Semantic Segmentation on Cityscapes Dataset Using SegNet, U-Net, and FCN: A Study Approach

Shreyanshu Bhushan Lee Dong-gyu
Graduate School of Artificial Intelligence, Kyungpook National University
{bhushan,dglee}@knu.ac.kr

Abstract

Deep learning has been extremely effective when working with images as information and is presently at a phase where it works better compared to humans in numerous utilization cases. The main issues that people have been keen on addressing with computer vision are image classification, object segmentation and detection. There are several pre-existing models that already achieves excellent performance on cityscapes dataset. Due to this it becomes very difficult for someone who wants to work on semantic segmentation, to choose the right model with correct parameters. In this paper, we present a full detailed report on semantic segmentation on Cityscapes dataset. For this study, we use three different pre-existing models: SegNet, U-Net, and Fully Convolutional Networks (FCN) with VGG-16 as the encoder. Afterwards, we perform several experiments on it and compare them each other. In addition, we also use some traditional and modern techniques in order to increase the performance accuracy of the models. Extensive experiments and analysis shows that the best pixel accuracy is achieved by SegNet model with some certain hyper-parameters, having an score of ***84.73%****. Code and outputs are available at* https://github.com/shreyanshu09/Semantic-Segmentation-Cityscapes.

1. Introduction

In advanced image processing and computer vision, image segmentation is the method involved with dividing a computerized image into various sections. The objective of the division is to simplify and additionally change the representation of a picture into something more significant and simpler to examine. Image segmentation is regularly used to locate objects and boundaries in pictures. It is the most common way of allocating a label to each pixel in a picture to such an extent that pixels with a similar name share specific qualities. The result of image segmentation is a bunch of sections that aggregately cover the whole image or a bunch of shapes extracted from the picture. Every one of the pixels in a region is comparable regarding some characteristic or registered property, like texture, color, or intensity. Adjacent regions are significantly different colors with respect to the equivalent characteristic. When applied to a stack of pictures, normally in medical imaging, the resulting contours after image segmentation can be utilized to make 3D recreations.

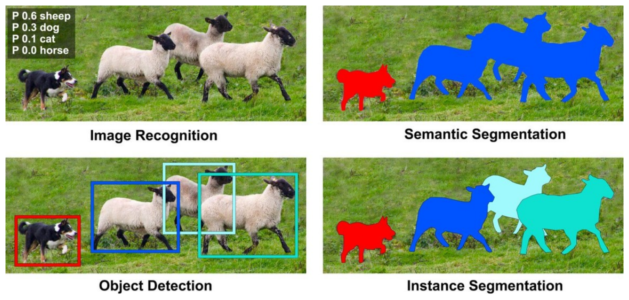

Figure 1. Difference between semantic segmentation and instance segmentation. Image taken from google image.

One of the pre-dominant methodologies for this innovation is deep learning and it falls under the classification of Artificial Knowledge where it can act or adopt the thought process of a human. Regularly, the actual framework will be set with hundreds or perhaps large numbers of information to make the training session to be more productive and quick. The major task of image segmentation is to group every pixel in a picture belonging to a specific class and subsequently can be considered as a classification problem for each pixel. Generally, there are two main types of image segmentation: semantic segmentation and instance segmentation. Fig. 1 shows the difference between image recognition, semantic segmentation, object detection, and instance segmentation. Semantic segmentation is the most common way of classifying every pixel belonging to a particular label. It doesn't distinctive across various examples of a similar item. For example, in case there are 2 cats in a picture, semantic division gives the same label to all the pixels of both cats. Instance segmentation is different from semantic segmentation, as it gives a novel name to each occurrence of a specific item in the picture. As can be found in the picture of 3 dogs and each dog is assigned different colors i.e different labels. With semantic segmentation, every one of them would have been assigned a similar color. There are many applications of image segmentation such as handwriting recognition to extract the text from the handwritten documents, google portrait mode to separate the foreground from background in an image, YouTube stories for different backgrounds while creating stories, self-driving cars, and many more.

In this paper, we use three different pre-existing models such as SegNet, U-Net, and FCN with VGG-16 as the encoder for image segmentation task on cityscapes dataset. Furthermore, we use several techniques in order to improve the accuracy of the pre-existing model such as traditional and modern data augmentation, different activation function, different optimizers, and learning rate. The main contributions of this paper are as follows:

- We perform extensive experiments with several pre-existing models for semantic segmentation on cityscapes dataset and present a full detailed comparison between them.
- We also use different techniques on each model for further enhancement of the performance.

2. Related Work

There are many works already done on semantic segmentation on cityscapes dataset. In this section, we will describe a brief introduction about all the pre-existing models used in this study.

2.1. SegNet

SegNet [1] is intended to be productive architecture for pixel-wise semantic division. It is motivated by street scene understanding applications that require the capacity to show appearance (street, building), shape (cars, pedestrians), and comprehend the spatial relationship (context) between various classes like street and sidewalk. In average street scenes, most of the pixels have a place with huge classes like street, building and thus the network should create smooth division.

Fig. 2 depicts the overall architecture of SegNet model. It has an encoder network and a relating decoder network, followed by a last pixel-wise classification layer. The encoder network comprises of 13 convolutional layers which correspond to the initial 13 convolutional layers in the VGG16 network intended for object classification. They dispose of the fully connected layers for holding higher resolution feature maps at the deepest encoder output. This additionally lessens the quantity of boundaries in the SegNet encoder network altogether (from 134M to 14.7M) when contrasted with other recent structures. Each encoder layer has a comparing decoder layer and subsequently, the decoder network has 13 layers. The last decoder output is taken care of to a multi-class soft-max classifier to create class probabilities for every pixel freely.

2.2. Fully Convolutional Network

The overall architecture of a CNN comprises a few convolutional and pooling layers followed by a few fully connected layers toward the end. The paper of Fully Convolutional Network [3] contends that the last completely associated layer can be considered as doing a 1x1 convolution that covers the whole region. Hence the last dense layers can be supplanted by a convolution layer accomplishing a similar outcome. In any case, presently the upside of doing this is the size of input need not be fixed any longer. While including dense layers, the size of info is obliged, and henceforth when an alternate measured information is given it must be resized. But supplanting a dense layer with convolution, this requirement doesn't exist. Additionally, when a greater size of the picture is given as information the result delivered will be a feature map and in addition to a class yield like for typical info estimated picture. Likewise, the noticed conduct of the last element map addresses the heatmap of the necessary class i.e the situation of the article is featured in the feature map. Since the result of the component, the map is a heatmap of the necessary article it is legitimate data for our utilization instance of division.

Since the feature map got at the result layer is a down examined because of the arrangement of convolutions performed, we would need to up-example it utilizing an additional method. Bilinear up sampling works however the paper proposes utilizing learned up testing with deconvolution which can even become familiar with a non-linear upsampling. The down part of the network is called an encoder and the upsampling part is known as a decoder. This is an example we will see in numerous structures i.e decreasing the size with encoder and afterward up inspecting with the decoder. In an ideal world, we would not want to downsample using pooling and keep a similar size all through however that would prompt a tremendous measure of boundaries and would be computationally infeasible. In our study, we use VGG-16 trained on ImageNet classification [2] as the encoder.

2.3. U-Net

U-Net [4] expands on top of the fully convolutional network. It was built for clinical purposes to track down growths in the lungs or the cerebrum. It likewise comprises an encoder that downsamples the input picture to a feature

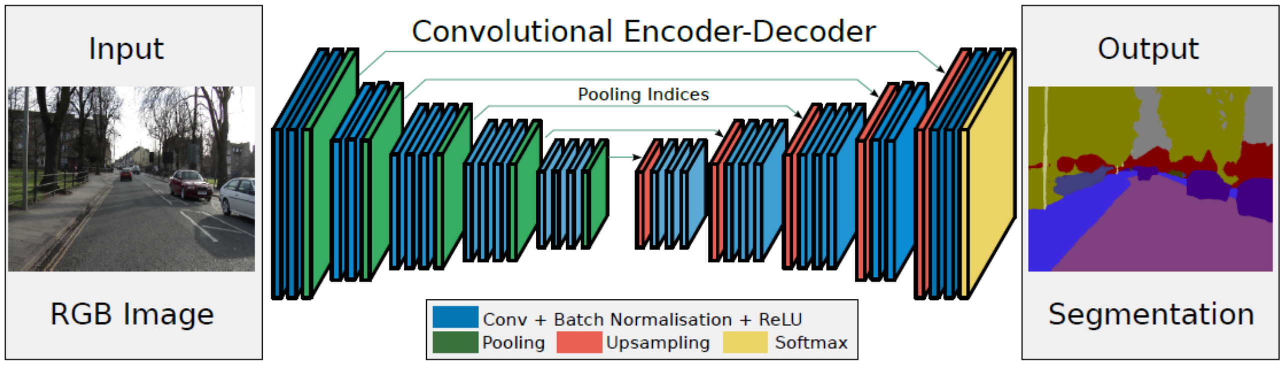

Figure 2. Architecture of SegNet model.

map and the decoder which upsamples the element guide to include picture size utilizing learned deconvolution layers.The fundamental contribution of the U-Net design is the easy route connections. We saw above in FCN that since we downsample a picture as a component of the encoder, we lost a ton of data that can't be effectively recovered in the encoder part. FCN attempts to address this by taking data from pooling layers before the last component layer.

U-Net proposes another way to deal with taking care of this data deficit issue. It proposes to send data to each upsampling layer in the decoder starting from the corresponding downsampling layer in the encoder. In this way, we can get better data while keeping the calculation low. Since the layers toward the start of the encoder would have more data they would reinforce the upsampling activity of the decoder by giving fine details relating to the input pictures accordingly, further developing the outcomes a lot. The paper additionally proposed utilization of a clever misfortune work which we will examine underneath.

3. Method

In this section, we have describe about all those techniques used along with the pre-existing models in order to increase their accuracy.

3.1. Data Augmentation

It is a technique use to increase the amount of data by by adding somewhat altered duplicates of previously existing data. There are various methods to do that but in our work we have focus on few such methods such as random crop, horizontal flips, normalization, and color jittering. Data Augmentation isn't simply used to prevent overfitting. As a general rule, having an enormous dataset is crucial for the performance of both machine learning (ML) and Deep Learning (DL) models. We can work on improving the performance of the model by enlarging the data. It implies that Data Augmentation is additionally useful for upgrading the model's performance and plays an important role for better accuracy of the models. In this study, we use all the mentioned data augmentation techniques in order to further boost the performances of the model.

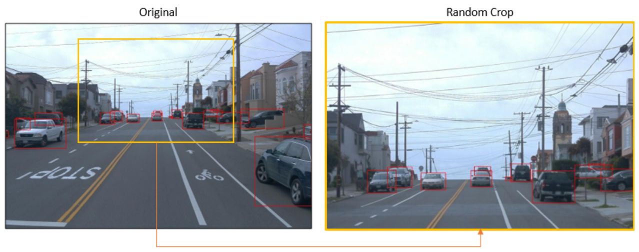

Figure 3. An example of random crop technique. Image taken from google images.

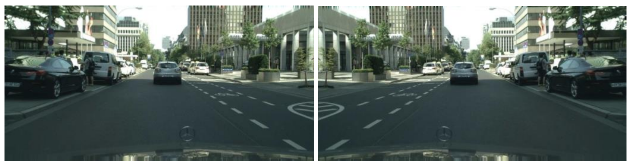

Figure 4. An example of horizontal flip technique. Image taken from cityscapes dataset.

3.1.1 Random Crop

Random crop is a data augmentation strategy wherein we make a random subset of an original picture. This assists our model to generalize better because the objects of interest we need for our models to learn are not always completely noticeable in the picture or similar scale in our training data. Fig. 3 shows an example of random crop technique.

For example, when we are making a deep learning model that recognizes vehicles in a street. We might have a camera streaming pictures to an item discovery model revealing

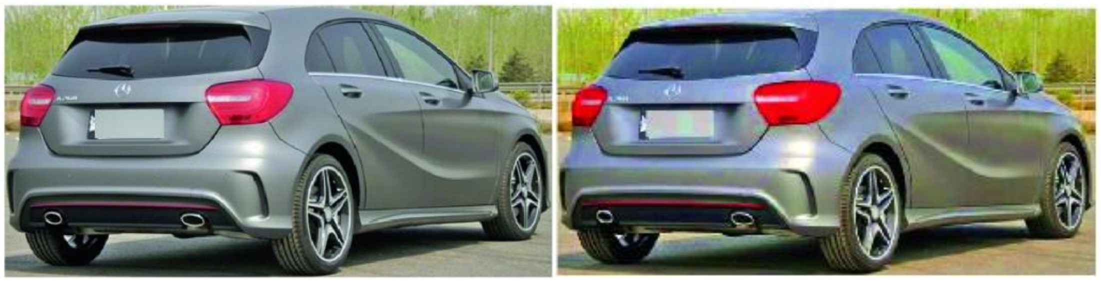

Figure 5. An example of color jittering technique with randomly change of saturation, hue, and brightness level of image. Image taken from google images.

when a vehicle is apparent. The vehicles are not always completely in the frame, nor are they continually a similar distance from the camera. A random crop is an extraordinary decision as an expansion procedure for this situation. Because of this, we use random crop with padding of four to further improve the performances of each model.

3.1.2 Horizontal Flips

Flipping implies rotating a picture on a horizontal or vertical axis. In horizontal flip, the flipping will be on the vertical axis, and in Vertical flip, the flipping will be on the horizontal axis. Turning around the whole lines and sections of picture pixels in a horizontal direction is called horizontal flip augmentation. We further use this technique to add more varieties in the dataset. Fig. 4 depicts an example of horizontal flip technique performed on the cityscapes dataset.

3.1.3 Normalization

Normalization is a data pre-handling method used to carry the mathematical data to a typical scale without distorting its shape. Generally, when we input the information to a machine or deep learning calculation, we will quite often change the values to a reasonable scale. The reason we normalize is somewhat to guarantee that our model can sum up appropriately. It makes the Optimization quicker in light of the fact that normalization doesn't permit the weights to explode everywhere and confines them to a specific range.

3.1.4 Color Jittering

The color jitter is one of the image data augmentation strategies, and it means to randomly change the lightness, hue, and saturation of the picture. HSL (Hue, Saturation, Lightness) is another representation of the RGB model. It has a curved geometric color structure that is the beginning of a cylinder, and the lightness and saturation are straight lines and the chromaticity has a round shape.

Hue (color) represents the color and is liable for the point of HSL based on the cylinder. 0 degrees is red, 120 degrees is green, and 240 degrees is blue. It has a range from 0 to 360 (degrees). Saturation is more distinctive color is called to be more saturated, and color more likely an achromatic shading, for example, gray or white or black is supposed to be less saturated. When referring to a shading with high saturation, the expression 'dark' is regularly utilized. Alternately, when referring to a shading with low saturation, the expression 'blurred' is regularly utilized. It has a range of 0 and 100%. Lightness literally means the brightness of a color. It has a range between 0 and 100%. Fig. 5 shows an example of color jittering technique applied on an image. Left side image is the original image and right side image is the outcome after applying different hue, saturation, and lightness value. Likewise, We also randomly apply different values of hue, saturation, and lightness to different images for increasing the dataset which gradually increases the performance of the model.

3.2. Activation Function

An activation function is a function that is added into an artificial neural network to assist the network with learning complex patterns in the data. When compared with a neuron-based model that is in the human brain, the activation function is toward the end concluding what is to be fired to the following neuron. Pre-existing architectures uses a certain type of activation function such as SegNet uses “Relu” activation function and U-Net uses “ELU” activation function. We also try to replace those activation functions with different one to see the effect on the performance of the model.

3.3. Optimizers

Optimizers integrate with the loss function and model parameters by refreshing the model in response of the result of the loss function. In less difficult terms, optimizers shape and mold the model into its most precise possible structure by futzing with the loads. The loss function is the manual

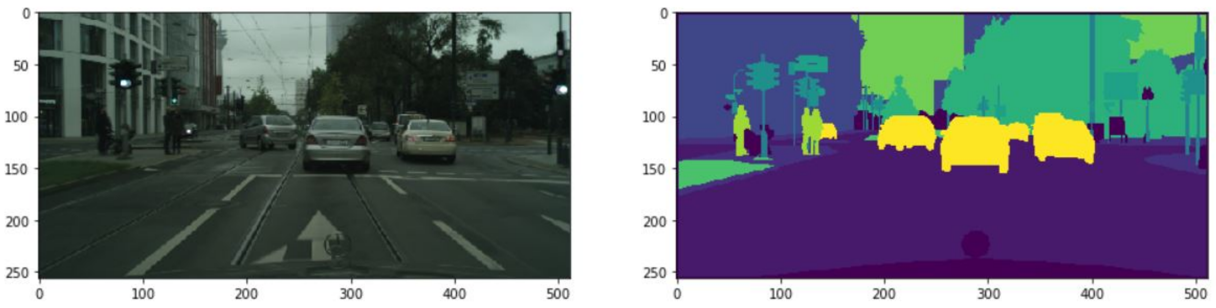

Figure 6. An example from the cityscapes dataset with it's true label

for the terrain, telling the optimizer when it's moving in the right direction or wrong direction. It is used to maximize the efficiency or to minimize the loss. Different architecture uses different optimizers such as some uses SGD optimizer and some uses Adam optimizer. To see the effect on the performance, we also change the optimizer and report the results.

3.4. Learning Rate

The learning rate is a configurable hyper-parameter utilized in the preparation of neural networks that has a small value in positive, somewhere in the range of 0.0 and 1.0. The learning rate controls how rapidly the model is adjusted to the problem. Small learning rates require additional training epochs given the more small changes made to the loads each update, while bigger learning rates bring about quick changes and require less preparing ages.

A learning rate that is too enormous can make the model converge excessively fast to a sub-optimal solution, though a learning rate that is too little can make the process get stuck. The challenge of training deep learning neural networks includes cautiously choosing the learning rate. It could be the most significant hyper-parameter for the model. So, we also try with different learning rates on each model to see the performance.

4. Experiments

To validate the performance of different models before and after applying any modification. We conduct extensive experiments on cityscapes dataset. In this section, we first specify the experimental setup details which includes environment, dataset, models, hyper-parameters, and metrics used in this study, followed by the experimental results which shows the quantitative and qualitative analysis of each model and a full detailed comparison.

4.1. Experimental Setup

We perform our experiments on a single GPU (NVIDIA TITAN RTX) having a memory of 24GB with CUDA version 10.2. Our implementation is totally based on tensorflow with Python 3.6 as the programming language. Few programs, we run on colab while others on PyCharm environment.

4.1.1 Dataset

We use the cityscapes dataset which comprises of diverse metropolitan road scenes across 50 distinct cities at different years as well as ground truths for multiple vision tasks including semantic division, instance-level segmentation, and stereo pair disparity inference. Fig. 6 depicts an example from cityscapes dataset with it's true label. For segmentation tasks, Cityscapes gives dense pixel-level annotations for 5000 images at 1024 * 2048 resolution pre-split into training (2975), validation (500) and test (1525) sets. Label annotations for segmentation tasks range across 30 plus classes such as person, car, truck, road, etc, generally experienced during driving scene discernment.

4.1.2 Models

As discussed in the related work section, we use three different pre-existing models such as SegNet, U-Net, and FCN with VGG-16 as the encoder block. We use these models with different hyper-parameters and see the performances. Also, we combine these models with some regularization techniques.

4.1.3 Hyper-parameters

In this study, we use different parameters for different models to study the outcome from each of them. For SegNet and U-Net, we run our programs on both ReLu and ELU activation function, Adam and SGD optimizer, categorical cross-entropy loss, with and without data augmentation techniques, and different learning rates (10^{-4}, 10^{-2}, 10^{-5}). For FCN, we use the learning rate schedule, if steps is less than 10,000 then learning rate is 10^{-4}, if steps is between 10,000 and 20,000 then learning rate is 10^{-5}, if learning

Model	Activation Function(%)	Optimizer(%)	Learning Rate(%)	Data Augmentation(%)	Accuracy(%)
SegNet	ReLu	Adam	10^{-4}	No	81.74
SegNet	ELU	Adam	10^{-4}	Yes	**84.73**
SegNet	ELU	Adam	10^{-2}	Yes	79.08
SegNet	ELU	SGD	10^{-4}	Yes	65.31
SegNet	ELU	Adam	10^{-5}	Yes	79.64
U-Net	ReLu	Adam	10^{-4}	No	81.42
U-Net	ELU	Adam	10^{-4}	Yes	82.65
U-Net	ELU	Adam	10^{-2}	Yes	33.06
U-Net	ELU	SGD	10^{-4}	Yes	56.32
U-Net	ELU	Adam	10^{-5}	Yes	75.71
FCN-8s	-	-	Lr schedule	Yes	83.92

Table 1. Pixel accuracy of each model with different hyper-parameters on cityscapes dataset. Lr schedule means Learning Rate Schedule which is discussed in the hyper-parameter section. Dash(-) means they use multiple functions and optimizers. Bold number indicates the best score.

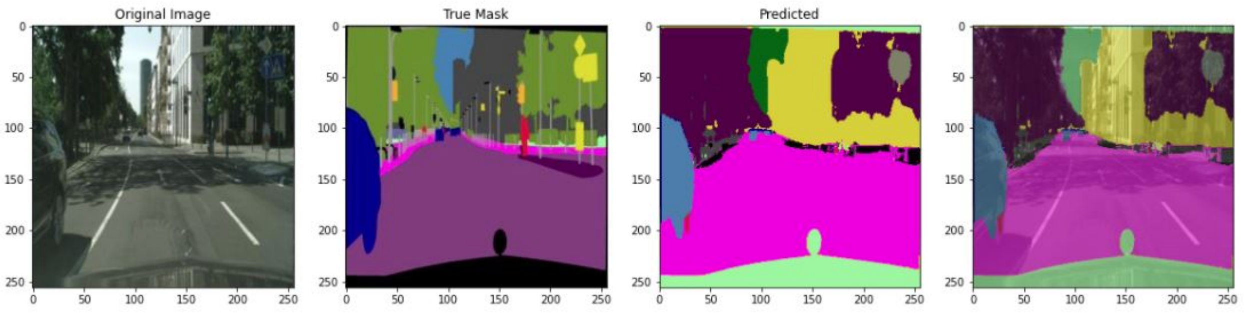

Figure 7. Output from the SegNet Model

rate is between 20,000 and 40,000 then learning rate is 3 * 10^{-6}, and for every other value learning rate is 10^{-6}. Also for FCN, we use data augmentation techniques and all other parameters are taken from the paper [2]. We run SegNet and U-Net model for 10 epochs and FCN for 6 epochs. For more clarification, we have added the configurations for each model in the github repository whose link is mentioned in the abstract section.

4.1.4 Metrics

We use the pixel accuracy as the evaluation metric to assess a semantic segmentation which reports the percent of pixels in the picture which were accurately classified. It is the easiest to understand the conceptual meaning as compare to other evaluation metrics. The pixel precision is normally reported for each class independently just as worldwide across all classes. While considering the per-class pixel precision we're basically evaluating a binary mask; a true positive addresses a pixel that is accurately anticipated to have a place with the given class though a true negative addresses a pixel that is effectively distinguished as not having a place with the given class.

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{1}$$

Here, TP is the frequency at which both the actual value and the predicted value are True, and TN is the frequency at which both the actual value and the predicted value are False. FP means the frequency at which the actual value is False or the predicted value is True, and FN means the frequency at which the actual value is True or the predicted value is False.

4.2. Experimental Result

We conduct our experiments on four basis for each model. First, we run the model by changing the activation function such as we use ReLu and ELU for our study. Second, we run the model by changing the optimizer. In our case, we use Adam and SGD. Third, we run the model by changing the learning rates to a greater and smaller value and fourth by adding and not adding data augmentation

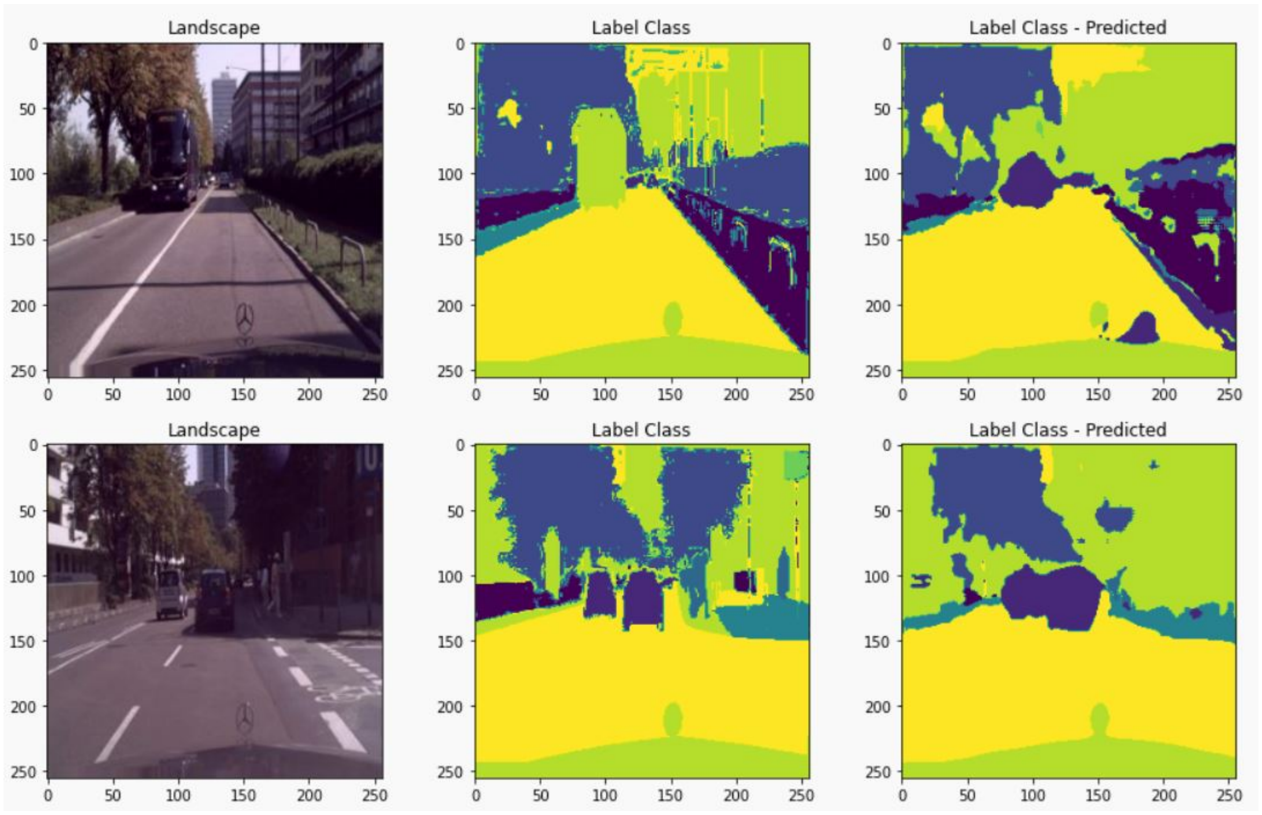

Figure 8. Output from the U-Net Model

techniques. First, we will discuss the quantitative analysis on the results we got from different models followed by qualitative analysis.

4.2.1 Quantitative Analysis

Tab. 1 shows the pixel accuracy of each model with different hyper-parameters. First, we run the SegNet model with all different variants. The best accuracy was achieved by ELU activation function with 10^{-4} learning rate along with Adam optimizer and with data augmentation techniques, getting a score of 84.73%. However, changing the optimizer to SGD was not a good idea as it's performance drastically decreases to 65.31%. Increasing or decreasing the learning rates to 10^{-2} and 10^{-5} respectively, gives almost score 79.08% and 9.64%, which is not a bad performance. Running the SegNet model with ReLu activation function, Adam optimizer, with learning rate of 10^{-4}, and without data augmentation techniques gives a score of 81.4%. This performance degradation shows the importance of data augmentation technique for semantic segmentation. Afterwards, For U-Net model, the best performance was achieved by the same parameters as we get for the SegNet model. With ELU function, Adam optimizer, 10^{-4} learning rate, and with data augmentation, it achieves a score of 82.65%. However, changing the learning rate to 10^{-2} gives a very bad performance accuracy of 33.06% only. And changing the optimizer to SGD with ELU function and 10^{-4} learning rate and with data augmentation technique gives a score of 56.32% which is not as bad as with 10^{-2} learning rate. Also, when we try to increase the learning rate to 10^{-5}, it gives an accuracy of 75.7%, which is a descent accuracy. And with ReLu activation function, Adam optimizer, 10^{-4} learning rate and without data augmentation technique gives a score of 81.42%. Lastly, we perform for FCN model with learning rate scheduling and with data augmentation techniques, it gives a score of 83.92% which is a very good score as compare to other models.

After comparing all the models with different hyper-parameters, it looks like model with ELU activation function, Adam optimizer, 10^{-4} learning rate and by using data augmentation, performances of model increases.

4.2.2 Qualitative Analysis

Fig. 7 shows the output of an image from the SegNet model, where the first image is the original image, second is it's true mask, third is the predicted segmentations in the image and fourth is the output we got from the model. It

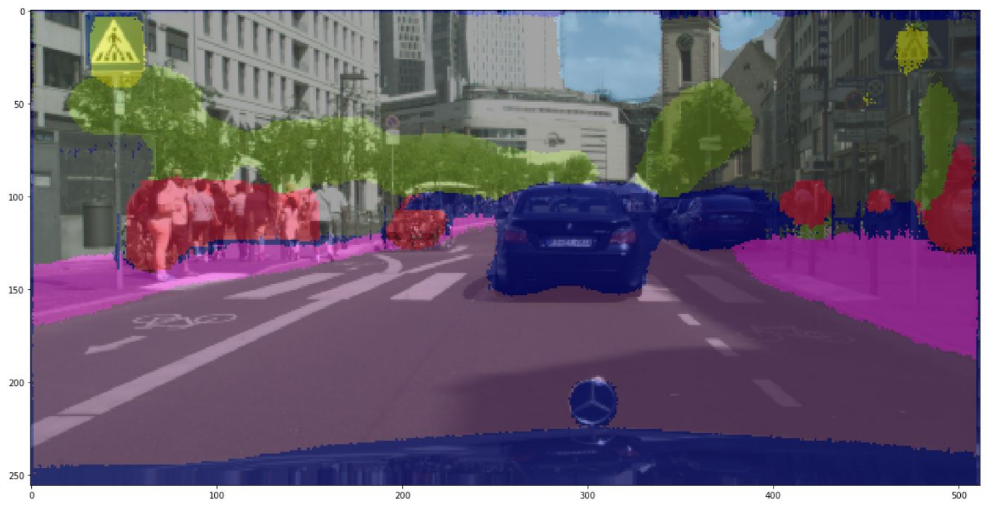

Figure 9. Output from the FCN Model

can be clearly seen that the SegNet model has done a good job here. It's not perfect as it failed to detect pedestrian and traffic signals. Fig. 8 shows the output of two images from the U-Net model, where the first column is the original image, second is the true mask or label class image, and third is the predicted label class from the U-Net model. The performance of U-Net model is also not bad as compare to SegNet model. The predicted class label is almost equal to label class. However, Fig. 9 shows the output from the FCN model and this model successfully distinguish between pedestrians, sign boards, and cars. Overall it shows a good performance for semantic segmentation.

5. Conclusion

In this work, we study three different pre-existing models such as SegNet, U-Net, and FCN with VGG-16 as the encoder for image segmentation task on cityscapes dataset. Further, we try different techniques in order to improve that performance. We also presented a full detailed comparison between three different DCNNs run with different hyper-parameters. Additionally, We use different techniques in order to improve the pre-existing models and compare the performance of the models with each other.

After comparing each model on every technique, we found that the SegNet model with Adam optimizer, learning rate of 10^{-4}, ELU activation function, and with data augmentation techniques gives the best result as compare to other models, and it achieves the highest score of **84.73**%.

6. Future Work

In future, we will try to conduct more experiments with different techniques. Due to limited resources such as number of GPUs and memory, it takes more time to train a single model. and because of short time period, we couldn't test more different techniques and models.

References

[1] Vijay Badrinarayanan, Ankur Handa, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv preprint arXiv:1505.07293*, 2015. 2

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 2, 6

[3] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 2

[4] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 2