Atrous Spatial Pyramid Pooling for Semantic Segmentation

Darren Mei
Stanford University
Stanford, CA
dmei@cs.stanford.edu

Aditya Khandelwal
Stanford University
Stanford, CA
akhand@cs.stanford.edu

Abstract

For this project, we present a deep learning framework that aims at solving semantic image segmentation of urban environments. To achieve this task, our deep learning framework utilizes atrous spatial pyramid pooling. Training and evaluation is conducted on the Cityscapes dataset [12], which contains a diverse set of stereo video sequences recorded in street scenes from 50 different cities. Following an overview of prior semantic segmentation techniques and an implementation of SegNet, this project reviews in-depth how atrous convolution can be applied to solve semantic segmentation. Using the DeepLabv3 framework, numerous experiments on atrous convolution structures are conducted, and the final result receives a 72.53% mean IoU score on the Cityscapes validation set.

1. Introduction

In this paper, we are concerned with pixel-level semantic segmentation of urban street images using Deep Convolutional Neural Networks (DCNNs). Pixel-wise image segmentation is a challenging and demanding task in computer vision and image processing. At the same time, semantic segmentation of images is an essential task required to make autonomous machines of the future - from robots to self-driving cars - function. In this task, we label regions of an image according to what is being shown. More specifically, the goal of semantic image segmentation is to label each pixel of an image with a corresponding class of what is being represented. Since we are predicting a class for every pixel in the image, this task is commonly referred to as dense prediction.

1.1. Applications

Semantic segmentation of images has useful applications in many areas of research and industry. Some of the most common ones are:

1. Machine vision
   (a) Pedestrian and Road Segmentation
   (b) Video surveillance
   (c) Traffic control systems
2. Medical imaging
3. Content-based image retrieval

Although this is not an exhaustive list by any means, our motivation to work on this important area of research has been very much influenced by the number of different ways our approach can be utilized in various distinct and independent fields of work. Moreover, since image segmentation requires huge domain knowledge database that does not currently exist for each possible class that one might find in an image, we were curious to understand the advantages and limitations of recent work in the area and hypothesize our implementation of a DCNN as a solution to this problem.

1.2. Early Naive Implementations

The earliest examples of image segmentation aimed to classify each pixel in an image using a binary thresh-holding method. This meant that pixels above a certain intensity threshold were classified into the same category. More robust methods, such as k-means clustering, balanced histogram thresh-holding, etc., were subsequently created to provide multi-class classification capabilities. However, since these methods rely on high-level image features and do not take into account any contextual information, they were poor at consistently accurate classification. Furthermore, for tackling more complex data, such as frames of a video, or challenges, such as real-time segmentation, these methods presented several operational and runtime challenges.

1.3. Neural Network Approach

Recent breakthroughs in the field of computer vision have led to unprecedented success in procedural image segmentation tasks. Fully Convolutional Neural Networks

(FCNN) [4] were proposed in 2014 to learn basic image segmentation tasks. Learning a deconvolution network on top of VGG-16 [7] convolutional network which consists of deconvolution and unpooling layers was implemented in 2015 to identify detailed structures and handle objects in multiple scales naturally. Since then, pyramid networks have achieved state-of-the-art semantic segmentation accuracy scores. For our research, DeepLabv3 [11] became an important starting point as we went ahead and introduced several variations of the current DeepLab model in order to benchmark and test its performance.

1.4. Problem Statement

For our project, we wanted to implement DeepLabv3 in PyTorch and use it to test accuracy and mean IoU on images in the Cityscapes dataset. Our input consists of images and we wanted to generate output image maps that classified each pixel in every image into one of the 19 potential classes, as described in the Cityscapes dataset. Moreover, after looking at some of the DeepLab results, we wanted to experiment with the DeepLabv3 network by introducing more layers and tweaking some of the hyperparameters to yield better results when it came to segmenting global features in images, and provide benchmark results of these model improvements in this paper.

2. Related Work

There are various approaches to semantic segmentation problems, and can be broadly broken up into traditional methods and deep learning solutions. Additionally, there are different categories of deep learning approaches, such as the widely adopted Fully Convolutional Network, Encoder-Decoder Networks, and Pyramid Networks.

2.1. Traditional Methods

More traditional approaches to semantic segmentation utilize concepts such as Decision Forests and Conditional Random Fields. Decision Forests consist of decision trees which have classifications at the leaves of each tree, and are effective at both classification and clustering.[1] Conditional Random Fields are a graph structure used when the label of an input is dependent on the labels of nearby inputs. Since semantic segmentation typically has many pixels of the same class clustered in various groups throughout the image, CRFs have been a popular way to model the input image.[2] Additionally, they have even been used in deep learning approaches in combination with convolutional networks.[3] However, due to the slow training and inference speeds of CRFs they have not been as common in recent publications.

2.2. Fully Convolutional Networks

One of the first deep learning approaches to Semantic Segmentation used an end-to-end, pixels-to-pixels convolutional network.[4] With transfer learning using previous classification networks like AlexNet and VGG net, FCNs were able to achieve up to 62.2% mean IoU on the PASCAL VOC dataset[5], a common benchmark for semantic segmentation. To do this, the FCN upsamples from the dense feature maps to get an output of the exact same size as the input. However, an issue with the FCN was brought up in a later work known as ParseNet[6] which claimed that the FCN model loses a more global context of the image by using small feature maps. To solve this, ParseNet normalizes the initial feature maps and concatenates the normalization with the standard FCN. In doing so, it better understands features at a larger scale and also performed better with a 69.8% mean IoU on the PASCAL VOC dataset.

2.3. Encoder-Decoder Networks

Building off of FCNs, other architectures utilize a similar first half but have a more complex upsampling network to better map pixel-wise probabilities. Successful examples of these architectures include both DeconvNet and U-Net, the latter of which was proposed for the medical imaging community.[7][8] These "encoder-decoder" networks can also be seen as encoding information into dense feature maps through a standard convolution network like VGG-16 and then decoding this information via unpooling and upconvolutional layers. By adding a more complex upconvolution process, it better approximates pixel-wise class probabilities and can better segment an image. These architectures have achieved up to 72.5% mean IoU on the PASCAL VOC dataset.

2.4. Segnet

This project used a baseline modeled after SegNet, another variation of an Encoder-Decoder Network. We chose a PyTorch implementation of SegNet developed by Badrinarayanan et al. at the Machine Intelligence Lab at University of Cambridge [9]. The main reason this baseline was used is because other implementations like DeconvNet[7] have many more parameters due to their use of fully connected convolutional layers. This would require additional computational resources and time and SegNet was shown to perform comparably.

SegNet uses a four layer "flat" architecture. Each encoder performs dense convolutions, ReLU non-linearity, a non-overlapping max pooling with a 2×2 window and down-sampling. Each decoder upsamples its input using the memorized pooled indices and convolves it with a trainable filter bank. No ReLU non-linearity is used in the decoder. The encoder and decoder filters are also untied to

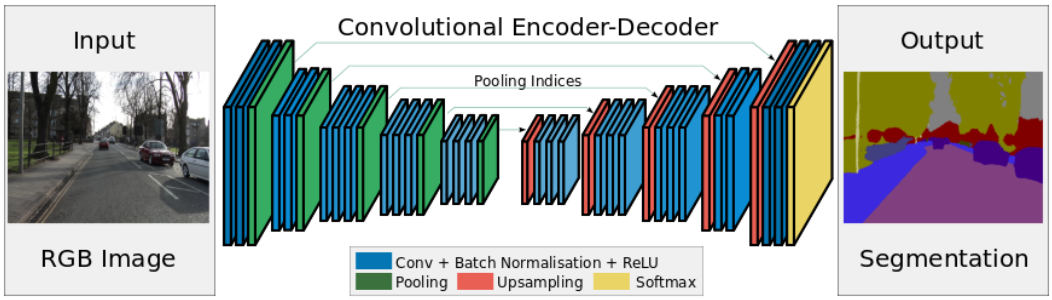

Figure 1: Visualization of SegNet Layers that produce a mapping from pixels in RGB Images to Segmentation Class and Category Labels

provide additional degrees of freedom to minimize the objective. The final layer is a softmax classifier (with no bias term) which classifies each pixel independently.

A highlight of this baseline architecture is its ability to produce smooth segment labels when compared with local patch based classifiers. This is due to deep layers of feature encoding that employ a large spatial context for pixel-wise labelling. Both qualitative and numerical accuracy of the SegNet for outdoor and indoor scenes is very competitive, even without use of any CRF post-processing.

2.5. Pyramid Networks

The current state-of-the-art implementations for semantic segmentation utilize pyramid pooling to identify features at different scales.[10] This further utilizes the global context of the image because more information can be captured within these pyramid pooling layers. In PSPNet, an input to the layer is convolved with four different scales or pyramid levels, which are each then run through a 1x1 convolutional layer to reduce the dimension of context representation. By then directly upsampling these low-dimensional feature maps, different feature scales can be concatenated to capture various sizes of objects in the image. Using this pyramid architecture, PSPNet and its successor DeepLabv3[11], the architecture implemented in this project, are able to achieve over 80% mean IoU on the Cityscapes Dataset.

3. Data

While there are numerous semantic segmentation benchmarks such as the previously mentioned PASCAL VOC, this project utilized the Cityscapes Dataset for training and evaluation.[12] The Cityscapes Dataset consists of urban scenes recorded in 50 different cities, and is meant to focus on a visual understanding of complex urban street scenes. The results are evaluated on 19 classes out of the 30 visual classes annotated due to the rarity of 11 classes in the dataset. The annotations also occur at fine and coarse levels. In this project we only use the fine pixel-wise annotations, of which there are 2975 training and 500 validation images. Although there are 1525 test images, since Cityscapes is a benchmark these images do not have corresponding public annotations. Each image in the dataset has resolution 1024x2048.

Figure 2: Example from the Cityscapes Dataset. The above image is the input image, and the below image is the corresponding fine annotation. *Note: Although there are borders between instances of classes that does not matter for the purposes of semantic segmentation evaluation.*

Since the Cityscapes Dataset provides semantic, instance, and panoptic segmentation annotations, the ground truth files needed to be converted from polygonal annotations into semantic segmentation annotations before training. The Cityscapes Benchmark Suite[13] provides an encoding function which converts the original annotations into per-pixel ground truth class encodings.

Other models utilized the coarse annotations for training, and even augmented the dataset through random scaling and flipping. However, for the scope of this project the 2975 training examples were deemed enough to train our model,

and no data augmentation was used.

4. Methods

Our main approach to semantic segmentation follows the architecture described in DeepLabv3 [11], utilizing atrous convolution and spatial pyramid pooling to extract features at varying scales.

4.1. Atrous Convolution

Atrous convolution, also known as dilated convolution, uses convolution filters which have a larger filter size but only sample from select pixels.

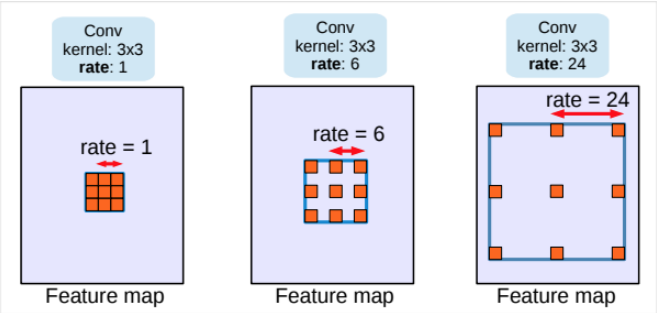

Figure 3: Atrous convolution with kernel size 3 3 and different rates. Standard convolution corresponds to atrous convolution with rate = 1.

As seen in Figure 3, atrous convolution is able to capture contextual information at a larger scale without additional parameters. This can also be seen in the following equation describing the output of an atrous convolution. Given an input feature map x and a filter w, the output y at index i will look like[11]:

$$y[i] = \sum_k x[i + r \cdot k]w[k]$$

In the above equation, r denotes the atrous rate. The atrous rate r can also be seen as 1 divided by the frequency of seeing a nonzero element. If $r = 6$, then that means 1 in every 6 elements in the filter has a non-zero weight. A standard convolution can then be seen as a special case where $r = 1$, so there is no space in between each nonzero weight.

4.2. Cascaded Atrous Convolution

Similar to previously discussed works, DeepLabv3 additionally uses an initial encoder structure to obtain feature maps of the data. Instead of using VGG-16 like other implementations[7][5], DeepLabv3 chose to use ResNet[14], but added atrous convolutions in some of the blocks. Each ResNet block consists of three 3x3 convolutions, with the last convolution having a stride of 2 except in the last block. The reason for using ResNet is due to deep networks having issues with vanishing gradients. Furthermore, by using atrous convolution and increasing the stride in the last convolution, it is easier to capture long range information in deeper blocks. Though DeepLabv3 also goes into more detail regarding changing the sizes of each of the three convolutional layers in each block based on a Multi-grid parameter, we did not focus on this method because we wanted to pay more attention to the Atrous Spatial Pyramid Pooling Layer. As a result, while four cascading ResNet blocks were used in the final architecture, only one of them utilized atrous convolution with the atrous rate set to 2.

4.3. Atrous Spatial Pyramid Pooling

One of the main differences between DeepLabv3 and other implementations is the use of an Atrous Spatial Pyramid Pooling or ASPP layer before upsampling. An Atrous Spatial Pyramid Pooling layer is similar to the pyramid structure discussed earlier along with PSPNet[10], except that the convolutions are atrous convolutions. Each input into the ASPP layer is run through multiple convolutions of the same filter size but with varying dilation (space in between each weight). There are two 1x1 convolutions in addition to the atrous convolutions in the pyramid, because one of the 1x1 convolutions is on the global pooling result of the input to maintain the fine annotation details. The padding for each convolution is calculated as follows to ensure that the output shape matches the input shape. While for a kernel size of 3 the padding remains the same as the dilation parameter, for the other kernel sizes used in this project it needed to be calculated. If the input window size is defined as W, the filter size as F, the padding as P, and the stride as S, then the output size O is defined as:

$$(\mathrm{W} - F + 2P)/S + 1 = O$$

For Atrous Convolution, the filter size F can also be calculated in terms of the kernel size K and dilation parameter D:

$$F = (K - 1) * D + 1$$

Since the input window size should be equal to the output window size and the stride is set to 1, then we calculated the necessary padding for a given kernel size and dilation as:

$$P = \frac{(K - 1) * D}{2}$$

With this structure, features at varying distances from each other are still related in this layer, and will have a greater impact on the final pixel-wise prediction.

4.4. Model Architecture

As seen in Figure 4, the final DeepLabv3 architecture receives the input RGB image, convolves it in Conv1 with

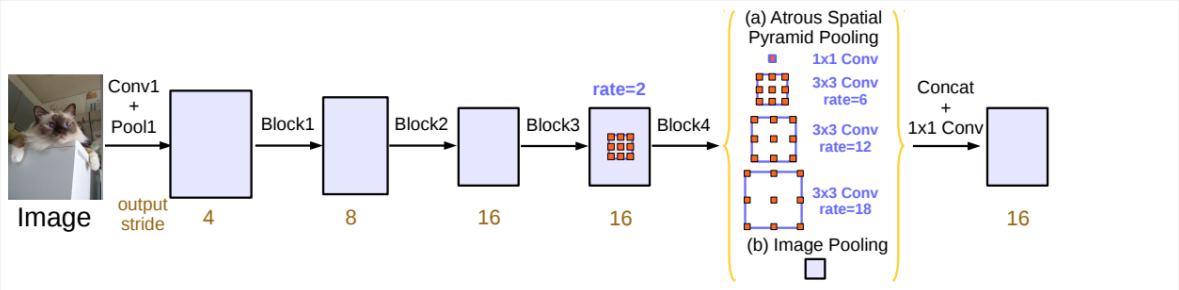

Figure 4: DeepLabv3 Architecture combining an initial ResNet structure with one Atrous Convolution ResNet Block with an Atrous Spatial Pyramid Pooling layer.

64 filters of size 7, stride 2, and padding of 3. After batch normalization, the ReLU of the output is then pooled with a 3x3 filter with stride 2 and padding of 1. This initial convolutional and pooling layer is then fed into the first of the four ResNet blocks. Each ResNet block consists of varying numbers of layers, each of which do three convolutions in sequence. The first convolution has a 1x1 filter, which is then batch normalized and fed through a filter of size 3x3. This is then batch normalized and sent through another convolution with a 1x1 filter which also expands the output dimension by a factor of 2. After a final batch normalization, the output is then either added with the original input to complete the skip connection, or with a downsampled version of the input if the output dimension is less than the input dimension.

During this project, we utilized the ResNet-101 structure[14] with pretrained weights on ImageNet[15] in order to speed up the training process. Since the weights for Block1-Block4 in Figure 4 were pretrained on ImageNet, most of our attention was to on the implementation of the ASPP layer. As will be discussed in the next section, we altered various portions of the ASPP layer to determine the best possible configuration.

Finally, we used Pixel-wise Cross Entropy Loss for this model. Pixelwise cross entropy loss is calculated as the log loss summed over all 19 classes as such:

$$\text{Loss} = -\sum_{\text{classes}} y_{\text{true}} \log(y_{\text{pred}})$$

Since the Cityscapes dataset disregarded classes that did not appear often enough, then the Pixel-wise Cross Entropy Loss is a valid loss function. This is because by evaluating predictions for each pixel vector (across the 19 classes) individually, we are equally learning across every pixel in the image. Other implementations also assumed there was not a class imbalance in the Cityscapes dataset, and used Pixel-wise Cross Entropy Loss as well.

5. Experiments

We conducted several experiments by introducing variations to the DeepLabv3 model. The task of all of these experiements was to understand how applying these transformations changed the mean IoU score which was the evaluation metric that we were most interested in. For each model architecture we evaluated on, we found the average IoU ratio for each of the 19 classes across the validation set. Then, the mean IoU score was calculated by averaging the individual IoU scores for each of the 19 classes.

5.1. Evaluation

In order to evaluate our DCNN implementations, we used benchmarking metrics as provided by the Cityscapes Dataset Challenge [12]. To assess performance, we rely on the standard Jaccard index, commonly known as the PASCAL VOC IoU metric.

$$\text{IoU} = \frac{\text{True Pos.}}{\text{True Pos.} + \text{False Pos.} + \text{False Neg.}}$$

5.2. Baseline Model: SegNet

As a baseline model for our task, we used a PyTorch implementation of SegNet that we finetuned for our dataset. Our task was to classify all pixels in each image correctly into one of 19 classes (as provided in the Cityscapes dataset) and we were evaluating on the mean IoU metric, as described above. For our loss function, we used Cross-Entropy loss, and for optimization we used Stochastic Gradient Descent. We evaluated this model after running it for 7500 iterations. While SegNet was able to produce smooth classifications for all pictures, the mean IoU of generated image maps compared to the ground truths was 0.4379. Since this mean IoU score is comparable to SegNet implementations evaluated on datasets like the Pascal VOC, we were satisfied with the results and were also confident that our DeepLabv3 variations would perform much better on the Cityscapes dataset.

Method	Road	Sidewalk	Building	Wall	Fence	Pole	T. Light	T. Sign	Veg.	Terr	Sky	Ped.	Rider	Car	Truck	Bus	Train	Mbike	Bike	**mIoU**
SegNet	94.99	69.19	85.72	25.32	32.57	44.57	0.06	50.04	88.41	48.27	**89.46**	59.98	0.01	84.77	0.0	0.04	0.0	0.0	58.74	43.79
DL-1	95.83	76.83	89.44	**45.45**	**50.29**	**48.74**	55.33	66.23	89.67	55.48	86.96	74.17	41.56	91.18	41.36	67.69	**51.35**	44.72	69.26	65.43
DL-2	**96.99**	**78.33**	**89.56**	41.46	47.21	48.60	**56.58**	**68.83**	**90.39**	**58.07**	87.70	**75.67**	**51.53**	**92.09**	**52.53**	**70.78**	43.90	**56.57**	**71.75**	**67.29**
DL-3	96.19	75.76	88.01	36.10	41.43	44.00	41.27	65.80	89.43	50.52	88.25	72.41	37.74	89.56	44.96	55.04	35.97	44.29	61.98	60.99
DL-4	93.97	76.18	87.82	39.47	46.42	41.78	45.38	55.68	85.86	55.52	76.62	70.39	46.66	89.16	45.45	57.75	18.83	33.90	67.20	59.69
DL-1*	97.75	82.14	91.33	51.68	57.82	53.27	64.12	73.30	91.45	60.69	93.22	78.31	56.69	93.79	67.40	80.83	49.75	59.86	74.71	72.53

Table 1: Per-class results on Cityscapes Validation Set. All DeepLabv3 models listed were trained for 40 epochs except for DL-1*, which was trained for 150 epochs. The best IoU scores amongst the first four methods are bolded. *Note: T. Light=Traffic Light, T.Sign=Traffic Sign, Veg.=Vegetation, Ped.=Pedestrian*

5.3. DeepLabv3: Original Implementation

The original ASPP layer of DeepLabv3 follows the architecture defined in Figure 4. For our implementation, we trained with a mini-batch size of 8 and 16. While a mini-batch size of 16 trained faster (albeit requiring additional computational resources), we found that a mini-batch size of 8 received a better validation mean IoU, due in part to a larger batch size resulting in worse generalization.[16]

We found the ideal base learning rate to be 0.007, and throughout training we also used a learning rate decay schedule. Since the base learning rate is 0.007, the learning rate for a given iteration i out of a max number training iterations m is defined as:

$$\text{Learning Rate} = 0.007 * (1 - \frac{i}{m})^{0.9}$$

5.4. DeepLabv3: Additional Models

In addition to training the original model of DeepLabv3, we wanted to experiment with various ASPP layers. This stems from wanting a better combination of dilation and filter sizes for the atrous convolutions to capture important contextual information for the Cityscapes Dataset.

We implemented DL-2, which is a variation of DeepLab, where the dilations for ASPP layers were changed to 3, 6 and 9 respectively. Since the dilation sizes were decreased, we hypothesized that this would result in better accuracy for finer details in images, such as motorbikes, traffic lights, and pedestrians.

Furthermore, we implemented a filter size of 4 instead of the default 6, and tweaked the dilations in successive ASPP layers to 8, 16, 24. The idea here was similar to DL-2. Since we were interersted in the final validation set results, we wanted to understand the trade-offs of having lower filter and dilation sizes for ASPP layers than was proposed originally.

Finally, we decided to add an additional ASPP layer to experiment with the learn more about the information that is being encapsulated by each layer. Our hypothesis here was that having more ASPP layers will improve the mean IoU score. Therefore, we tested DL-4 with atrous convolutional layer dilations of 6, 12, 18, and 24.

5.5. Quantitative Results

In the results shown in Table 1, the model DL-1 corresponds to the original architecture of the DeepLabv3 model discussed earlier. DL-2 is a similar DeepLabv3 model except with dilations of 3, 6, and 9. DL-3 corresponds to the DeepLabv3 model with a filter size of 4 and dilations of 8, 16, and 24. Finally, DL-4 is the model with 4 atrous convolutions, each with a filter size of 3 and with dilations 6, 12, 18, and 24.

Besides viewing the mean IoU on the 500 validation samples, we also evaluated each model on the 2975 training samples. As seen in Table 2, it is clear that after 40 epochs the models have not overfit to the training set. However, the best performing model appears to be DL-2, which has both higher training and validation accuracies after 40 epochs.

Method	**mIoU on Training Set**	**mIoU on Val Set**
DL-1	69.2	65.43
DL-2	70.97	67.29
DL-3	66.3	63.40
DL-4	64.8	59.69
DL-1*	77.95	72.53

Table 2: Training and Validation Set mean IoU scores for DeepLabv3 Models. DL-1* corresponds to model DL-1 trained for 150 epochs.

Though we experimented with various ASPP layer architectures, one potential reasoning for this is that with the larger filter and dilation sizes no additional information was added. This is because as the dilation increases in atrous convolution, it can reduce the number of valid filter weights. In other words, as the dilation increases then the number of weights that are applied to valid features instead of padded zeros decreases. An extreme example can be seen if a 3x3 filter is given a dilation size such that the resulting field of view is the same size as the input window. In this case, it is effectively performing the same function as a 1x1 filter, because the middle weight of the 3x3 is the only weight which impacts a majority of the result.

As a result, there is a tradeoff between the amount of information captured in an atrous convolution and the number of valid filter weights. This explains why the mean IoU is

lower for methods such as DL-4, because while the kernel size is larger and should be capturing more information, it is likely that with padding the weights are not properly learning values. Ideally the global average pooling will lower the impact of having too large of a filter, but with the input to the ASPP layer being 33x33, the dilation parameter can not be too large. This is again seen in DL-4 because the additional atrous convolution in the ASPP layer has too large of a dilation parameter. However, since there are additional parameters to train in both DL-3 and DL-4, another possibility is that both models could eventually outperform DL-1.

Given the small input window size of 33x33, it also makes sense why DL-2 with an ASPP layer with dilations of 3, 6, and 9 performed better than DL-1. By halving each of the dilation values, we were still able to infer context (at a smaller scale) and also ensured that more of the weights affected each element of the convolution.

Although DL-2 had the largest mean IoU score after 40 epochs, we initially trained DL-1 for 150 epochs after first implementing the original DeepLabv3 model. From this, we received a mean IoU score of 72.53% for what is denoted as DL-1* in Tables 1 and 2. This score is 5% less than the score reported in DeepLabv3 [11], but their method also utilized data augmentation to increase their training set and improve generalization.

5.6. Qualitative Results

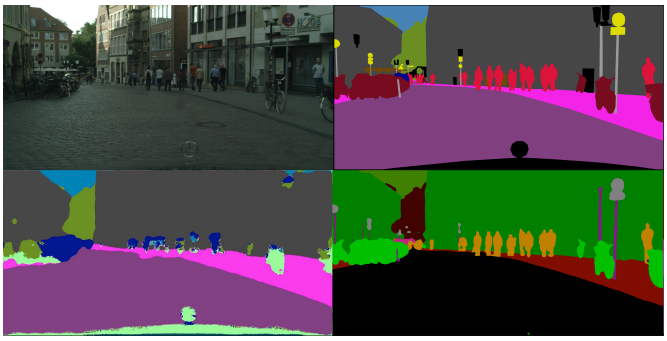

Figure 5: Top Left: Input Validation Image. Top Right: Ground Truth Fine Annotation. Bottom Left: SegNet Model Output. Bottom Right: DL-1* Model Output

Figure 5 shows the large qualitative improvement DL-1* had over our SegNet baseline when compared to the input image and ground truth annotation. Though the color correspondences are different between the ground truth and our model output, it is clear that DL-1* properly identifies pedestrians, bikes and traffic signs. Moreover, the edges between different classes are smoother when compared to the SegNet output for 7500 iterations. This means that our quantitative analysis was in tune with the qualitative results.

It is important to notice that DL-1* does better on some classes than on others. For instance, DL-1* completely ignores the segmentation of the car on which the camera is mounted.

Figure 6: Example of Errors in DL-1* Model Output. Left: DL-1* Model Output. Right: Ground Truth Annotation

Figure 6 highlights additional qualitative errors in the model prediction. Though DL-1* properly segments the road and sidewalk from pedestrians, errors occur with particular edge cases like the stroller annotated in brown in right image of Figure 6. Since there are likely not enough cases with such classes in the training set, having this in the validation set causes issues in the predicted output. In the model output, pixels in the top half of the pedestrian pushing the stroller is incorrectly segmented as a biker, likely due to the wheels on the stroller. The stroller itself is also segmented as part of a bicycle but also as part of the road.

Lastly, certain classes with clear boundaries such as traffic signs, poles, and traffic lights are not as finely defined in the model output when compared to the ground truth annotation. A potential reason for this is because of the wide range of the atrous convolutions, which can cause for small objects with clear boundaries to not be as well outlined. A potential next step to identify the cause of this issue would be qualitatively evaluating these boundaries on the output of the DL-2 model trained for 150 epochs, because it has smaller dilation parameters.

6. Conclusion

In this paper, we present improvements to DeepLabv3 to segment high level features in the Cityscapes dataset images more accurately. We experiment with various tweaks to the DeepLabv3 model including but not limited to adding more Atrous Spatial Pyramid Pooling Layers and changing the filter and dilation sizes. We then benchmark these results for the mean IoU metric as provided in the Cityscapes dataset and analyze the results we obtained for our changes to the model.

Since time and computing resources were serious constraints, we were only able to train each of the variations of DeepLabv3 models for no more than 40 epochs. Although we observed vast improvements over our baseline model, we would have hoped to run each of the models for more epochs (closer to 150). Nevertheless, our experiments were useful in bringing us closer to understanding the way to de-

rive feature maps that can be used to obtain global and fine-grain features from images during semantic segmentation.

6.1. Future Considerations

A natural continuation to our work is running the training network for more epochs. In particular, given DL-2's better performance than DL-1 after 40 epochs, the next step would be to also train DL-2 for 150 epochs and evaluate its performance on the validation set. We plan on adding more layers to the network with varying dilution and kernel sizes to explore the relationship between a deeper network and image segmentation results. We also plan to investigate pre-processing the data with naive clustering methods, as well as following similar data augmentation techniques used in DeepLabv3. This could help the model learn global features and filter out smaller anomalies.

Once we complete this aspect of the work, we would want to implement a more sophisticated work that has been proposed. For instance, we want to implement and experiment DeepLabv3+, an extension of DeepLabv3 with an additional decoder structure used to refine the segmentation results. Similar to other encoder-decoder models discussed earlier, this would help in our results, especially along object boundaries. With this addition we would ideally see a better validation mean IoU on the Cityscapes Dataset.

7. Contributions

Darren worked on the DeepLabv3 implementation and tweaks to the architecture. Additionally, he worked on quantitative analysis of the results and on generating qualitative results from Deeplabv3.

Aditya worked on the baseline model - understanding the implementation of SegNet, training the model on the Cityscapes dataset train set and evaluating and visualizing results.

8. Acknowledgements

We would like to thank Cityscapes for providing an exciting dataset and helpful benchmark tools (Cityscapes Github Repo), as well as Pragnesh Shah for his SegNet starter code (SegNet Github Repo). Additionally, we would like to thank Chenxi Liu for his DeepLabv3 framework (DeepLabv3 Github Repo).

References

[1] Shotton, J., Johnson, M., Cipolla, R. (2008, June). Semantic text on forests for image categorization and segmentation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1-8). IEEE.

[2] Teichmann, M. T., Cipolla, R. (2018). Convolutional CRFs for semantic segmentation. *arXiv preprint arXiv:1805.04777*.

[3] Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*.

[4] Long, J., Shelhamer, E., Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431-3440).

[5] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2), 303-338.

[6] Liu, W., Rabinovich, A., Berg, A. C. (2015). Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*.

[7] Noh, H., Hong, S., Han, B. (2015). Learning deconvolution network for semantic segmentation. In Proceedings of the IEEE international conference on computer vision (pp. 1520-1528).

[8] Ronneberger, O., Fischer, P., Brox, T. (2015, October). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (pp. 234-241). Springer, Cham.

[9] Badrinarayanan, V., Kendall, A., Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12), 2481-2495.

[10] Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J. (2017). Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2881-2890).

[11] Chen, L. C., Papandreou, G., Schroff, F., Adam, H. (2017). Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*.

[12] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., ... Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3213-3223).

[13] Cordts, M. (2019). Cityscapes Benchmark Suite, https://github.com/mcordts/cityscapesScripts

[14] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[15] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248-255). Ieee.

[16] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836.*