Introduction:

This project focuses on identifying drivable areas and detecting lanes on the road. Our main task involves adjusting the model using the dataset.

Data Preprocessing:

1. Random Perspective Transformation:
   This transformation simulates changes in the camera's perspective, including rotation, scaling, shearing, and translation. It is applied with random parameters:
   - **degrees:** Random rotation in the range of -10 to 10 degrees.
   - **translate:** Random translation in the range of -0.1 to 0.1 times the image dimensions.
   - **scale:** Random scaling in the range of 0.9 to 1.1 times the original size.
   - **shear:** Random shearing in the range of -10 to 10 degrees.
   - **perspective:** A slight random perspective distortion.
2. HSV Color Augmentation:
   - This changes the hue, saturation, and value of the image.
   - Random gains for hue, saturation, and value are applied.
   - The hue is modified by rotating the color wheel.
   - The saturation and value are adjusted by multiplying with random factors.
   - This helps to make the model invariant to changes in lighting and color variations.
3. Image Resizing:
   - If the Images are not in the specified size, the images are resized to a fixed size (640x360) using cv2.resize.
4. Label Preprocessing:
   - The labels (segmentation masks) are thresholded to create binary masks. This means that pixel values are set to 0 or 255 based on a threshold (usually 1 in this case).
   - The binary masks are also inverted to create a binary mask for the background.
   - These binary masks are converted to PyTorch tensors for use in training the semantic segmentation model.

Loading Pretrained Parameters:

The pretrained model is loaded using pytorch. The entire network is loaded with its pretrained weights.

```
[ ]  import TwinLite as net


     model = net.TwinLiteNet()
     model = torch.nn.DataParallel(model)
     model = model.cuda()
     model.load_state_dict(torch.load('best.pth'))
```

```
     <All keys matched successfully>
```

Model Fine-Tuning:

We have Fine-Tuned the pretrained model. We did not add any new layers to the model. We trained the whole model on our dataset.

Hyperparameters:

Learning rate = 5e-4

Epochs = 30

Weight decay = 5e-4

```
     from tqdm import tqdm
     from loss import TotalLoss

     lr = 5e-4
     optimizer = torch.optim.Adam(model.parameters(), lr, (0.9, 0.999), eps=1e-08, weight_decay=5e-4)

     criteria = TotalLoss()
```

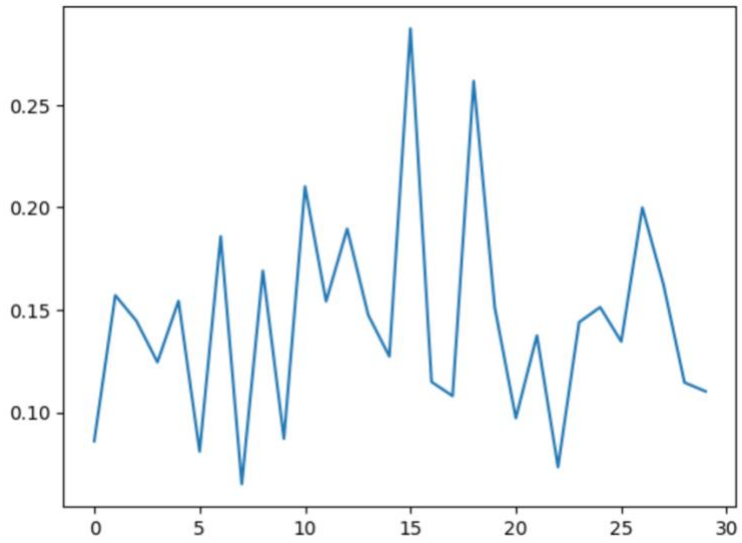
```
[ ]  args = dict()

     args["lr"] = lr
     args["max_epochs"] = 30
     args["onGPU"] = True
```

LOSS:

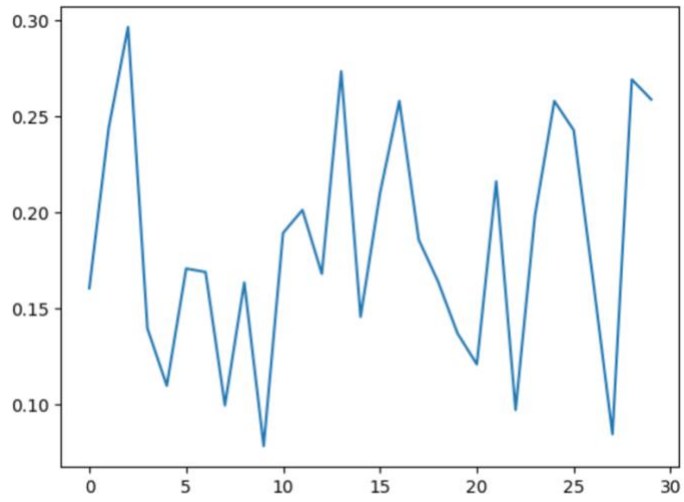
```
Training loss for last batch: 0.11452271789312363
Validation loss for last batch: 0.26914864778518677
----------------------------------------
Epoch: 30/30
Learning rate: 1e-08

        29/29      0.06289      0.04731       0.1102: 100%|██████████| 20/20 [00:05<00:00,  3.64it/s]

Average Training Loss: 0.13972755856812
Average Validation Loss: 0.18667576710383096

Training loss for last batch: 0.11020036041736603
Validation loss for last batch: 0.25874409079551697
----------------------------------------
```

Results:

Training Loss:



Validation Loss:



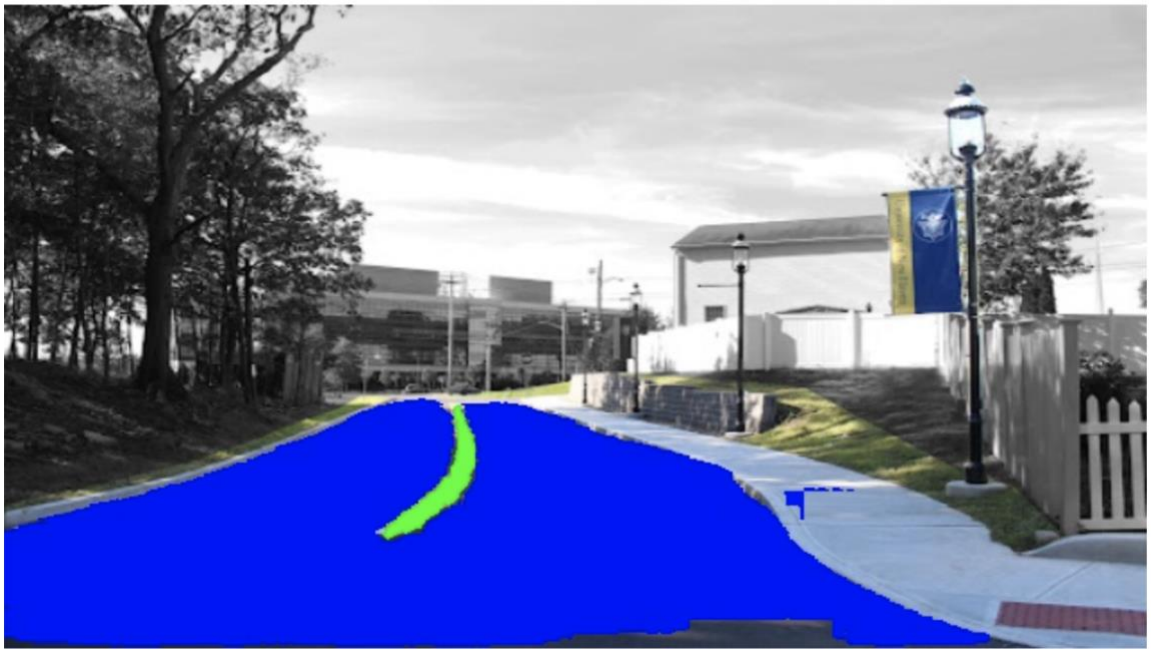
Metrics:

- Evaluation metrics are pixel accuracy and IoU(Intersection over Union).
- We have achieved an accuracy of 96.3% for Driving area segment.
- We have achieved an accuracy of 98.4% for Lane Line segment.

Input Image:

Output Image:

```
from PIL import Image


# Blue area is drivable area, green lines are lanes
Image.open("unh1.jpg")
```



Links: Click here to go the Google Colab where we worked on.