1 Brief Description of Programs

1.1 split_data() function

The split_data() function in a user defined for splitting the training data into train,test, and validation data. These images are stored in data/images/train, data/images/test, and data/images/val folders respectively. It also splits the corresponding labels into label folders.

1.2 my_Kitti_Dataset class

This is the user defined Dataset class tohelp load data into the system. The norm parameter specifies 2 different techniques I tried, with and without normalization of input class, but they do not create much of a difference.

1.3 find_mean_std() function

Function used to find mean and standard deviation of Dataset when using normalization.

1.4 data_preproc() function

Function used to create dataloaders for train, test, validation sets. It uses normalization of data.

1.5 data_preproc_no_norm() function

Function used to create dataloaders for train, test, validation sets without using normalization

1.6 FCN32 class

The class used to implement FCN32 using pretrained ResNet-18. I added 100 padding to the first layer, and removed the last 2 layers. Added a average pool layer and a 1x1 kernel convolution layer. Finally, added a ConvTranspos2d Layer to bring back the size by x32. I used the wkentaro implementation of FCN32 which was linked in the homework pdf as a reference.

1.7 FCN16 class

The class used to implement FCN16 using pretrained ResNet-18. Most of the steps are the same as FCN32. Additionally, I took the output of the conv_4x and combined it with the upsampled features from the next average layer. I used the wkentaro implementation of FCN16 which was linked in the homework pdf as a reference.

1.8 label_accuracy_score() function

This function is used to calculate the accuracy and mean IOU, from the images and labels passed to it. This code is also from the wkentaro github link.

1.9 train() function

This is the main training function that is used to perform training using the FCN32 and FCN16 class models. I am also running the net on the validation set at every epoch to find if the model is overfitting. The main reference for this code is the last homework HW4.

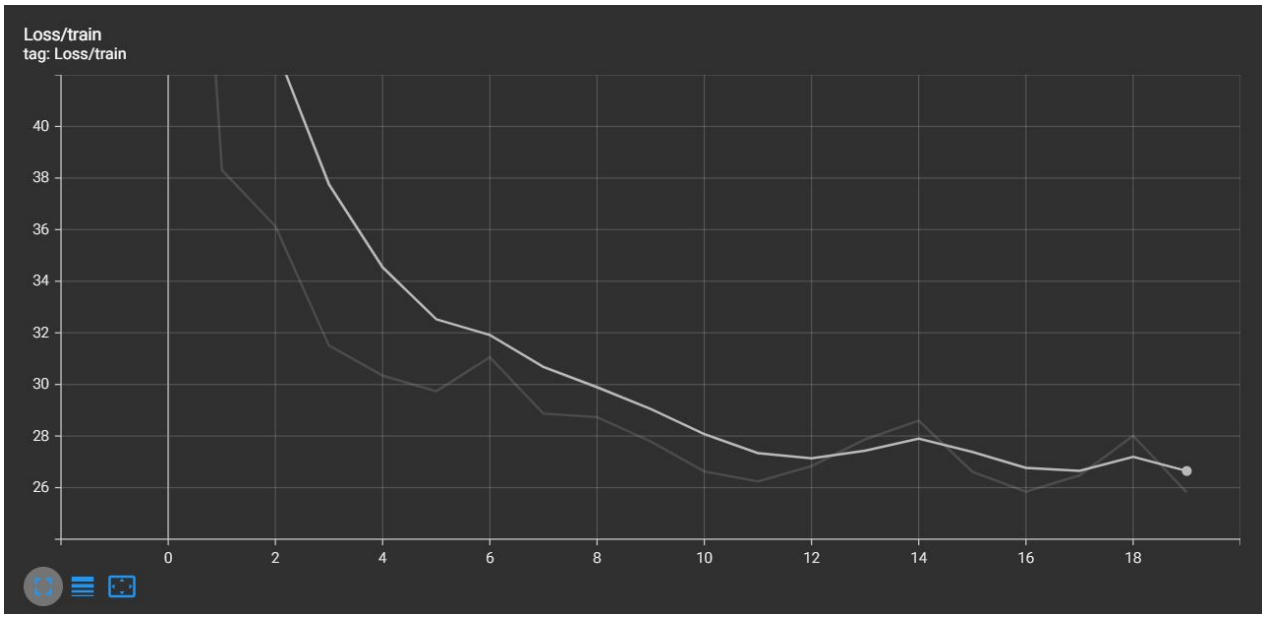
1.10 predict_an_image() function

This function is used to run the trained model on a particular image, by default the inp.png. It creates an image with pixels coloured in according to labels.py and saves it as out.png.

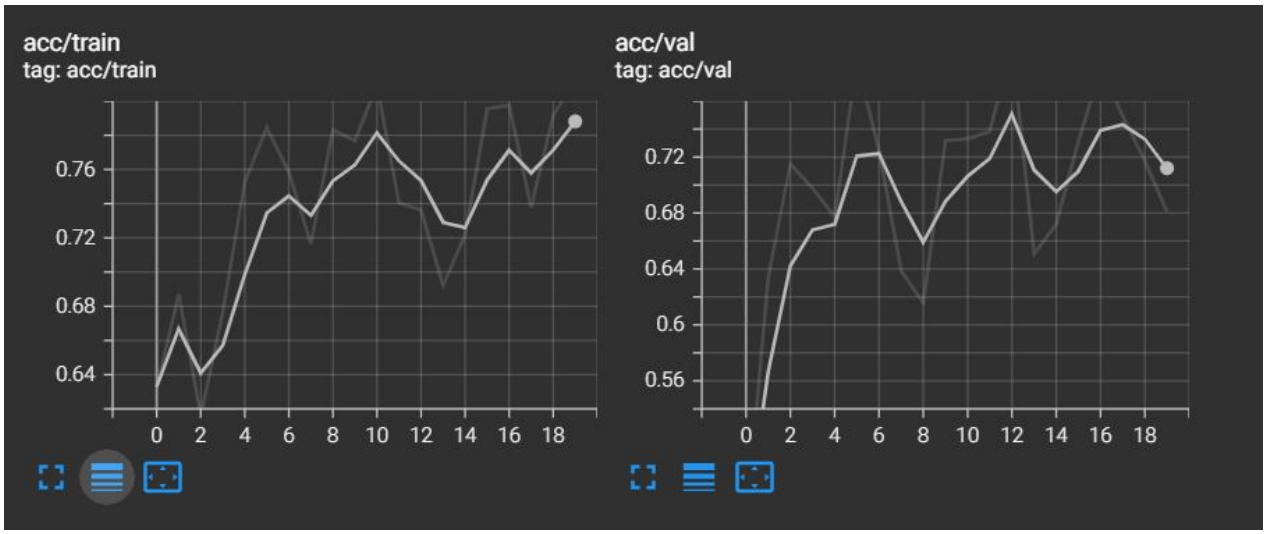
1.11 predict_test() function

Function to print out mean IOU and accuracy over a given dataloader.

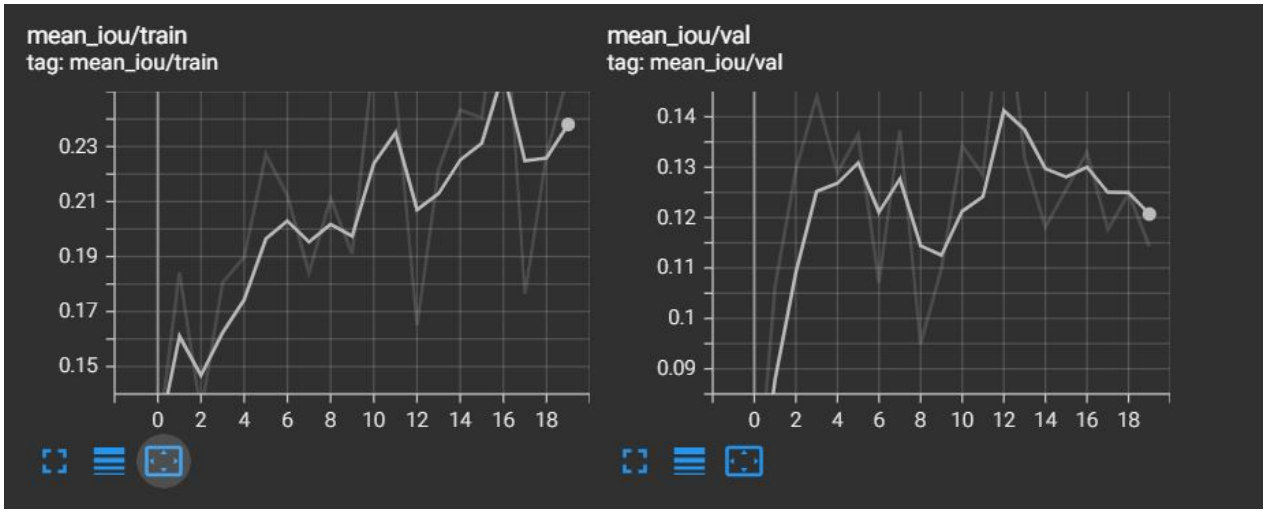
2 Evolution of Loss, Accuracy, and MIOU:



Training Loss over Epochs



Training and Validation Accuracy



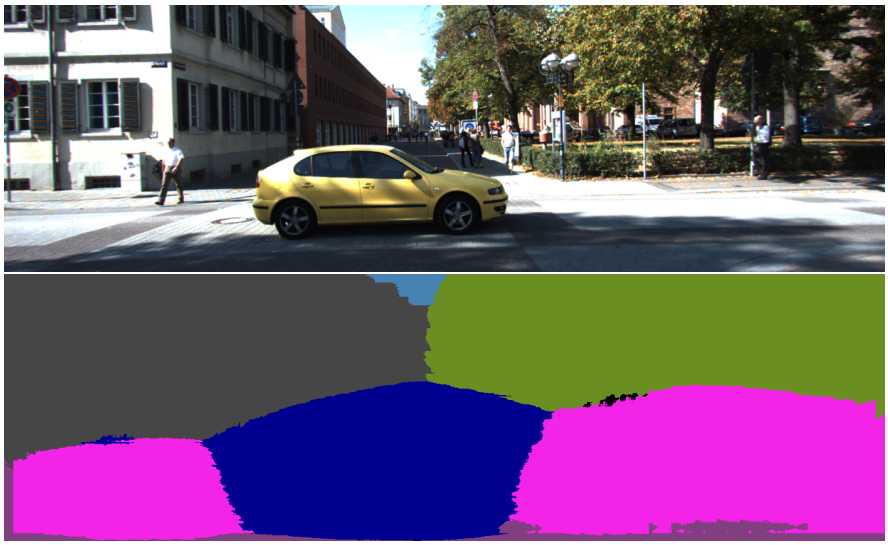
Training and Validation Mean IoU

3 Summary and Discussion:

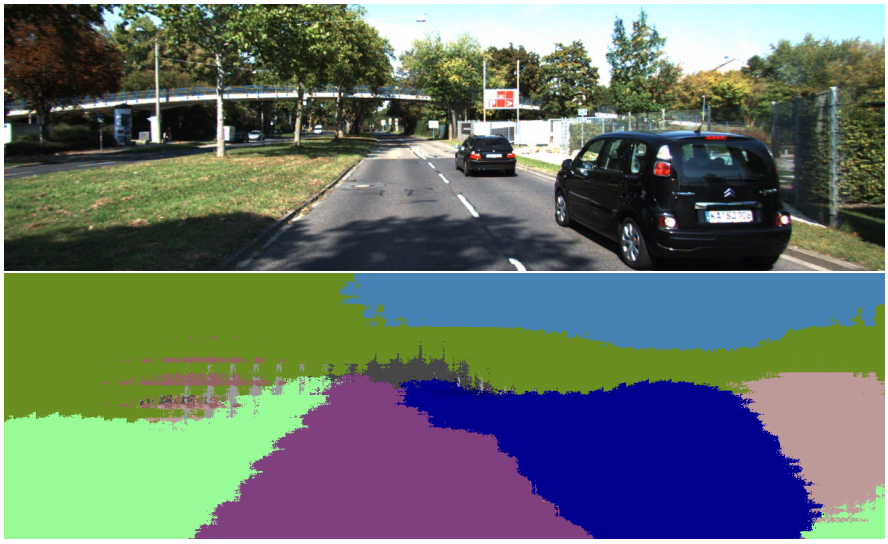
I achieved the following results for FCN32 and FCN16 models.

Model	Maximum Mean IOU	Validation Accuracy	Test Accuracy
FCN32	0.141	69.94	71.83
FCN16	0.211	76.00	79.30

3.1 FCN32 Examples of Classification:

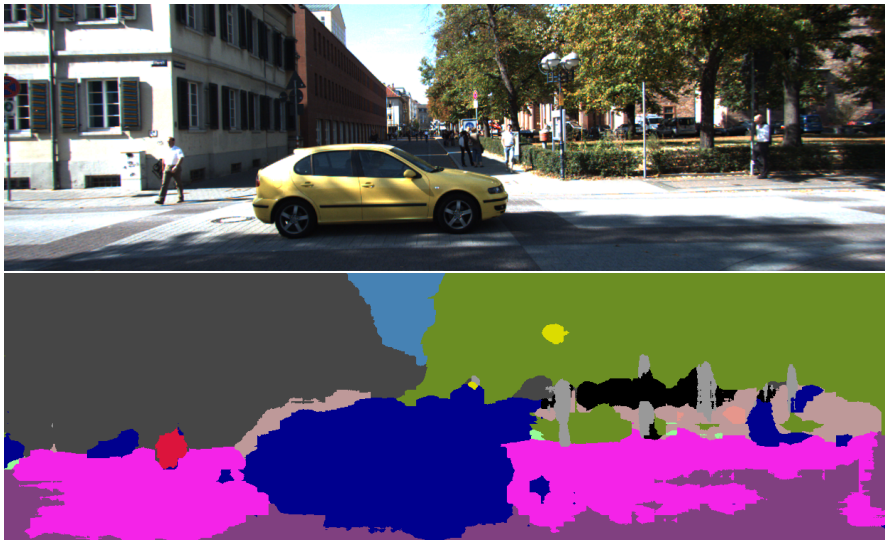


Example 1 of Prediction

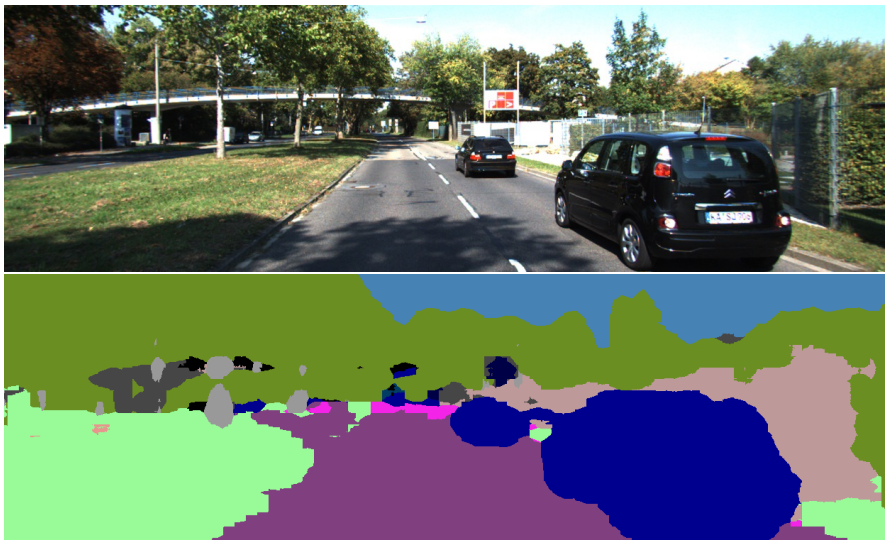


Example 2 of Prediction

3.2 FCN16 Examples of Classification:



Example 1 of Prediction



Example 2 of Prediction

3.3 Conclusion:

As we can see visually and quantitatively, the FCN16 model works better than the FCN32.