Report 3:PSPNet Architecture and Results

Adithya Narayan
Manipal, Karanataka

Abstract

In this report, the architecture of the PSPNet's building blocks along with its performance are gone over. Additionally, concepts such as atrous convolutions , auxiliary losses and residual blocks are covered in relation to the ResNet 101 architecture used in the paper.

Keywords: PSPNet, ResNet101, Semantic Segmentation, Architecture

1. Introduction

The PSPNet builds on a traditional dilated fully-connected convolutional network for semantic segmentation by using pyramidal pooling for parsing pixel level information and generating pixel wise classifications for an input image[1].

It boasts improved performance on[1] the cityscapes dataset along with other similar segmentation datasets[1]. Since this report is ultimately building towards the cityscapes dataset, this architecture achieves a 78.4% IoU score which is nearly 7% greater than the next highest performing architecture[1]. Additionally, for a significant improvement in performance, the computational cost of the architecture is not much higher that the base dilated Fully-Convolutional-Network used as the base. With this in mind, the architecture of the network is gone over in the upcoming sections.

2. Architecture

The overall structure of the PSPNet consists of 4 major blocks as can be seen in Fig.1.

It begins with the input block to feed in the data into the CNN. This is followed by a typical CNN. According to the paper, this CNN block is a modified ResNet101 with atrous convolutions and auxiliary loss functions.

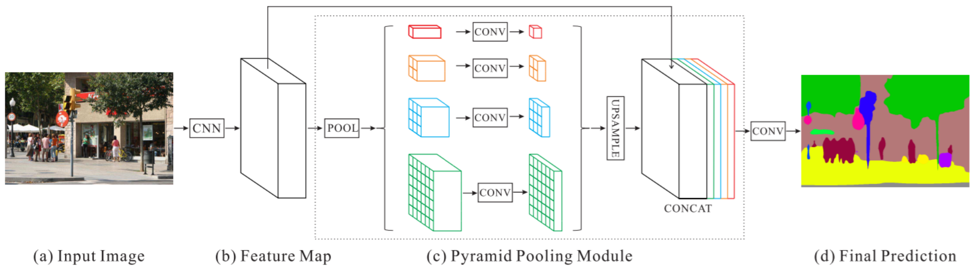


Figure 1: PSPNet Architecture

This generates the base feature map which is 1/8 of the original image. This feature map is then fed into the pyramidal pooling module. Here, the feature map is pooled at different scales and then passed through a 1×1 convolutional block to rescale the channel depth to $1/N$ times the number of levels in the pyramid[1]. This reduces the context information in the blocks. These outputs from the convolutional blocks are then up-sampled and concatenated together to generate the final output for the pyramidal pooling block.This is followed by a simple convolutional layer to generate the final pixel-wise classifications.

2.1. ResNet101

This particular section discusses the architecture of the ResNet used along with information on ResNet's usage of residual blocks. Additionally, the use of atrous convolutions(or dilated convolutions) and auxiliary loss functions will also be gone over in this particular section. While the normal ResNet101 architecture does not feature a atrous convolutions or auxiliary losses, this particular paper uses a modified version of ResNet101[2].

2.1.1. Residual Blocks

To understand what exactly a ResNet does, we begin by understanding the residual block itself. The residual block essentially consists of a skip connection that adds the activation output of a prior layer to the linear output of the next convolutional layer[6]. Then this sum is passed through the activation function. Mathematically, if $A^{[l]}$ represents the activation of the l^{th} layer and $Z[l]$ represents the output of the convolutional layer,then, the output of the residual block will be given by,

$$Z^{[l+2]} = W^{[l+2]}A^{[l+1]} + b^{[l+2]} \tag{1}$$

$$g(Z) = g(Z^{[L+2]} + A^{[l]}) \tag{2}$$

In effect, the residual block adds a 'skip connection' from layer l to layer $l+2$. This being said, this architecture is not fixed. Different papers include different skipping lengths. However, in the original ResNet paper, it was experimentally found that skip connections over a single layer did not achieve much[6]. This can be seen in Fig.2.

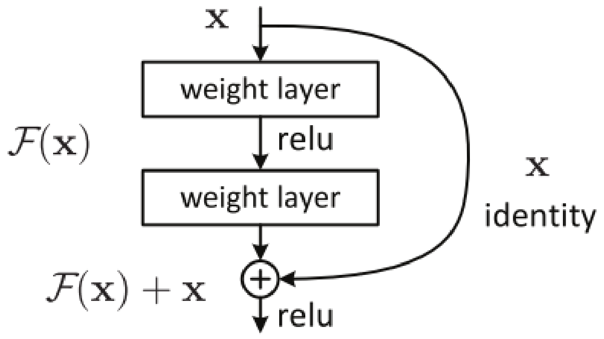


Figure 2: A sample residual block

Generally, particularly in most modern architectures, skip connections are fairly common. The advantage of it being that it allows for the training of deeper networks. Deeper neural networks often face the problem of 'vanishing gradient'.

2.1.2. Dilated Convolutions

An interesting feature of this modified ResNet is its use of dilated convolutions. Instead of using typical convolutional layers which have a small receptive field, dilated convolutions are used.

Dilated convolution are similar to typical convolutions except that instead of using adjacent elements in the input layer, the elements of the kernel are spaced out before multiplying with the input kernel[3]. Mathematically, we can compare this to standard convolution as shown below where equation(3) represents standard convolution and equation(4) represents dilated convolution[3],

$$(F * P)(p) = \sum_{s+t=p} F(s)k(t) \tag{3}$$

$$(F *_l P)(p) = \sum_{s+lt=p} F(s)k(t) \tag{4}$$

The idea behind this is that using a spread out version of convolution would increase the receptive field of the convolutional layer[4]. A visualization of this can be seen in Fig.3

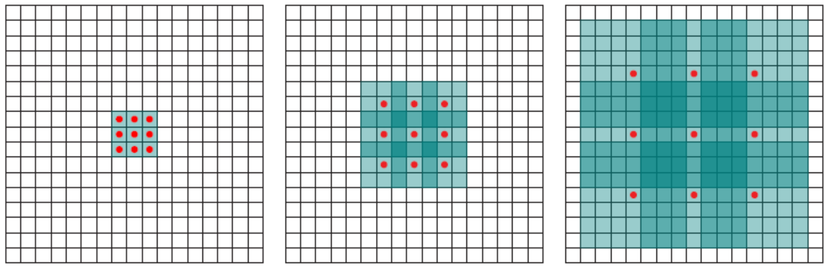


Figure 3: Example of Dilated Convolution's receptive field

As can be seen for quite a number of papers, for instance this one[5], dilated networks often produce better segmentation results across the board.

2.1.3. Auxillary Loss

Apart from the primary loss computed at the end of the ResNet101, this modified version also includes a auxiliary loss after the 'resb22' residual block of the network as can be seen in Fig.4.

Similar to GoogLeNet, this is used as a way to again address the problem of vanishing gradients deep networks often face. In general, the idea here is that earlier layers of the network may learn certain context information that is valuable for the actual segmentation process[2]. As a consequence, during the training stage of the process, we compute a weighted loss which combines both the auxiliary loss as well as the final loss. In this particular case, we take $\alpha = 0.4$ times the auxiliary loss plus the main loss[1]. It is also noted that the auxiliary loss is only used during the training process and is removed during inference.

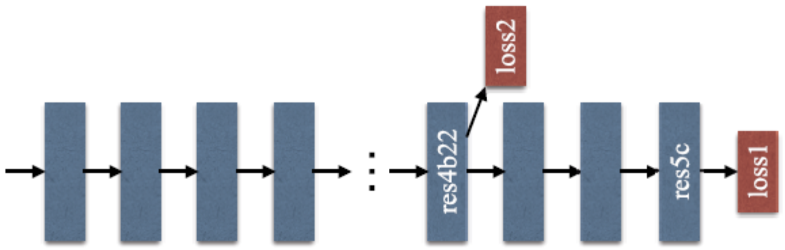


Figure 4: Auxiliary loss used in the model

After the modified ResNet101 block, the feature map from the last convolutional block of the ResNet is taken and four seperate pooling operations are performed with the first being Global Average Pooling and the rest being standard pooling operations. In the next section, we discuss this.

2.2. Pyramidal Pooling Module

The Pyramidal Pooling Module involves a multi-level pyramid which pools the feature map from the ResNet at different scales. In a general neural network, Global Average Pooling typically serves as an alternative to the Flatten() operation in a CNN's fully-connected layer and sometimes even replaces the fully-connected layer entirely. In Global Average Pooling, every channel in the layer is averaged. This results in an output with the same number of channels but with a 1×1 height and width[7]. This is shown in the Fig.5.

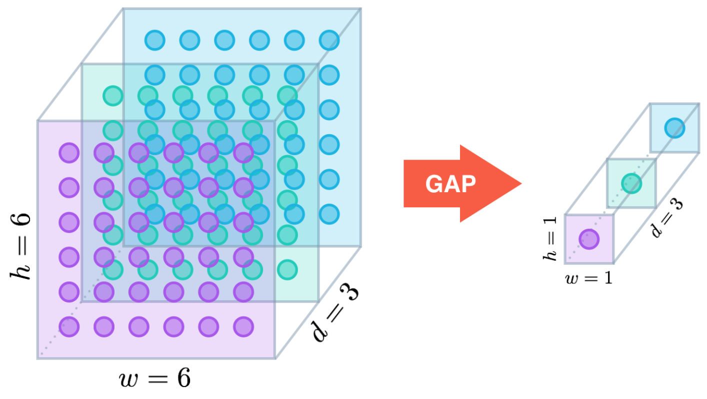


Figure 5: Global Average Pooling

It has been shown in this paper[8] that GAP offers information helpful for object localization as well. However, in this case, it is used as a baseline global contextual prior[1].

Besides, GAP, the blocks in (c) also perform standard average pooling at different scales. Once this is performed, the outputs of the pooling operations compressed to reduce the context information. Through this, the channel depth is reduced to $1/N$ of its original size(N being the depth of the pyramidal module). Then, the outputs are upsampled using bilinear interpolation to the size of the feature map and concatenated to the original feature map generated by the modified ResNet101. Note that while the depth of the Pyramidal module can be anything, this particular paper implements a 4 layer deep pyramid with the pooling kernel sizes which generate output bins of 1×1, 2×2,3×3 and 6×6.

2.3. Final Block

One the output is generated by the pyramidal module, we pass it as an input to a convolutional layer which generates the final output prediction classes. This particular architecture uses a "poly" learning rate[6] defined by,

$$lr_{current} = lr_{base} * (1 - \frac{iter}{iter_{max}})^{power} \tag{5}$$

Here, the paper recommends $power = 0.9$ and base learning rate $lr_{base} = 0.01$. In the next section, the ablation study results of the paper is briefly discussed.

3. Ablation Study

In this particular section, the results from the ablation study of this particular architecture is discussed in terms of pooling, auxiliary loss, network depth and data-augmentation.

3.1. Max Pooling,Average Pooling and DR

In this they replace the ResNet101 with a dilated ResNet 50 baseline and perform comparisons by adding/removing max/average pooling layers in the network. Additionally, the effect of dimensionality reduction using 1×1 convolutions are touched upon. In general average convolutions have better IoU values as well as accuracy. Additionally, the addition of the pyramidal scheme also improves the aforementioned metrics. The best performance comes from a combination of average pooling with dimensionality reduction along with a pyramidal scheme. The results are summarized in the table below.

Method	Mean IoU(%)	Pixel Acc.(%)
ResNet50-Baseline	37.23	78.01
ResNet50+B1+MAX	39.94	79.46
ResNet50+B1+AVE	40.07	79.52
ResNet50+B1236+MAX	40.18	79.45
ResNet50+B1236+AVE	41.07	79.97
ResNet50+B1236+MAX+DR	40.87	79.61
ResNet50+B1236+AVE+DR	**41.68**	**80.04**

Figure 6: A comparison of MP, AVGP and DR

3.2. Auxiliary Loss

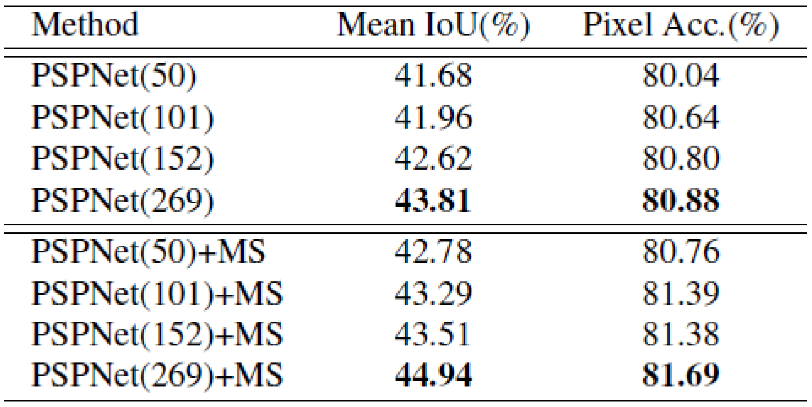
Experimentally, different values of were tested with a baseline of no auxiliary loss. The selected value of $\alpha = 0.4$ was chosen since it gave the best performance.

Loss Weight α	Mean IoU(%)	Pixel Acc.(%)
ResNet50 (without AL)	35.82	77.07
ResNet50 (with $\alpha = 0.3$)	37.01	77.87
ResNet50 (with $\alpha = 0.4$)	**37.23**	**78.01**
ResNet50 (with $\alpha = 0.6$)	37.09	77.84
ResNet50 (with $\alpha = 0.9$)	36.99	77.87

Figure 7: Effect of different values of α

3.3. Depth and multi-scale testing

As is fairly common, deeper networks performed better than shallower ones. This holds true even during the multi-scale testing process. Again, the results are summarized below,

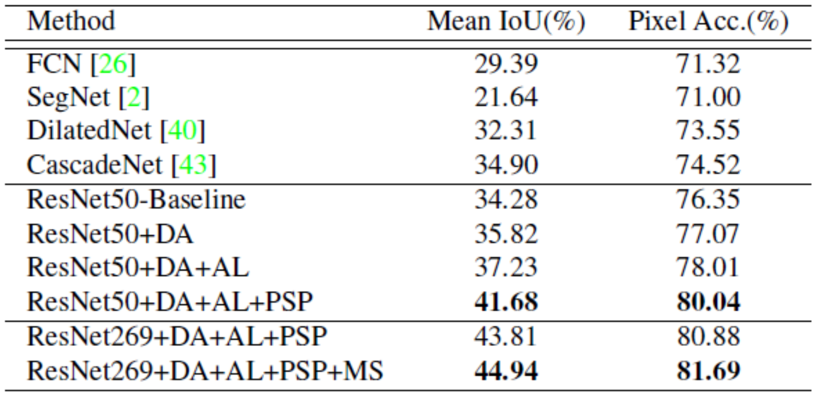


Method	Mean IoU(%)	Pixel Acc.(%)
PSPNet(50)	41.68	80.04
PSPNet(101)	41.96	80.64
PSPNet(152)	42.62	80.80
PSPNet(269)	**43.81**	**80.88**
PSPNet(50)+MS	42.78	80.76
PSPNet(101)+MS	43.29	81.39
PSPNet(152)+MS	43.51	81.38
PSPNet(269)+MS	**44.94**	**81.69**

Figure 8: Effect of network depth

3.4. Data Augmentation and PSP

In general, datasets with cropping used for data augmentation performed better than those without. Additionally, data augmentation combined with the PSP module performed better than all prior architectures as can be seen in the table.

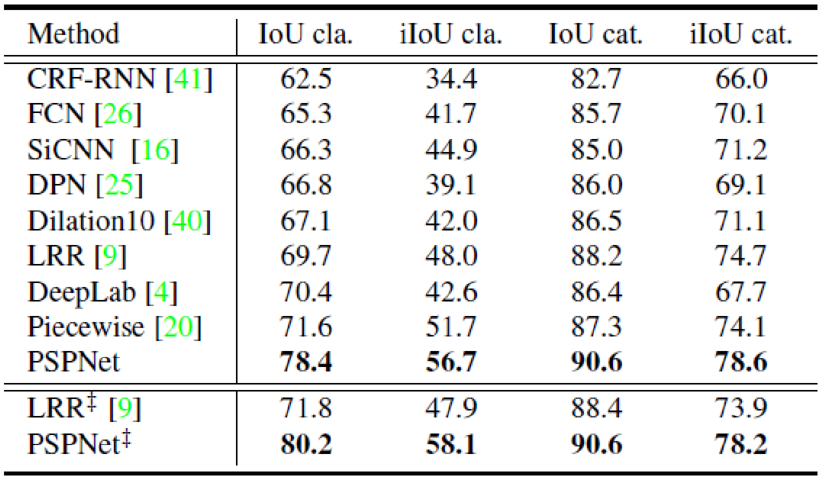


Method	Mean IoU(%)	Pixel Acc.(%)
FCN [26]	29.39	71.32
SegNet [2]	21.64	71.00
DilatedNet [40]	32.31	73.55
CascadeNet [43]	34.90	74.52
ResNet50-Baseline	34.28	76.35
ResNet50+DA	35.82	77.07
ResNet50+DA+AL	37.23	78.01
ResNet50+DA+AL+PSP	**41.68**	**80.04**
ResNet269+DA+AL+PSP	43.81	80.88
ResNet269+DA+AL+PSP+MS	**44.94**	**81.69**

Figure 9: Effect of Data Augmentation and the PSP module

4. Performance on the Cityscapes dataset

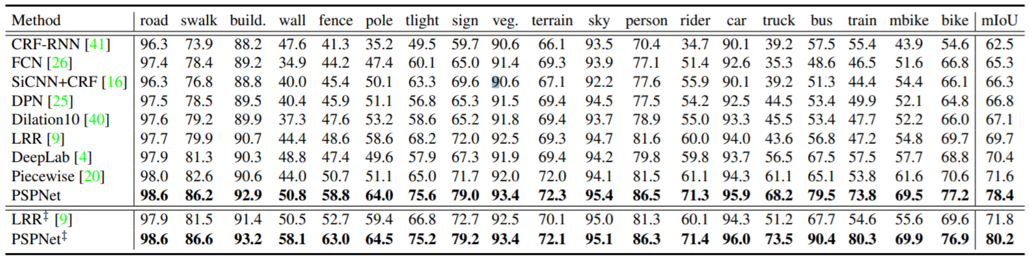
Since this project is building towards its performance on the cityscapes dataset, that is gone over in this section. In general, over 2975 training images, 500 validation images and 1525 testing images, the PSPNet outperform all prior architectures used for semantic segmentation. In both the coarse and fine mask datasets, this holds true. This can be seen in the table below. Note that ++ represents its performance on the coarse labels.



Method	IoU cla.	iIoU cla.	IoU cat.	iIoU cat.
CRF-RNN [41]	62.5	34.4	82.7	66.0
FCN [26]	65.3	41.7	85.7	70.1
SiCNN [16]	66.3	44.9	85.0	71.2
DPN [25]	66.8	39.1	86.0	69.1
Dilation10 [40]	67.1	42.0	86.5	71.1
LRR [9]	69.7	48.0	88.2	74.7
DeepLab [4]	70.4	42.6	86.4	67.7
Piecewise [20]	71.6	51.7	87.3	74.1
PSPNet	**78.4**	**56.7**	**90.6**	**78.6**
LRR‡ [9]	71.8	47.9	88.4	73.9
PSPNet‡	**80.2**	**58.1**	**90.6**	**78.2**

Figure 10: Performance on the Cityscapes dataset

ADditionally, the table below shows its performance per class on the cityscapes dataset.



Method	road	swalk	build.	wall	fence	pole	tlight	sign	veg.	terrain	sky	person	rider	car	truck	bus	train	mbike	bike	mIoU
CRF-RNN [41]	96.3	73.9	88.2	47.6	41.3	35.2	49.5	59.7	90.6	66.1	93.5	70.4	34.7	90.1	39.2	57.5	55.4	43.9	54.6	62.5
FCN [26]	97.4	78.4	89.2	34.9	44.2	47.4	60.1	65.0	91.4	69.3	93.9	77.1	51.4	92.6	35.3	48.6	46.5	51.6	66.8	65.3
SiCNN+CRF [16]	96.3	76.8	88.8	40.0	45.4	50.1	63.3	69.6	90.6	67.1	92.2	77.6	55.9	90.1	39.2	51.3	44.4	54.4	66.1	66.3
DPN [25]	97.5	78.5	89.5	40.4	45.9	51.1	56.8	65.3	91.5	69.4	94.5	77.5	54.2	92.5	44.5	53.4	49.9	52.1	64.8	66.8
Dilation10 [40]	97.6	79.2	89.9	37.3	47.6	53.2	58.6	65.2	91.8	69.4	93.7	78.9	55.0	93.3	45.5	53.4	47.7	52.2	66.0	67.1
LRR [9]	97.7	79.9	90.7	44.4	48.6	58.6	68.2	72.0	92.5	69.3	94.7	81.6	60.0	94.0	43.6	56.8	47.2	54.8	69.7	69.7
DeepLab [4]	97.9	81.3	90.3	48.8	47.4	49.6	57.9	67.3	91.9	69.4	94.2	79.8	59.8	93.7	56.5	67.5	57.5	57.7	68.8	70.4
Piecewise [20]	98.0	82.6	90.6	44.0	50.7	51.1	65.0	71.7	92.0	72.0	94.1	81.5	61.1	94.3	61.1	65.1	53.8	61.6	70.6	71.6
PSPNet	**98.6**	**86.2**	**92.9**	**50.8**	**58.8**	**64.0**	**75.6**	**79.0**	**93.4**	**72.3**	**95.4**	**86.5**	**71.3**	**95.9**	**68.2**	**79.5**	**73.8**	**69.5**	**77.2**	**78.4**
LRR‡ [9]	97.9	81.5	91.4	50.5	52.7	59.4	66.8	72.7	92.5	70.1	95.0	81.3	60.1	94.3	51.2	67.7	54.6	55.6	69.6	71.8
PSPNet‡	**98.6**	**86.6**	**93.2**	**58.1**	**63.0**	**64.5**	**75.2**	**79.2**	**93.4**	**72.1**	**95.1**	**86.3**	**71.4**	**96.0**	**73.5**	**90.4**	**80.3**	**69.9**	**76.9**	**80.2**

Figure 11: Per class performance on the Cityscapes dataset

References

[1] Zhao, Hengshuang, et al. ”Pyramid scene parsing network.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[2] https://towardsdatascience.com/review-pspnet-winner-in-ilsvrc-2016-semantic-segmentation-scene-parsing-e089e5df177d

[3] Yu, Fisher, and Vladlen Koltun. ”Multi-scale context aggregation by dilated convolutions.” arXiv preprint arXiv:1511.07122 (2015).

[4] https://erogol.com/dilated-convolution/

[5] Yu, Fisher, Vladlen Koltun, and Thomas Funkhouser. ”Dilated residual networks.” Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[6] He, Kaiming, X. Zhang, Shaoqing Ren and Jian Sun. “Deep Residual Learning for Image Recognition.” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): 770-778.

[7] https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization/

[8] http://cnnlocalization.csail.mit.edu/Zhou-Learning-Deep-Features-CVPR-2016-paper.pdf