REPORT FOR ASSIGNMENT 3

Task 1: A Section explaining the project

i. Introduction of the dataset.

The dataset has been manually labeled for object detection tasks in a driving environment. It contains 101 images, each with a resolution of 960*720 pixels, and each pixel has been assigned to one of 32 object classes.
In this dataset, the "void" label is used to indicate an area in the image that is ambiguous or irrelevant for the specific object detection or classification task in the driving environment. For example, if there is an area in the image where it's difficult to distinguish between two or more object classes, or if there is an area that is not relevant to the task (such as the sky or a nearby building), it may be labeled as "void" to indicate that it should be ignored by the machine learning model.

ii. The structure of FCN-16s, UNet and DeepLabV2 and how to build them up.

Fully Convolutional Networks (FCN-16s):
The FCN-16s model is based on a VGG16 architecture and consists of a feature extractor followed by two 1*1 convolutional layers and two transposed convolutional layers.
The feature extractor has five blocks of convolutional layers, each with multiple convolutional layers, ReLU activation functions, and max pooling operations. The output of the feature extractor is passed through two 1*1 convolutional layers to generate score maps with the same spatial resolution as the feature maps. These score maps are then up-sampled using two transposed convolutional layers, with the second layer incorporating the score maps from the second-to-last layer of the feature extractor to improve localization accuracy. The final output is a segmentation map with a class label for each pixel in the input image.

U-shaped encoder-decoder network (UNet):
The UNet model has an encoder-decoder structure with skip connections between corresponding layers in the encoder and decoder paths. The model consists of four encoder blocks (enc1, enc2, enc3, enc4), a middle block (middle), and four decoder blocks (dec4, dec3, dec2, dec1), each composed of convolutional and up-convolutional blocks.
The encoder blocks gradually reduce the spatial resolution of the input image, while the decoder blocks gradually increase the spatial resolution of the feature maps. The middle block consists of a convolutional block that reduces the spatial resolution of the feature maps further, followed by a transposed convolutional block that increases the spatial resolution to the same size as the encoder feature maps.

DeepLab series (DeepLabV2):

The DeepLabV2 model has three main components for semantic segmentation. The backbone network has three blocks of convolutional layers, followed by batch normalization, ReLU activation, and max pooling.
The first block has a 7*7 convolutional layer with 64 output channels, while the next two blocks have 3*3 convolutional layers with 128 and 256 output channels, respectively.
The ASPP module uses four 3-3 convolutional layers with 256 output channels and different dilation rates of 1, 6, 12, and 18 to capture context information at multiple scales from the backbone network.
The decoder network has a single 1x1 convolutional layer that maps the features output by the ASPP module to the desired number of output classes. It up-samples the resulting feature map to the original image size using bilinear interpolation.

Task 2: A Section explaining the rationale of the design of the models

i. The hyperparameters and techniques used in the project.

We choose optimizers and learning rate schedulers to explore the performance of different combinations of these two hyperparameters. On the other side, we use Cross validation technique to find the best hyperparameters and use data augmentation technique to improve the performance of the models.

ii. For each technique, describe how it affects your model’s performance and analyze the reasons why the technique can boost or decrease the performance of your models.

Cross-Validation:

This technique boosts performance of our model, improves hyperparameter tuning, and increases robustness to data distribution. On the one hand, cross-validation helps to reduce overfitting, which occurs when a model is too complex and performs well on the training data but poorly on unseen data. By evaluating a model on multiple folds of the data, cross-validation can give a better estimate of the model's true performance on unseen data. On the other hand, cross-validation can help to optimize the model's hyperparameters by evaluating the model's performance on different combinations of hyperparameters. This can help to identify the hyperparameters that give the best performance on average across all the folds. However, it might also increase training time and, in some cases, lead to overfitting the validation set.

Data Augmentation:

This technique also improves the performance and robustness of our models by training them on a larger and more diverse set of examples. It applies transformations such as rotation, cropping, scaling, flipping, and color changes to images, which can create new examples that are similar to the original images but with variations in appearance. Since the dataset used in this project is very small, data augmentation can be particularly useful.

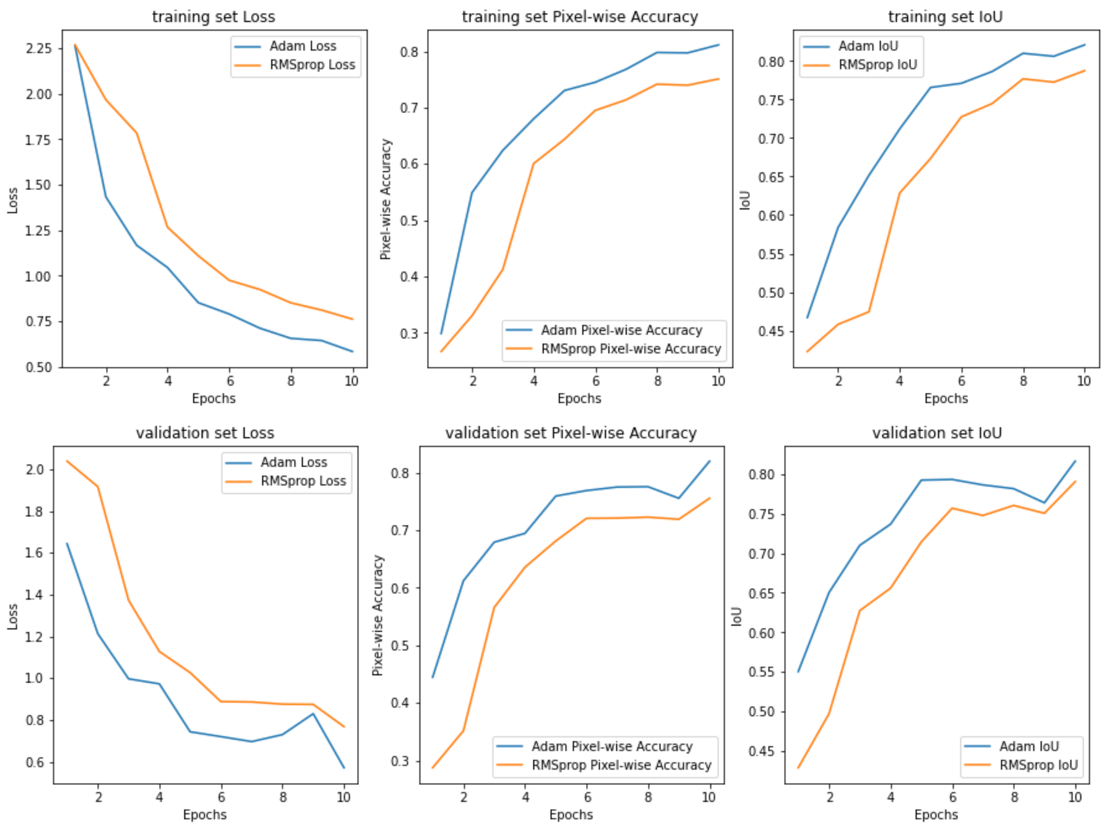
iii. Do you ever have overfitting or underfitting issue while training the models? How do you deal with overfitting or underfitting issue?

Yes, we have overfitting issue while training the models. The overfitting problem cannot be solved fundamentally, it can only be alleviated. Thus, we use data augmentation technique to improve the training process, which increase the size of the training set by adding additional copies of the training set to improve the generalization of the model. So, we manually change the hyperparameters to get a better result on test part.

Task 3: A section describing and explaining your results

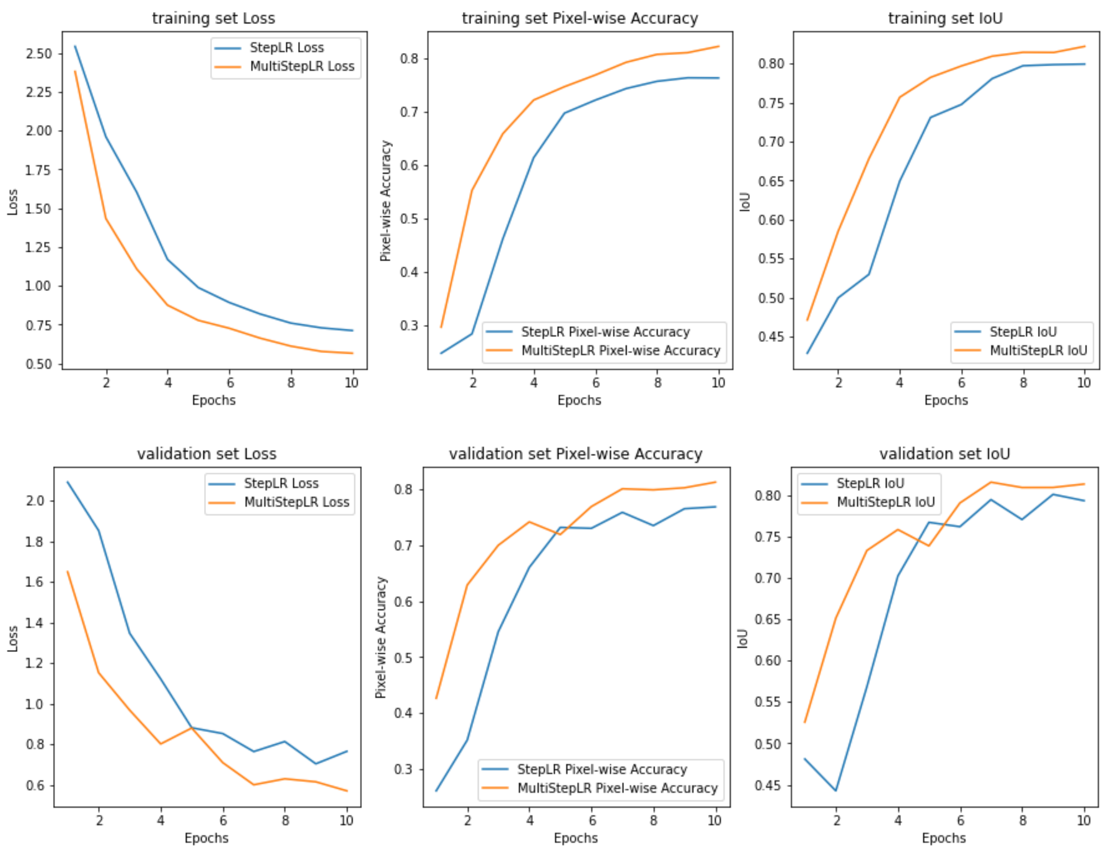
i. Describe and explain the training loss, validation loss and testing accuracy of all

FCN-16s (optimizers):



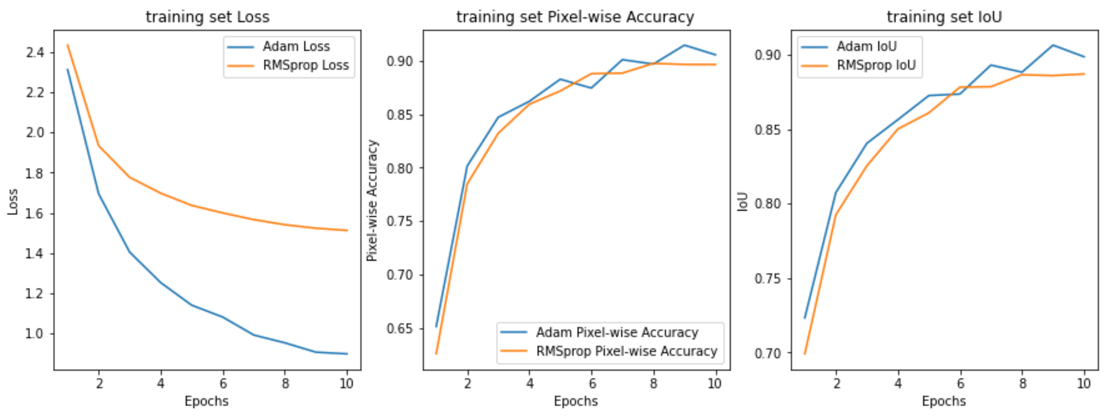
We find that the value of train loss function of both optimizers decrease steadily in the train process, but the value of validation loss function suddenly rise at 9th epoch, causing the overfitting issue. Overall, val_loss is slightly less than train_loss for both optimizers' models, which indicate that the training process is normal. And the accuracy and IoU of Adam is larger than RMSprop. Thus, Adam is better than RMSprop.

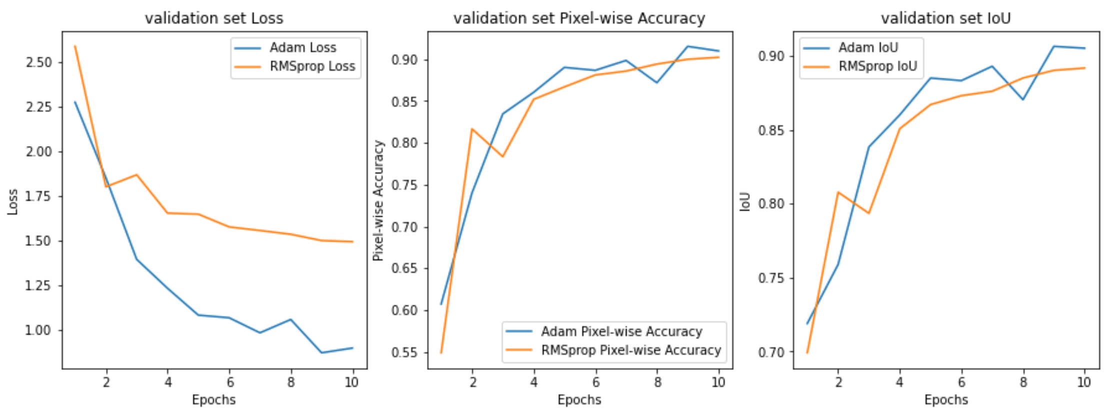
FCN-16s (learning rate schedulers):



We find that the value of train and validation loss function of both learning rate schedulers decrease steadily in the train process. Although the accuracy fluctuates during training, the overall trend is upward. At last, the accuracy and IoU of MutiStepLR is larger than StepLR. Thus, MutiStepLR is better than StepLR.

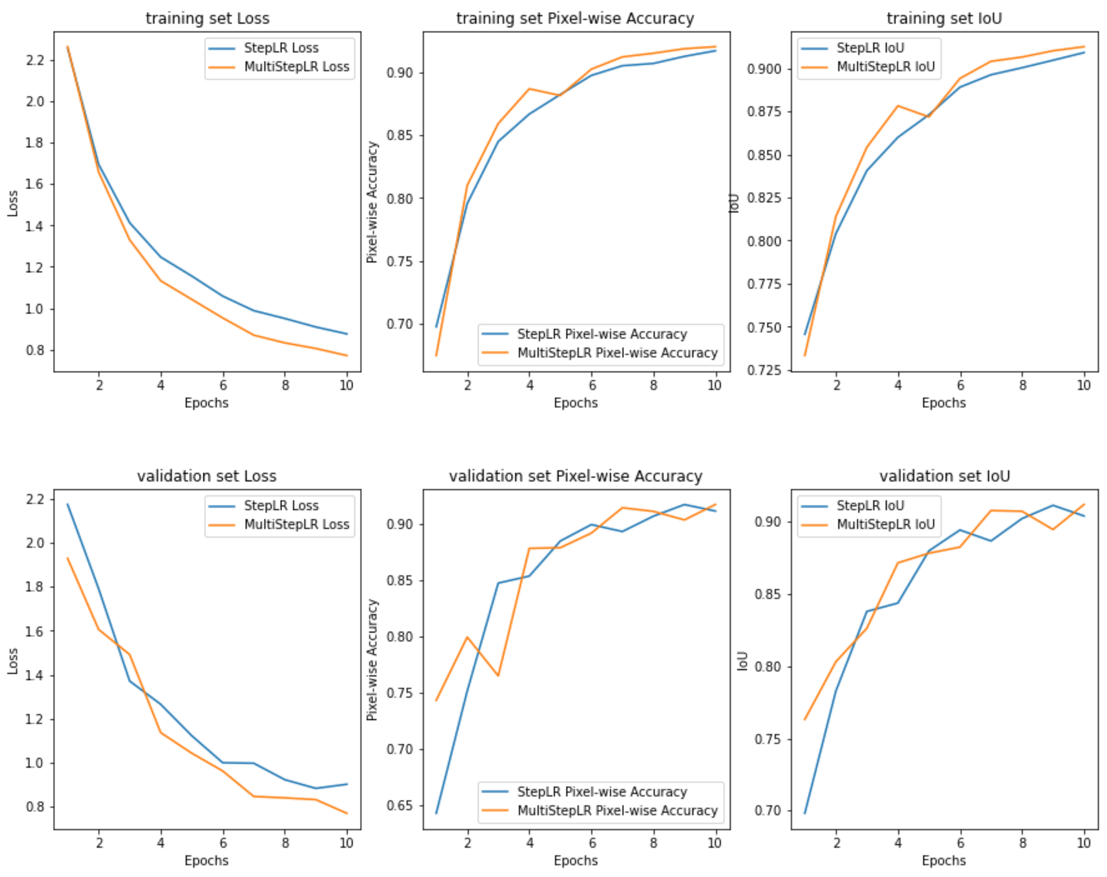
UNet (optimizers):





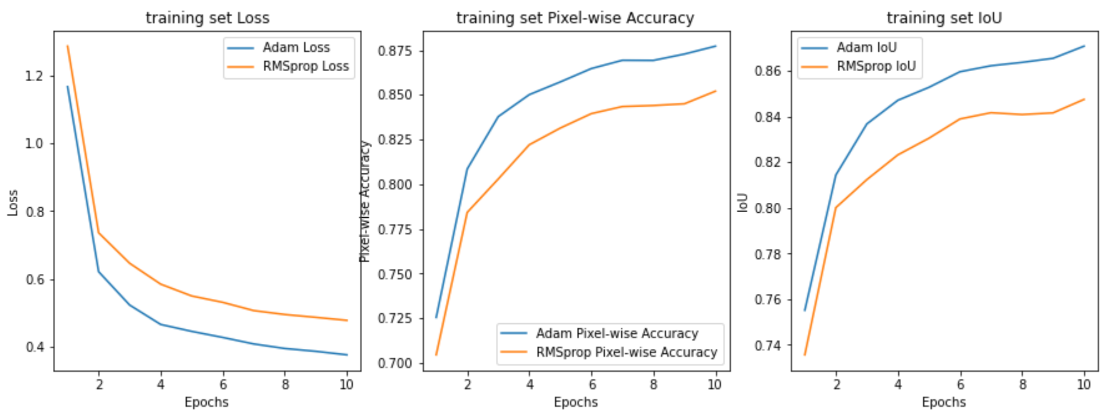
We find that the value of train loss function of both optimizers decrease steadily in the train process. The value of accuracy and IoU of both optimizers rising volatility. At last, the accuracy and IoU of Adam is larger than RMSprop. Thus, Adam is better than RMSprop.

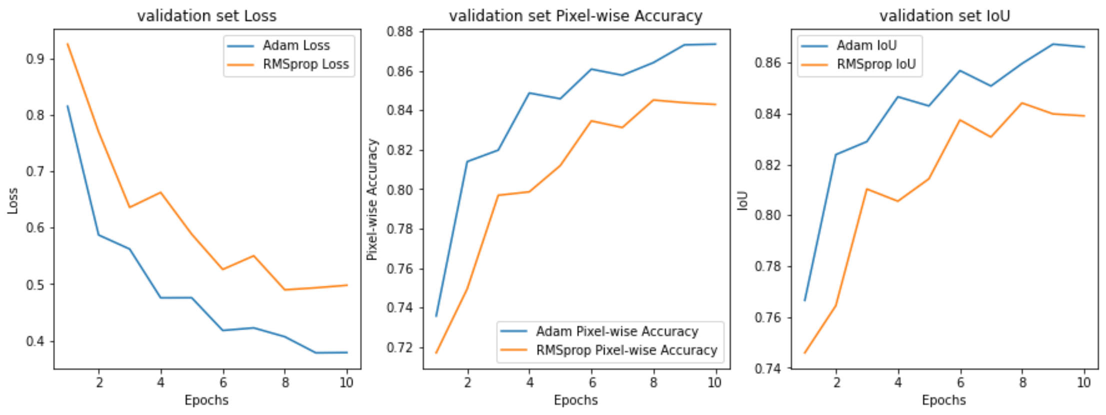
UNet (learning rate schedulers):



We find that the value of train and validation loss function of both learning rate schedulers decrease steadily in the train process. The accuracy and IoU of MutiStepLR is larger than StepLR. Thus, MutiStepLR is better than StepLR

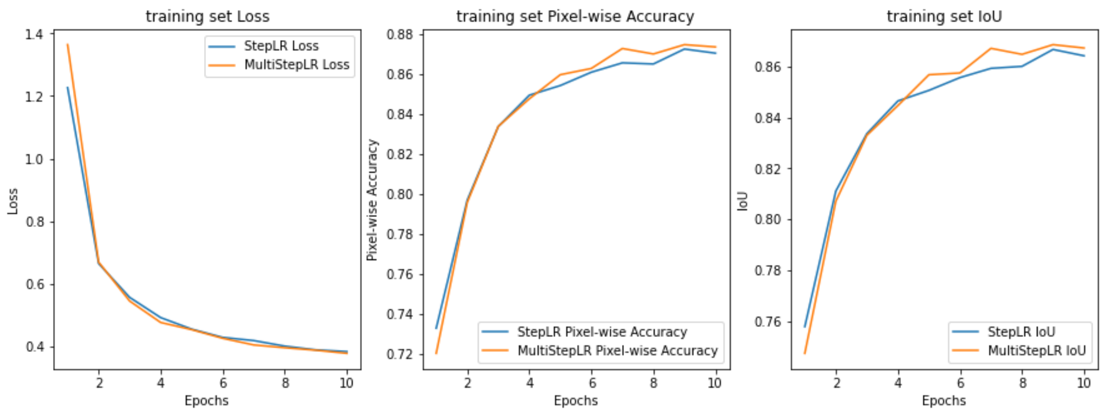
DeepLabV2 (optimizers):

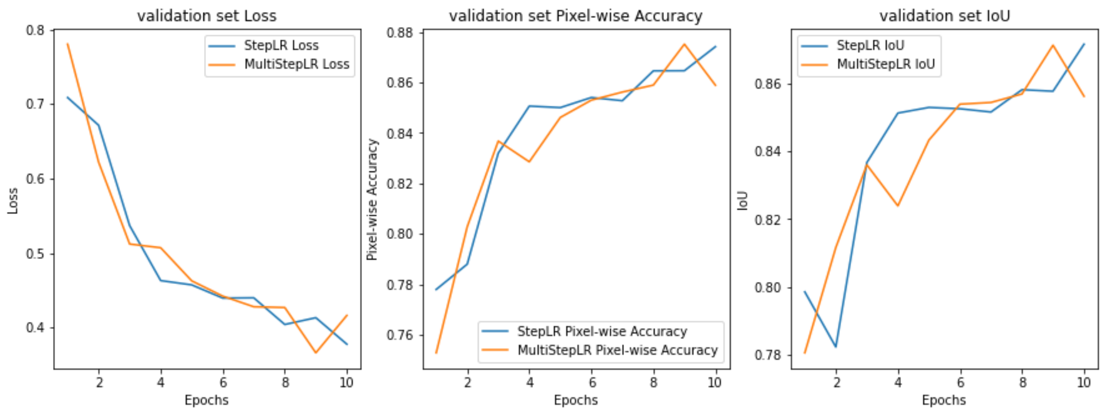




We find that the value of train loss function of both optimizers decrease steadily in the train process. The value of validation loss function fluctuates down at the same time. And the accuracy and IoU of Adam is larger than RMSprop. Thus, Adam is better than RMSprop.

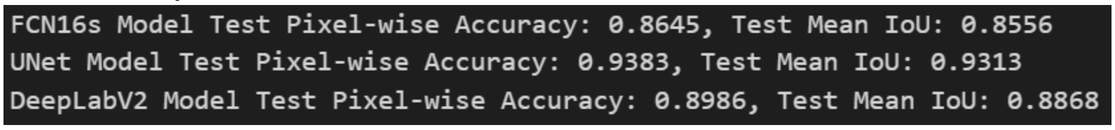
DeepLabV2 (learning rate schedulers):





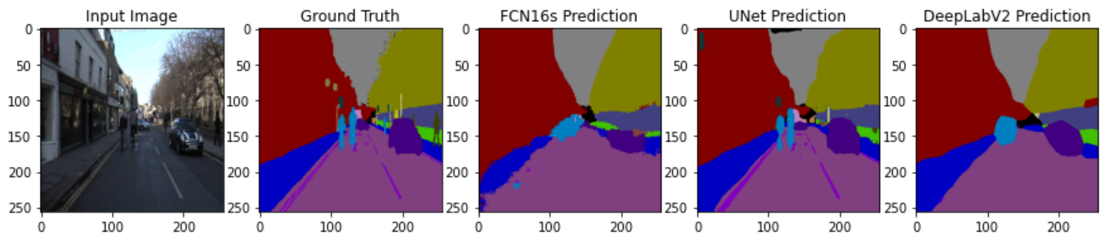
We find that the value of train loss function of both optimizers decrease steadily in the train process, but the value of validation loss function suddenly rise at 9th epoch, causing the overfitting issue. Overall, val_loss is slightly less than train_loss for both optimizers' models, which indicate that the training process is normal. The accuracy and IoU of MutiStepLR is larger than StepLR. Thus, MutiStepLR is better than StepLR.

Test accuracy and mean IoU



```
FCN16s Model Test Pixel-wise Accuracy: 0.8645, Test Mean IoU: 0.8556
UNet Model Test Pixel-wise Accuracy: 0.9383, Test Mean IoU: 0.9313
DeepLabV2 Model Test Pixel-wise Accuracy: 0.8986, Test Mean IoU: 0.8868
```

ii. Segmentation results of FCN-16s, UNet and DeepLabV2.



According to the output image above, we could clearly find that UNet model perform best in this project.

iii. Personal analysis for FCN-16s, UNet and DeepLabV2.

<u>**FCN-16s**</u>:

Pros:

FCN-16s uses a fully convolutional architecture, allowing it to handle images of any size.

It is computationally efficient and can produce segmentation results in real-time.

FCN-16s lacks the ability to handle finer details and edges in images due to its down-sampling and up-sampling process.

It requires a significant amount of training data to produce good results.

<u>**UNet:**</u>

Pros:
UNet has a unique architecture that allows it to capture both contextual information and fine-grained details in images.
It has shown good performance in medical image segmentation tasks and has been widely used in the field.
UNet can be trained on a small amount of data and still produce good results.
Cons:
UNet is computationally expensive and requires a lot of GPU memory during training.
It is not suitable for handling images of different sizes.

DeepLabV2:
Pros:
DeepLabV2 uses dilated convolutions to capture information at multiple scales, allowing it to handle images with fine details and edges.
It has shown good performance in various segmentation tasks and has been used in many real-world applications.
DeepLabV2 can be fine-tuned on small datasets to produce good results.
Cons:
DeepLabV2 is computationally expensive and requires a significant amount of training data to produce good results.
It is not suitable for handling images of different sizes.

In terms of comparison, FCN-16s is computationally efficient and can handle images of any size, but it lacks the ability to handle finer details and edges in images. UNet, on the other hand, can capture both contextual information and fine-grained details in images, but it is computationally expensive and not suitable for handling images of different sizes. Finally, DeepLabV2 uses dilated convolutions to handle images with fine details and edges, but it is also computationally expensive and requires a significant amount of training data to produce good results. Since the dataset in this project is small and the image are all the same size, therefore, UNet model perform best.

Task4: The final conclusions

i. Your own reflection on this project, for example, what you have learned via this project, which part you could do better next time, etc.

In this project, we learned how to build FCN-16s, UNet and DeepLabV2 model, using cross validation and data augmentation to explore the options of different hyperparameters. The most important part is that I learned how to improve the training process with small dataset. By applying different transformations to the original training dataset, we could create an almost infinite amount of new training samples. On the other hand, sometimes we found that different choice of hyperparameters has a great impact on the training and validation loss function. In order to train the model more efficiently, it is also important to understand the background of the hyperparameters.