Street View Segmentation using FCN models

Yen-Kai Huang
Department of Computer Science
Stanford University
nykh@stanford.edu

Vivian Yang
Department of Electrical Engineering
Stanford University
viviany@stanford.edu

Abstract

We fine-tune a Fully Convolutional Network model and implemented a Dilated Convolution Network to evaluate their performance on a novel street-level imagery dataset, Mapillary Vistas Dataset, for street view images semantic segmentation. The Mapillary Vista dataset is harder to segment than all the other dataset released before. However, even dealing with the very complicated dataset, our performance of FCN-32s model can still reach up to 80.9% overall accuracy and 23.4% per-class accuracy. Qualitatively the FCN model was able to label objects occupying big areas on the street view image. However, more work needs to be done to improve the recognition for smaller objects.

1. Introduction

Semantic Segmentation is an important task in the field of computer vision. While similar to normal object recognition task, the goal of semantic segmentation is not to find a bounding box of recognized objects but to label each pixel according to which object or class it belongs to. Other names that refer to the same or similar task include "scene labeling" [5]. Semantic segmentation can come in many different settings. When the requirement is only to label each pixel according to class, it is very accurately described by other names such as "pixel-level Labeling" [11], or "pixelwise classification" [8]. When only the category is not enough and each object instance must also be identified (in the case of objects that can appear together but are separate, such as person, car, etc), the setting must also assign instance number to each pixel. This setting is often called instance-level segmentation or instance segmentation [6]. In this project we limit our scope to producing a class for each of the pixel.

Street view segmentation is an application of the semantic segmentation on street view images. Street view image segmentation plays a very important role in the context of autonomous driving. The use case often requires captioning on a video stream, but this is not our focus in this project and we will evaluate our system in an offline setting.

In this experiment we started with a Fully-Convolutional Network model that was previously trained on the VOC2011 image segmentation task and fine-tuned it for a newly released street view dataset to explore the performance of a widely-used model on the challenging posed by a new, complex dataset. We used overall accuracy as well as average accuracy across classes to benchmark the performance because they are easy to compute and provides an intuitive approximation of human perception of the performance. We achieved 80.9% overall accuracy and 23.4% average accuracy across class when trained using the full Mapillary Vista dataset on a cloud GPU instance. The gap between the overall and per-class accuracy is shown in qualitative assessment of output, the model is able to label pixels that belong to classes occupying majority area on an street view image, such as road, vegetation, sky, and building. However the system often misclassify or outright ignore smaller objects.

In order to overcome a problem of disappearing of fine feature, we experimented with the Dilated Convolution [16]. However, because each test image has a very large dimension, it became very time consuming to train the model on.

2. Related Work

General semantic segmentation has been studied since the beginning of computer vision as one of the setting for object recognition and scene understanding. Among the general semantic segmentation task specification and datasets, there was the Pascal Visual Object Classes Challenge of 2011[4]. In this task each pixel of the 2501 training images can be from 20 different classes (plus one "background" class). The challenge was well-accepted and included such classes as "person", "car", and "train" which are relevant to street view scene.

Street view segmentation is an application of the semantic segmentation on street view images. This is an important task when developing, for example, autonomous driving system, where the system must recognize and identify

lane, other car, and static or moving objects on the road to avoid. Cityscapes dataset [2] collected 20,000 images taken in many cities of Germany with a fixed camera whose labeling included 30 classes total. Among the classes are flat surfaces like "road" and "sidewalk", vehicles subclassed as "car", "truck", human pedestrians and riders, and objects like "vegetation" and "pole". Cityscapes provided a reasonable common reference point for different street view segmentation system to benchmark their performance. However due to the limited diversity in geographical locations, camera size, and lighting conditions due to time of day and weather, some raised questions to the ability to generalize for a system trained on the dataset, which inspired the creation of Mapillary Vistas dataset used in our work [10]. The Mapillary Vistas dataset is described in more details in Section-4.1.

Krizhevsky et al. re-introduced to the attention of academics for Deep Convolutional Networks as a viable solution to compute vision task [7]. One of the Convolutional Network variant [15] developed by the Visual Geometry Group, Oxford won the ImageNet [13] challenge in 2014. However, VGG14 suffers from a time complexity problem as the network repeatedly computes convolution over adjacent windows that are highly redundant. For this reason there has been many research to speed up the training and prediction of the convolutional network. Girshick et al. proposed combining an efficient region-proposal algorithm to identify objects and compute convolution for the regions. Girshick then achieved further speed-up by computing convolution of the whole image and then crop out the regions of interest (ROI) [12]. The speed up in the convolution network was very dramatic, until the region-proposal part of the system became the bottleneck. Thus He et al. proposed using another branch of neural network to compute the region proposal, which achieved good result on segmentation on Cityscapes dataset and a 5 frame per second throughput [6].

Another convolutional network that performed well when applied to the semantic segmentation task was the Fully Convolutional Network [14]. A FCN, designed by Shelhamer et al. The key insight of the FCN was that by replacing the fully-connected layers in traditional neural network with only convolution and transposed convolution layers, the network is able to achieve better performance with much fewer parameters. One of the important part of architecture is a skip-architecture that allows "coarse feature" in deep layers to propagate and combine with "fine features" in a shallow layer, which produces more accurate segmentation. Some of the extensions to FCN introduce variants to convolution that allows this combination without the skip layers. Chen et al. introduced "atrous convolution", or "convolution with upsampled filters" [1] to increase the receptive field of filter without introducing more parameters. Similarly, Yu and Kolton introduced dilated convolution to to aggressively increase the receptive field of kernel without introducing parameters or subsampling [16]. Another architecture that has proven to perform well on semantic segmentation was the Conditional Random Fields as Recurrent Neural Networks, whose pairwise potential can be computed to approximate a Recurrent Neural Network.

3. Method

In this project we apply the Fully-Convolutional Network model on the semantic segmentation task. To overcome some problem inherent to the FCN model, we then introduced Dilated Convolution to the model.

3.1. Fully Convolutional Network

A Fully Convolution Network (FCN) [14] is an end-to-end, pixels-to-pixels learning model, which can output a pixel-wise prediction and has been widely used for various segmentation tasks. The model differs from traditional model because it excludes any fully-connected layer and instead rely completely on convolution and transposed operation. As shown in Figure 1, the FCN model first perform many layers of convolution on the image to extract a multi-scale feature representation of the image, with the dimension (H_i, W_i, C_i), where C_i is the number of channels or kernels. The layers are stacked from shallow layers to deep. Shallower layers have dimension that are closer to original image, while the deeper layers will be much smaller in height and width but have many more channels. Finally, the last layer performs a transposed convolution that increases the dimensions to (H, W, C'), whose height and width are the same as the input image, yet the array on each pixel represents a likelihood distribution that the pixel belongs to each class. To be more precise, Figure 2 draws out the actual number of convolutions and max-pooling that each layer has.

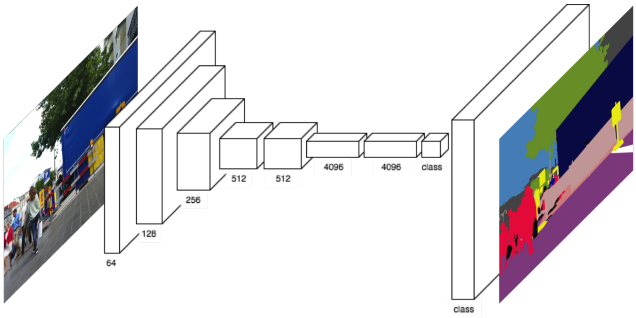

Figure 1. Fully Convolutional Network

The particular architecture we used in this project is FCN-32. The 32 indicates that in the pipeline it performs five max-pooling to reduce the size of image by $1/32$. This

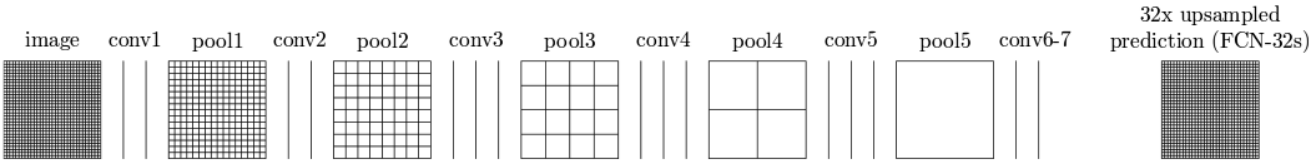

Figure 2. FCN-32s layers

architecture is very efficient to compute but can result in losing fine-grained features.

3.2. Dilated Convolution Network

Yu and Kolton introduced dilated convolution [16]. This variant of convolution supports exponentially expanding receptive fields without losing resolution or coverage, by "dilating" the kernel by $l - 1$ pixels, as shown in Figure 3.

Our Dilated Convolution Network architecture follows the general architecture of Fully-Convolutional Network, but introduced a context module. The context module contains dilated convolutions to aggressively increase the receptive field of the kernel without down-sampling the image. In the last layer, the network performs a transposed convolution just like in FCN to generate pixel labelings, the architecture parameters is summarized in Table 1.

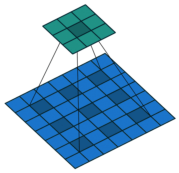

Figure 3. Diagram of a Dilated Convolution with dilation factor $l = 2$

4. Experiment

In this project we repurpose a Fully Connected Network model pre-trained on a general semantic segmentation task through fine tuning, and evaluate on the preprocessed dataset.

In previous study, segmentation trained on VOC2011 FCN-32s model can successfully segment simple images. The pretrained model reaches a high accuracy. However, the segmentation images contain only a few main objects, e.g. one or two animals, people or vehicles. The model has not been experimented on complicated dataset, where one image has about ten or more objects to segment.

4.1. Dataset

In our experiment, we used the Mapillary Vistas Dataset [10]. Mapillary Vistas dataset is a very new street-level imagery dataset, including images with corresponding, instance-specific, and pixel-accurate annotations. This dataset has much more diverse in geographical locations, weather circumstances (sun, rain, snow, fog, haze), and day times (dawn, daylight, dusk, and even night) than previous datasets, such as VOC2011 and Cityscape Dataset. It contains 25,000 high-resolution images (split into 18,000 training, 2,000 validation, and 5,000 testing images) and it has 100 object categories, 60 of those instance-specific.

As previously mentioned, this dataset was inspired by the more widely-used Cityscapes street view dataset, while addressing some of the problem of Cityscapes. The result is a dataset that is much closer to the real-world application. One difference between the two dataset is in the way the images are collected. Cityscapes dataset collected its images in 23 cities in Germany through a mounted camera in a single car. This ensures the image resolution and aspect-ratio are all identical, while the distribution of objects in the street can also be similar. Mapillary Vista, on the other hand, collects images from user upload of its cellphone map application. The result is a very diverse set of resolutions and aspect ratios. The Mapillary Vista images are also taken all around the world, with only a few images belonging to the same location. Another significant difference is in the granularity of labeling between the two datasets. As shown in Figure 5, it can be seen that Mapillary Vista labeling are much more fine-grained. It does not have the black (unlabeled) pixels of Cityscapes, and the general contour of objects are preserved much better. In this sense Mapillary Vista has a much better labeling quality than the Cityscapes. These differences can mean the system will have to learn a more generalizeable set of model to account for the fine-grained labeling and different image conditions. Information of different datasets is summarized in Table 2.

Due to the novelty of the Mapillary Dataset, it has no previous implementation. We repurposed a data loader for the VOC task on the dataset to correctly arrange and load the data. Because the size of the training images are very large (averaging over 1000 pixels in height and width), we first crop a 1024×1024 square from the center of image, and then shrink it down to 256×256 pixels. The same is done for the label but instead of shrinking we used down-sampling. This ensures bilinear scaling doesn't introduce wrong label values. We also wrote a visualization script to

Table 1. Dilated Convolution Network

Layer	1	2	3	4	5	6	7	Final	
# of Convolution	2	2	3	3	3	1	1	1	
Kernel Size	3	3	3	3	3	7	1	1	
Dilation	1	1	1	1	2	4	1	1	
Output Channel	64	128	256	512	512	4096	4096	N	
Max Pool	Yes	Yes	Yes	No	No	No	No	No	
Context Layer	1	2	3	4	5	6	7	8	Final
# of Convolution	2	1	1	1	1	1	1	1	1
Kernel Size	3	3	3	3	3	3	3	3	1
Padding	1	2	4	8	16	32	64	1	0
Dilation	1	2	4	8	16	32	64	1	1
Output Channel					N				

* At the end, need to do a ConvTranspose with Kernel Size = 16, Stride = 8, Padding = 4, Output Channel = N
* N = number of class

better print out the results.

Figure 4. Mapillary Vistas Dataset

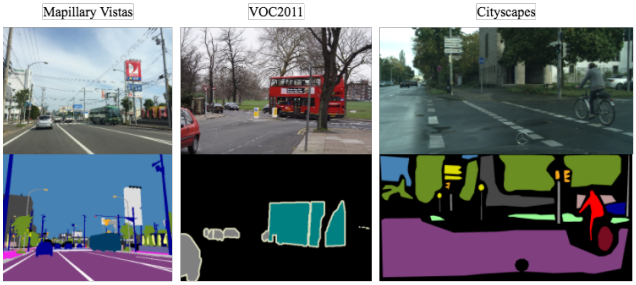

Figure 5. The Labeling of Three Datasets

Table 2. Comparision of Three Datasets

	VOC 2011	CityScapes	Mapillary
Training set	1112	19,500	18,000
Validation set	1111	500	2,000
Class	20	30	65
Instance label	Yes	Yes	Yes
Location	N/A	Limited	Diverse
Size	Varies	Fixed	Varies

4.2. Evaluation Measures

In our experiment, we used two popular evaluation measures [3] to judge the performance of our result: the Overall Pixel (OP) accuracy measures the proportion of correctly labeled pixels; the Per-Class (PC) accuracy measures the proportion of correctly labeled pixels for each class and then averages over the classes.

To express the computation formula of OP and PC accuracy, we assume that L is the number of classes, C_{ij} is the number of pixels having ground-truth label i and whose prediction is j, and $G_i = \sum_{j=1}^{L} C_{ij}$ is the total number of pixels labeled with i. Then:

$$OP = \frac{\sum_{i=1}^{L} C_{ii}}{\sum_{i=1}^{L} G_i}$$

$$PC = \frac{1}{L} \sum_{i=1}^{L} \frac{C_{ii}}{G_i}$$

Intuitively, this measures diagonal of a confusion matrix over the number of pixels. The overall pixel accuracy in particular provides an intuitive approximation for qualitative perception when the image is viewed in its overall shape and not its details. A system can achieve a high overall accuracy by learning to label objects occupying big areas in the image, while ignoring objects that appear rarely and are small. On the other hand, per-class accuracy requires objects each class to be classified accurately, so such a system will score low.

4.3. Result

4.3.1 Pretrained VOC2011 FCN-32s model

First, we ran the "model Caffe to Torch" script [1] provided by K. Wada to transform the Caffe pretrained model into Torch

[1] Pytorch-FCN: https://github.com/wkentaro/pytorch-fcn

usable document. Then, we evaluate the performance of this pretrained model on VOC2011 validation set and get the following result. This result shows the validity of the pretrained model, which allow us to attempt transfer learning based on this model.

Table 3. Pretrained Model Evaluation

Learning Rate	Epoch	Accuracy	Class Accuracy
1e-10	9	90.48537	76.47010

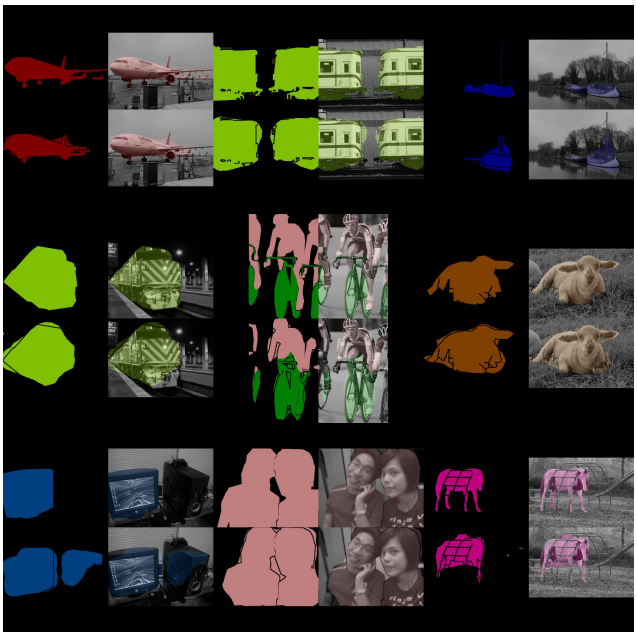

Figure 6. Best Performance on VOC2011 using FCN-32s Model

4.3.2 Finetuned FCN-32s model

To finetune the pretrained VOC2011 FCN-32s model on our Mapillary Vistas Dataset, we read over the Finetune example script [2] provided by J. Johnson and implement some of the main concept into our own finetune script. In this part we need to rewrite both training and evaluation script to fit FCN-32s model with the Mapillary Dataset. The training based on fine-tuning comes in two phases: in the first phase only the last few layers are replaced and trained to account for the new output shape (from 20 classes to 65 classes). In the second phase the whole network is trained.

We experimented by growing the number of training images. For the training subset with 5000 images we varied the hyperparameter to search for the best combination. In the end we arrived at a batch size of 16, learning rate of 1×10^{-4} for the fine-tuning phase and 1×10^{-6} for the second phase. After this we trained on the full training set of 18,000 images. The results are shown in Table 4 and 5. The learning curve corresponding to the best hyperparameter when trained on 5000 is shown in Figure 8, while the learning curve of full training data is shown in Figure 10. As shown in the learning curves, the training loss can drop to the magnitude of 1×10^4. We can then visualize the weights, as shown in Figure 7 and 9.

Table 4. 5000 Training Data

Entire Model MaxIter	Entire Model LR	Accuracy	Class Accuracy
6000	1e-07	72.5748	9.74099
6000	1e-08	68.7077	9.03626
10000	1e-04	79.8610	19.72758
10000	1e-05	78.9662	19.63397
10000	1e-06	76.8709	15.89977

* with Last Layer MaxIter = 1000, Last Layer LR = 1e-04, Decay = 0.0005, Batch Size = 16

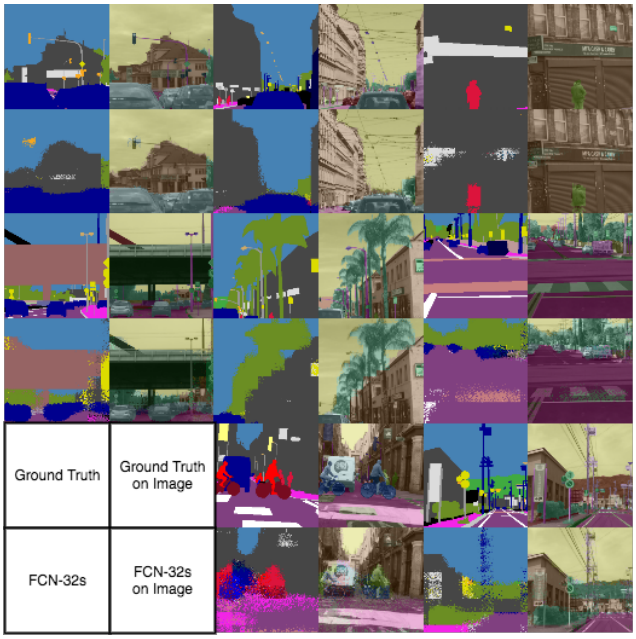

Figure 7. The Output of the Best Result (Learning Rate: 1e-4)

Table 5. 18000 Training Data

Entire Model MaxIter	Entire Model LR	Accuracy	Class Accuracy
23000	1e-04	80.8653	23.39519

* with Last Layer MaxIter = 1000, Last Layer LR = 1e-04, Decay = 0.0005, Batch Size = 16

[2] Pytorch-finetune: `https://gist.github.com/jcjohnson/6e41e8512c17eae5da50aebef3378a4c`

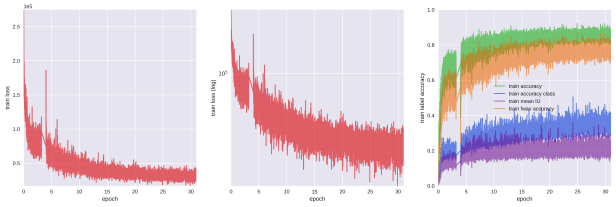

Figure 8. The Learning Curve of the Best Result

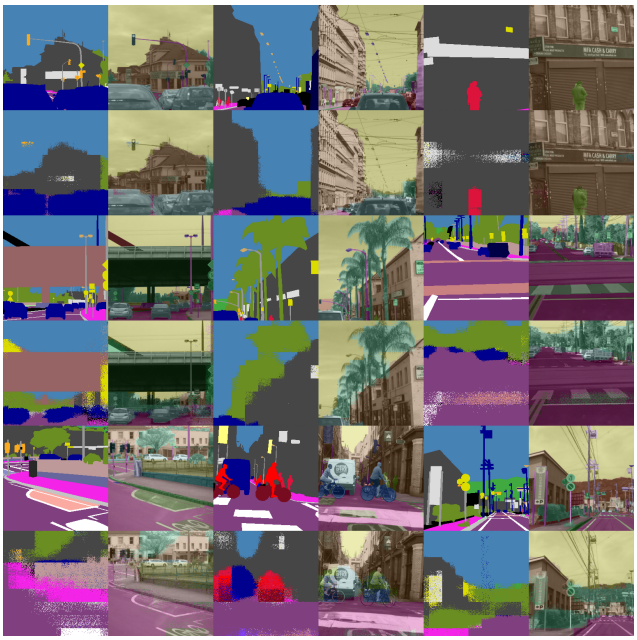

Figure 9. The Output of 18000 Training Data

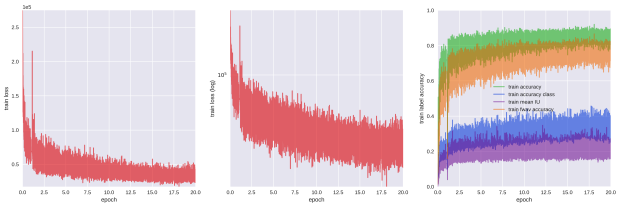

Figure 10. The Learning Curve of 18000 Training Data

4.3.3 Dilated Convolution model

Since comparing with CamVid, VOC2012 and KITTI Dataset, Cityscapes shares more similarities with Mapillary Vistas Dataset, we transformed the Caffe written Dilated Convolution model for Cityscape [3] into Pytorch code and implemented the model on Mapillary Vistas Dataset. However, because the images used for dilated convolutional network is in its full size, plus the lack of a pre-trained model, the training time of this model is very long, denying us a satisfactory converging result even after 6 hours of training. The resulting accuracy is very law, and when visualized the output labels are similar to gaussian noise, which indicates the network has not fully converged

[3]Caffe-Dilation: `https://github.com/fyu/dilation`

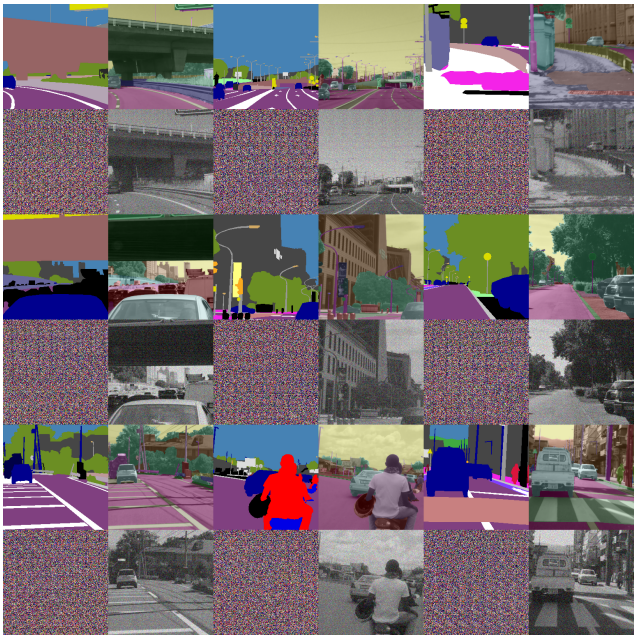

Figure 11. Dilated Convolution Failure Result

5. Conclusion

In the result achieved so far, our FCN model were able to correctly label objects that occupy big areas in a common street view image to achieve high overall accuracy. These include sky, road, cars, buildings, and pedestrians. This partially proves the feasibility of the task. However, the boundaries are still not very cohesive, which can be seen as a weakness of the FCN-32s model itself.

Another difficulty posed by the Mapillary Vistas segmentation task is the fact that it labeled some objects with very small or narrow dimensions, such as bird, street light, and sign poles, which can disappear during the down-sampling process. Some possible solutions to overcome these shortcomings are FCN-8s, Mask R-CNN, and dilated convolution network.

5.1. FCN-8s

The structure of FCN-8s is actually quite similar to FCN-32s model we used. However, in FCN-8s, images only need to go through three max-pooling layers, which will reduce the size of image by 1/8. Therefore, it only need to backward convolution with an output stride of 8, which will improve segmentation detail and reach a better performance.

5.2. Transfer Learning from Cityscapes

In this experiment we based our model on a FCN-32 model pre-trained on VOC 2012 segmentation task by simple availability, however intuitively it should be much better to fine-tune a model that was pre-trained on the Cityscapes dataset considering its similarity to the Mapillary. An experiment worth conducting is to see if a fine-tuned model based on Cityscapes dataset can outperform out model.

5.3. Dilated Convolutional Network

In our experiment we attempted training a dilated convolutional network but was unable to make it converge. One future direction would be to implement a more efficient Dilated Convolutional Network and train it on the dataset fully to evaluate its result.

5.4. Mask R-CNN

Mask R-CNN [6] is a very new modification of the Faster R-CNN model [12], which in terms extends on the R-CNN approach. This architecture was developed for the instance segmentation task, and has been applied on the Microsoft COCO object classification task [9] as well as Cityscapes Dataset [2] and has shown to outperform current state-of-the-art. Therefore, we think this method would be worthy of further testing on the very new and complicated Mapillary Vistas Dataset to see if the method can still perform well.

References

[1] 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.

[2] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. *ArXiv e-prints*, Apr. 2016.

[3] G. Csurka, D. Larlus, and F. Perronnin. What is a good evaluation measure for semantic segmentation? *BMVC*, 27, Sept. 2013.

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[5] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, Aug 2013.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. *ArXiv e-prints*, Mar. 2017.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[8] H. Li, R. Zhao, and X. Wang. Highly efficient forward and backward propagation of convolutional neural networks for pixelwise classification. *CoRR*, abs/1412.4526, 2014.

[9] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft COCO: Common Objects in Context. *ArXiv e-prints*, May 2014.

[10] Mapilarry Research. Mapillary vistas dataset, May 2017.

[11] P. H. O. Pinheiro and R. Collobert. Weakly supervised semantic segmentation with convolutional networks, 2014.

[12] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *ArXiv e-prints*, June 2015.

[13] 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. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[14] E. Shelhamer, J. Long, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. *IEEE Pattern Analysis and Machine Intelligence (PAMI)*, May 2016.

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

[16] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2015.