National University of Sciences and Technology
School of Electrical Engineering and Computer Science
Department of Computing

CS893: Advanced Computer Vision Spring 2022

Assignment 3

Semantic Segmentation

Instructor: Dr. Muhammad Moazam Fraz

Submitted by: Muhammad Salman Akhtar
Registration No: 398895

1 Introduction

Semantic segmentation is aimed at classifying each pixel in an image to its corresponding class. It is one of the dense prediction tasks in computer vision domain with enhanced complexity as compare to simple classification tasks in which an image is identified into a single dominating class. Semantic segmentation is essential for visual scene understanding since it not only distinguishes the objects among classes in the image but also identifies their spatial locations which are useful for finding out their relationships. It achieves this by grouping the pixels of localized image patches into respective classes thus creating a segmentation mask. Segmentation is beneficial in several diverse domains such as enabling autonomous vehicles to understand their environment, creating image filters (background blurring), satellite image mapping (segmenting roads, water bodies, crops) and augmenting medical image diagnostics (X-ray and MRI scans).

2 Dataset

The dataset is a subset of 'Cityscapes' dataset which is a largescale dataset of street scene images from different cities with pixel-level annotations. This subset dataset comprises of 367 training images along with their masks and 101 test images along with their corresponding masks. However, training dataset is further split into 20% validation dataset as required by the assignment. The dataset is annotated into 12 classes (11 objects and 01 background) as depicted by figure 1. The dataset suffers from class imbalance since the bigger objects such as sky, buildings, roads and trees dominate the scene as compare to smaller objects such as poles, signals and pedestrians. Figure 2 summarizes the class representation of each object in the combine dataset.

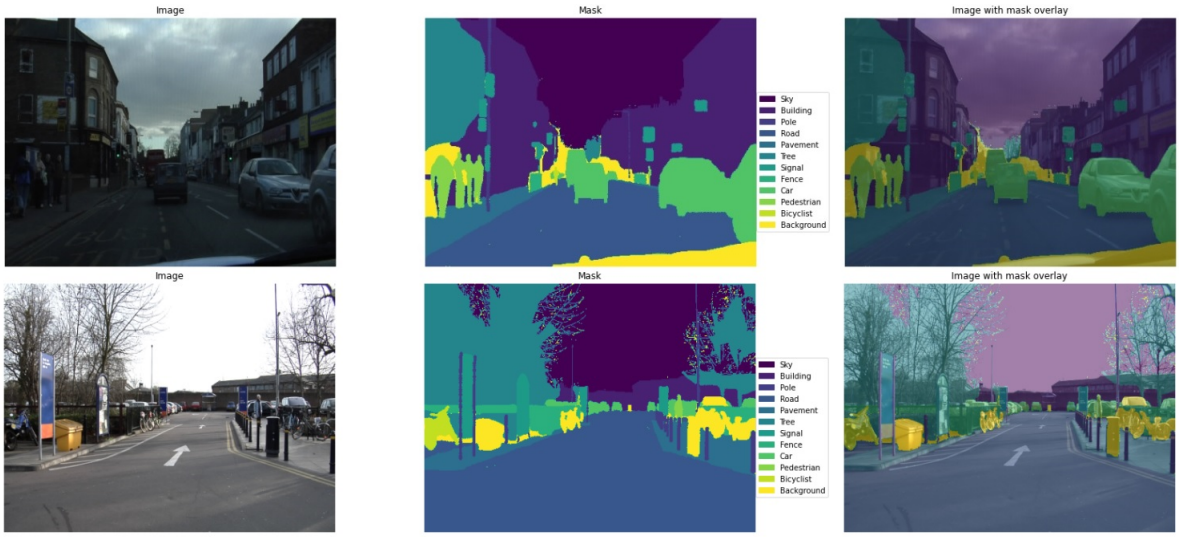

Figure 1: Sample Dataset Pictures

3 Existing Methods for Segmentation

Before the dawn of deep learning era, semantic segmentation was performed through techniques such as clustering, partial differential equation-based methods, watershed transformation and graph partitioning methods. However, since deep learning has taken the center stage in computer vision, these methods have become somewhat obsolete since they do not generalize well to the large-scale real-world images. Convolutional Neural Networks have been used ubiquitously in modern deep learning-based segmentation methods.

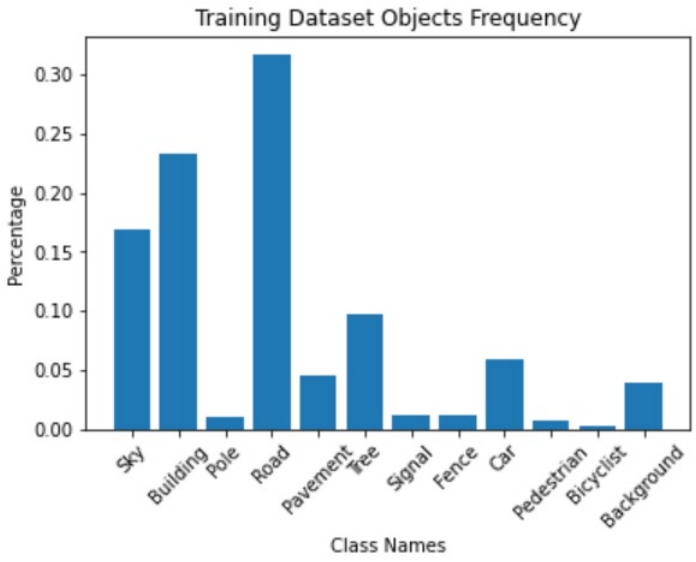

Figure 2: Training Dataset Class Distribution

One of the earliest such architecture was Fully Convolutional Network (FCN) (1) in which features are learned through cascaded convolution layers and features maps are successively down-sampled. The prediction is done on the up-sampled feature map and accuracy can be improved by concatenating the feature maps from earlier down-sampled layers. U-Net (2) is another encoder decoder-based architecture which was proposed for medical images. For up-sampling the U-Net introduced transposed convolutions which incorporated learnable up-sampling technique in the model. DeepLab v3+ (3) introduced Atrous Spatial Pyramid Pooling block for handling multi-scale images. ParseNet (4) proposed the incorporation of global context for enhancing the model accuracy.

4 Model Architecture

For this assignment, I have adopted two architectures. For first architecture, I have used U-Net with the VGG16 backbone. For second architecture, I have used a variation of U-Net with the backbone of DenseNet 121.

4.1 Architecture 1

In architecture 1, U-Net is implemented with VGG16 as a backbone network (encoder). VGG16 backbone pre-trained on 1.3 million images (imagenet dataset) for transfer learning. The encoder module comprises of 05 convolution blocks where each convolution block doubles the number of filters while halving the spatial dimension. The decoder module also comprises of similar convolution blocks. However, in decoder blocks instead of simple convolutions, transposed convolutions have been used for up-sampling. Moreover, the up-sampled feature maps are concatenated with the feature maps from corresponding encoder block followed by couple of convolution layers with Batchnormalization and ReLU activation. Figure 3 shows the architecture for this model.

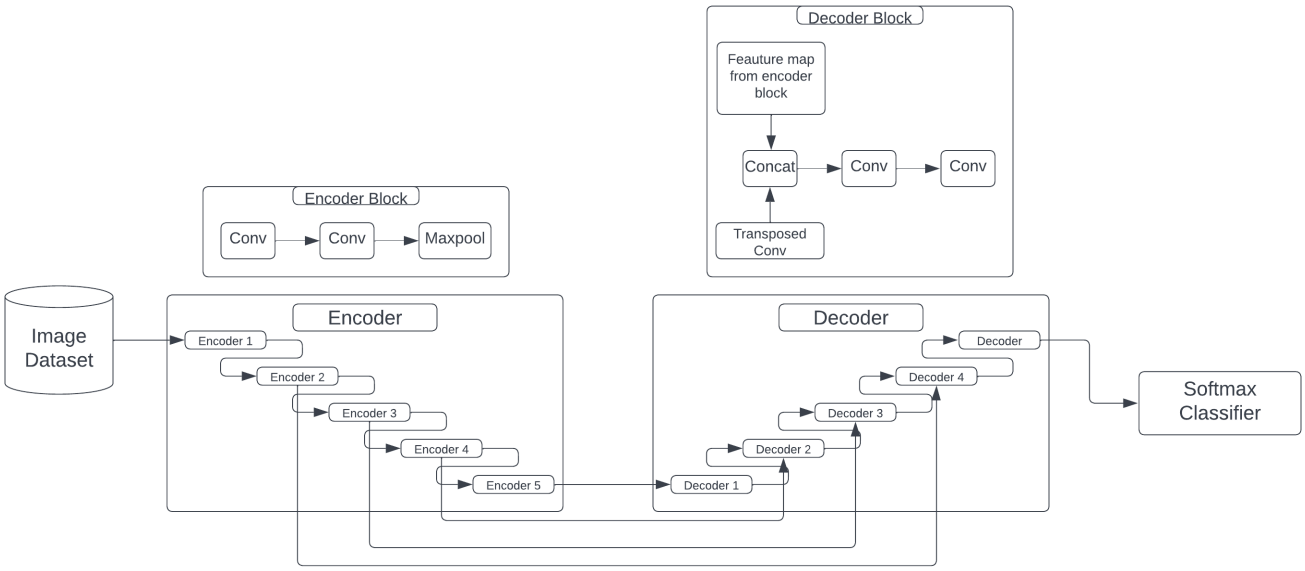

Figure 3: Architecture 1 (VGG16 backbone

4.2 Architecture 2

In architecture 2, a variation of U-Net is implemented with DenseNet 121 as a backbone network (encoder). DenseNet backbone pre-trained on imagenet dataset was used for transfer learning. The encoder module comprises of 04 convolution groups where each group has multiple convolution blocks. The number of filters are sequentially increased while the spatial dimensions of the feature maps are reduced with successive convolution groups. Within each group, the feature maps are shared with next convolution block by concatenation. Figure 4 shows the architecture for this model.

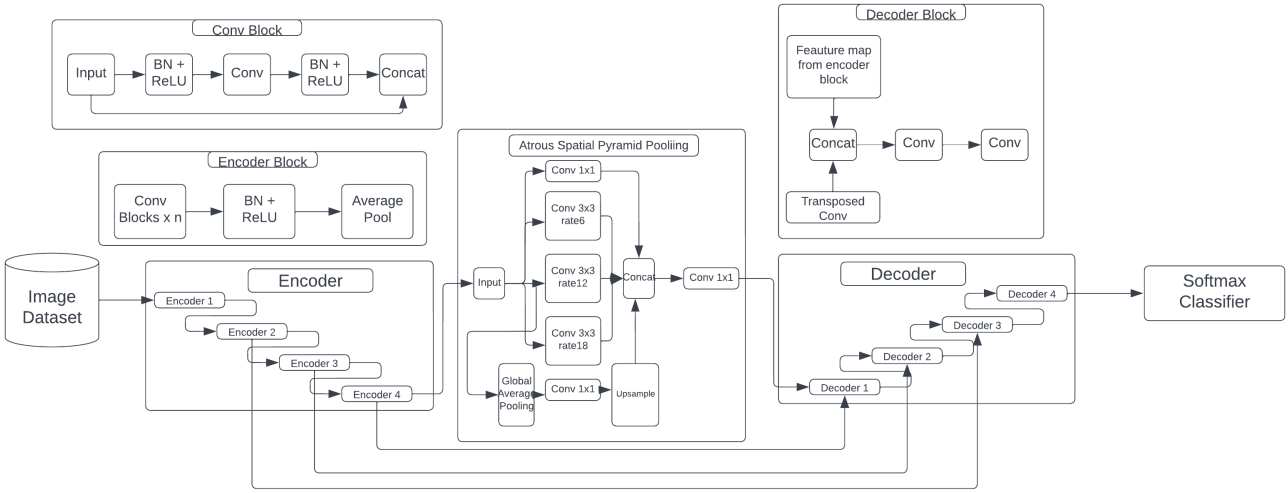

Figure 4: Architecture 2 (DenseNet121) backbone

5 Model Training

Both models are trained on Google Colab platform on allotted GPU. A custom data generator class was created using tensorflow modules that is capable of generating dataset batches (images, masks) of desired sizes. Moreover, this dataset generator is also capable of image augmentation on the fly. For image augmentation 'albumentations' library was used for incorporating random rotation, scaling, shifting and brightness variation. Table 1 summarizes the augmentation operations that were applied to the training dataset.

Table 1: Augmentation Parameters

Horizontal Flip	Random Crop	Rotation(deg)	Shift	Brightness
0.5 prob	256x256	5	[0.8, 1.2]	0.5 prob

Architecture 1 was trained using simple softmax loss function for 150 epochs using Adam optimizer. However, for architecture 2 a modified loss function was used to handle dataset imbalance which makes the ordinary loss functions (softmax loss, dice loss) rather ineffective. Therefore, a combination of weighted softmax and dice loss were implemented. The weights were calculated using effective number estimation as suggested in (5). The weights are calculated such that less represented class is assigned more weight. This penalizes the model more when it wrongly predicts the less represented class thus forcing the model to learn even the less represented classes. Through experimentation it was found that a combination of loss functions (weighted softmax + dice loss) were more effective. Equation 1 is used for calculating custom loss function for training the model (Architecture 2). Here w is the weight while α is the parameter for tweaking the importance given to weighted softmax and dice loss. Table 2 shows the training parameters while figure 5 and figure 6 depicts the training graphs for architecture 1 and architecture 2 respectively. The reason for sudden rise in the loss at epoch number 40 for architecture is due to the reason that the value of α in equation 1 was changed from 0.8 to 0.9.

$$loss = \alpha * w * (-Y * \log(\hat{Y})) + (1 - \alpha) * (1 - \frac{2 * Y * \hat{Y}}{Y + \hat{Y}}) \tag{1}$$

Table 2: Training Parameters

Model	Batch Size	Epochs	Learning Rate	Optimizer	Average Epoch Time	Loss Function
Architecture1	32	100	0.001 (reduced by factor of 0.5)	Adam	12 secs	Softmax
Architecture2	32	180	0.001 (reduced by factor of 0.5)	Adam	15 secs	Custom

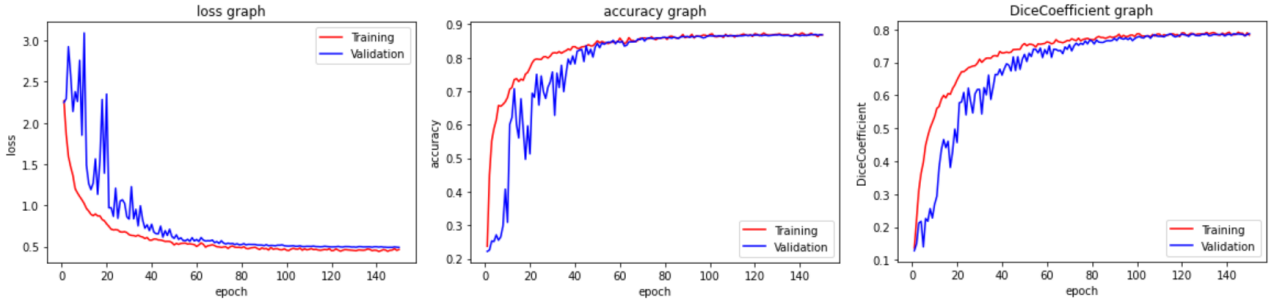

Figure 5: Training Graph (Architecture 1)

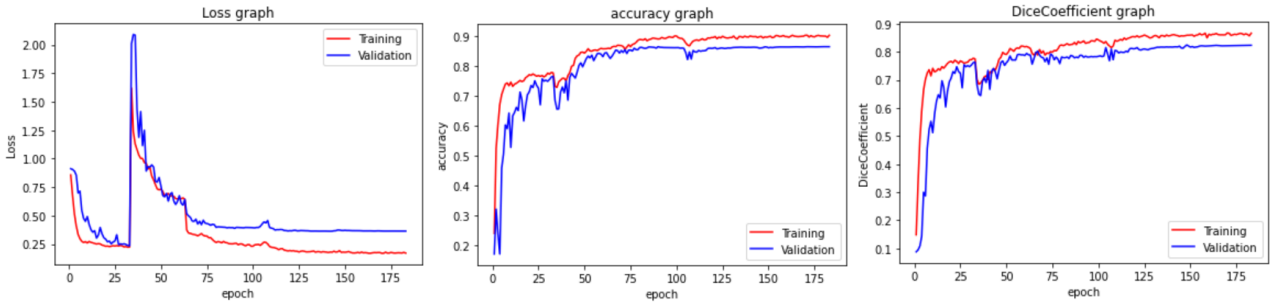

Figure 6: Training Graph (Architecture 2)

6 How to run the code

The models are implemented using the python programming language in Jupyter notebook and corresponding "Semantic_Segmentation_Assignment3.ipynb' file can be accessed from github link. During programming, it was tried to follow the modular and object-oriented approach. The code expects that extracted dataset folder "A3_Dataset" is present in the same directory with the code file. All the cells should be run sequentially from top to bottom as per instructions in the notebook. The notebook also includes the supporting functions that are required for training and testing the model.

7 Results and Discussion

Both models were evaluated based on number of metrics. Table 4 shows the comparison of architecture 1 and architecture 2 against accuracy and Dice Coefficient. It can be seen that Architecture 2 has better accuracy as well as Dice Coefficient as compare to architecture 1. The accuracy is the measure of how accurately the model predicted each pixel in the image without regard for class imbalance. Dice Coefficient is a better metric for semantic segmentation task because it measures the intersection over union between the predicted mask and ground truth. Thus, for segmentation which requires dense predictions it better summarizes the results.

Table 3: Result Comparison (without class distinction)

Metric	Architecture 1	Architecture 2
Dice Coefficient	0.7704	0.8467
Accuracy	0.8526	0.8918

For multiclass segmentation task it would be even better to analyze the sensitivity, specificity and F1 score of individual classes. The sensitivity also known as recall indicates the number of correctly identified positive pixels divided by total number of positive pixels for that class. Table 4 shows that for architecture 1 which has been trained on simple softmax loss performs very poorly for under represented classes like pole, signal, fence, pedestrian and bicyclist. Similarly, the model also fails to perform satisfactorily with respect to F1 score for the under-represented classes. F1 score combines the precision as well as recall by taking their harmonic mean thus summarizing the results better. Looking closely, it can be identified that the Architecture trained on simple softmax loss ignored the under-represented classes and failed to correctly predict even a single instance of these classes. However, architecture 1 performs very nicely on specificity which indicates the number of correctly identifying the negative pixels divided by total number of negative pixels predicted for that class. The architecture 2 was trained on a combination of weighted softmax and dice losses. Table 4 indicates the improved results for under-represented classes in terms of sensitivity and F1 score which basically indicates that now our model has also learned the under-represented classes as well and predicting number of correct instances of these classes. A significant improvement can be observed in the pavement,

fence, pedestrian and bicyclist classes whose sensitivity has been increased from 0 to 0.92, 0.75, 0.6 and 0.79 respectively. A positive effect is also seen for signal class whose sensitivity has been increased to 0.25 which although still less yet better predicted as compare to architecture 1. The pole class does not have any appreciable effect and it is still not identified by the model. This could be due to the reason that poles are very thinly represented in the images and thus difficult to detect. Moreover, pedestrian class show low F1 score despite having relatively high recall which indicates a greater number of false positives for this class. This could possibly due to the reason that pedestrian is represented by relatively small cluster of pixels but have a greater number of instances and thus model tends to falsely predict this class confusing it with other similar class (bicyclist).

Table 4: Result Comparison (individual classes)

Class	**Architecture 1**			**Architecture 2**		
	Sensitivity	**Specificity**	**F1 score**	**Sensitivity**	**Specificity**	**F1 score**
Sky	0.9776	0.9948	0.9635	0.9678	0.9962	0.9655
Building	0.8438	0.9775	0.8831	0.8457	0.9878	0.8989
Pole	0	0.9999	0	0.08346	0.9899	0.06322
Road	0.9626	0.9855	0.9633	0.9549	0.9958	0.9719
Pavement	0.8830	0.9827	0.8536	0.9223	0.9862	0.8901
Tree	0.9748	0.9618	0.9008	0.9637	0.9786	0.9303
Signal	0	0.9999	0	0.2596	0.9965	0.3242
Fence	0	0.9999	0	0.6881	0.9826	0.6234
Car	0.7905	0.9772	0.5122	0.6623	0.9959	0.6729
Pedestrian	0	0.9999	0	0.6053	0.9856	0.3600
Bicyclist	0	0.9999	0	0.78984	0.9888	0.6876
Background	0.6478	0.9535	0.2488	0.0779	0.9966	0.1176

The qualitative results also endorse the quantitative results. It can be observed from figure 7, figure 8 and figure 9 that Architecture 1 failed to detect signal, fence, pedestrian and bicyclist. Architecture 2 detects all the annotated classes with reasonable degree visual results. However, there are also significant miss classifications as well. As can be seen in scene 1, the model partially classified the bicyclist as a pedestrian.

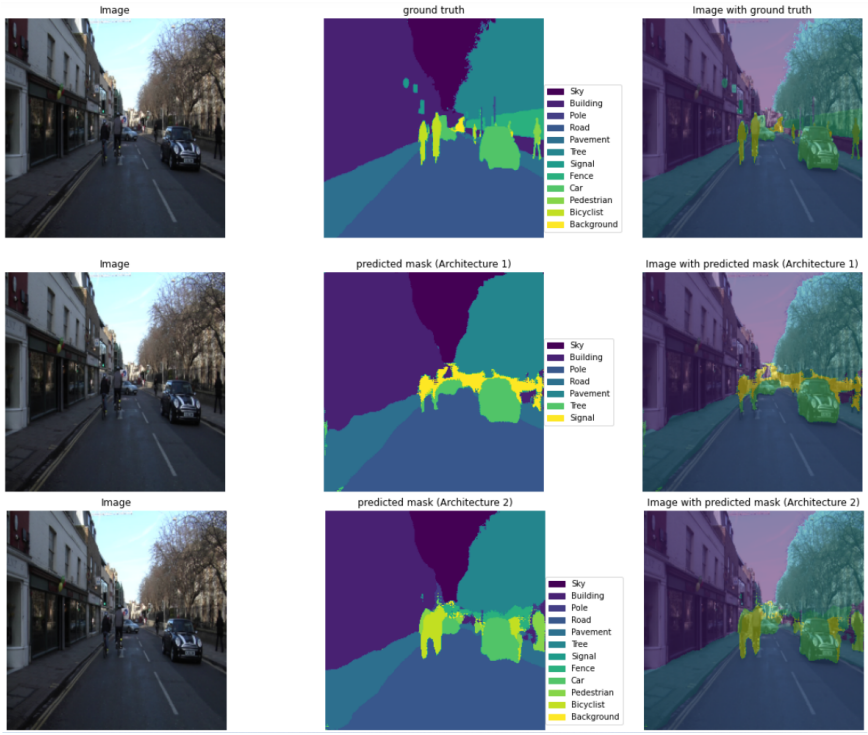

Figure 7: Scene 1

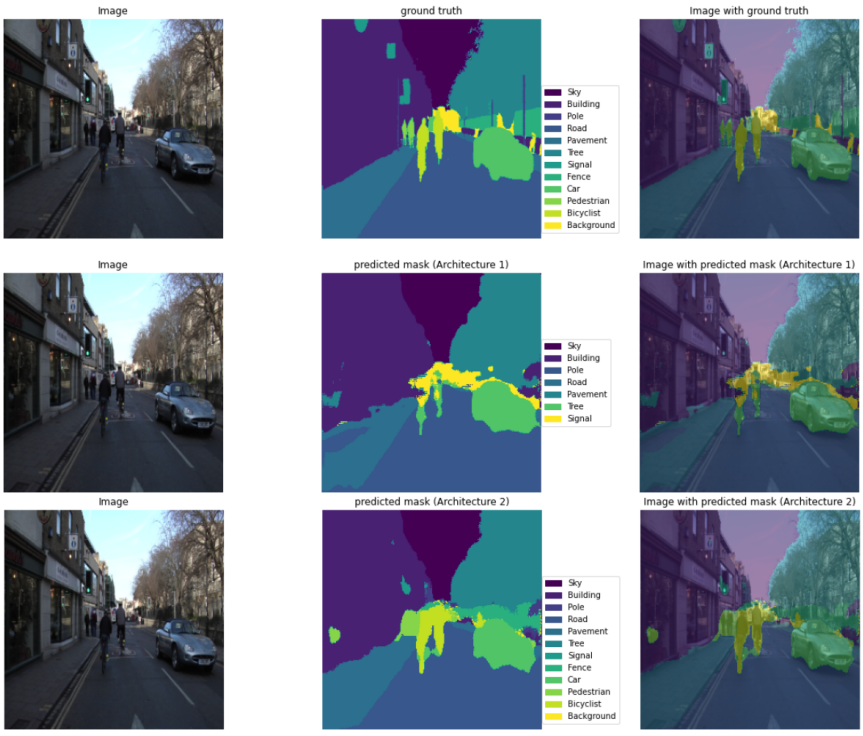

Figure 8: Scene 2

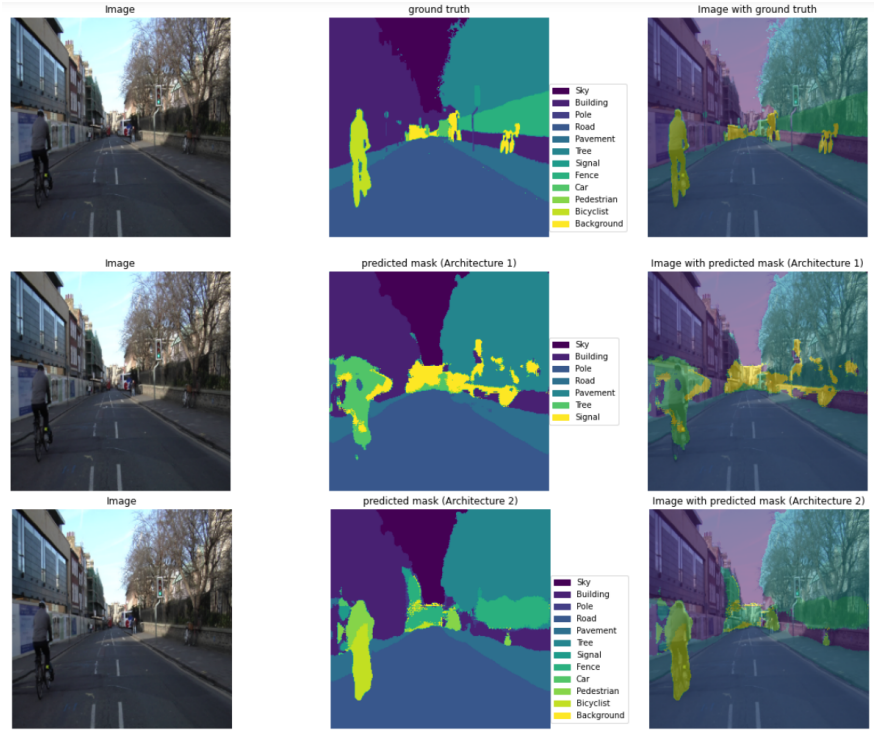

Figure 9: Scene 3

8 Conclusion

In this assignment, two separate architectures (U-Net and U-Net plus ASPP) with separate backbone networks (VGG16 and DenseNet121) pre-trained on imagenet dataset were implemented. Both models were trained using a subset of cityscapes dataset using a custom data generator capable of augmenting the images and masks simultaneously. Architecture 1 was trained on simple softmax loss while architecture 2 was trained using a combination of weighted softmax and dice loss. Comparing the results, it was revealed that architecture 1 although achieved a respectable accuracy and dice coefficient score but failed to detect all the annotated classes. However, the architecture 2 performed reasonably well on most of the classes thus justifying the use of modified loss function and change in model architecture.

References

[1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[3] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[4] W. Liu, A. Rabinovich, and A. C. Berg, "Parsenet: Looking wider to see better," *arXiv preprint arXiv:1506.04579*, 2015.

[5] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 9268–9277.