SEMANTIC SEGMENTATION ON CITYSCAPES DATASET

Rishabh Saxena
University of California, Irvine

ABSTRACT

Semantic segmentation is a Computer Vision task that aims to classify each pixel of an image to one of the predefined labels comprising of the range of objects that can be present in the image scene. One of the major use-case of this task is to improve visual understanding of complex urban street scenes, which is required in a wide range of applications such as autonomous driving. The Cityscapes dataset, is a widely accepted to capture the complexity of real world urban scenes. In this project, we aim to develop some models based on well known deep learning architectures to implement semantic segmentation on the Cityscapes dataset.

1 BACKGROUND

Semantic Segmentation is specific task in Computer Vision that comes under a broader set of tasks widely referred to as Image Segmentation. In this section we will briefly explain the different kinds of tasks under Image Segmentation.

1.1 IMAGE SEGMENTATION

The general task of image segmentation refers to identifying the subject and the background from a given image. However, this basic task is can be more concretely divided into different tasks such as Semantic Segmentation and Instance Segmentation.

Figure 1: Segmentation Input Example. Source: (Mallick, 2018)

1.2 SEMANTIC SEGMENTATION

Semantic Segmentation refers to the task of labeling each pixel given an image to one of predefined classes. The predefined classes will be a list of objects or elements expected to be found in our image.

Figure 2: Semantic Segmentation Example. Source: (Mallick, 2018)

1.3 Instance Segmentation

Instance Segmentation refers to the task of identifying different ”instances” of a particular object in an image. An example could be to identify to location of all instances of people in an image. Even though it is similar to object detection, the output for instance segmentation is the contour of the object instance, and unlike semantic segmentation we do not label ever pixel.

Figure 3: Instance Segmentation Example. Source: (Mallick, 2018)

1.4 Panoptic Segmentation

This is a combination of Semantic and Instance Segmentation. Each pixel of input is assigned a label however if multiple instances of a class are found then we know which pixel belongs to which pixel.

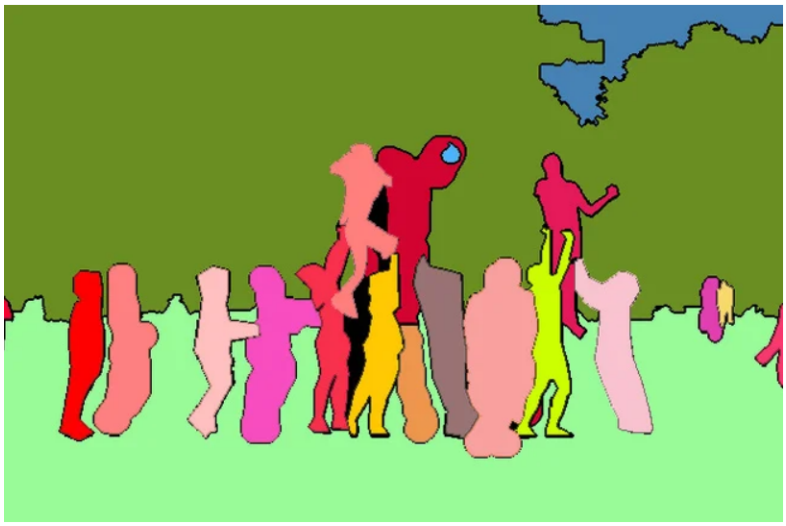

Figure 4: Panoptic Segmentation Example. Source: (Mallick, 2018)

2 MOTIVATION

The ability for a computer algorithm to determine and locate all the different objects in an image is an extremely interesting, useful and challenging problem. One of the many ways such an algorithm can be used is for autonomous driving, where the car needs to decide where are all the different objects located in its vision. The potential benefits of solving such a complex problem and understanding the nuances of the architectures development to achieve this is the primary goal of this project.

3 DATASET DESCRIPTION

The Cityscapes Dataset (Cordts et al., 2016) is widely regarded as one of the best public datasets for urban street understanding. The dataset consists of 5000 finely images annotated images with an additional 20,00 coarsely annotated images for additional training requirements. For the purposes of this project we intend to only train on the finely labeled images. The dataset creators have provided instance labels as well as semantic labels for all images. For the purposes of the semantic segmentation task, we will only deal with the semantic label masks.

3.1 SEGMENTATION CLASSES

Classes in the dataset have been divided into groups, which represent a collection of objects like vehicle, construction etc. Each group contains multiple classes within it. This helps is associating a additional level of detail with the images. The table below lists the various image labels:

Table 1: Cityscapes Image Class Labels

GROUP	CLASSES
flat	road, sidewalk, parking, rail track
human	person, rider
vehicle	car, truck, bus, on rails, motorcycle, bicycle, caravan, trailer
construction	building, wall, fence, guard rail, bridge, tunnel
object	pole, pole group, traffic sign, traffic light
nature	vegetation, terrain
sky	sky
void	ground, dynamic, static

3.2 Dataset Splits

The authors of the dataset have carefully divided the 5000 finely labeled images into training, validation and test sets. The training set consists of 2975, while the validation set contains 500 images and the test set contains 1525 images. The images of in each split contain images captured in different cities with a good mix of suburban and urban areas in all splits. Additionally, the images in all splits capture different weather conditions to help develop really robust systems that can handle the real world challenges of this problem.

4 Exploratory Data Analysis

The colab notebook contains the entire EDA that has been performed till now. Here we will highlight two things. The collab notebook is part of this document as an appendix.

4.1 Vieweing the Dataset

We have created a small function that fetches 9 random images from the dataset and displays them on the screen along with its label mask overlayed on top. Screenshot of this is being attached as reference.

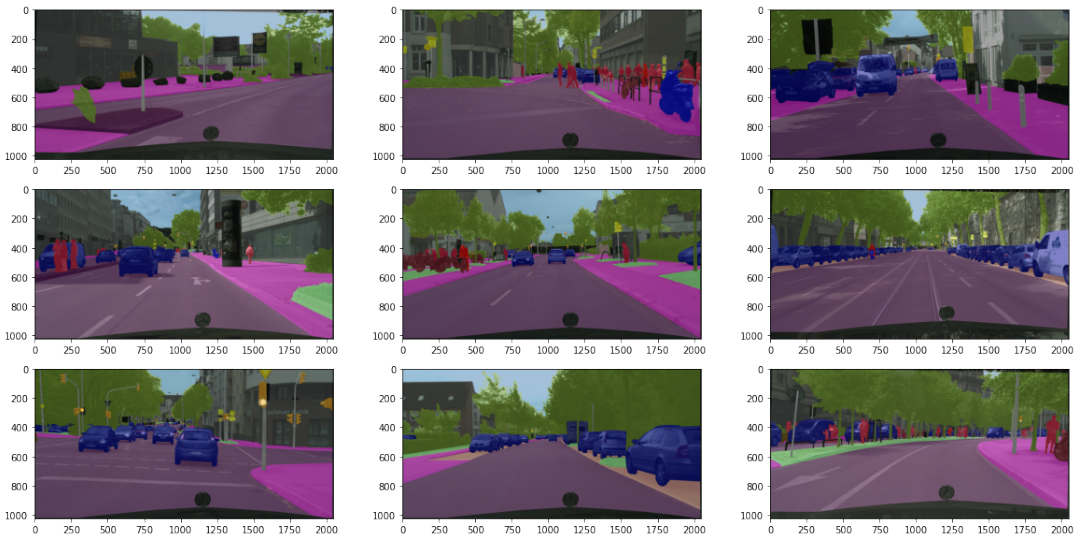

Figure 5: Training Images with Masks overlayed on top. Source: Collab Notebook

4.2 Visualizing disparity in labels

A batch of 100 images was processed from the training dataset, and a frequency count of label pixels were generated. It was found that there is a huge disparity in the labels frequency as some of the very few labels occour a lot in the training dataset.

5 Model Architectures

In this section we will discuss some of the architectures that are widely used for the segmentation task.

5.1 Challenges with Imagenet Classification Architectures

The Imagenet classification task brought the advent of Convolutional Neural Networks for Image Classification. Architectures like the Resnet architecture(He et al., 2016) have human like performance for the classification task.

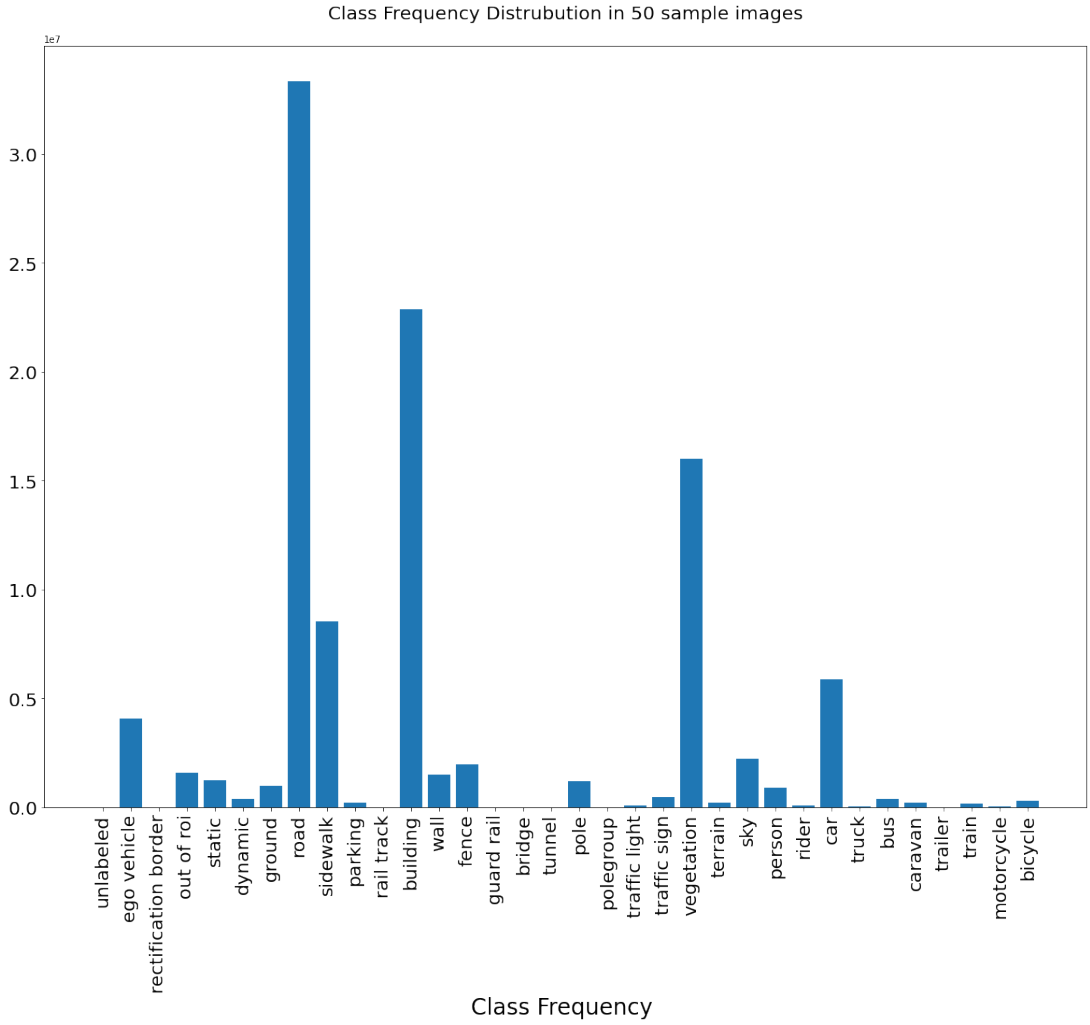

Figure 6: Histogram plot of label frequency. Source: Collab Notebook

These architectures depend on shrinking the image in its spatial dimension as the images is processed through layers of convolution operation. This leads us to down-sampling the images to make class predictions. However, for the task of segmentation, we need to provide a label for each pixel of the input image which requires our output to be of the same dimension as our input(height and width). This poses a problem in adoption of such architectures for the task of image segmentation.

5.2 U-Net Architecture

This architecture was introduced in the following two papers(Long et al., 2019) & (Ronneberger et al., 2015). This is going to be the architecture that we plan to follow for our implementation of the Segmentation Task.

One of the main operations behind U-Net (and most segmentation FCN based architectures) is the concept of up-sampling. Up sampling refers to an operation of increasing the dimensions of the input image. This forms a core fundamental for architectures designed for segmentation purposes as the models first down-sample a given input image, and then up-samples the image to obtain the output. Each of these components are also referred to as decoder and encoder.

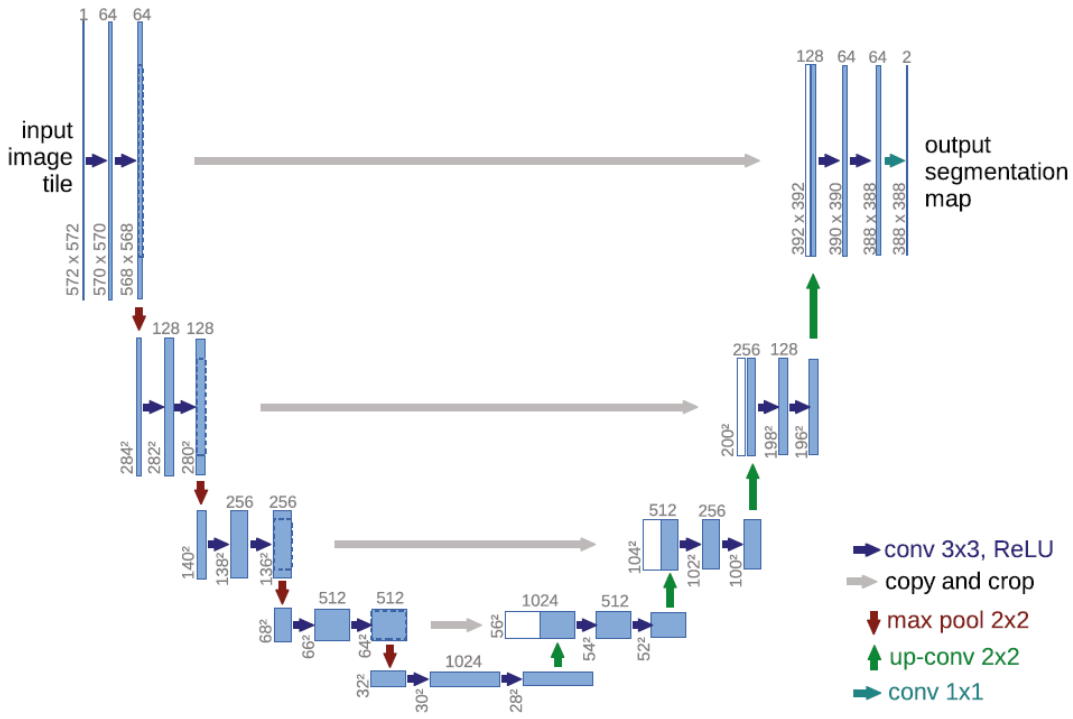

Figure 7: U-Net Architecture Illustration. Source: (Ronneberger et al., 2015)

6 Implementation Details

6.1 Technologies & Framework

The project implementation was done using the PyTorch Framework for Deep Learning which provides very useful predefined DataLoading wrappers for the Cityscapes dataset.

6.2 Code Repository

The entire code base for the project can be found in the following GitHub Repo:

https://github.com/rsaxena07/cityscapesSemanticSegmentation (Clickable Link)

6.3 Data Pre-Processing

Pytorch contains a predefined Dataset Class for Cityscapes Dataset in the torchvision library. We have created an additional wrapper class (`loader/cityscapesDataset.py`) inheriting from that class to implement the following functionality:

- Perform the same random crop augmentation on input image and label mask for training.
- Create a sub-sample of the dataset to improve training speed. This also helps in writing small unit tests for other modules.
- Encode various void classes to a singular value so that they do not interfere during training. The reduce the number of classes from **34** to the **19** important classes.

The original dataset images are of size 1024 X 2048 which are very big for a CNN to train on. Scaling down the images also reduces the finer details from the images which could negatively impact our model performance. The above code helps us load and augment our training images to prepare them for training. The data augmentations implemented are converting images and labels to tensors and performing random crops of 256 X 512 and finally converting them to a tensor.

6.4 Architecture Implementation

U-Net architecture was implemented in PyTorch as shown in Fig 7. To reduce the complexity of the model and the number of training parameters, the number of downsampling layers were reduced 3 (as opposed to 4 in Fig 7). The architecture has a total of 20 Layers having 14 Conv-2D, 3 MaxPooling and 3 Upsampling layers. The activation function used is ReLU. The implementation can be found in the `models/unet.py` in the git repo.

6.5 Loss Functions & Evaluation Metrics

The code for the Loss Functions and Evaluation Metrics was implemented in `loss/loss_metrics.py`.

6.5.1 Loss Functions

The following 2 Loss functions were used for the training experiments:

- **Cross Entropy Loss:** Cross Entropy Loss is one of the most well known Loss Functions for Multi Class Classification problems. It involves using a Softmax function of the model response and then using the negative Log Likelihood to estimate the loss values.
- **Dice Loss:** Dice Loss is a well known loss function and used extensively for Image segmentation problems. In our implementation, we have used the following definition of the Dice Loss function:

  $$DL(prediction, labels) = 1 - DC(prediction, labels)$$

  Where DL() is defined as the Dice Loss and DC() is defined as the Dice Coefficient

6.5.2 Evaluation Metrics

The following 2 Evaluation metrics were used to evaluate the trained models:

- **Accuracy:** Accuracy is defined as the number of pixels classified correctly upon the total number of the pixels in the image. In our implementation we ignore the pixels that belong to the void class in the actual labels. Therefore, Accuracy (Acc) can also be defined as:

  $$Acc() = \frac{TP + TN}{TP + FP + FN + TN}$$

- **Intersection Over Union:** Intersection Over Union is another eval metrics that is particularly useful when most of the samples in our dataset are True Negatives. It calculates the ratio intersection and union of the predictions and actual labels. Therefore IoU can be defined as:

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

- **Dice Coeffecient:** The dice coefficient is a simplified version of IoU which estimates a similar ratio. It is used to calculate the DiceLoss and provides a measure of how accurate we are at predicting a class in the input image. The definition used in our implementation is:

  $$D = \frac{2\sum_{i=1}^{n} P_i.L_i}{\sum_{i=1}^{n} L_i + \sum_{i=1}^{n} P_i}$$

TP - True Positive
TN - True Negative
FN - False Positive
FP - False Positive
L_i - Predictions Vector
L_i - Actual Label Vector

7 Model Training

Here we will explain the various experiments that we have tried.

7.1 Experiment 1: High Regularization

This was the first experiment I conducted and realized that I was not able to reduce the Loss more than 1.4. After a few epochs the model is not learning much and this happens because I had used a very high weight decay for this problem (0.01). Setting such a high weight decay that model was not able to train well and it was underfitting only predicting one of the 3 main classes for all pixels. The graph below shows the Loss Function trend on training.

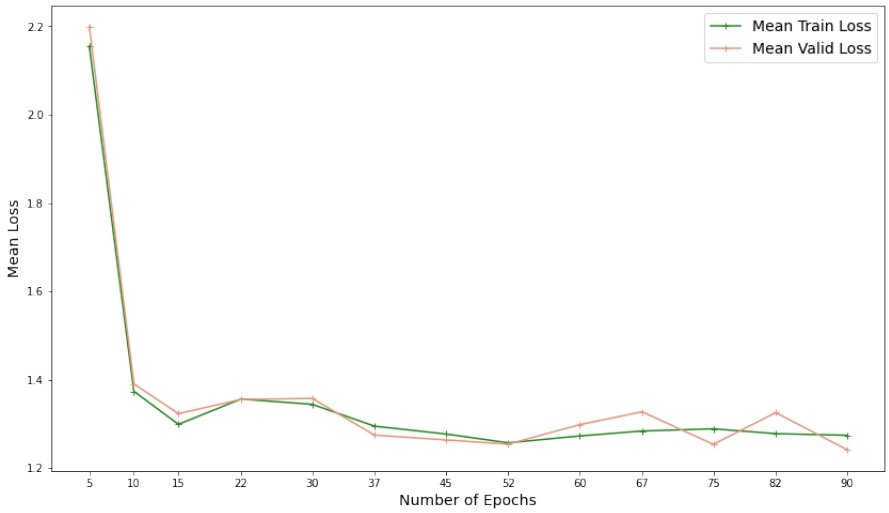

Figure 8: Experiment 1 - Training/Validation Loss Variation

7.2 Experiment 2: Cross Entropy Loss

For the next experiment we used the cross entropy loss with a reduced weight decay parameter. This model was able to learn pretty well and shows the best performance across all the experiments. Training decision chosen were:

- Loss Function: Cross Entropy Loss
- Optimizer: Adam ((Kingma & Ba, 2014))
- Hyper-parameters:
  - Total Epochs: 90
  - Batch Size: 16
  - Learning Rate: 1e-3, 3e-4, 1e-4 (for 60, 40, 20 epochs).
  - Weight Decay (L2 Regularization): 1e-5

Figure 9 shows the trend of training and validation loss during the training. The best performance is seen for epoch 105 and therefore this epoch is picked as the best model.

Figure 10 shows the improvement trend on validation epochs of our model. It can be seen that first it start to recognize simple classes like road, vegetation, building and sky (epoch 14). Then in around 60 epochs it gets good at recognizing vehicles. In the last image it can be seen that it is recognizing humans quite well.

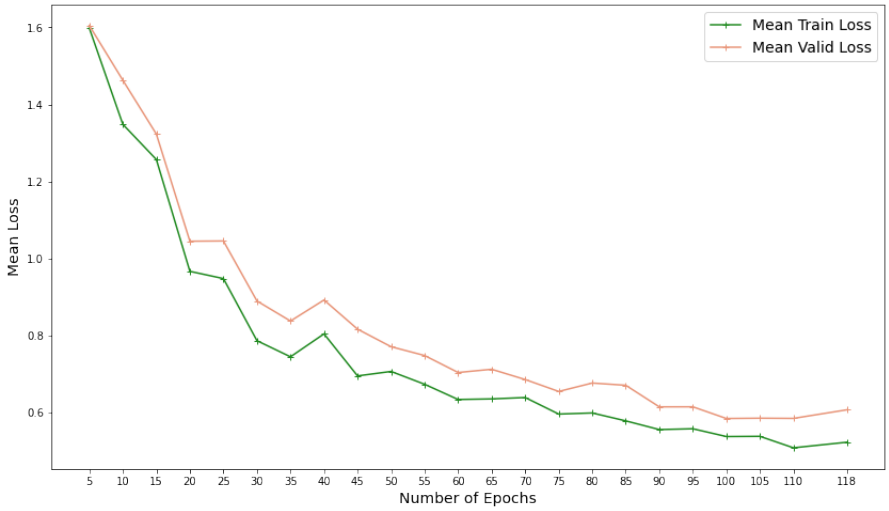

Figure 9: Training and Validation Loss vs. number of epochs

7.3 Experiment 3: Using Weighted Cross Entropy Loss

Using a weighted Cross Entropy Loss functions is very common solution to the problem of class imbalance in training. In this experiment we have assigned weights to each class which are inversely proportional to their occurrence in our training set. To calculate this we have randomly sampled 50 images from the training set and calculated the total number of samples of each class. Finally we use these frequencies to calculate the weight which are inversely proportional.

On training such a model, even after training for 45 epochs, the training and validation loss was observed to be very high (Figure 11). This meant that the model was not able to learn very well. On further inspection of the images obtained for the validation set (Figure 12), it can be seen that they presence of weights negatively impacts the model, as the model starts predicting rare classes almost everywhere. For eg. in a lot of images it starts predicting the human and shrub class everywhere just because getting it right sometimes is very valuable.

We feel further engineering is required to determine the optimum values of these weights, however, we can try other experiments for this project rather than trying to determine other weights.

7.4 Experiment 4: Using Dice Loss

Using Dice Loss we observe that there was a lot of problems in learning from the data. This could be due to some mistake in the implementation, however after referring to multiple different sources and trying out a lot of different experiments for a Week we could still not get the model to train properly and learn useful features.

8 Model Evaluation

In the following section we will analyse the performance of the model trained in experiment 2 on our evaluation metrics. Our analysis is only on this experiment as we were able to identify this as the best performing models among our experiments by looking at the predictions and validation loss values.

8.1 Accuracy

The trend for accuracy with increasing epochs can be seen in Figure 13. It can be seen that the best accuracy is obtained for the model on epoch 118 with an accuracy of **84.43%** on the validation set of 500 images.

(a) Epoch 4

(b) Epoch 14

(c) Epoch 25

(d) Epoch 59

(e) Epoch 75

(f) Epoch 96

(g) Epoch 118

Figure 10: Model predictions on validation images for different epochs. Image from Left to Right: Model Prediction, Input Image, Actual Label Masks

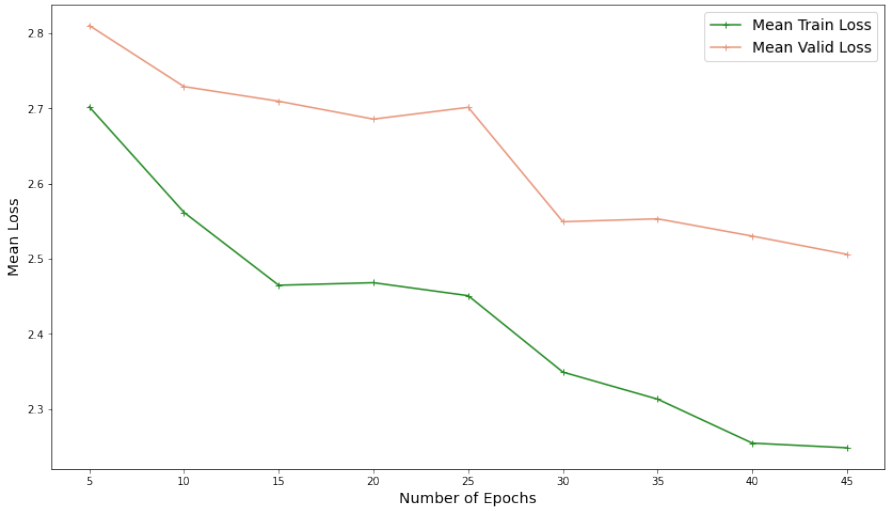

Figure 11: Training and Validation Loss vs. number of epochs

(a) Epoch 8

(b) Epoch 20

(c) Epoch 36

(d) Epoch 44

Figure 12: Model predictions on validation images for different epochs. Image from Left to Right: Model Prediction, Input Image, Actual Label Masks

8.2 Dice Coefficient

The trend for Dice Coefficient with increasing epochs can be seen in Figure 14. It can be seen that the best accuracy is obtained for the model on epoch 80 with an mean dice coefficient of **58.36%** on the validation set of 500 images. This metric is much lower because the there are still many classes which our model is not good at predicting.

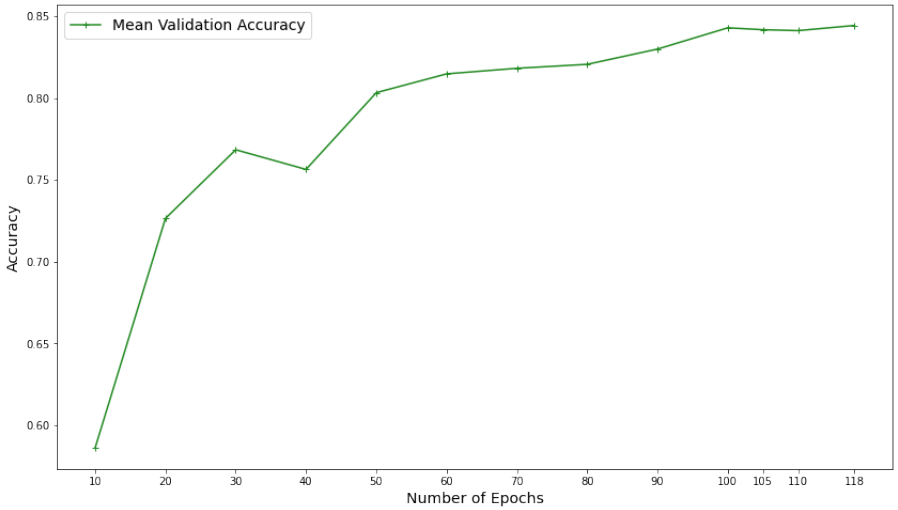

Figure 13: Trend for Accuracy vs. number of epochs

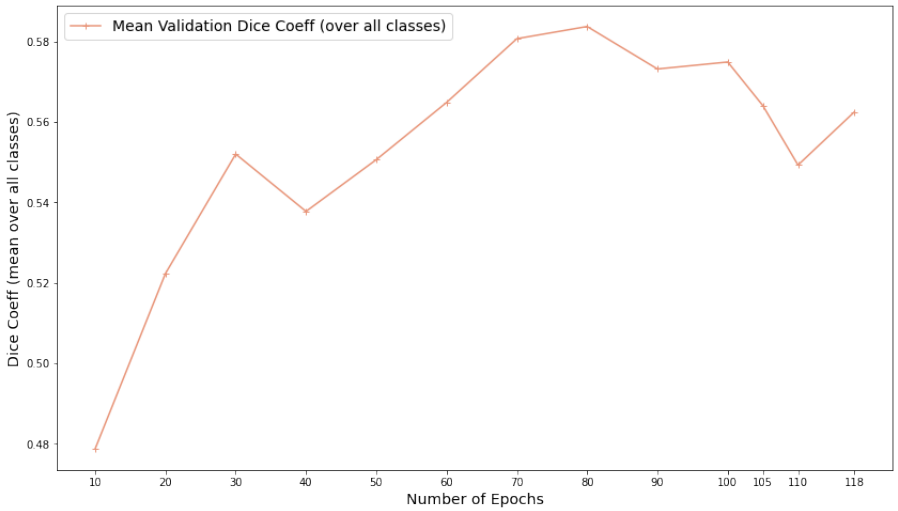

Figure 14: Trend for Dice-Coefficient vs. number of epochs

9 Future Scope

Following section will list some of the improvements that can be made to this project in the future.

9.1 Improving the Dice Loss Training

We would like to improve the results of our Dice Loss training such that it can hopefully help us train som better models for this problem.

9.2 PSP-Net Model

Pyramid Scene Parsing is another widely used segmentation architecture introduced in (Zhao et al., 2016). It has the ability to incorporate global context of an image to improve the segmentation results and eliminates the need for post processing on the predication image. This is a model that we would like to explore for further improving this project.

10 CONCLUSION

In this project we have successfully trained a U-Net based Deep Convolutional Neural Network for the task of semantic segmentation on urban street scenes. Our model gives an accuracy of **84.46%** and mean Dice Coeff (IoU) of **58.36%**

REFERENCES

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

J Long, E Shelhamer, and T Darrell. Fully convolutional networks for semantic segmentation. 2019.

Satya Mallick. Image segmentation, Nov 2018. URL `https://www.learnopencv.com/image-segmentation/`.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4.

Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. *CoRR*, abs/1612.01105, 2016. URL `http://arxiv.org/abs/1612.01105`.

A APPENDIX

Project Submission Details:

The dataset can be found here: Cityscapes Dataset Google Drive Link (Click Here)

Final Model Weights Drive Link (Click Here)

GitHub Code Repo: https://github.com/rsaxena07/cityscapesSemanticSegmentation (Clickable Link)

Collab Notebooks:

EDA Notebook

Final Training/Eval Notebook (Click Here)