2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE 2021)

# Fully Convolutional Network Variations and Method on Small Dataset

Tianyou Hu*
Chongqing University
Chongqing, China
1253110957@qq.com

Yancong Deng†
University of California, San Diego
San Diego, the United States
Yad002@eng.ucsd.edu

Yuwei Deng‡
Beijing University of Posts and Telecommunications
Beijing, China
2017212765@bupt.edu.cn

Anmin Ge§
Huazhong University of Science and Technology
Wuhan, China
598053111@qq.com

*Abstract*— **Fully Convolutional Network (FCN) labels each pixel in an image with its category by up-sampling convolutional layer to the exact shape of input image. This paper presents a detailed evaluation on Fully Convolutional Network variations and method on small dataset. The paper mainly discusses three FCN models based on VGG16, containing FCN-32s, FCN-16s and FCN-8s, which are different in their up-sample multiple and process of fusing skipped layers. FCN based on ResNet and vanilla Convolutional Neural Network (CNN) are discussed as well for comparative experiment. Because of the small dataset, FCN method is quite different from the general, therefore arguments containing kernel size and up-sample method are tuned to increase accuracy for each kind of model. Arguments with highest accuracy are picked for comparative experiment among different kinds of model, which are FCN based on VGG16, ResNet and vanilla CNN. Mean Intersection over Union (mIoU) metric is computed as well to contrast segmentation performance among models and among classes. Loss, accuracy and mIoU after 300 epochs of training are compared. Optimize processes of models are recorded to evaluate converge trend. Among all models implemented in our experiment, FCN-8s stands out, reaching the accuracy of 86.79% after 300 epochs, only by training a small dataset including 367 train images and 101 test images.**

***Keywords - Fully Convolutional Network; Semantic Segmentation; CNN; VGG16; ResNet; Small Dataset.***

## I. INTRODUCTION

FCN makes predictions for each pixel while CNN makes predictions for the whole image. The process of FCN can be simply concluded in 3 steps: convolve several times with max-pooling, fuse different layers (some FCNs do not have this step) and up-sample.

By fusing different layers, coarse, higher layer's information and fine, lower layer's information are combined and integrated. FCN models based on VGG16 discussed in this paper contain FCN-32s, FCN-16s and FCN-8s. FCN-32s does not have fusion process, while FCN-16s fuses 2 skipped layers and FCN-8s fuses 3 skipped layers. FCN based on ResNet does not fuse, but it combines deep and shallow information by residual algorithm, which occurs in its CNN parts. FCN based on vanilla CNN is just like FCN-32s, because it does not fuse skipped layers either. Their only difference is their CNN structure, about which FCN based on vanilla is more simple.

The situation of training based on small dataset is quite different from normal one, therefore in order to obtain higher accuracy, arguments are tuned and performances of FCNs based on different CNNs are compared.

## II. RELATED WORK

Frank Rosenblatt invented perceptron algorithm in 1958 at the Cornell Aeronautical Laboratory. He realized perceptron by making Mark I Perceptron machine[2]. Perceptron algorithm, a linear classifier, can classify a vector of numbers to a specific class. But there are too many weights and connections.

CNN reduces weights and connections by using convolution kernel as weight variables. The first ever "convolutional network" was the Neocognitron by Fukushima[3]. The neocognitron was based on the idea of simple and complex cells. The simple cells of neocognitron basically perform a convolution and the complex cells perform average pooling. The standard reference for CNNs is from LeCun et al., "Object Recognition with Gradient Based Learning"[4]. Since that time, there have been many improvements and extensions — things like max pooling & batch normalization.

Fully Convolutional Network was then implemented to output multi-dimension array instead of 1-dimension array, which CNN outputs. For semantic segmentation, the FCN performance was greatly improved by Dr. Shelhamer, Dr. Long and Dr. Darrell published their approach[5]. The FCNs introduced in this paper are mainly based on their model.

2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE) | 978-1-7281-8319-0/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICCECE51280.2021.9342059

978-1-7281-8319-0/21/$31.00 ©2021 IEEE

## III. Architecture of Fully Convolutional Networks

### A. *Architecture of FCN based on VGG16*

As mentioned in Section I, FCN model takes mainly 3 steps: convolution with max-pooling, fusing different layers, and up-sampling.

The detailed structure of FCN based on VGG16 is described in Figure 1. Each convolve calculation will not change the height and width of image, but pool layer will. Therefore, Dr. Shelhamer, Dr. Long and Dr. Darrell use vertical lines to represent convolve layers and grid of image to represent pool layer [5]. Each vertical line means convolves once. Information of channel is not described in this figure, which means this figure only focus on height and width of layers.

The FCN structure is divided into 5 blocks and 2 convolutions, which is specifically described in Figure 1. In the first block, the input image is convolved twice and pooled once. Conv1 includes 2 convolutional layers. The process of second block and first block are same. Each one of block3 to block5 contains 3 convolutional layers and a pool layer. The detailed shape of these 5 blocks refers VGG16 model, which is one of the most important image classification model. After these 5 blocks, there are 2 convolutional layers, which are conv6 and conv7 in the figure.

About fusion, this step combines semantic information from deep, coarse layer with appearance information from shallow, fine layer. Each pool makes height and width twice smaller than before. There are 5 pools in model in total, so the output height and width of block 5 will be $2^5$ times smaller than input images of block 1. So our task is to up-sample higher layer to make it match the lower layer so that they can be added together. What's more, the output channel of each block is different, so they have to be convolved to same channel for matching. In a word, the key of fusion is to convert different layers to the same shape and make addition possible. As we can see in Figure 1, FCN-32s does not have a fusion part. FCN-16s fuse conv7 and pool4, while FCN-8s fuse conv7, pool4 and pool3. One thing we have to know is that "$\mathbf{n}\times$" means n times up-sampling in this figure.

About up-sampling, it is simply up-sample the fusion to the same height and width as input image, however, the output channel will be n-classes, since the ground truth's channel is n-classes. As you can see in the figure 1, FCN-32s up-samples conv7 32 times. FCN-16s up-samples fusion 16 times, while FCN-8s up-samples fusion 8 times.

Up-sampling does not change channel dimension, it only changes height and width. But how? Approaches to up-sample include the following steps:

- un-pooling, recover max values and fill over positions with 0;
- interpolation, interpolate values in settled ways;
- deconvolution;
- dilated convolution;

The animation of deconvolution and dilated convolution can be viewed on Github website of https://github.com/vdumoulin/conv_arithmetic to show how these 2 up-sample methods works.

Now the general structure of FCN based on VGG16 is clear. However, channel, kernel size, stride and fusion parts are still need to be determined. Channels of 5 blocks are cited from VGG16 model, which can be seen in Figure 2. The fully

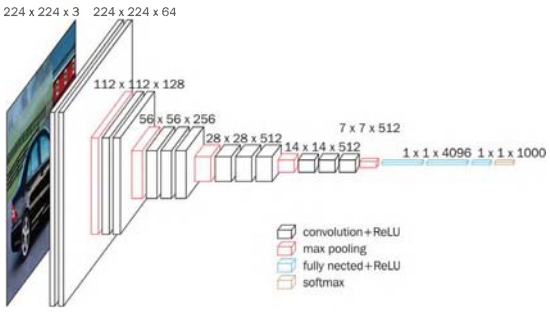



Figure 2: Structure of VGG16 [6].

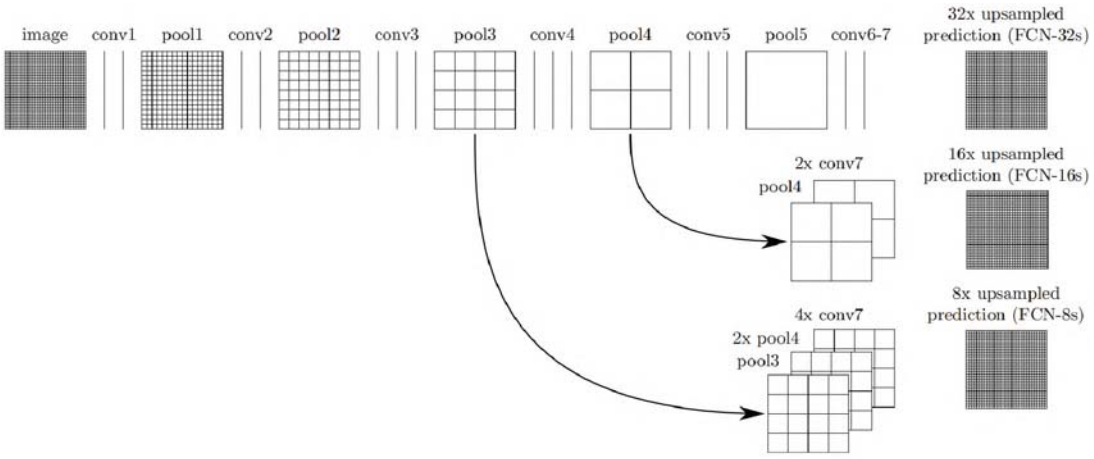



Figure 1: The architecture of FCN based on VGG16 [5].

connected layer and softmax in Figure 2 are not utilized. FCNs based on VGG16 input a batch of images with shape 224×224×3, then repeat convolution and max pooling until the shape is 7×7×512. Convolution layers are set to use kernel size of (3, 3), activation function of ReLu and "same padding". ReLu function converts any values< 0 to 0. Maxpool layers' kernel size and strides are all set to (2, 2).

Kernel size of conv6 and conv7 are set respectively (7,7) and (1,1) used deconvolution up-sample method. However, the accuracy returns to only 38.73%. Final setting of conv6, conv7 and up-sample method are obtained by tuning, which will be discussed in Section 0 later.

### B. *Architecture of FCN based on ResNet*

FCN based on ResNet replaces 5 blocks in Figure 1 with ResNet without fully connected layer. ResNet does not fuse different layers, but its residual algorithm combines shallow and deep information. Principle of residual learning is shown in Figure 3. It inputs x and output F(x), and fuses F(x) and x as input to the next layer, by which it overcome the problem that too much layers will cause lower accuracy. When x and F(x) have same shape they can be added simply, whereas x should be convolved to the same shape as F(x) for adding x and F(x) when their shape is different. ResNet50, ResNet101 and ResNet152 by Dr. Kaiming He, Dr.Xiangyu Zang, Dr. Shaoqing Ren et al.[7] are implemented for this task.

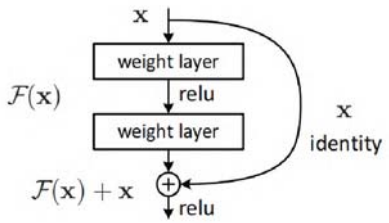



Figure 3: Principle of residual learning.

### C. *Architecture of FCN based on vanilla*

FCN based on vanilla CNN replaces 5 blocks in Figure 1 with structure in Figure 4. The vanilla CNN of this paper[8] contains 4 convolution layers, first 3 of which are followed by Maxpooling layer but the last does not. Therefore, architecture of FCN based on vanilla CNN is just structure of Figure 4 followed by conv6 and conv7 in Figure 1. FCN based on vanilla CNN does not fuse any layers.

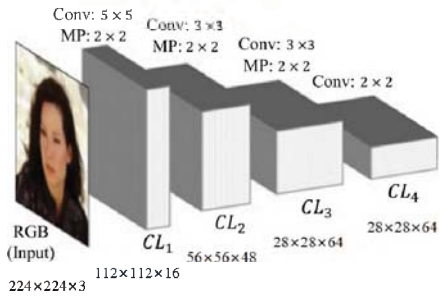



Figure 4: Vanilla CNN without fully connected layer [8].

## IV. EXPERIMENT

### A. *Build code environment*

This task was implemented based on python3.8 using Jupyter Notebook, and it utilized Tensorflow as the backend and Keras as front end.

About FCN Training, with limited resources local computer each epoch takes approximately 350 seconds in total. For each model of this paper, it takes about 100 epochs to converge to a close value, which lead to 10 hour of training time. To speed up the calculation, cloud computation on Google Colab was used. By created a folder outside of the Google Drive folder for data storage, copied contents from folder in Google Drive to the folder we created, and set run time type from None to GPU, which is utilized as hardware accelerator[9], each epoch run-time can be reduced to 3 seconds. In comparison to the local machine, it is a revolutionary progress.

After training the model, we can save the weights to h5 type file, so that we can upload our trained weights to Google Drive and then just call these weights next time.

### B. *Dataset and CNN weights used for experiment*

The dataset was downloaded from the website of https://drive.google.com/file/d/0B0d9ZiqAgFkiOHR1NTJhWVJMNEU/view, which contains 101 test images and 367 train images with their annotation images. The images are all about first view driving scenes. They were reshaped to 224×224 after called from path, since this is the image shape used in VGG16. Pre-trained VGG16 weight is also used so that it can be utilized for the FCN-VGG16 training. On the other hand, ResNet could be implemented by functions in "Keras.applications", containing weights and structure.

### C. *Train*

First, trained images and annotations were shuffled. Then each of trained images and annotations was divided to 85% training part and 15% validation part. Thus 311 samples were trained and 56 samples were validated.

For training, all models should be given an optimizer, which is SGD in this task, then compile and fit. Fit means feed the model with data. The parameter setting is the same for all models, which is shown in TABLE I. .

TABLE I. SETTING OF ARGUMENTS FOR TRAINING

| function | argument | setting |
|---|---|---|
| SGD | learning rate | 0.01 |
| | decay | $5^{-4}$ |
| | momentum | 0.9 |
| compile | loss | categorical crossentropy |
| | optimizer | SGD |
| | metric | accuracy |
| fit | batch size | 32 |
| | epochs | 300 |
| | verbose | 2 |

TABLE II. TUNING ARGUMENTS OF FCN-8S. THE FAR LEFT COLUMN OF COMBINATION PERFORMS BEST.

| | FCN-8s | | | | | |
|---|---|---|---|---|---|---|
| kernel of conv6 | (3,3) | (3,3) | (3,3) | (7,7) | (7,7) | (7,7) |
| kernel of conv7 | (1,1) | (1,1) | (3,3) | (1,1) | (1,1) | (1,1) |
| upsample method | interpolation | interpolation | interpolation | interpolation | deconvolution | deconvolution |
| parameters in total | 33,735,284 | 33,735,284 | 34,118,500 | 117,611,364 | 135,058,124 | 135,058,124 |
| if load VGG weights | no | yes | no | no | no | yes |
| epochs | 300 | 300 | 300 | 300 | 300 | 300 |
| validation loss | 0.4518 | 12.5761 | 0.5400 | 0.5901 | 1.8364 | 1.7259 |
| validation accuracy | 0.8679 | 0.2198 | 0.8530 | 0.8511 | 0.3424 | 0.3873 |

### D. *Tune arguments of FCNs*

For FCN based on VGG16, containing FCN-8s, FCN-16s, FCN-32s, the cited paper[5] use deconvolution method, which returns to low accuracy. However, bilinear interpolation shows better performance for small dataset. About kernel size of conv6 and conv7, different kernels containing (7, 7), (1, 1), (3,3) were used for experiment. Kernel size and output channel in these 2 layers will have a significant influence on the amount of parameters. Several different training of FCN-8s were experimented, which can be seen in TABLE II. , suggesting that smaller kernel size reduce the parameters, leading to lower loss and higher accuracy. Besides deconvolution method performance bad, therefor it is not suitable for small dataset training. In addition, loading VGG weights increases accuracy for deconvolution type model. However, for interpolation ones, interpolation with loading VGG weights will cause validation stuck at first epoch. The third column from left to right gives an example. Actually for all combination of conv6 and conv7's kernel in this table, the validation loss and accuracy will stuck at first epoch if interpolating and loading VGG weight. By comparing the performance of different parameters, the best combination is settled for FCN-8s. That is kernel (3,3) for conv6, kernel (1,1) for conv7, interpolation and not loading VGG weights.

Kernel of FCN-16s were also tuned, which can be seen in TABLE III. . However, (3, 3) for conv7 is a little bit better than (1, 1). Kernel size of (3,3) is the most common one nowadays, therefore kernel shape of FCN-16s and FCN-32s are all set (3,3), while FCN-8s changes kernel shape of conv7 to (1,1) in order to reach higher accuracy. Our choice of these parameters may not be the best, however, it does increase the model's accuracy a lot. The accuracy of FCN-8s is increased from 34.24% to 86.79% by tuning combination of different arguments.

For FCN based on ResNet, ResNet50, ResNet101 and ResNet152 are implemented. Kernel size of conv6 and conv7 are all set (3,3). However, for ResNet101, we find that if conv6 is removed and kernel size of conv7 is modified to (1,1), accuracy will increase from 67.43% to 74.21%. Therefor 3 experiments, each of is trained 300 epochs, are implemented for all 3 ResNet models. One of experiments has conv6 and conv7 with kernel size of (3,3), one removes conv6, and the rest one removes conv6 and converts kernel of conv7 from (3,3) to (1,1), which is shown in TABLE IV. , comparing segmentation accuracy of different situations. The results shows that removing conv6, remaining 3×3 kernel of conv7 decreases accuracy a lot, while other 2 situation reach much higher accuracy. For ResNet50 and ResNet152, combination of 3×3 conv6 and 3×3 conv7 perform best, whereas removing conv6 and changing kernel of conv7 to 1×1 performs best for ResNet101.

TABLE III. TUNING KERNEL SIZE OF CONV7 FOR FCN-16S.

| | FCN-16s | |
|---|---|---|
| kernel of conv6 | (3,3) | (3,3) |
| kernel of conv7 | (3,3) | (1,1) |
| up-sample method | interpolation | interpolation |
| epochs | 300 | 300 |
| loss | 0.5461 | 0.5376 |
| accuracy | 0.8479 | 0.8452 |

TABLE IV. TUNING FCNS BASED ON RESNET.

| | ResNet50 | ResNet101 | ResNet152 |
|---|---|---|---|
| 3×3 conv6<br>3×3 conv7 | **47.70%** | 67.43% | **77.32%** |
| conv6 removed<br>3×3 conv7 | 35.94% | 53.78% | 40.37% |
| conv6 removed<br>1×1 conv7 | 46.97% | **74.02%** | 76.99% |

For FCN based on vanilla CNN, which has the simplest structure, combination of 3×3 conv6 and 3×3 conv7 is picked for experiment, because this combination is better than others in TABLE IV. in most situations. As a result, FCN based on vanilla CNN reaches accuracy of 81.76%.

FCNs based on ResNet50, ResNet101, ResNet152 and vanilla CNN with highest accuracy are picked for comparison with FCN-32s, FCN-16s and FCN-8s, which means all FCNs have been tuned for comparison in Section IV.E.

### *E. Comparative result*

This section will compare performance of tuned FCNs based on VGG16, ResNet and vanilla CNN. FCNs based on VGG16 include FCN-32s, FCN-16s and FCN-8s. FCNs based on ResNet include ResNet50, 101 and 152. Their values on last epoch will be compared firstly, then converge trend will be shown in figure.

Comparative metric on last epoch consists of loss, accuracy and mean Intersection over Union. The way to compute loss is by cross entropy algorithm. Accuracy is calculated on 56 validation images, which is mentioned in Section IV.C. Mean Intersection over Union (mIoU) metric is also referred to as the Jaccard index, which is essentially a method to quantify the percent overlap between the target mask and our prediction output. There are 2 methods to get IoU. The first formula is down.

$$IoU = \frac{\text{target} \bigcap prediction}{\text{target} \bigcup prediction}$$

Target is just ground truth and prediction is what we predicted from model. The **intersection** (target ∩ prediction) is comprised of the pixels found in both the prediction mask and the ground truth mask, whereas the **union** (target ∪ prediction) is simply comprised of all pixels found in either the prediction or target mask. IoU can be also calculated by equation below.

$$IoU = \frac{TP}{TP + FP + FN}$$

TP means intersection of target's true index and prediction's true index. FP means intersection of target's False index and prediction's True index. FN represents intersection of target's True index and predictions False index. TP is True Positive. FP is False Positive and FN is False Negative. Besides TN means True Negative. These 4 abbreviation are used in confusion matrix, which compare actual class with predicted class. The confusion matrix can be seen in Figure 5 to better understand these abbreviations. This paper makes use of the second formula to compute Intersection over Union for each class, obtaining detailed information of TP, FP and FN. The number of pixels that each of them includes are recorded, thereby predictions and ground truth are better compared.

The comparative result of FCNs are shown in TABLE V. , contrasting their loss, accuracy and mIoU metrics. Among models, FCN-8s stand out with highest values of 3 metrics. Vanilla CNN with simplest structure shows better performance than all ResNet based FCNs and FCN-32s, suggesting that too complicated structure is not suitable for small dataset training.

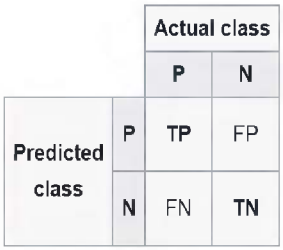



Figure 5. Confusion Matrix

TABLE V. FINAL COMPARISON WHEN EPOCH IS 300.

| FCN models | loss | accuracy | mIoU |
|---|---|---|---|
| based on ResNet50 | 8.4256 | 47.70% | 5.9% |
| based on ResNet101 | 0.8977 | 74.02% | 24.9% |
| based on ResNet152 | 0.7802 | 77.32% | 27.5% |
| based on vanilla CNN | 0.6086 | 81.76% | 32.4% |
| FCN-32s | 0.8689 | 78.68% | 26.5% |
| FCN-16s | 0.5461 | 84.79% | 37.1% |
| **FCN-8s** | **0.4518** | **86.79%** | **39.1%** |

One important point is that the channel of ground truth and original image is 3. They value from 0 to 11. But the channel of prediction by Keras is 12, which means prediction have 12 layers with value 0 or 1. To compute their IoU, prediction should be converted from 12 channels type to 3 channels type, otherwise ground truth should be converted to 12 channels type.

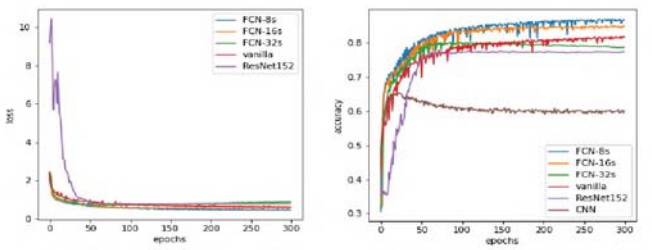
Another important thing to emphasize is that the IoU score is calculated for each class separately and then averaged over all classes to provide a global, mean IoU score of our semantic segmentation prediction. The weight of each class is the same. Which means if one class only have little percent of pixel but its IoU is really low compared to other classes, it will have a big influence on mIoU. As a result, the mIoU may be low because of its effect. This dataset has 12 classes totally. By using this metric, FCNs' results vary, which are summarized in 0.

Comparing the IoU among classes, ground category obtains highest accuracy. Basically by feeding this dataset, models can only recognize big objects like ground and building, but not small objects like pedestrian. Comparing IoU among models, FCN-8s is extraordinary in most of categories, showing much better performance than other models. FCN-16s recognizes sky best and FCN based on vanilla recognizes vegetation best. FCN-32s performs best on pole and signboard but its accuracy is useless at all for recognizing objects of these 2 categories. FCN based on ResNet is not reliable, because all of them cannot recognize car, which is significant for self-driving car. Besides their mIoU is relatively low compared to other models.

TABLE VI. DETAILED INTERSECTION OVER UNION OF ALL CLASSES OF ALL MODELS.

| category | FCN-8s | FCN-16s | FCN-32s | based on vanilla | based on ResNet50 | based on ResNet101 | based on ResNet152 |
|---|---|---|---|---|---|---|---|
| pole | 0.0% | 0.0% | **0.1%** | 0.0% | 0.0% | 0.0% | 0.0% |
| signboard | 0.0% | 2.2% | **3.0%** | 2.6% | 0.0% | 0.0% | 0.0% |
| people riding bicycles | **5.4%** | 2.3% | 0.5% | 0.0% | 0.0% | 0.0% | 0.0% |
| others | **5.7%** | 5.8% | 5.1% | 0.5% | 0.0% | 0.0% | 0.0% |
| pedestrians | **7.5%** | 2.1% | 0.1% | 0.0% | 0.0% | 0.0% | 0.0% |
| fence | **31.7%** | 30.9% | 2.7% | 9.9% | 0.0% | 0.0% | 18.0% |
| sky | 45.3% | **50.1%** | 47.4% | 37.6% | 21.8% | 36.8% | 39.2% |
| car | **51.5%** | 37.7% | 22.4% | 33.2% | 0.0% | 0.0% | 0.0% |
| building | **73.5%** | 72.7% | 51.5% | 63.3% | 0.0% | 59.9% | 64.5% |
| side walk | **73.8%** | 69.8% | 64.3% | 67.4% | 0.0% | 49.1% | 49.9% |
| vegetation | 83.3% | 81.5% | 32.9% | **84.7%** | 0.0% | 68.9% | 74.2% |
| ground | **91.9%** | 90.1% | 87.9% | 90.1% | 49.3% | 84.1% | 84.3% |
| mIoU | 39.1% | 37.1% | 26.5% | 32.4% | 5.9% | 24.9% | 27.5% |

Converge trend of models will be shown in Figure 6, comparing their loss and accuracy of each epoch. To make the graph clearer, FCN based on ResNet50 and ResNet101 are not included, because they return to lowest accuracy and mIoU. The Loss trend is opposite to accuracy.




(a) Loss variation trend. (b) Accuracy variation trend.

Figure 6. Comparative variation of FCN models. (a) shows loss and (b) shows accuracy.

Figure 6 (a) shows that FCN based on ResNet 152 has an extremely high loss at the beginning, corresponding to its relatively lower accuracy in Figure 6 (b). Because loss reflects deviation of predictions and ground truth, loss converge trend is just opposite from accuracy converge trend.

In Figure 6 (b), a CNN model training CIFAR-10 is contained as well. The CNN has 3 convolutional layers, each of which is followed by a maxpooling layer. The output channels of 3 convolutional layers are 16, 32 and 64.

There's a fully connected layer at the end. It can be seen that FCN-32s and CNN has overfitting problem while others do not. FCN-8s and FCN-16s are not overfitting while FCN-32s do, suggesting that FCN based on VGG16 should fuse skipped layers to overcome overfitting problem. To our surprise, FCN based on vanilla, which has simplest structure, performs better than FCN-32s and all FCNs based on ResNet. It is not overfitting as well. FCN based on ResNet is not suitable for small dataset training.

### *F. Visualization*

Visualize function was writing to see the segmentation effect of models. First test image of dataset was taken as an example to compare ground truth and prediction, which is shown in Figure 7 in next page.

FCNs based on ResNet are unable to recognizing cars, which is purple in Figure 7. The boundary lines of sky, building, and road are too curving in segmentation of ResNet based models. FCN based on ResNet50 only classifies pixel to sky and ground, performing worst among models.

FCN based on vanilla CNN has better performance than FCNs based on ResNet, but it recognizes too many cars.

FCN based on VGG16 perform best. FCN-8s predicted boundary of trees better than FCN-16s, whereas it did a bad job on blue class segmentation, which is fence. About cars, all of them cannot get exact shape, but they can recognize approximate size. FCN-32s predicted the left car much bigger than the actual size, and the other two models do not have this problem. FCN-8s predicted right car's shape best. All of the models cannot recognize small objects like pedestrians and people riding bicycles. They can only recognize big objects and boundaries of these objects cannot be predicted precisely. However, the best of 3 models, FCN-8s, can relatively predict road and cars precisely. With assistance of lane detection algorithm base on Hough Transform, the self-driving cars can

go straight along its lane and not collide other cars by implementing FCN-8s.

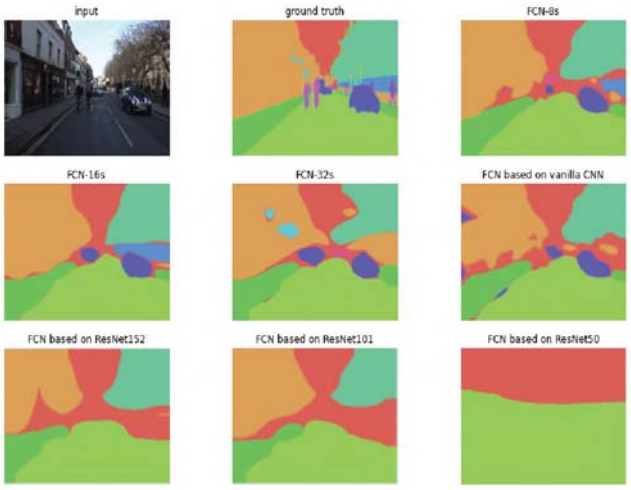



Figure 7. Visualization of all FCNs.

## V. CONCLUSION

This paper discusses FCN variation and method on small dataset. The cited paper[5] advocating using deconvolution as up-sample method, however, that does not work well on small dataset. Bilinear interpolation method shows much higher accuracy on small dataset, which is shown is TABLE II. . Besides, loading VGG16 weights decreases accuracy when implementing bilinear interpolation, meaning initializing weights performs better. FCN-8s' accuracy increases from 34.24% to 86.79% by tuning, which is the most dramatic changes. FCNs based on ResNet are tuned as well, which is summarized in Table 3.

We pick tuned models for comparative result in Section IV.E, which contrasts loss, accuracy and mIoU at last epoch firstly, then compare IoU of each classes of models to see more details. At last this section compare converge trend of models. Loss is obtained by cross entropy algorithm. mIoU represents intersection divided by union, which means overlapped pixels divided by pixels which are occupied by both ground truth and predictions. FCN-8s shows highest accuracy among all models, reaching accuracy of 86.79% and mIoU of 39.1%, shown in Table 4. Table 5 shows IoU of each classes, suggesting that small dataset training can only make computer recognize big objects like sky, road, sidewalk and cars. Small objects like pedestrian, signs and fence can hardly be recognized. The converge trend are shown in Figure 6, showing that FCN based on ResNet has high loss at the beginning. FCN-32s has overfitting problem, which is the same as CNN. As FCN based on VGG16 as well, FCN-16s and FCN-8s overcome the problem by fusing skipped layers.

Visualization of each model is summarized in Figure 7. Though FCN based on vanilla CNN has higher metric value, it recognizes too many cars. FCNs based on VGG16 perform best in visualization.

In conclusion, tuned FCN-8s invented in "Fully Convolutional Networks for Semantic Segmentation" performs best in our experiments, but it should utilize bilinear interpolation instead of deconvolution for up-sampling, which is more suitable for training based on small dataset. Tuned FCN-8s reaches accuracy of 86.79% and mIoU of 39.1% only by training 311 images, which can recognize cars and relatively detect shape of sky, road and buildings precisely.

## REFERENCES

[1] Ibrahem Kandel, Mauro Castelli, The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset, ICT Express, 2020, ISSN 2405-9595.

[2] F. Rosenblatt, "Perceptron Simulation Experiments," in Proceedings of the IRE, vol. 48, no. 3, pp. 301-309, March 1960, doi: 10.1109/JRPROC.1960.287598.

[3] K. Fukushima, S. Miyake and T. Ito, "Neocognitron: A neural network model for a mechanism of visual pattern recognition," in IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-13, no. 5, pp. 826-834, Sept.-Oct. 1983, doi: 10.1109/TSMC.1983.6313076.

[4] Lecun Y , Haffner P , Léon Bottou, et al. Object Recognition with Gradient-Based Learning[C]// Shape, Contour and Grouping in Computer Vision. 1999.

[5] E. Shelhamer, J. Long and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 4, pp.640-651,1April2017,doi:10.1109/TPAMI.2016.2572683.

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

[7] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[8] Y. Wu, T. Hassner, K. Kim, G. Medioni and P. Natarajan, "Facial Landmark Detection with Tweaked Convolutional Neural Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 12, pp. 3067-3074, 1 Dec. 2018, doi: 10.1109/TPAMI.2017.2787130.

[9] Google Colab is very slow compared to my PC. (2019). Retrieved September 16, 2020, from stack overflow: https://stackoverflow.com/questions/49360888/google-colab-is-very-slow-compared-to-my-pc