural network. In *NIPS*, pages 488–495, 1991. 1, 2

[66] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and Learn: Unsupervised Learning using Temporal Order Verification. In *ECCV*, 2016. 6

[67] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *CVPR*, pages 3376–3385, 2015. 1, 2, 11

[68] 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. 2, 9

[69] D. Munoz, J. A. Bagnell, and M. Hebert. Stacked hierarchical labeling. In *Computer Vision–ECCV 2010*, pages 57–70. Springer, 2010. 2

[70] S. Nickell, C. Kofler, A. P. Leis, and W. Baumeister. A visual approach to proteomics. *Nature reviews Molecular cell biology*, 7(3):225–230, 2006. 6

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

[72] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016. 6

[73] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba. Ambient sound provides supervision for visual learning. In *ECCV*, pages 801–816, 2016. 6

[74] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 6

[75] P. H. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene parsing. In *ICML*, 2014. 2

[76] L. Pinto, D. Gandhi, Y. Han, Y. Park, and A. Gupta. The curious robot: Learning visual representations via physical interactions. In *ECCV*, pages 3–18, 2016. 6

[77] J. C. Platt and R. Wolf. Postal address block location using a convolutional locator network. In *NIPS*, 1993. 1, 2

[78] D. Ramanan. Learning to parse images of articulated bodies. In *NIPS*. 2007. 1

[79] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 2015. 5, 6

[80] C. Russell, P. Kohli, P. H. Torr, et al. Associative hierarchical crfs for object class image segmentation. In *ICCV*. IEEE, 2009. 2

[81] A. Saxena, S. H. Chung, and A. Y. Ng. 3-d depth reconstruction from a single still image. *IJCV*, 76(1), 2008. 1

[82] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *Int. Journal of Computer Vision (IJCV)*, January 2009. 1, 2

[83] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 2, 4, 8, 9, 11, 12

[84] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 4, 6

[85] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1), 2014. 6, 9

[86] D. Teney and M. Hebert. Learning to extract motion from videos in convolutional neural networks. *CoRR*, abs/1601.07532, 2016. 1

[87] J. Tighe and S. Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. In *Computer Vision–ECCV 2010*, pages 352–365. Springer, 2010. 2

[88] Z. Tu and X. Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *TPAMI*, 32(10), 2010. 2

[89] X. Wang, D. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *CVPR*, 2015. 1, 2, 4, 11, 12

[90] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV 2015*, pages 2794–2802, 2015. 6

[91] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *CVPR*, 2016. 1

[92] R. Xiaofeng and L. Bo. Discriminatively trained sparse code gradients for contour detection. In *NIPS*, 2012. 12

[93] S. Xie, X. Huang, and Z. Tu. Convolutional pseudo-prior for structured labeling. *arXiv preprint arXiv:1511.07409*, 2015. 2, 7, 10, 11

[94] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015. 1, 2, 8, 12, 13, 14

[95] M. Xu and F. Alber. Automated target segmentation and real space fast alignment methods for high-throughput classification and averaging of crowded cryo-electron subtomograms. *Bioinformatics*, 29(13):i274–i282, 2013. 6

[96] J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *CVPR*. IEEE, 2012. 2

[97] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. 2

[98] G. Zhang, J. Jia, T.-T. Wong, and H. Bao. Consistent depth maps recovery from a video sequence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(6):974–988, 2009. 6

[99] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. *ECCV*, 2016. 6

[100] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. *arXiv*, 2016. 6

[101] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015. 2, 7, 10, 11