A GoogLeNet Implementation of FCN for Semantic Segmentation in TensorFlow

Tianxiao Zhao, Yang Wang, Junxun Luo

KTH Royal Institute of Technology

Abstract. Fully Convolutional Network (FCN) is regarded as groundwork for semantic segmentation tasks, and nowadays most of the state-of-the-art methods in this area are built up based on this idea. This paper serves as an attempt to dig deeper into the capability of a reimplementation of FCN in TensorFlow. We replace the pre-trained model VGG16 net with GoogLeNet InceptionV3 and add upsampling layers on the top. Then we fine-tune the whole net with training data from PASCAL VOC 2012 and utilize skip different architectures. Our FCN-GoogLeNet performance is not as good as the original paper and we provide some inspections on the reasons behind. To test its capability with more classes, we also train and validate our FCN on dataset MIT Scene Parsing, which contains 150 object classes in total. A drop of performance in this case indicates that such datasets are beyond the ability of our FCN.

Keywords: Semantic Segmentation, Fully Convolutional Networks, Transfer Learning, GoogLeNet, Upsampling, Skip Architecture

1 Introduction

Semantic segmentation is one of the most popular research areas of computer vision, which is still challenging nowadays. In theory, it combines two tasks, one is image segmentation and the other is object classification. It clusters parts of images together which belong to the same object class. With semantic segmentation, we can finally get a pixel-by-pixel semantic annotation of the image.

Compare to object detection, semantic segmentation is a big improvement and advantage in terms of getting different and pixel-wise object instances However, there are a couple of problems and challenges. The difficulty is mainly in the following aspects: 1) Object level: Due to different illumination, angle of view and distance, the same object in the image might be far different . 2) Class level: Objects of the same class might be different, and objects of different classes might be similar. For example, a pedestrian in front of a car which visually divides the car into two parts. 3) Background level: Clean background helps the segmentation, but in fact, the background is usually complex, which might be misleading.

2 Background

Before the breakthrough of deep learning, there are a couple of popular methods of image segmentation. Threshold segmentation[1] is one of the most basic methods of image segmentation, it divide the pixels based on the difference of color or grayscale of the pixel.Edge segmentation is to detect some points on the edge then generate a close segmentation area using some certain algorithm.The active contour model[2], also known as the Snake model, formulate the segmentation into an energy-minimize problem to find the edges. The watershed algorithm is a regional segmentation method based on morphology. The basic idea is to regard the gradients of the image as a hypothetical terrain surface, in which the gradient of flat area is small, regarded as the basin, and the boundary of large gradients is regarded as the ridge. The regional growth method[3] is also a commonly used regional segmentation technique. The regional growth method is also a commonly used regional segmentation technique. The basic idea is to first define a growth criterion and then look for a seed pixel in each of the divided regions. Random forest[4] which contains multiple decision trees, is used as a classifier. Image segmentation based on graph theory is to map the image as a weighted undirected graph in which the pixels are treated as nodes. The weight of the edge between nodes corresponds to the non-similarity measurement between two pixels. Cutting of these edges depends on the energy function. Markov Random Fields is an undirected probability graph model used for image segmentation. The idea is to give the pixel a random value and then classify the pixel by probability.

After the breakthrough of deep learning, a series of semantic segmentation methods based on convolution neural network, especially fully convolutional network (FCN) have been proposed and has reached a big progress. The paper Fully Convolutional Networks for Semantic Segmentation (2015)[5] is one of the milestone in this research area. Using FCN, the size of the input image can be arbitrary. It transfer the current state-of-art classification networks by fine-tuning to the segmentation task. It also define a skip architecture combining semantic information from a deep and coarse layer to improve the classification accuracy and detailed segmentations.

After FCN, SegNet[6], Dilated Convolutions[7], DeepLab v1[8] and v2[9], RefineNet[10], PSPNet[11], Large Kernel Matters[12], and DeepLab v3[13] were proposed successively and has improved the pixel accuracy from 67% (FCN) to 85% (DeepLab v3).

3 Approach

The approach we took in this paper originated from the idea of fine-tuning. Given a trained classification deep network on a certain dataset, we could change the last one or a few layers and tailor it to train for another classification dataset. It would save time instead of training from scratch and equally importantly, it would work.

In this paper, we pushed this even further. We used a pre-trained inceptionV3 as a warm start and modified the network to do a segmentation task. The feasibility of this approach had already been proven in [5]. In this section we would like to further explain how we implemented this approach and what changes we made.

3.1 Segmentation Architecture

As pointed out in the previous paper[5], we first converted the network to a FCN, meaning we needed to get rid of the fully-connected layer at the very end of the network and make that layer convolutional, see Fig. 1. Fortunately, the network model we got from TensorFlow Model Zoo was very well-defined. It was already a FCN because every layer was a convolutional layer and it had an option to do a ”spatial squeeze”, making the output of the network a one-hot label (1D array) instead of a tensor.

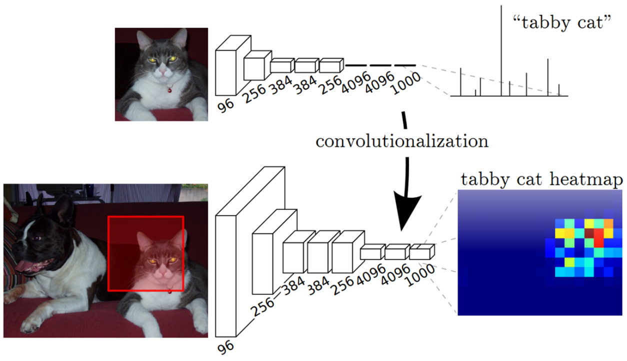

Fig. 1: Illustration about how to convert a classification net into a fully convolutional net.

The next step was to define the upsampling network and connect it to the end of the pre-trained network. As the network went deeper, the output to the next layer got smaller. To do a pixel-wise labeling, an upsampling process was needed. We used convolutional-transpose layer offered in the slim API to structure this part and calculated the filter size and stride according to the formulas we learned from the lectures.

After this part was done, we ran some preliminary test to evaluate our network and the method. The result labeled image was, in a sense, coarse and in low resolution so we figured that adding the skip structure was necessary. The skip structure was a intuitive way to combine the stack of features collected along the depth of the network. When the layer was close to the output (deep), the outcome of the layer had a low resolution and it contained global information about what it was. On the other hand, shallow layer had a high-resolution output and the local information of where it was. Without this skip structure, the output of the segmentation task would be more like the label of the classification task populating the whole image. Therefore, instead of upsampling the output of the classification network directly to the image size, we upsampled it a bit, making it the same size as a shallower layer, fused them, and repeated this process. In the previous paper, they basically took the outputs of three selected layers and fused them. We did the same for comparison, see Fig. 2.

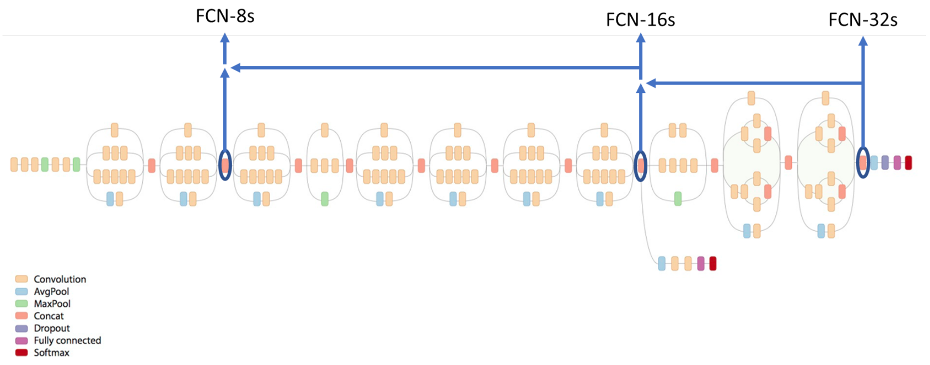

Fig. 2: Skip architectures we experimented on FCN-GoogLeNet model. With different numbers of layers fused into final prediction, we can obtain FCN-32s, 16s and 8s accordingly. The endpoints we extracted layer information from are illustrated above.

3.2 Changes from the Previous Paper

There were 3 highlights for our project. We first decided to move the project from Caffe to TensorFlow for its simplicity and popularity. Secondly, we chose GoogLeNet, InceptionV3 in this paper, as our pre-trained classification network, a different starting point from previous work. And additionally, we tried this method on another segmentation dataset, MIT Scene Parsing.

TensorFlow Framework TensorFlow is a popular Deep Learning framework for both novice and expert and it is growing very fast. During the course of our project, we happened to witness a few major releases, most importantly version 1.1 and the full-on use of a high-level API, slim. These major updates gave us more powerful tools to set up trainings and simplify the definition of each layer, however, it also brought us challenges for lack of documentation and examples to learn from. We had to keep up with the updates. And even if we did, sometimes the examples we found on the Internet did not, creating gaps between versions and APIs. These were some of the major issues when we started our project.

But when everything was settled and stable, TensorFlow offered many projects to learn from, both from Google and the Internet. We could examine other people's work and try to push our own. And most importantly, defining the structure of a network became easy and clear. With the help of high-level API, like slim, we were able to create layers neatly. And the advantage would be more obvious when there were repeated structure in the network.

InceptionV3 By the time we started our project, InceptionV4 was already released. We chose InceptionV3 for it was mentioned in previous work and comparable to other networks doing the same job, because they were from the same time. It was also more developed in that it had many tutorials, examples, and applications to learn from.

The major difference between InceptionV3 and VGG was the inception module, making InceptionV3 "go deeper" while the computation was efficient. The structure seemed more abstract and less intuitive given the fact that it was, to some extent, optimized for computational time.

The advantage of starting with a pre-trained network was that it saved a huge amount of time, given the fact that these networks were trained parallel with many first-class GPUs for a long time.

MIT Scene Parsing In previous work, three different datasets (PASCAL VOC, NYUDv2 and SIFT Flow), which contains 20, 40 and 33 semantic categories respectively, were used to test FCN's performances on semantic segmentation tasks. Here, we consider to take a step further by replacing these datasets with MIT Scene Parsing dataset. This dataset expands its object classes to 150, and allows appearance for multiple objects in the same image. Using this challenging dataset, we plan to test FCN's capability to handle large amounts of data at one time.

4 Experiments

4.1 Basic Setup

Based on the approach mentioned above, we choose GoogleNet-InceptionV3 as our pre-trained model, convolutionalize it, add upsampling layers on the top and fine-tuned it from end to end. And we choose PASCAL VOC 2012 as our dataset, which contains 1464 images for training and 1449 images for validation, including 20 different object classes. The networks are trained and fine-tuned for 100000 epochs on PDC with NVDIA Tesla K80 acceleration. It takes around 6 hours to fine-tune only the output layer of pre-trained model along with upsampling layers while keeping other layers fixed, and around 16 hours to fine-tune all layers by back propagation through the whole net. Fig. 3 and Fig. 4 illustrate the evolution of pixelwise training loss for fine-tuning all layers of a FCN-GoogLeNet-8s model trained on different datasets.

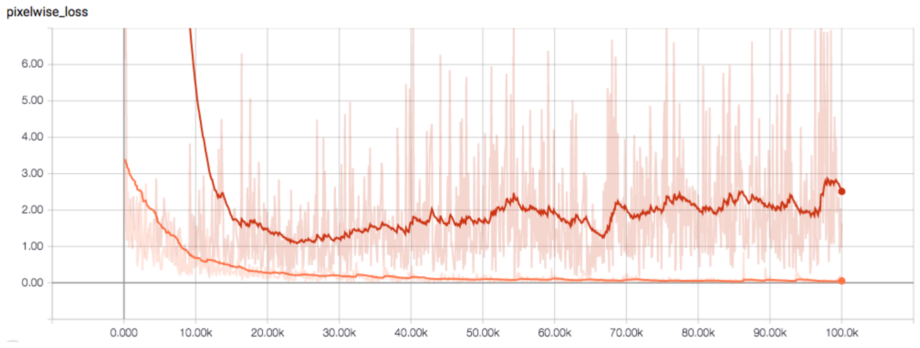

Fig. 3: Pixelwise training loss (orange) and validation loss (red) of FCN-8s trained on PASCAL VOC 2012. Faded-color curve represents original data points while solid line represents a smoothed loss.

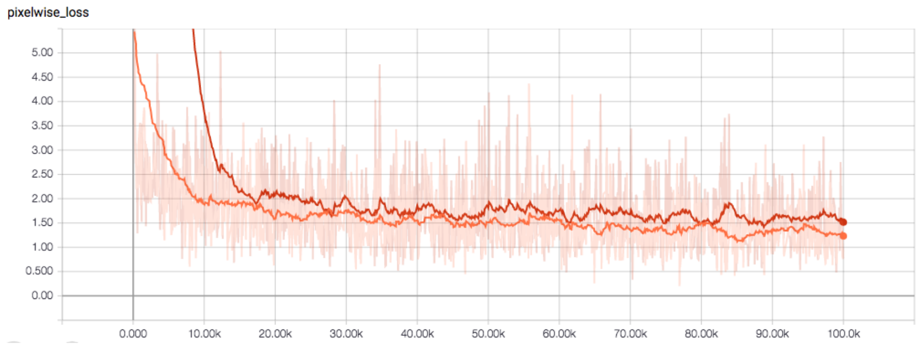

Fig. 4: Pixelwise training loss (orange) and validation loss (red) of FCN-8s trained on MIT Scene Parsing. Faded-color curve represents original data points while solid line represents a smoothed loss.

4.2 Optimizer Setup

Compared to the original implementation, we train our networks using Adam optimizer with a minibatch size of 3 resized images. Since GoogLeNet is deeper and more complicated than VGG net, a relatively larger learning rate, which in this case is set to 0.00032, is necessary to guarantee gradient flow throughout the whole network. Besides, dropout is included with a 0.85 keep-probability.

4.3 Datasets

To further test FCN's performance and generalization, we also train a FCN-GoogLeNet model on the dataset MIT Scene Parsing, which covers 150 object classes. This dataset contains 20000 images for training and 2000 images for validation, and usually multiple objects in different classes appear in the same image, which makes it more challenging for FCN to do semantic segmentation work. Other parameters are kept unchanged during the training process.

4.4 Results

We demonstrate some of our experiment results based on what dataset they are trained on. For simplification, all the training and validation images are resized to 224×224, and no more preprocessing operations are employed afterwards. We also evaluate our FCN-GoogLeNet model's skip architectures and compare them in a parallel manner.

PASCAL VOC Fig. 5 and Fig. 6 demonstrate two groups of experimental results, with colorbars locating besides them. These results are post-processed by simple colorization. Each color in the colorbar corresponds to a specific semantic category. Along with background and ambiguous region, there are 22 different categories in total, and usually each image only contains one to two various classes of objects.

One significant discovery from above is a gradual refinement of results as more skip layers are included in the network structures. When no skip layers are added, which makes up a FCN-32s net, results show up with coarser details and boundary information, but objects are at least placed in a globally correct position; As more information from different layers is fused into prediction, finer details and shape information will be captured and appear. For instance, we can easily identify sheep's legs and the hand bar of the kid's bicycle from FCN-8s results in Fig. 5 and Fig. 6, but when it comes to FCN-32s results, those parts are more blurry and warp more easily.

Another issue about the results is that points from similar objects are easier to be misclassified even though their outlines look quite alike those of ground truth, e.g. some parts of the sheep in Fig. 5 are misclassified as cow since they have similar shape and size.

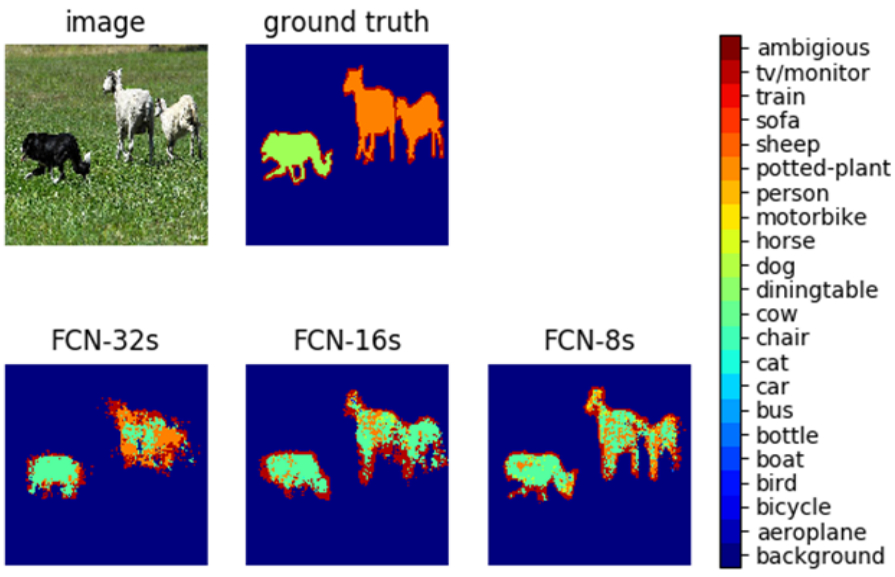

Fig. 5: Experiment results of different skip architectures.

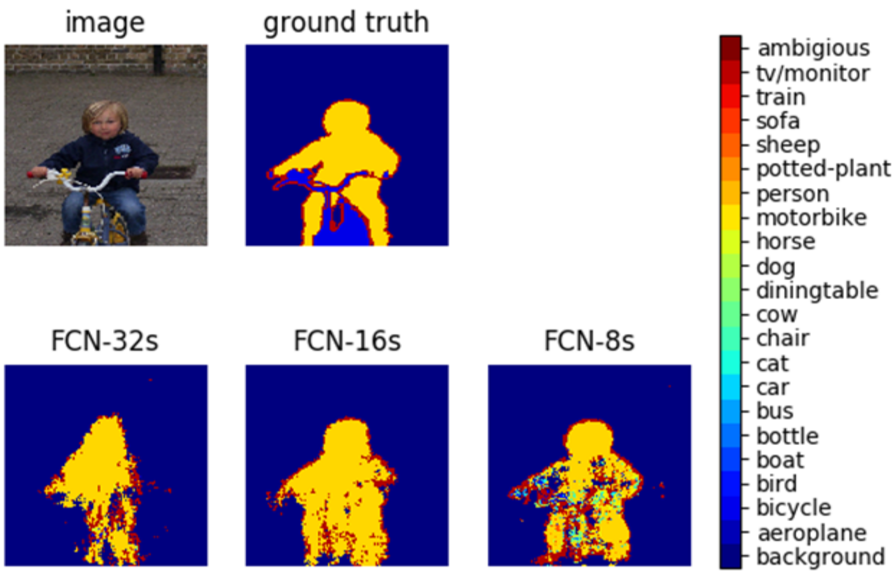

Fig. 6: Experiment results of different skip architectures.

MIT Scene Parsing We provide two acceptable results of MIT Scene Parsing dataset here - Fig. 7 and Fig. 8, and both of them are predicted by a FCN-8s net and post-processed by conditional random field. Since too many object classes are included, the colorbar on the right is disabled for legibility.

In general, we can expect a drop of performance when FCN is faced with a dramatic increase of semantic categories. As is seen in Fig. 7 and Fig. 8, some objects are missing in the final prediction, such as those orange objects in Fig. 7 and light bulbs in Fig. 8. Besides, edge details are badly ruined even if information from several layers is already integrated into prediction.

Same as the results of PASCAL VOC dataset, our FCN-GoogLeNet is more sensitive to specific object classes than others. Furnitures, humans and animals are easier to be segmented reliably in this case.

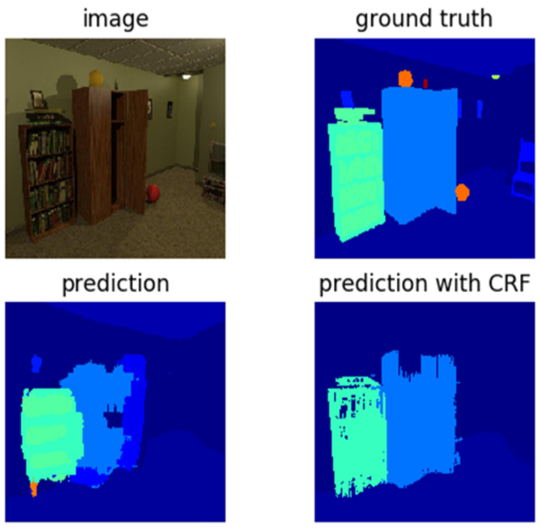

Fig. 7: Experiment results of MIT Scene Parsing.

Metrics Apart from those image segmentation results above, we also calculate four metrics from common semantic segmentation and scene parsing evaluations as what original paper did. We define the four metrics here, same as [5], for clarity:

Let n_{ij} be the number of pixels of class i predicted to belong to class j; n_{cl} denotes total object classes; $t_i = \sum_j n_{ij}$ denotes the total number of pixels of class i. Based on these definitions, we compute metrics as following:

- pixel accuracy: $\sum_i n_{ii} / \sum_i t_i$
- mean accuracy: $(1/n_{cl}) \sum_i n_{ii}/t_i$

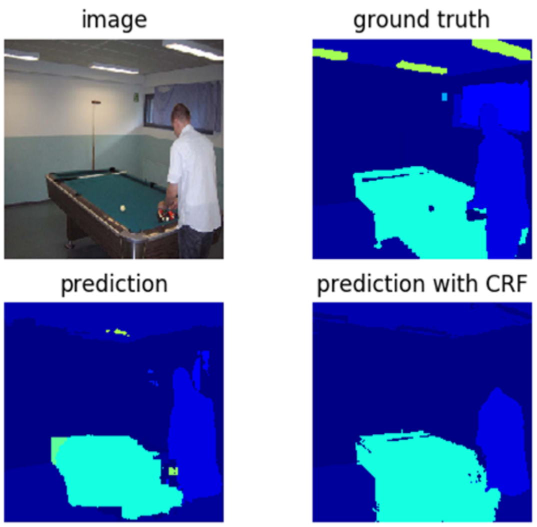

Fig. 8: Experiment results of MIT Scene Parsing.

- mean IOU: $(1/n_{cl}) \sum_i n_{ii}/(t_i + \sum_j n_{ji} - n_{ii})$
- frequency weighted IOU: $(\sum_k t_k)^{-1} \sum_i t_i n_{ii}/(t_i + \sum_j n_{ji} - n_{ii})$

We compute and list the four metrics for different skip FCNs (for PASCAL VOC dataset), as is shown in Table 1. The results confirm skip architecture's effectiveness as all the metrics improve in value when more skip layers are used. We also compare these metrics of FCN-GoogLeNet with the statistics from the previous paper[5]. It seems that using GoogLeNet as pre-train model generally fails to achieve in similar results as using VGG16 net previously.

Table 1: Metrics comparison of different skip FCNs and pre-trained models trained on PASCAL VOC dataset.

	pre-trained model	pixel acc.	mean acc.	mean IOU	f.w. IOU
FCN-32s	GoogLeNet	72.3	22.5	20.2	60.2
FCN-16s	GoogLeNet	73.6	28.5	25.8	62.3
FCN-8s	GoogLeNet	74.9	29.7	26.8	64.4
FCN-32s	VGG16 in [5]	89.1	73.3	59.4	81.4
FCN-16s	VGG16 in [5]	90.0	75.7	62.4	83.0
FCN-8s	VGG16 in [5]	90.3	75.9	62.7	83.2

5 Analysis

As is shown in previous sections, there exist several major problems in our GoogLeNet implementation. We will analyze possible reasons and discuss visible solutions for them in this section.

First of all, when dealt with semantic segmentation, GoogLeNet seems to be a suboptimal choice compared with VGG16 net since it gives out much lower metrical results than those from the previous paper. The corresponding reason can be multi-faceted. According to the training and validation loss evolution in Fig. 3, a slight overfitting problem may exist in this case and cause a degraded generalization. The frequent oscillation of validation loss implies that a too small batch size and a too large initial learning rate are to be blame. Plus, in GoogLeNet, the size of output doesn't change much after a bunch of consecutive layers, which means the extracted information may be quite similar to each other. Therefore, when applying a skip architecture in network, information from only three different layers may not be enough to guarantee accuracy in the final prediction. Naturally, in order to solve this, we could try out a better initialization of parameters, e.g. a larger batch size or a smaller learning rate, and more skip layers can be added into the network structure. If necessary, data augmentation can be employed as well.

Another issue is about the extremely low value of mean accuracy and mean IOU, as is shown in Table 1. This may be mainly caused by a relatively small validation set we use. The equations of mean accuracy and mean IOU require to operate a division by the number of semantic categories to average through all classes. Due to a memory issue, we are restricted to evaluate our FCNs on a shuffled smaller validation set with a size of 900 rather than 1449. Using a smaller validation set is equivalent to test our skip FCNs on a dataset with fewer semantic categories, which makes metrics calculation concerned with number of classes have a tendency to give out smaller values. We believe that an access to more computational resources may address this problem directly.

Besides, we notice that there are some misclassifications in the final predictions even though sometimes they obtain accurate edge details and positions. This can be explained by the fact that object with many examples in the dataset are easier to be correctly identified. For instance, vehicles, humans and animals are easier to be classified since the network is intensively trained on training images with these objects. Objects that have few examples in dataset and share certain features with dominating ones are easily attracted to a wrong category. To handle this problem, one may collect more training data or perform data augmentation for those scarce objects to balance quantities of training images for each class in principal.

When changing PASCAL VOC dataset to MIT Scene Parsing, we observe a worse performance for FCN on semantic segmentation tasks. As is mentioned before, this dataset contains 150 semantic categories, which turns out to be beyond FCN-GoogLeNet's capability. Also the fact that usually multiple objects present in each image makes learning more difficult.

6 Conclusion

Fully convolutional network is proven to be an effective tool for semantic segmentation tasks. We reimplement this network structure in Tensorflow and replace its pretrained model with GoogLeNet InceptionV3. Generally, its performance is acceptable but not as good as the results in [5], which implies that VGG net has some strengths over GoogLeNet in semantic segmentation. Also, a performance drop can be expected when semantic categories increase in a exploding rate.

As for future work, more experiments, such as grid search or random search, will be conducted to investigate parameter effects on performance. A better initialization will be picked out to solve overfitting problem for PASCAL VOC dataset. Moreover, tricks such as data augmentation and momentum can be tried out to further increase its generalization. To enhance its performance on MIT Scene Parsing, we plan to study some extensions of our FCNs, e.g. FCN-CRF model. Lastly, we slightly suspect changing deep learning frameworks could have an influence on results. A performance comparison of FCN-GoogLeNet in Caffe and TensorFlow may give us some clues about it.

References

1. Bittel, S., Kaiser, V., Teichmann, M., Thoma, M.: Pixel-wise segmentation of street with neural networks. CoRR **abs/1511.00513** (2015)
2. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. International journal of computer vision **1**(4) (1988) 321–331
3. Levine, M.D., Shaheen, S.I.: A modular computer vision system for picture segmentation and interpretation. IEEE Transactions on Pattern Analysis and Machine Intelligence (5) (1981) 540–556
4. Ho, T.K.: Random decision forests. In: Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on. Volume 1., IEEE (1995) 278–282
5. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 3431–3440
6. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. arXiv preprint arXiv:1511.00561 (2015)
7. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)
8. Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected crfs. arXiv preprint arXiv:1412.7062 (2014)
9. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. arXiv preprint arXiv:1606.00915 (2016)
10. Lin, G., Milan, A., Shen, C., Reid, I.: Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation. arXiv preprint arXiv:1611.06612 (2016)
11. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. arXiv preprint arXiv:1612.01105 (2016)
12. Peng, C., Zhang, X., Yu, G., Luo, G., Sun, J.: Large kernel matters–improve semantic segmentation by global convolutional network. arXiv preprint arXiv:1703.02719 (2017)
13. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)