ParseNet: Looking Wider to See Better

Wei Liu
UNC Chapel Hill
wliu@cs.unc.edu

Andrew Rabinovich
MagicLeap Inc.
arabinovich@magicleap.com

Alexander C. Berg
UNC Chapel Hill
aberg@cs.unc.edu

Abstract

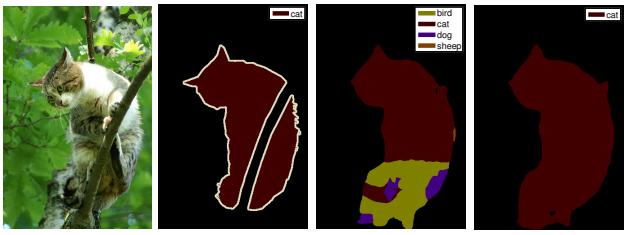
We present a technique for adding global context to deep convolutional networks for semantic segmentation. The approach is simple, using the average feature for a layer to augment the features at each location. In addition, we study several idiosyncrasies of training, significantly increasing the performance of baseline networks (e.g. from FCN [19]). When we add our proposed global feature, and a technique for learning normalization parameters, accuracy increases consistently even over our improved versions of the baselines. Our proposed approach, ParseNet, achieves state-of-the-art performance on SiftFlow and PASCAL-Context with small additional computational cost over baselines, and near current state-of-the-art performance on PASCAL VOC 2012 semantic segmentation with a simple approach. Code is available at https://github.com/weiliu89/caffe/tree/fcn.

1. Introduction

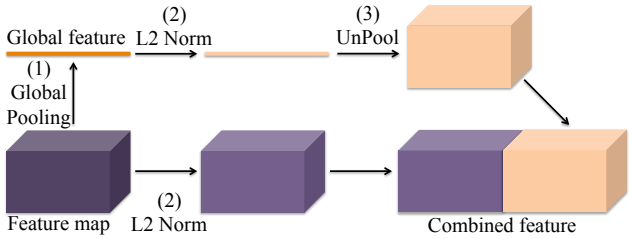
Image segmentation and object recognition have been long standing problems in psychology, computer vision and machine learning. Semantic segmentation, largely studied in the last 10 years, merges segmentation with recognition to produce per pixel semantic labeling of images. Recently, with the emergence of deep learning, highly promising approaches have emerged and delivered encouraging results on this problem. In this work, we design an end-to-end simple and effective convolutional neural network, *ParseNet*, for semantic segmentation. This is complemented by extensive experiments on the relevant choices in design and training methodology.

To achieve per pixel labeling, as opposed to per image labeling, fully convolution network (FCN), adapted from the image level network [15, 27, 26], was introduced and demonstrated impressive level of performance [19]. FCN, in its original incarnation, is oblivious to context. In particular, FCN disregards global information about an image, thus ignoring potentially useful scene-level information of semantic context. Two similar looking patches are indistinguishable by the network if considered in isolation. However, by adding contextual information, as shown in Fig. 1, it can greatly help clarify local confusions. In classical computer vision, before the current deep-learning revolution, such context was studied in detail [29, 23]. There, context information was incorporated as a post processing step. Starting from the FCN architecture, several approaches [2, 24, 18, 31], combining graphic model such as conditional random field (CRF), have been proposed to introduce global context and structured information into a FCN. Although these methods are powerful, they require an elevated amount of *black magic* (expertise in managing the idiosyncrasies of training methodology and parameters) to train successful networks, and may lead to more complex architecture and to more difficult or at least more time-consuming training and inference.

As one of our goals is to develop a simple and robust architecture for semantic segmentation, we revisit global context by terms of aggregating local features over the whole



(a) Image (b) Ground truth (c) FCN output (d) ParseNet output



(e) ParseNet contexture module overview.

Figure 1: ParseNet uses the extra global context feature to help clarify the local confusion.

image. Previous research [20] shows that by concatenating feature from the whole image with features on local patches, it is not necessary to apply CRF smoothing afterwards, as the whole image feature already encodes the smoothness information. [28, 21] show that simply adding such context features helps in the detection and segmentation tasks. However, these approaches use separate (image/patch) network to do it, not trained jointly.

For our network, due to the FCN properties, we can directly pool out whole image feature from a feature map of our network and combine it with each individual position, as shown in Fig. 1, and the whole system can be trained end-to-end. We can apply this technique to any feature maps within a network if that is desired. Notice that features from some layers have much larger scale (in terms of the values in each dimension) than those in other layers, making it difficult to directly combine them for prediction. Instead, we first l_2 normalize features from each layer, followed by applying an appropriate scaling factor, which is also learnt through backpropagation. In sections 4 we demonstrate that these simple operations, by adding global feature pooled from a feature map with appropriate scaling, are sufficient to significantly improve performance over the basic FCN architecture, resulting in accuracy on par with methods [2] that use detailed structure information for post processing. That said, we do not advocate ignoring other structure information, instead we posit that adding global feature is a simple and robust method to improve FCN performance by considering contextual information. In fact, our network can be combined with explicit structure output prediction, e.g. a CRF, to potentially further increase performance.

The rest of the paper is organized as follows. In Section 2 we review the related work. Our proposed approach is described in Section 3 followed by extensive experimental validation in Section 4. We conclude our work and describe future directions in Section 5.

2. Related Work

Attributed to the large learning capacity of deep neural networks [15, 27, 26], convolutional neural networks (CNN) have become a powerful tool not only for whole image classification, but also for object detection and semantic segmentation [4, 27, 28, 9, 6]. Followed by the *proposal + post-classification* scheme [30, 1], CNNs achieve state-of-the-art results on object detection and segmentation tasks. As a caveat, even though a single pass through the networks used in these systems is approaching or already past video frame rate for individual patches, these approaches require classifying hundreds or thousands of patched per image, and thus are still slow. [10, 19] improve the computation by applying convolution to the whole image once, and then pool features from the final feature map of the network for each region proposal or pixel to achieve comparable or even better results. Yet, these methods still fall short of including whole image context and only classify patches or pixels locally. Our ParseNet is built upon the fully convolutional network architecture [19] with a strong emphasis on including contextual information in a simple approach.

For tasks such as semantic segmentation, using context information [23, 25, 14] from the whole image can significantly help classifying local patches. [20] shows that by concatenating features from the whole image to the local patch, the inclusion of post processing (i.e. CRF smoothing) becomes unnecessary; the image level features already encode the smoothness. Along this line, [28] verify that by concatenating features extracted from a whole image to the features for each proposal, they can greatly improve the object detection performance. [21] demonstrate that by using the "zoom-out" features, which is a combination of features for each super pixel, region surrounding it, and the whole image, they can achieve impressive performance for the semantic segmentation task. [8] goes further by combining features from different layers of the network to do the prediction.

The above mentioned approaches pool features differently for local patches and the whole image, making it difficult to train the whole system end-to-end. Exploiting the FCN architecture, ParsetNet can directly use global average pooling from the final (or any) feature map, resulting in the feature of the whole image, and use it as context. Our experiments (Fig. 3) confirm that ParseNet can capture the context of the image and thus improve local patch prediction results.

Notice that there is another line of work that attempts to combine graphical models with CNNs to incorporate both context and smoothness priors. [2] first uses a FCN to estimate the unary potential, then applies a fully connected CRF to smooth the predictions spatially. As this approach consists of two decoupled stages, it is difficult to train the FCN properly to minimize the final objective of smooth and accurate semantic segments. A more unified and principled approach is to incorporate the structure information during training directly. [24] propagates the marginals computed from the structured loss to update the network parameters, [18] uses piece-wise training to make learning more efficient by adding a few extra piece-wise networks, while [31] convert CRF learning to recurrent neural network (RNN) and use message passing to do the learning and inference. However, we show that our method can achieve comparable accuracies, with a simpler – hence more robust – structure, while requiring only a small amount of additional training/inference time.

3. ParseNet

3.1. Global Context

Context is known to be very useful for improving performance on detection and segmentation tasks in general, and using deep learning in particular. [21, 28] and references therein, illustrate how context can be used to help in different tasks. As for semantic segmentation, it is essentially doing per pixel or per patch classification, which is difficult if only local information used. However, the task becomes much simpler if we can also provide the classifier with contextual information about the whole image. Although theoretically, features from higher level layers of a network have very large receptive fields (e.g. fc7 in FCN with VGG has a 404×404 pixels receptive field), in practice the size of receptive fields at higher levels is much smaller. As shown in [32], the actual sizes are about 25% of the theoricatical ones at higher levels, thus preventing the model from making global decisions. Explicitly adding features from the whole image is needed and is rather straightforward within the FCN architecture. Specifically, we use global average pooling and pool the context features from the last layer or any layer if that is desired. The quality of semantic segmentation is greatly improved by adding the global feature to local feature map, either with early fusion [1] or late fusion as discussed in Sec. 3.2. For example, Fig 1 has misclassified a large portion of the image as bird since it only used local information, however, adding contextual information in the loop, which might contain strong signal of cat, corrects the mistake. Experiment results on VOC2012 and PASCAL-Context dataset also verify our assumption. Compared with [2], the improvement is similar as of using CRF to postprocess the output of FCN.

In addition, we also tried to follow the spatial pyramid idea [17] to pool features from increasingly finer subregions and attach them to local features in the sub-regions, however, we did not observe significant improvements. We conjecture that it is because the receptive field of high level feature maps is larger than those sub-regions. However features pooled from the whole image is still beneficial.

3.2. Early Fusion and Late Fusion

Once we get the global context feature, there are two general standard paradigms of using it with the local feature map. First, the *early fusion*, illustrated in in Fig. 1 where we unpool (replicate) global feature to the same size as of local feature map spatially and then concatenate them, and use the combined feature to learn the classifier. The alternative approach, is *late fusion*, where each feature is used to learn its own classifier, followed by merging the two predictions into a single classification score [19, 2].

[1] we use unpool operation by simply replicating the global feature horizontally and vertically to have the same size as the local feature map.

There are cons and pros for both fusion methods. If there are no extra operation with the combined feature, *early fusion* is quite similar to *late fusion* as pointed out in [8]. With *late fusion*, there might be case when both individual feature cannot recognize but combining them might and there is no way to recover it once each makes the predictions. There are strategies, such as double fusion [16], to combine strength from both. However, we do not explore much in this paper but leave it for future work. Our experiments show that both method works more or less same if we normalize the feature properly for early fusion case.

With *early fusion*, we can add extra capacity, nonlinearity or dimensionality reduction (1×1 convolution layer), to apply nonlinear transformations to the combined feature. Notice this leads to increases memory and computation. Table 7 compares different strategies of combining the features, and it turns out proper l_2 normalization works better than a 1×1 convolution layer.

While merging the features, one must be careful to normalize each individual feature to make the combined feature work well; in classical computer vision this is referred as the cue combination problem. As shown in Fig. 2, we extract a feature vector at a position combined from increasing higher level layers (from left to right), with lower level feature having a significantly larger scale than higher level layers. As we show in Sec. 4.2, by naively combining features, the resultant feature will not be discriminative, and heavy parameter tuning will be required to achieve sufficient accuracy. Instead, we can first l_2 normalize each feature and also possibly learn the scale parameter, which makes the learning more stable. We will describe more details in Sec. 3.3.

3.3. l_2 Normalization Layer

As discussed above and shown in Fig. 2, it is necessary to combine two (or more) feature vectors, which generally have different scale and norm. Naively concatenating features leads to poor performance as the "larger" features dominate the "smaller" ones. Although during training, the weight might adjust accordingly, it requires very careful tuning of parameters and depends on dataset, thus goes against the robust principle. For example, as shown in Table 4, on PASCAL-Context dataset, we can combine feature upto conv4 and achieve state-of-the-art performance. However, if we further add conv3, the network collapses. We find that by normalizing each individual feature first, and also possibly learn to scale each differently, it makes the training more stable and sometimes improves performance.

l_2 norm layer is not only useful for feature combination. As was pointed out above, in some cases —late fusion also works equally well, but only with the help of l_2 normalization. For example, if we want to use lower level feature to learn classifier, as demonstrated in Fig. 2, some of the features will have very large norm. It is not trivial to learn

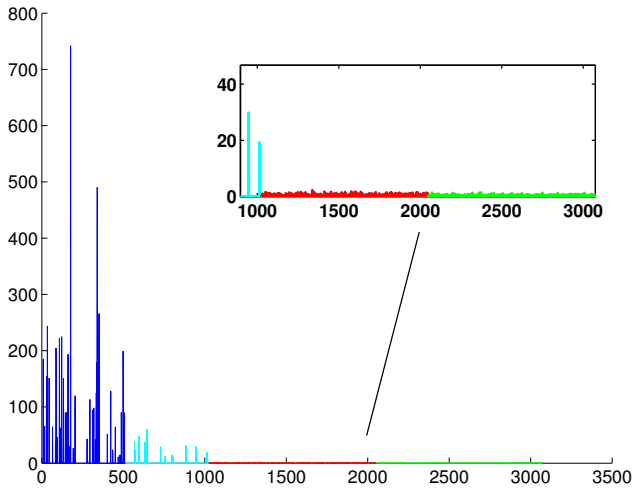


Figure 2: Features are in different scale. We show the features for a position from conv4_3, conv5_3, fc7 and pool6 when we concatenate them together.

with it without careful weight initialization and parameter tuning. A work around strategy is to apply an additional convolutional layer [2, 8] and use several stages of finetuning [19] with much lower learning rate for lower layer. This again goes against the principle of simply and robustness. In our work, we apply l_2-norm and learn the scale parameter for each channel before using the feature for classification, which leads to more stable training.

Formally, let ℓ be the loss we want to minimize. Here we use the summed softmax loss. For a layer with d-dimensional input $\mathbf{x} = (x_1 \cdots x_d)$, we will normalize it using l_2-norm[2] with Eq. 1

$$\hat{\mathbf{x}} = \frac{\mathbf{x}}{||\mathbf{x}||_2} \tag{1}$$

where $||\mathbf{x}||_2$ is the l_2 norm of $\mathbf{x}$ as defined in Eq. 2

$$||\mathbf{x}||_2 = \Big(\sum_{i=1}^{d} |x_i|^2\Big)^{1/2} \tag{2}$$

Note that simply normalizing each input of a layer changes the scale of the layer and will slow down the learning if we do not scale it accordingly. For example, we tried to normalize a feature s.t. l_2-norm is 1, yet we can hardly train the network because the features become very small. However, if we normalize it to e.g. 10 or 20, the network begins to learn well. Motivated by batch normalization [12] and PReLU [11], we introduce a scaling parameter γ_i, for each channel, which scales the normalized value:

$$y_i = \gamma_i \hat{x}_i \tag{3}$$

[2]We have only tried l_2 norm, but can also potentially try other l_p norms.

The number of extra parameters is equal to total number of channels, and are negligible and can be learned with backprogation. Indeed, by setting $\gamma_i = ||\mathbf{x}||_2$, we could recover the l_2 normalized feature, if that was optimal. Notice that this is simple to implement as the normalization and scale parameter learning only depend on each input feature vector and do not need to aggregate information from other samples as batch normalization does. We notice that by introducing the scaling factor, we can boost the performance significantly when combining the global context feature with the local feature map on PASCAL VOC2012.

During training, we use backpropagation and chain rule to compute derivatives with respect to scaling factor γ and input data $\mathbf{x}$

$$\frac{\partial \ell}{\partial \hat{\mathbf{x}}} = \frac{\partial \ell}{\partial \mathbf{y}} \cdot \gamma \tag{4}$$

$$\frac{\partial \ell}{\partial \mathbf{x}} = \frac{\partial \ell}{\partial \hat{\mathbf{x}}} \Big(\frac{\mathbf{I}}{||\mathbf{x}||_2} - \frac{\mathbf{x}\mathbf{x}^T}{||\mathbf{x}||_2^3} \Big) \tag{5}$$

$$\frac{\partial \ell}{\partial \gamma_i} = \sum_{y_i} \frac{\partial \ell}{\partial y_i} \hat{x}_i \tag{6}$$

For our case, we need to do l_2-norm per each pixel in a feature map instead of the whole. We can easily extend the equations by doing it elemental wise as it is efficient.

4. Experiment

In this section, we mainly report results on two benchmark datasets: VOC2012 [3] and PASCAL-Context [22]. VOC2012 has 20 object classes and one background class. Following [19, 2], we augment it with extra annotations from Hariharan *et al.* [7] that leads to 10,582, 1,449, and 1,456 images for training, validation, and testing. PASCAL-Context [22] fully labeled all scene classes appeared in VOC2010. We follow the same training + validation split as defined and used in [22, 19], resulting in 59 object + stuff classes and one background classes with 4,998 and 5105 training and validation images. All the results we describe below use the training images to train, and most of the results are on the validation set. We also use the PASCAL official evaluation server to report results on VOC2012 test set. We use Caffe [13] and fine-tune ParseNet from VGG-16 network [26] for different dataset.

4.1. Best practice of finetuning

First, we try to reproduce the state-of-the-art systems' results. As we know parameters are important for training/finetuning network, we tried to explore the parameter space and achieve better baseline performance.

PASCAL-Context We start from the public system FCN-32s PASCAL-Context. Notice that it uses the accumulated gradient and affine transformation tricks that were introduced in [19]. As such, it can deal with any input image

of various sizes without warping or cropping it to fixed size, which can distort the image and affect the final segmentation result. Table 1 shows our different versions of reproduced baseline results. Baseline A uses the exactly same protocol, and our result is 1.5% lower. In Baseline B, we tried more iteration (160k vs. 80k) of finetuning and achieved similar performance to the reported one. Then, we modified the network a bit, i.e. we used "xavier" initialization [5], higher base learning rate (1e-9 vs. 1e-10), and lower momentum (0.9 vs. 0.99), and we achieved 1% higher accuracy as shown in Baseline C. What's more, we also remove the 100 padding in the first convolution layer and observed no significant difference but network trained slightly faster. Furthermore, we also used "poly" learning rate policy (base_lr$\times(1-\frac{\text{iter}}{\text{max_iter}})^{\text{power}}$, where power is set to 0.9.) as it is proved to converge faster than "step", and thus can achieve 1.5% better performance with the same iterations (80k). All experimental results on PASCAL-Context are shown in table 1.

PASCAL-Context	Mean IoU
FCN-32s [3]	35.1
Baseline A	33.57
Baseline B	35.04
Baseline C	36.16
Baseline D	**36.64**

Table 1: Reproduce FCN-32s on PASCAL-Context. There are various modifications of the arcitecture that are described in Section 4.1.

PASCAL VOC2012 We carry over our experience and parameters from training models on PASCAL-Context to VOC2012. We tried both FCN-32s and DeepLab-LargeFOV[4]. Table 2 shows the reproduced baseline results. DeepLab is very similar to FCN-32s, and our reproduced result is 5% better (64.96 vs. 59.80) using the parameters we used in PASCAL-Context. DeepLab-LargeFOV uses the filter rarefication technique (atrous algorithm) that has much less parameters and is faster. We also use the same parameters on this architecture and can achieve 3.5% improvements. The gap between these two models is not significant anymore as reported in [2]. Later on, we renamed DeepLab-LargeFOV Baseline as ParseNet Baseline, and ParseNet is ParseNet Baseline plus global context.

Until now, we see that parameters and details are important to get best performance using FCN models. Below, we report all our results with the reproduced baseline networks.

[3] https://gist.github.com/shelhamer/80667189b218ad570e82#file-readme-md

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

VOC2012	Mean IoU
DeepLab [2]	59.80
DeepLab-LargeFOV [2]	62.25
DeepLab Baseline	64.96
DeepLab-LargeFOV Baseline	**65.82**

Table 2: Reproduce DeepLab and DeepLab-LargeFOV on PASCAL VOC2012.

4.2. Combining Local and Global Features

In this section, we report results of combining global feature and local feature on three dataset: SiftFlow, PASCAL-Context, and PASCAL VOC2012. For simplicity, we use pool6 as the global context feature, conv5 as conv5_3, conv4 as conv4_3, and conv3 as conv3_3 through the rest of paper. **SiftFlow** is a relatively small dataset that only has 2,688 images with 33 semantic categories. We do not use the geometric categories during training. We use the FCN-32s network with the parameters found in PASCAL-Context. Instead of using two stages of learning as done in [19], we combine the feature directly from different layers for learning. As shown in Table 3, adding more layers can normally improve the performance as lower level layers have more detailed information. We also notice that adding global context feature does not help much. This is perhaps due to the small image size (256×256), as we know even the empirical receptive field size of fc7 in FCN is about the same (e.g. 200×200), thus pool6 is essentially a noop.

	pixel acc.	mean acc.	mean IU	f.w. IU
FCN-16s [19]	85.2	51.7	39.5	76.1
fc7	85.1	44.1	35.4	75.6
pool6 + fc7	85.7	43.9	35.5	76.4
pool6 + fc7 + conv5	85.4	51.4	38.7	76.3
pool6 + fc7 + conv5 + conv4	**86.8**	**52.0**	**40.4**	**78.1**

Table 3: Results on SiftFlow. Using early fusion can equally working as well as late fusion as used in [19]. Adding more layers of feature generally increase the performance. Global feature is not that helpful for SiftFlow as receptive field size of fc7 is large enough to cover most of the input image.

PASCAL-Context We then apply the same model on PASCAL-Context by concatenating features from different layers of the network. As shown in Table 4, by adding global context pool6, it instantly helps improve by about 1.5%, which means that context is useful here as opposed to the observation in SiftFlow. Context becomes more important proportionally to the image size. Another interesting observation from the table is that, without normalization, the performance keep increasing until we add conv5. How-

ever, if we naively keep adding conv4, it starts decreasing the performance a bit; and if we add conv3, the network collapses. Interestingly, if we normalize all the features before we combine them, we don't see such a drop, instead, adding all the feature together can achieve the state-of-the-art result on PASCAL-Context as far as we know.

	w/o Norm	w/ Norm
FCN-32s	36.6	N/A
FCN-8s	37.8	N/A
fc7	36.6	36.2
pool6 + fc7	38.2	37.6
pool6 + fc7 + conv5	39.5	39.9
pool6 + fc7 + conv5 + conv4	36.5	40.2
pool6 + fc7 + conv5 + conv4 + conv3	0.009	**40.4**

Table 4: Results on PASCAL-Context. Adding more layers helps if we normalize them accordingly beforehand.

PASCAL VOC2012 Since we have reproduced both network architecture on VOC2012, we want to see how does global context, normalization, and early or late fusion affect performance.

We start with using DeepLab Baseline, and try to add pool6 to it. As shown in Table 5 that it improve 2.6% by adding pool6 with normalization. Interestingly, without normalizing fc7 and pool6, we don't see any improvements. As opposed to what we observed from SiftFlow and PASCAL-Context. We hypothesize this is due to images in VOC2012 mostly have one or two objects in the image versus the other two datasets who have multiple labels per image, and we need to adjust the weight more carefully to make the context feature more useful.

DeepLab Baseline	w/o Norm	w/ Norm
fc7	64.96	64.92
pool6 + fc7	64.84	**67.49**

Table 5: Adding context with normalization of features helps on VOC2012 with DeepLab Baseline.

ParseNet Baseline performance is higher than DeepLab Baseline and it is faster, thus we switch to use it for most of the experimental comparison for VOC2012. As shown in Table 6, we observe a similar pattern as of DeepLab Baseline that if we add pool6, it is helping improve the performance by 3.61%. However, we also notice that if we do not normalize them and learn the scaling factors, its effect is diminished. Furthermore, we notice that early fusion and late fusion both work very similar. Figure 3 illustrates some examples of how global context helps. We can clearly see that without using context feature, the network will make many mistakes by confusing between similar categories as well as making spurious predictions. However, adding context solves this issue as the global context helps discriminate the local patches more accurately. On the other hand, sometimes context also brings confusion for prediction as shown in Figure 4. For example, in the first row, the global context feature definitely captured the spotty dog information that it used to help discriminate sheep from dog. However, it also added bias to classify the spotty horse as a dog. The other three examples have the same issue. Overall, by learning to weight pool6 and fc7 after l_2 normalization helps improve the performance greatly.

Layers	Norm (Y/N)	Early or Late (E/L)	Mean IoU
fc7	N	NA	65.82
fc7	Y	NA	65.66
pool6 + fc7	N	E	65.30
pool6 + fc7	Y	E	69.43
pool6 + fc7	Y	L	**69.55**
pool6 + fc7	N	L	69.29

Table 6: Add context for ParseNet Baseline on VOC2012, compare w/ or w/o normalization, and early/late fusion.

Table 7 compares different strategy of combining fc7 and pool6. We notice that if we normalize the feature that l_2-norm is 1, and simply concatenate them together, the results decrease. Perhaps, using a higher learning rate for the classification layer may improve the performance, but we did not carry this experiment out. However, if we apply 1×1 convolution to combine these two features, the performance is the same as if we did not normalize the features and is better than concatenating fc7 and pool6 directly (65.30). Yet, the best results comes from normalizing both features but scale them to 10 and adjust the scale during training, increasing the accuracy by 3% over the 1×1 convolution approach.

Norm	1x1 conv	concatenate
1	66.7	62.58
10	58.87	**69.43**
NA	66.39	65.30

Table 7: Compare different ways of combining features for early fusion.

We also tried to combine lower level feature as was done with PASCAL-Context and SiftFlow, but no significant improvements using either *early fusion* or *late fusion* were observed. We believe it is because the fc7 of ParseNet Baseline is the same size as of conv4, and including lower level feature will not help much as they are not sufficiently discriminative.

System	bkg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	*mean*
FCN-8s[19]	-	76.8	34.2	68.9	49.4	60.3	75.3	74.7	77.6	21.4	62.5	46.8	71.8	63.9	76.5	73.9	45.2	72.4	37.4	70.9	55.1	62.2
Hypercolumn[8]	-	68.7	33.5	69.8	51.3	70.2	81.1	71.9	74.9	23.9	60.6	46.9	72.1	68.3	74.5	72.9	52.6	64.4	45.4	64.9	57.4	62.6
TTI-Zoomout-16[21]	89.8	81.9	35.1	78.2	57.4	56.5	80.5	74.0	79.8	22.4	69.6	53.7	74.0	76.0	76.6	68.8	44.3	70.2	40.2	68.9	55.3	64.4
DeepLab-CRF-LargeFOV[2]	**92.6**	83.5	36.6	**82.5**	62.3	**66.5**	85.4	78.5	83.7	**30.4**	72.9	**60.4**	**78.5**	75.5	**82.1**	**79.7**	**58.2**	**82.0**	48.8	73.7	63.3	**70.3**
ParseNet Baseline	92.3	82.6	36.1	76.1	59.3	62.3	81.6	79.5	81.4	28.1	70.0	53.0	73.2	70.6	78.8	78.6	51.9	77.4	45.5	71.7	62.6	67.3
ParseNet	92.4	**84.1**	**37.0**	77.0	**62.8**	64.0	**85.8**	**79.7**	**83.7**	27.7	**74.8**	57.6	77.1	**78.3**	81.0	78.2	52.6	80.4	**49.9**	**75.7**	**65.0**	69.8

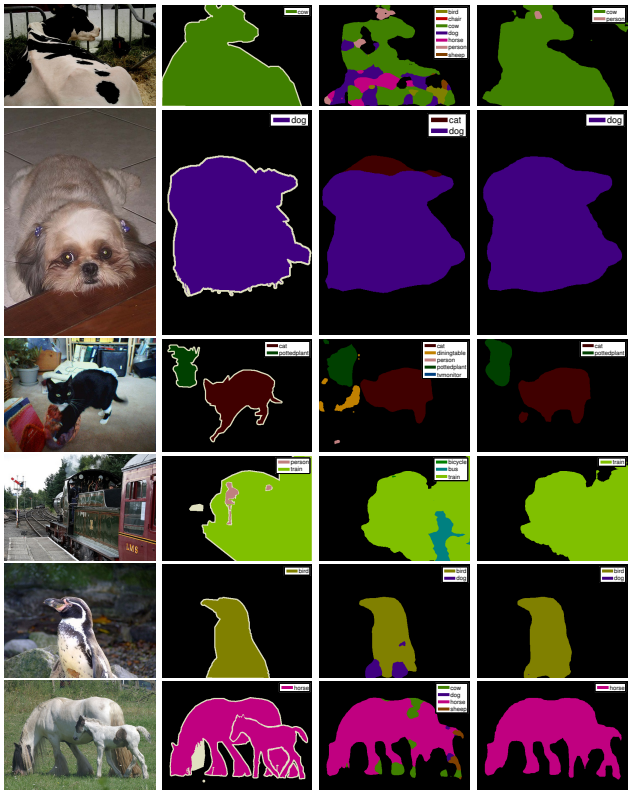
Table 8: PASCAL VOC2012 test Segmentation results.

We also tried the idea similar to spatial pyramid pooling where we pool 1×1 global feature, 2×2 subregion feature, and 4×4 subregion feature, and tried both *early fusion* and *late fusion*. However, we observed no improvements. We conjecture that the receptive field of the high level feature map (e.g. fc7) is sufficiently large that subregion global feature does not help much. However we have not explored this on lower level layers, maybe it is helpful for low level feature maps.

Finally, we also use two models ParseNet Baseline [5] and ParseNet [6], and compare our results on VOC2012 test set. As shown in Table 8, we can see that our baseline result is already higher than many of the existing methods due to proper finetuning. By adding the global context feature, we achieve performance that is within the standard deviation of the one [2] using fully connect CRF to smooth the outputs and perform better on more than half of categories. Again, our approach is much simpler to implement and train, hence is more robust. Using *late fusion* has almost no extra training/inference cost.

5. Conclusion

In this work we presented ParseNet, a simple fully convolutional neural network architecture that allows for direct inclusion of global context for the task of semantic segmentation. As part of developing and analyzing this approach we provided analysis of many architectural choices for the network, discussing best practices for training, and demonstrated the importance of normalization and learning weights when combining features from multiple layers of a network. By themselves, our practices for training significantly improve the baselines we use before adding global context. The guiding principle in the design of ParseNet is simplicity and robustness of learning. Results are presented on three benchmark datasets, and are state of the art on SiftFlow and PASCAL-Context, and near the state of the art on PASCAL VOC2012. On PASCAL VOC2012 test set, segmentation results of ParseNet is within the standard deviation of the DeepLab-LargeFOV-CRF, which suggests that adding global feature has similar effect of using CRF to do post-process smoothing. Given the simplicity and easy of training, we find these results very encouraging. In our on going work, we are exploring combining our technique with structure training/inference as done in [24, 18, 31].



(a) Original Image (b) Ground truth (c) ParseNet Baseline output (d) ParseNet output

Figure 3: Global context helps for classifying local patches.

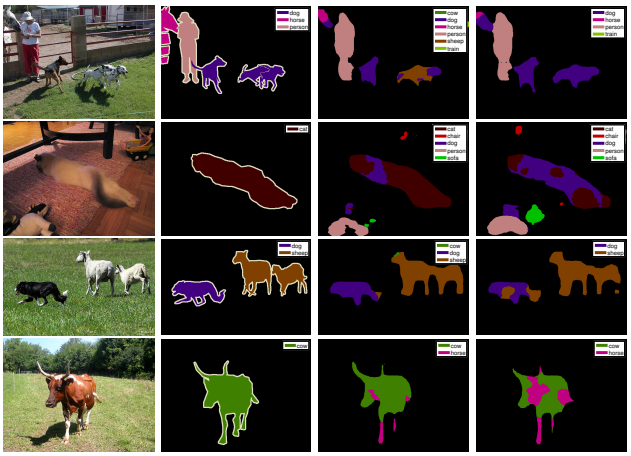
References

[1] J. Carreira and C. Sminchisescu. Cpmc: Automatic object segmentation using constrained parametric min-cuts. *PAMI*, 2012. 2

[2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep con-

[5] http://host.robots.ox.ac.uk:8080/anonymous/LGOLRG.html

[6] http://host.robots.ox.ac.uk:8080/anonymous/56QLXU.html



(a) Original Image (b) Ground truth (c) ParseNet Baseline output (d) ParseNet output

Figure 4: Global context confuse local patch predictions.

volutional nets and fully connected crfs. *arXiv:1412.7062*, 2014. 1, 2, 3, 4, 5, 7

[3] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 2014. 4

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2

[5] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, 2010. 5

[6] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *ECCV*, 2014. 2

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

[8] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. *arXiv:1411.5752*, 2014. 2, 3, 4, 7

[9] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *ECCV*, 2014. 2

[10] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *arXiv:1406.4729*, 2014. 2

[11] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv:1502.01852*, 2015. 4

[12] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015. 4

[13] Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org, 2013. 4

[14] P. Kohli, P. H. Torr, et al. Robust higher order potentials for enforcing label consistency. *IJCV*, 2009. 2

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

[16] Z.-z. Lan, L. Bao, S.-I. Yu, W. Liu, and A. G. Hauptmann. Double fusion for multimedia event detection. In *International conference on Advances in Multimedia Modeling*, 2012. 3

[17] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 3

[18] G. Lin, C. Shen, I. Reid, et al. Efficient piecewise training of deep structured models for semantic segmentation. *arXiv:1504.01013*, 2015. 1, 2, 7

[19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *arXiv:1411.4038*, 2014. 1, 2, 3, 4, 5, 7

[20] A. Lucchi, Y. Li, X. Boix, K. Smith, and P. Fua. Are spatial and global constraints really necessary for segmentation? In *ICCV*, 2011. 2

[21] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. *arXiv:1412.0774*, 2014. 2, 3, 7

[22] 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. 4

[23] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *CVPR*, 2007. 1, 2

[24] A. G. Schwing and R. Urtasun. Fully connected deep structured networks. *arXiv:1503.02351*, 2015. 1, 2, 7

[25] 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. *IJCV*, 2009. 2

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014. 1, 2, 4

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv:1409.4842*, 2014. 1, 2

[28] C. Szegedy, S. Reed, D. Erhan, and D. Anguelov. Scalable, high-quality object detection. *arXiv:1412.1441*, 2014. 2, 3

[29] A. Torralba. Contextual priming for object detection. *IJCV*, 2003. 1

[30] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 2013. 2

[31] 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. *arXiv:1502.03240*, 2015. 1, 2, 7

[32] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014. 3