Download the Repository

Repository Link

- This is our team's repository. This repository contains all the necessary code that we worked on and it also contains the dataset that we annotated.
- You do not need to do anything like uploading and adjusting the paths. Just run the cells sequentially.
- All the necessary commands are written in this notebook itself

```
!git clone https://github.com/pijush2022/Lane_detection
```

```
fatal: destination path 'road-detection' already exists and is not an
empty directory.
```

Install the Requirements

- Install all the python dependencies
- After Installing dependencies, Restart the runtime. If you do not restart the runtime, the python will throw "module not found error"

```
!pip install -r road-detection/TwinLiteNet/requirements.txt
```

```
Requirement already satisfied: certifi==2023.7.22 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 1)) (2023.7.22)
Requirement already satisfied: charset-normalizer==3.3.2 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 2)) (3.3.2)
Requirement already satisfied: colorama==0.4.6 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 3)) (0.4.6)
Requirement already satisfied: contourpy==1.2.0 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 4)) (1.2.0)
Requirement already satisfied: cycler==0.12.1 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 5)) (0.12.1)
Requirement already satisfied: dnspython==2.4.2 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 6)) (2.4.2)
Requirement already satisfied: elephant==0.12.0 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 7)) (0.12.0)
Requirement already satisfied: filelock==3.13.1 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 8)) (3.13.1)
```

```
Requirement already satisfied: fonttools==4.44.0 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 9)) (4.44.0)
Requirement already satisfied: fsspec==2023.10.0 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 10)) (2023.10.0)
Requirement already satisfied: idna==3.4 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 11)) (3.4)
Requirement already satisfied: Jinja2==3.1.2 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 12)) (3.1.2)
Requirement already satisfied: joblib==1.2.0 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 13)) (1.2.0)
Requirement already satisfied: kiwisolver==1.4.5 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 14)) (1.4.5)
Requirement already satisfied: MarkupSafe==2.1.3 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 15)) (2.1.3)
Requirement already satisfied: matplotlib==3.7.1 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 16)) (3.7.1)
Requirement already satisfied: mpmath==1.3.0 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 17)) (1.3.0)
Requirement already satisfied: neo==0.12.0 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 18)) (0.12.0)
Requirement already satisfied: networkx==3.2.1 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 19)) (3.2.1)
Requirement already satisfied: numpy==1.24.3 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 20)) (1.24.3)
Requirement already satisfied: opencv-python==4.7.0.72 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 21)) (4.7.0.72)
Requirement already satisfied: packaging==23.2 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 22)) (23.2)
Requirement already satisfied: Pillow==9.5.0 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 23)) (9.5.0)
Requirement already satisfied: pyparsing==3.1.1 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 24)) (3.1.1)
Requirement already satisfied: python-dateutil==2.8.2 in
```

```
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 25)) (2.8.2)
Requirement already satisfied: python-etcd==0.4.5 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 26)) (0.4.5)
Requirement already satisfied: PyYAML==6.0.1 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 27)) (6.0.1)
Requirement already satisfied: quantities==0.14.1 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 28)) (0.14.1)
Requirement already satisfied: requests==2.31.0 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 29)) (2.31.0)
Requirement already satisfied: scikit-learn==1.3.2 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 30)) (1.3.2)
Requirement already satisfied: scipy==1.10.1 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 31)) (1.10.1)
Requirement already satisfied: six==1.16.0 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 32)) (1.16.0)
Requirement already satisfied: sympy==1.12 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 33)) (1.12)
Requirement already satisfied: threadpoolctl==3.2.0 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 34)) (3.2.0)
Requirement already satisfied: torch==2.1.0 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 35)) (2.1.0)
Requirement already satisfied: torchdata==0.7.0 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 36)) (0.7.0)
Requirement already satisfied: torchelastic==0.2.2 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 37)) (0.2.2)
Requirement already satisfied: torchtext==0.16.0 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 38)) (0.16.0)
Requirement already satisfied: torchvision==0.16.0 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 39)) (0.16.0)
Requirement already satisfied: tqdm==4.66.1 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 40)) (4.66.1)
Requirement already satisfied: typing_extensions==4.8.0 in
/usr/local/lib/python3.10/dist-packages (from -r
```

```
road-detection/TwinLiteNet/requirements.txt (line 41)) (4.8.0)
Requirement already satisfied: urllib3==2.0.7 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 42)) (2.0.7)
Requirement already satisfied: webcolors==1.13 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 43)) (1.13)
Requirement already satisfied: yacs==0.1.8 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 44)) (0.1.8)
Requirement already satisfied: zipp==3.15.0 in
/usr/local/lib/python3.10/dist-packages (from -r
road-detection/TwinLiteNet/requirements.txt (line 45)) (3.15.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.1.105 in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r road-
detection/TwinLiteNet/requirements.txt (line 35)) (12.1.105)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.1.105
in /usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r
road-detection/TwinLiteNet/requirements.txt (line 35)) (12.1.105)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.1.105 in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r road-
detection/TwinLiteNet/requirements.txt (line 35)) (12.1.105)
Requirement already satisfied: nvidia-cudnn-cu12==8.9.2.26 in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r road-
detection/TwinLiteNet/requirements.txt (line 35)) (8.9.2.26)
Requirement already satisfied: nvidia-cublas-cu12==12.1.3.1 in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r road-
detection/TwinLiteNet/requirements.txt (line 35)) (12.1.3.1)
Requirement already satisfied: nvidia-cufft-cu12==11.0.2.54 in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r road-
detection/TwinLiteNet/requirements.txt (line 35)) (11.0.2.54)
Requirement already satisfied: nvidia-curand-cu12==10.3.2.106 in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r road-
detection/TwinLiteNet/requirements.txt (line 35)) (10.3.2.106)
Requirement already satisfied: nvidia-cusolver-cu12==11.4.5.107 in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r road-
detection/TwinLiteNet/requirements.txt (line 35)) (11.4.5.107)
Requirement already satisfied: nvidia-cusparse-cu12==12.1.0.106 in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r road-
detection/TwinLiteNet/requirements.txt (line 35)) (12.1.0.106)
Requirement already satisfied: nvidia-nccl-cu12==2.18.1 in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r road-
detection/TwinLiteNet/requirements.txt (line 35)) (2.18.1)
Requirement already satisfied: nvidia-nvtx-cu12==12.1.105 in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r road-
detection/TwinLiteNet/requirements.txt (line 35)) (12.1.105)
Requirement already satisfied: triton==2.1.0 in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r road-
detection/TwinLiteNet/requirements.txt (line 35)) (2.1.0)
```

```
Requirement already satisfied: nvidia-nvjitlink-cu12 in
/usr/local/lib/python3.10/dist-packages (from nvidia-cusolver-
cu12==11.4.5.107->torch==2.1.0->-r
road-detection/TwinLiteNet/requirements.txt (line 35)) (12.4.127)
```

Copy Dataset from Repository

- Our repository contains dataset.zip in datasets folder in the repository. copy that zip file to root

```
!cp road-detection/datasets/dataset.zip ./
```

Unzip the file

```
!unzip dataset.zip

Archive:  dataset.zip
replace dataset/test/images/road_image_160.png? [y]es, [n]o, [A]ll,
[N]one, [r]ename: N
```

Import the all the required libraries

```
import torch
import cv2
import torch.utils.data
import torchvision.transforms as transforms
import numpy as np
import os
import random
import math
from matplotlib import pyplot as plt
import torch.nn as nn
```

Image transformation functions

- By paper author

```
def augment_hsv(img, hgain=0.015, sgain=0.7, vgain=0.4):
    """change color hue, saturation, value"""
    r = np.random.uniform(-1, 1, 3) * [hgain, sgain, vgain] + 1  #
random gains
    hue, sat, val = cv2.split(cv2.cvtColor(img, cv2.COLOR_BGR2HSV))
    dtype = img.dtype  # uint8

    x = np.arange(0, 256, dtype=np.int16)
    lut_hue = ((x * r[0]) % 180).astype(dtype)
    lut_sat = np.clip(x * r[1], 0, 255).astype(dtype)
    lut_val = np.clip(x * r[2], 0, 255).astype(dtype)

    img_hsv = cv2.merge((cv2.LUT(hue, lut_hue), cv2.LUT(sat, lut_sat),
```

```
cv2.LUT(val, lut_val))).astype(dtype)
    cv2.cvtColor(img_hsv, cv2.COLOR_HSV2BGR, dst=img)  # no return
needed

def random_perspective(combination,  degrees=10, translate=.1,
scale=.1, shear=10, perspective=0.0, border=(0, 0)):
    """combination of img transform"""
    # torchvision.transforms.RandomAffine(degrees=(-10, 10),
translate=(.1, .1), scale=(.9, 1.1), shear=(-10, 10))
    # targets = [cls, xyxy]
    img, gray, line = combination
    height = img.shape[0] + border[0] * 2  # shape(h,w,c)
    width = img.shape[1] + border[1] * 2

    # Center
    C = np.eye(3)
    C[0, 2] = -img.shape[1] / 2  # x translation (pixels)
    C[1, 2] = -img.shape[0] / 2  # y translation (pixels)

    # Perspective
    P = np.eye(3)
    P[2, 0] = random.uniform(-perspective, perspective)  # x
perspective (about y)
    P[2, 1] = random.uniform(-perspective, perspective)  # y
perspective (about x)

    # Rotation and Scale
    R = np.eye(3)
    a = random.uniform(-degrees, degrees)
    # a += random.choice([-180, -90, 0, 90])  # add 90deg rotations to
small rotations
    s = random.uniform(1 - scale, 1 + scale)
    # s = 2 ** random.uniform(-scale, scale)
    R[:2] = cv2.getRotationMatrix2D(angle=a, center=(0, 0), scale=s)

    # Shear
    S = np.eye(3)
    S[0, 1] = math.tan(random.uniform(-shear, shear) * math.pi / 180)
# x shear (deg)
    S[1, 0] = math.tan(random.uniform(-shear, shear) * math.pi / 180)
# y shear (deg)

    # Translation
    T = np.eye(3)
    T[0, 2] = random.uniform(0.5 - translate, 0.5 + translate) * width
# x translation (pixels)
    T[1, 2] = random.uniform(0.5 - translate, 0.5 + translate) *
height  # y translation (pixels)

    # Combined rotation matrix
```

```
    M = T @ S @ R @ P @ C  # order of operations (right to left) is
IMPORTANT
    if (border[0] != 0) or (border[1] != 0) or (M != np.eye(3)).any():
# image changed
        if perspective:
            img = cv2.warpPerspective(img, M, dsize=(width, height),
borderValue=(114, 114, 114))
            gray = cv2.warpPerspective(gray, M, dsize=(width, height),
borderValue=0)
            line = cv2.warpPerspective(line, M, dsize=(width, height),
borderValue=0)
        else:  # affine
            img = cv2.warpAffine(img, M[:2], dsize=(width, height),
borderValue=(114, 114, 114))
            gray = cv2.warpAffine(gray, M[:2], dsize=(width, height),
borderValue=0)
            line = cv2.warpAffine(line, M[:2], dsize=(width, height),
borderValue=0)



    combination = (img, gray, line)
    return combination

!mkdir custom-dataset
!mkdir custom-dataset/test
!mkdir custom-dataset/train
!mkdir custom-dataset/validation

!mkdir -p custom-dataset/train/images
!mkdir -p custom-dataset/train/lane
!mkdir -p custom-dataset/train/segments
!mkdir -p custom-dataset/test/images
!mkdir -p custom-dataset/test/lane
!mkdir -p custom-dataset/test/segments
!mkdir -p custom-dataset/validation/images
!mkdir -p custom-dataset/validation/lane
!mkdir -p custom-dataset/validation/segments

# Copy only two images from each category in train
!cp dataset/train/images/road_image_0.png custom-dataset/train/images/
!cp dataset/train/lane/road_image_0.png custom-dataset/train/lane/
!cp dataset/train/segments/road_image_0.png custom-
dataset/train/segments/

# Copy only two images from each category in train
!cp dataset/train/images/road_image_1.png custom-dataset/train/images/
!cp dataset/train/lane/road_image_1.png custom-dataset/train/lane/
!cp dataset/train/segments/road_image_1.png custom-
dataset/train/segments/
```

```
# Copy only two images from each category in test
!cp dataset/test/images/* custom-dataset/test/images/
!cp dataset/test/lane/* custom-dataset/test/lane/
!cp dataset/test/segments/* custom-dataset/test/segments/


# Copy only two images from each category in validation
!cp dataset/validation/images/* custom-dataset/validation/images/
!cp dataset/validation/lane/* custom-dataset/validation/lane/
!cp dataset/validation/segments/* custom-dataset/validation/segments/
```

```
mkdir: cannot create directory ‘custom-dataset’: File exists
mkdir: cannot create directory ‘custom-dataset/test’: File exists
mkdir: cannot create directory ‘custom-dataset/train’: File exists
mkdir: cannot create directory ‘custom-dataset/validation’: File
exists
```

Custom Dataset Class

- This custom dataset class is based on the dataset class written by the author but with slight modifications like path. we have adjusted the path according to the google colab.

```
class MyDataset(torch.utils.data.Dataset):
    '''
    Class to load the dataset
    '''
    def __init__(self, transform=None, valid=False, test=False):
        '''
        :param imList: image list (Note that these lists have been
processed and pickled using the loadData.py)
        :param labelList: label list (Note that these lists have been
processed and pickled using the loadData.py)
        :param transform: Type of transformation. SEe Transforms.py
for supported transformations
        '''

        self.transform = transform
        self.Tensor = transforms.ToTensor()
        self.valid=valid
        if valid:
            self.root='custom-dataset/validation/images'
            self.names=os.listdir(self.root)
        elif test:
            self.root='custom-dataset/test/images'
            self.names=os.listdir(self.root)
        else:
            self.root='custom-dataset/train/images/'
            self.names=os.listdir(self.root)
```

```
    def __len__(self):
        return len(self.names)

    def __getitem__(self, idx):
        '''

        :param idx: Index of the image file
        :return: returns the image and corresponding label file.
        '''
        W_=640
        H_=360
        image_name=os.path.join(self.root,self.names[idx])

        image = cv2.imread(image_name)
        original_image = cv2.imread(image_name)
        label1 =
cv2.imread(image_name.replace("images","segments").replace("jpg","png"
), 0)
        label2 =
cv2.imread(image_name.replace("images","lane").replace("jpg","png"),
0)
        if not self.valid:
            if random.random()<0.5:
                combination = (image, label1, label2)
                (image, label1, label2)= random_perspective(
                    combination=combination,
                    degrees=10,
                    translate=0.1,
                    scale=0.25,
                    shear=0.0
                )
            if random.random()<0.5:
                augment_hsv(image)
            if random.random() < 0.5:
                image = np.fliplr(image)
                label1 = np.fliplr(label1)
                label2 = np.fliplr(label2)

        label1 = cv2.resize(label1, (W_, H_))
        label2 = cv2.resize(label2, (W_, H_))
        image = cv2.resize(image, (W_, H_))

        _,seg_b1 = cv2.threshold(label1,1,255,cv2.THRESH_BINARY_INV)
        _,seg_b2 = cv2.threshold(label2,1,255,cv2.THRESH_BINARY_INV)
        _,seg1 = cv2.threshold(label1,1,255,cv2.THRESH_BINARY)
        _,seg2 = cv2.threshold(label2,1,255,cv2.THRESH_BINARY)

        seg1 = self.Tensor(seg1)
        seg2 = self.Tensor(seg2)
        seg_b1 = self.Tensor(seg_b1)
```

```
        seg_b2 = self.Tensor(seg_b2)
        seg_da = torch.stack((seg_b1[0], seg1[0]),0)
        seg_ll = torch.stack((seg_b2[0], seg2[0]),0)
        image = image[:, :, ::-1].transpose(2, 0, 1)
        image = np.ascontiguousarray(image)

        return original_image, image_name,torch.from_numpy(image),
(seg_da,seg_ll)
```

Intialize a dataloader

- Intialize a dataloader with batch size 8
- Intialize train, test, validation datasets.

```
from torch.utils.data import DataLoader

train_dataloader = DataLoader(MyDataset(), batch_size = 2, shuffle =
True)
test_dataloader = DataLoader(MyDataset(test=True), batch_size = 8,
shuffle = True)
val_dataloader = DataLoader(MyDataset(valid=True), batch_size = 8,
shuffle = True)
```

Take only two samples

```
_, _, input, target = next(iter(train_dataloader))
```

Copy necessary files from repository

```
# Copy pretrained model from repository to root
!cp road-detection/TwinLiteNet/pretrained/best.pth ./

# Copy pytorch Neural Net from repo to root
!cp road-detection/TwinLiteNet/model/TwinLite.py ./

# Copy Loss function pytorch code from repo to root
!cp road-detection/TwinLiteNet/loss.py ./

# Copy all reqired constants from repo to root
!cp road-detection/TwinLiteNet/const.py ./

# Copy all val.py from repo to root
!cp road-detection/TwinLiteNet/val.py ./
```

Mini Version of Original Network

```
import torch.nn as nn
import torch.nn.functional as F
```

```
class MiniNet(nn.Module):
    def __init__(self):
        super(MiniNet, self).__init__()

        # Input Convolution
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, stride=2,
padding=1)
        self.bn1 = nn.BatchNorm2d(16)
        self.relu1 = nn.ReLU()

        # Downsampling
        self.pool = nn.MaxPool2d(kernel_size=3, stride=2, padding=1)

        # Intermediate Convolutions
        self.conv2 = nn.Conv2d(16, 32, kernel_size=3, stride=1,
padding=1)
        self.bn2 = nn.BatchNorm2d(32)
        self.relu2 = nn.ReLU()

        self.conv3 = nn.Conv2d(32, 64, kernel_size=3, stride=1,
padding=1)
        self.bn3 = nn.BatchNorm2d(64)
        self.relu3 = nn.ReLU()

        # Upsampling
        self.upconv1 = nn.ConvTranspose2d(64, 32, kernel_size=4,
stride=2, padding=1)
        self.bn4 = nn.BatchNorm2d(32)
        self.relu4 = nn.ReLU()

        self.upconv2 = nn.ConvTranspose2d(32, 16, kernel_size=4,
stride=2, padding=1)
        self.bn5 = nn.BatchNorm2d(16)
        self.relu5 = nn.ReLU()

        # Classifiers
        self.classifier1 = nn.Conv2d(16, 2, kernel_size=1)
        self.classifier2 = nn.Conv2d(16, 2, kernel_size=1)

    def forward(self, x):
        # Input Convolution
        x = self.relu1(self.bn1(self.conv1(x)))

        # Downsampling
        x = self.pool(x)

        # Intermediate Convolutions
        x = self.relu2(self.bn2(self.conv2(x)))
        x = self.relu3(self.bn3(self.conv3(x)))
```

```
        # Upsampling
        x = self.relu4(self.bn4(self.upconv1(x)))
        x = self.relu5(self.bn5(self.upconv2(x)))

        # Classifiers
        classifier1 = self.classifier1(x)
        classifier2 = self.classifier2(x)

        return classifier1, classifier2

# Instantiate the model
model = MiniNet().to("cuda")
```

Defining Optimizer and loss for training

```
from loss import TotalLoss

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

Defining Custom metrics for evaluation

```
from tqdm import tqdm

class SegmentationMetric(object):
    '''
    imgLabel [batch_size, height(144), width(256)]
    confusionMatrix [[0(TN),1(FP)],
                     [2(FN),3(TP)]]
    '''
    def __init__(self, numClass):
        self.numClass = numClass
        self.confusionMatrix = np.zeros((self.numClass,)*2)

    def pixelAccuracy(self):
        # return all class overall pixel accuracy
        # acc = (TP + TN) / (TP + TN + FP + TN)
        acc = np.diag(self.confusionMatrix).sum() /
self.confusionMatrix.sum()
        return acc


    def classPixelAccuracy(self):
        # return each category pixel accuracy(A more accurate way to
call it precision)
        # acc = (TP) / TP + FP
        classAcc = np.diag(self.confusionMatrix) /
(self.confusionMatrix.sum(axis=0) + 1e-12)
```

```
        return classAcc

    def meanPixelAccuracy(self):
        classAcc = self.classPixelAccuracy()
        meanAcc = np.nanmean(classAcc)
        return meanAcc

    def meanIntersectionOverUnion(self):
        # Intersection = TP Union = TP + FP + FN
        # IoU = TP / (TP + FP + FN)
        intersection = np.diag(self.confusionMatrix)
        union = np.sum(self.confusionMatrix, axis=1) +
np.sum(self.confusionMatrix, axis=0) - np.diag(self.confusionMatrix)
        IoU = intersection / union
        IoU[np.isnan(IoU)] = 0
        mIoU = np.nanmean(IoU)
        return mIoU

    def IntersectionOverUnion(self):
        intersection = np.diag(self.confusionMatrix)
        union = np.sum(self.confusionMatrix, axis=1) +
np.sum(self.confusionMatrix, axis=0) - np.diag(self.confusionMatrix)
        IoU = intersection / union
        IoU[np.isnan(IoU)] = 0
        return IoU[1]

    def genConfusionMatrix(self, imgPredict, imgLabel):
        # remove classes from unlabeled pixels in gt image and predict
        # print(imgLabel.shape)
        mask = (imgLabel >= 0) & (imgLabel < self.numClass)
        label = self.numClass * imgLabel[mask] + imgPredict[mask]
        count = np.bincount(label, minlength=self.numClass**2)
        confusionMatrix = count.reshape(self.numClass, self.numClass)
        return confusionMatrix

    def Frequency_Weighted_Intersection_over_Union(self):
        # FWIOU =     [(TP+FN)/(TP+FP+TN+FN)] *[TP / (TP + FP + FN)]
        freq = np.sum(self.confusionMatrix, axis=1) /
np.sum(self.confusionMatrix)
        iu = np.diag(self.confusionMatrix) / (
                np.sum(self.confusionMatrix, axis=1) +
np.sum(self.confusionMatrix, axis=0) -
                np.diag(self.confusionMatrix))
        FWIoU = (freq[freq > 0] * iu[freq > 0]).sum()
        return FWIoU


    def addBatch(self, imgPredict, imgLabel):
        assert imgPredict.shape == imgLabel.shape
        self.confusionMatrix += self.genConfusionMatrix(imgPredict,
```

```
imgLabel)

    def reset(self):
        self.confusionMatrix = np.zeros((self.numClass,
self.numClass))

class AverageMeter(object):
    """Computes and stores the average and current value"""
    def __init__(self):
        self.reset()

    def reset(self):
        self.val = 0
        self.avg = 0
        self.sum = 0
        self.count = 0

    def update(self, val, n=1):
        self.val = val
        self.sum += val * n
        self.count += n
        self.avg = self.sum / self.count if self.count != 0 else 0

@torch.no_grad()
def val(val_loader, model):

    model.eval()


    DA=SegmentationMetric(2)
    LL=SegmentationMetric(2)

    da_acc_seg = AverageMeter()
    da_IoU_seg = AverageMeter()
    da_mIoU_seg = AverageMeter()

    ll_acc_seg = AverageMeter()
    ll_IoU_seg = AverageMeter()
    ll_mIoU_seg = AverageMeter()
    total_batches = len(val_loader)

    total_batches = len(val_loader)
    pbar = enumerate(val_loader)
    pbar = tqdm(pbar, total=total_batches)
    for i, (_, _,input, target) in pbar:
        input = input.cuda().float() / 255.0
            # target = target.cuda()

        input_var = input
        target_var = target
```

```
        # run the mdoel
        with torch.no_grad():
            output = model(input_var)

        out_da,out_ll=output
        target_da,target_ll=target

        _,da_predict=torch.max(out_da, 1)
        _,da_gt=torch.max(target_da, 1)

        _,ll_predict=torch.max(out_ll, 1)
        _,ll_gt=torch.max(target_ll, 1)
        DA.reset()
        DA.addBatch(da_predict.cpu(), da_gt.cpu())


        da_acc = DA.pixelAccuracy()
        da_IoU = DA.IntersectionOverUnion()
        da_mIoU = DA.meanIntersectionOverUnion()

        da_acc_seg.update(da_acc,input.size(0))
        da_IoU_seg.update(da_IoU,input.size(0))
        da_mIoU_seg.update(da_mIoU,input.size(0))


        LL.reset()
        LL.addBatch(ll_predict.cpu(), ll_gt.cpu())


        ll_acc = LL.pixelAccuracy()
        ll_IoU = LL.IntersectionOverUnion()
        ll_mIoU = LL.meanIntersectionOverUnion()

        ll_acc_seg.update(ll_acc,input.size(0))
        ll_IoU_seg.update(ll_IoU,input.size(0))
        ll_mIoU_seg.update(ll_mIoU,input.size(0))

    da_segment_result =
(da_acc_seg.avg,da_IoU_seg.avg,da_mIoU_seg.avg)
    ll_segment_result =
(ll_acc_seg.avg,ll_IoU_seg.avg,ll_mIoU_seg.avg)
    return da_segment_result,ll_segment_result
```

Training The model

```
val_loss = []
train_loss = []

for i in list(range(1000)):
```

```
    model.train()
    model.to("cuda")
    output = model(input.cuda().float() / 255.0)

    # target=target.cuda()
    # print(target[0].size())
    optimizer.zero_grad()
    focal_loss, tversky_loss, loss = criterion(output,target)
    optimizer.zero_grad()
    train_loss.append(loss.item())
    # print(output.size())
    loss.backward()
    optimizer.step()
    if i % 50 == 0:
        print("loss: {loss:.5f}".format(loss = loss.item()))
        model.eval()
        example = torch.rand(1, 3, 360, 640).cuda()
        model = torch.jit.trace(model, example)
        print("Accuracy:")
        da_segment_results,ll_segment_results = val(train_dataloader,
model)

        msg =  'Driving area Segment: Acc({da_seg_acc:.3f})\n' \
                            'Lane line Segment:
Acc({ll_seg_acc:.3f})'.format(
                                da_seg_acc=da_segment_results[0],
                                ll_seg_acc=ll_segment_results[0])
        print(msg)
        print()
        print("Validation Evaluation:")
        da_segment_results,ll_segment_results = val(val_dataloader,
model)

        msg =  'Driving area Segment: Acc({da_seg_acc:.3f})    IOU
({da_seg_iou:.3f})    mIOU({da_seg_miou:.3f})\n' \
                            'Lane line Segment: Acc({ll_seg_acc:.3f})
IOU ({ll_seg_iou:.3f})  mIOU({ll_seg_miou:.3f})'.format(

da_seg_acc=da_segment_results[0],da_seg_iou=da_segment_results[1],da_s
eg_miou=da_segment_results[2],

ll_seg_acc=ll_segment_results[0],ll_seg_iou=ll_segment_results[1],ll_s
eg_miou=ll_segment_results[2])
        print(msg)

print("----------------------------------------------------------------
-")
```

```
loss: 1.21158
Accuracy:
```

```
100%|██████████| 1/1 [00:01<00:00,  1.36s/it]

Driving area Segment: Acc(0.856)
Lane line Segment: Acc(0.993)

Validation Evaluation:

100%|██████████| 3/3 [00:03<00:00,  1.12s/it]

Driving area Segment: Acc(0.798)    IOU (0.000)    mIOU(0.399)
Lane line Segment: Acc(0.981)    IOU (0.000)  mIOU(0.491)
------------------------------------------------------------

/usr/local/lib/python3.10/dist-packages/torch/jit/_trace.py:787:
UserWarning: The input to trace is already a ScriptModule, tracing it
is a no-op. Returning the object as is.
  warnings.warn(

loss: 0.91062
Accuracy:

100%|██████████| 1/1 [00:00<00:00, 11.03it/s]

Driving area Segment: Acc(0.856)
Lane line Segment: Acc(0.993)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  2.50it/s]

Driving area Segment: Acc(0.798)    IOU (0.000)    mIOU(0.399)
Lane line Segment: Acc(0.981)    IOU (0.000)  mIOU(0.491)
------------------------------------------------------------
loss: 0.70817
Accuracy:

100%|██████████| 1/1 [00:00<00:00,  9.49it/s]

Driving area Segment: Acc(0.858)
Lane line Segment: Acc(0.993)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  2.50it/s]

Driving area Segment: Acc(0.798)    IOU (0.000)    mIOU(0.399)
Lane line Segment: Acc(0.981)    IOU (0.000)  mIOU(0.491)
------------------------------------------------------------
loss: 0.51835
Accuracy:

100%|██████████| 1/1 [00:00<00:00,  9.17it/s]
```

```
Driving area Segment: Acc(0.681)
Lane line Segment: Acc(0.995)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  1.88it/s]

Driving area Segment: Acc(0.868)    IOU (0.469)    mIOU(0.659)
Lane line Segment: Acc(0.981)    IOU (0.000)  mIOU(0.491)
------------------------------------------------------------
loss: 0.28514
Accuracy:

100%|██████████| 1/1 [00:00<00:00,  6.28it/s]

Driving area Segment: Acc(0.753)
Lane line Segment: Acc(0.996)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  2.51it/s]

Driving area Segment: Acc(0.839)    IOU (0.414)    mIOU(0.616)
Lane line Segment: Acc(0.975)    IOU (0.016)  mIOU(0.495)
------------------------------------------------------------
loss: 0.12501
Accuracy:

100%|██████████| 1/1 [00:00<00:00, 10.71it/s]

Driving area Segment: Acc(0.954)
Lane line Segment: Acc(0.996)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  2.50it/s]

Driving area Segment: Acc(0.848)    IOU (0.439)    mIOU(0.633)
Lane line Segment: Acc(0.972)    IOU (0.018)  mIOU(0.495)
------------------------------------------------------------
loss: 0.10251
Accuracy:

100%|██████████| 1/1 [00:00<00:00, 10.26it/s]

Driving area Segment: Acc(0.901)
Lane line Segment: Acc(0.994)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  2.43it/s]
```

```
Driving area Segment: Acc(0.848)    IOU (0.440)    mIOU(0.633)
Lane line Segment: Acc(0.970)    IOU (0.018)  mIOU(0.494)
------------------------------------------------------------
loss: 0.08780
Accuracy:

100%|██████████| 1/1 [00:00<00:00, 10.35it/s]

Driving area Segment: Acc(0.862)
Lane line Segment: Acc(0.996)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  2.60it/s]

Driving area Segment: Acc(0.846)    IOU (0.431)    mIOU(0.629)
Lane line Segment: Acc(0.968)    IOU (0.016)  mIOU(0.492)
------------------------------------------------------------
loss: 0.07377
Accuracy:

100%|██████████| 1/1 [00:00<00:00,  6.60it/s]

Driving area Segment: Acc(0.970)
Lane line Segment: Acc(0.996)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  1.71it/s]

Driving area Segment: Acc(0.846)    IOU (0.414)    mIOU(0.621)
Lane line Segment: Acc(0.963)    IOU (0.013)  mIOU(0.488)
------------------------------------------------------------
loss: 0.06286
Accuracy:

100%|██████████| 1/1 [00:00<00:00,  6.28it/s]

Driving area Segment: Acc(0.955)
Lane line Segment: Acc(0.998)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  1.72it/s]

Driving area Segment: Acc(0.849)    IOU (0.396)    mIOU(0.614)
Lane line Segment: Acc(0.961)    IOU (0.018)  mIOU(0.490)
------------------------------------------------------------
loss: 0.05484
Accuracy:

100%|██████████| 1/1 [00:00<00:00, 10.62it/s]
```

```
Driving area Segment: Acc(0.973)
Lane line Segment: Acc(0.997)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  2.49it/s]

Driving area Segment: Acc(0.855)    IOU (0.390)    mIOU(0.615)
Lane line Segment: Acc(0.959)    IOU (0.017)  mIOU(0.488)
------------------------------------------------------------
loss: 0.04588
Accuracy:

100%|██████████| 1/1 [00:00<00:00,  3.45it/s]

Driving area Segment: Acc(0.913)
Lane line Segment: Acc(0.995)

Validation Evaluation:

100%|██████████| 3/3 [00:02<00:00,  1.25it/s]

Driving area Segment: Acc(0.848)    IOU (0.357)    mIOU(0.596)
Lane line Segment: Acc(0.961)    IOU (0.016)  mIOU(0.488)
------------------------------------------------------------
loss: 0.03685
Accuracy:

100%|██████████| 1/1 [00:00<00:00,  5.11it/s]

Driving area Segment: Acc(0.955)
Lane line Segment: Acc(0.996)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  2.33it/s]

Driving area Segment: Acc(0.855)    IOU (0.368)    mIOU(0.605)
Lane line Segment: Acc(0.956)    IOU (0.020)  mIOU(0.488)
------------------------------------------------------------
loss: 0.03707
Accuracy:

100%|██████████| 1/1 [00:00<00:00,  9.27it/s]

Driving area Segment: Acc(0.944)
Lane line Segment: Acc(0.996)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  2.41it/s]
```

```
Driving area Segment: Acc(0.860)    IOU (0.407)    mIOU(0.626)
Lane line Segment: Acc(0.952)    IOU (0.016)  mIOU(0.484)
------------------------------------------------------------
loss: 0.03035
Accuracy:

100%|██████████| 1/1 [00:00<00:00, 10.26it/s]

Driving area Segment: Acc(0.960)
Lane line Segment: Acc(0.998)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  2.46it/s]

Driving area Segment: Acc(0.849)    IOU (0.342)    mIOU(0.589)
Lane line Segment: Acc(0.957)    IOU (0.018)  mIOU(0.488)
------------------------------------------------------------
loss: 0.02830
Accuracy:

100%|██████████| 1/1 [00:00<00:00, 10.40it/s]

Driving area Segment: Acc(0.942)
Lane line Segment: Acc(0.997)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  2.47it/s]

Driving area Segment: Acc(0.850)    IOU (0.354)    mIOU(0.595)
Lane line Segment: Acc(0.957)    IOU (0.018)  mIOU(0.487)
------------------------------------------------------------
loss: 0.02667
Accuracy:

100%|██████████| 1/1 [00:00<00:00,  6.09it/s]

Driving area Segment: Acc(0.949)
Lane line Segment: Acc(0.987)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  1.73it/s]

Driving area Segment: Acc(0.850)    IOU (0.343)    mIOU(0.590)
Lane line Segment: Acc(0.957)    IOU (0.020)  mIOU(0.488)
------------------------------------------------------------
loss: 0.02541
Accuracy:

100%|██████████| 1/1 [00:00<00:00, 11.25it/s]
```

```
Driving area Segment: Acc(0.955)
Lane line Segment: Acc(0.998)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  2.63it/s]

Driving area Segment: Acc(0.849)    IOU (0.321)    mIOU(0.578)
Lane line Segment: Acc(0.958)    IOU (0.021)  mIOU(0.490)
------------------------------------------------------------------
loss: 0.02420
Accuracy:

100%|██████████| 1/1 [00:00<00:00,  9.43it/s]

Driving area Segment: Acc(0.953)
Lane line Segment: Acc(0.997)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  2.62it/s]

Driving area Segment: Acc(0.848)    IOU (0.336)    mIOU(0.586)
Lane line Segment: Acc(0.961)    IOU (0.021)  mIOU(0.491)
------------------------------------------------------------------
loss: 0.02297
Accuracy:

100%|██████████| 1/1 [00:00<00:00,  8.45it/s]

Driving area Segment: Acc(0.988)
Lane line Segment: Acc(0.998)

Validation Evaluation:

100%|██████████| 3/3 [00:01<00:00,  2.61it/s]

Driving area Segment: Acc(0.851)    IOU (0.354)    mIOU(0.596)
Lane line Segment: Acc(0.958)    IOU (0.020)  mIOU(0.489)
------------------------------------------------------------------
```

Evaluating the model on the test set

```
print("Test Evaluation:")
da_segment_results,ll_segment_results = val(test_dataloader, model)

msg =  '\nDriving area Segment: Acc({da_seg_acc:.3f})    IOU
({da_seg_iou:.3f})    mIOU({da_seg_miou:.3f})\n' \
                    'Lane line Segment: Acc({ll_seg_acc:.3f})    IOU
({ll_seg_iou:.3f})  mIOU({ll_seg_miou:.3f})'.format(

da_seg_acc=da_segment_results[0],da_seg_iou=da_segment_results[1],da_s
```

```
eg_miou=da_segment_results[2],

ll_seg_acc=ll_segment_results[0],ll_seg_iou=ll_segment_results[1],ll_s
eg_miou=ll_segment_results[2])
print(msg)
print("-----------------------------------------------------------------
-")
```

```
Test Evaluation:

100%|██████████| 3/3 [00:01<00:00,  2.43it/s]


Driving area Segment: Acc(0.834)    IOU (0.226)    mIOU(0.526)
Lane line Segment: Acc(0.976)    IOU (0.024)  mIOU(0.500)
-----------------------------------------------------------------
```