Clone Repository

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

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/requ
Requirement already satisfied: charset-normalizer==3.3.2 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteN
Requirement already satisfied: colorama==0.4.6 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/require
Requirement already satisfied: contourpy==1.2.0 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requir
Requirement already satisfied: cycler==0.12.1 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requirem
Requirement already satisfied: dnspython==2.4.2 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requir
Requirement already satisfied: elephant==0.12.0 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requir
Requirement already satisfied: filelock==3.13.1 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requir
Requirement already satisfied: fonttools==4.44.0 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requi
Requirement already satisfied: fsspec==2023.10.0 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requi
Requirement already satisfied: idna==3.4 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requirements.
Requirement already satisfied: Jinja2==3.1.2 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requireme
Requirement already satisfied: joblib==1.2.0 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requireme
Requirement already satisfied: kiwisolver==1.4.5 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requi
Requirement already satisfied: MarkupSafe==2.1.3 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requi
Requirement already satisfied: matplotlib==3.7.1 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requi
Requirement already satisfied: mpmath==1.3.0 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requireme
Requirement already satisfied: neo==0.12.0 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requirement
Requirement already satisfied: networkx==3.2.1 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/require
Requirement already satisfied: numpy==1.24.3 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requireme
Requirement already satisfied: opencv-python==4.7.0.72 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet
Requirement already satisfied: packaging==23.2 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/require
Requirement already satisfied: Pillow==9.5.0 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requireme
Requirement already satisfied: pyparsing==3.1.1 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requir
Requirement already satisfied: python-dateutil==2.8.2 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/
Requirement already satisfied: python-etcd==0.4.5 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requ
Requirement already satisfied: PyYAML==6.0.1 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requireme
Requirement already satisfied: quantities==0.14.1 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requ
Requirement already satisfied: requests==2.31.0 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requir
Requirement already satisfied: scikit-learn==1.3.2 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/req
Requirement already satisfied: scipy==1.10.1 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requireme
Requirement already satisfied: six==1.16.0 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requirement
Requirement already satisfied: sympy==1.12 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requirement
Requirement already satisfied: threadpoolctl==3.2.0 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/re
Requirement already satisfied: torch==2.1.0 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requiremen
Requirement already satisfied: torchdata==0.7.0 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requir
Requirement already satisfied: torchelastic==0.2.2 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/req
Requirement already satisfied: torchtext==0.16.0 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requi
Requirement already satisfied: torchvision==0.16.0 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/req
Requirement already satisfied: tqdm==4.66.1 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requiremen
Requirement already satisfied: typing_extensions==4.8.0 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNe
Requirement already satisfied: urllib3==2.0.7 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requirem
Requirement already satisfied: webcolors==1.13 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/require
Requirement already satisfied: yacs==0.1.8 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requirement
Requirement already satisfied: zipp==3.15.0 in /usr/local/lib/python3.10/dist-packages (from -r road-detection/TwinLiteNet/requiremen
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r roa
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r r
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r roa
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-det
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-de
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-de
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-
Requirement already satisfied: nvidia-cusolver-cu12==11.4.5.107 in /usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r roa
Requirement already satisfied: nvidia-cusparse-cu12==12.1.0.106 in /usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r roa
Requirement already satisfied: nvidia-nccl-cu12==2.18.1 in /usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r road-detect
Requirement already satisfied: nvidia-nvtx-cu12==12.1.105 in /usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r road-dete
Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dist-packages (from torch==2.1.0->-r road-detection/TwinLit
```

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
   creating: dataset/test/
   creating: dataset/test/images/
  inflating: dataset/test/images/road_image_160.png
  inflating: dataset/test/images/road_image_161.png
  inflating: dataset/test/images/road_image_162.png
  inflating: dataset/test/images/road_image_163.png
  inflating: dataset/test/images/road_image_164.png
  inflating: dataset/test/images/road_image_165.png
  inflating: dataset/test/images/road_image_166.png
  inflating: dataset/test/images/road_image_167.png
  inflating: dataset/test/images/road_image_168.png
  inflating: dataset/test/images/road_image_169.png
  inflating: dataset/test/images/road_image_170.png
  inflating: dataset/test/images/road_image_171.png
  inflating: dataset/test/images/road_image_172.png
  inflating: dataset/test/images/road_image_173.png
  inflating: dataset/test/images/road_image_174.png
  inflating: dataset/test/images/road_image_175.png
  inflating: dataset/test/images/road_image_176.png
  inflating: dataset/test/images/road_image_177.png
  inflating: dataset/test/images/road_image_178.png
  inflating: dataset/test/images/road_image_179.png
   creating: dataset/test/lane/
  inflating: dataset/test/lane/road_image_160.png
  inflating: dataset/test/lane/road_image_161.png
  inflating: dataset/test/lane/road_image_162.png
  inflating: dataset/test/lane/road_image_163.png
  inflating: dataset/test/lane/road_image_164.png
  inflating: dataset/test/lane/road_image_165.png
  inflating: dataset/test/lane/road_image_166.png
  inflating: dataset/test/lane/road_image_167.png
  inflating: dataset/test/lane/road_image_168.png
  inflating: dataset/test/lane/road_image_169.png
  inflating: dataset/test/lane/road_image_170.png
  inflating: dataset/test/lane/road_image_171.png
  inflating: dataset/test/lane/road_image_172.png
  inflating: dataset/test/lane/road_image_173.png
  inflating: dataset/test/lane/road_image_174.png
  inflating: dataset/test/lane/road_image_175.png
  inflating: dataset/test/lane/road_image_176.png
  inflating: dataset/test/lane/road_image_177.png
  inflating: dataset/test/lane/road_image_178.png
  inflating: dataset/test/lane/road_image_179.png
   creating: dataset/test/segments/
  inflating: dataset/test/segments/road_image_160.png
  inflating: dataset/test/segments/road_image_161.png
  inflating: dataset/test/segments/road_image_162.png
  inflating: dataset/test/segments/road_image_163.png
  inflating: dataset/test/segments/road_image_164.png
  inflating: dataset/test/segments/road_image_165.png
  inflating: dataset/test/segments/road_image_166.png
  inflating: dataset/test/segments/road_image_167.png
  inflating: dataset/test/segments/road_image_168.png
  inflating: dataset/test/segments/road_image_169.png
  inflating: dataset/test/segments/road_image_170.png
  inflating: dataset/test/segments/road_image_171.png
  inflating: dataset/test/segments/road_image_172.png
```

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
```

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='dataset/validation/images'
            self.names=os.listdir(self.root)
        elif test:
            self.root='dataset/test/images'
            self.names=os.listdir(self.root)
        else:
            self.root='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.

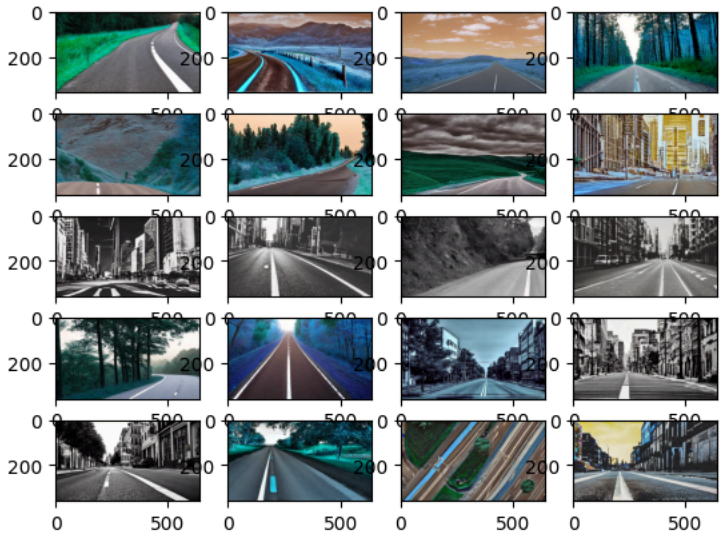
```
from torch.utils.data import DataLoader

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

Display images

- Show first sample of each mini-batch with size 8

```
# Printing the first sample of the each minibatch of size 8

plt.figure(figsize = (100, 100))

f, axarr = plt.subplots(5, 4)
i = 0
j = 0

for batch in train_dataloader:
  original_image, image_name, input, target = batch
  print(image_name[0])
  axarr[i, j].imshow(original_image[0])
  j += 1
  if j%4 == 0:
    i += 1
    j = 0

plt.show()
```

```
dataset/train/images/road_image_61.png
dataset/train/images/road_image_47.png
dataset/train/images/road_image_42.png
dataset/train/images/road_image_0.png
dataset/train/images/road_image_35.png
dataset/train/images/road_image_18.png
dataset/train/images/road_image_81.png
dataset/train/images/road_image_126.png
dataset/train/images/road_image_141.png
dataset/train/images/road_image_123.png
dataset/train/images/road_image_2.png
dataset/train/images/road_image_127.png
dataset/train/images/road_image_5.png
dataset/train/images/road_image_40.png
dataset/train/images/road_image_140.png
dataset/train/images/road_image_155.png
dataset/train/images/road_image_104.png
dataset/train/images/road_image_64.png
dataset/train/images/road_image_157.png
dataset/train/images/road_image_145.png
<Figure size 10000x10000 with 0 Axes>
```



Copy the required files from the repository to Root

```
# 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 ./
```

Load the pretrained model

```
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>
```

Intialize loss and optimizer.

- This is based on the original code from paper author

```
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


args
```

```
{'lr': 7.81e-06, 'max_epochs': 30, 'onGPU': True}
```

Intialize Polynomial Learning Rate Scheduler

- By Paper Author

```
def poly_lr_scheduler(args, optimizer, epoch, power=2):
    lr = round(args["lr"] * (1 - epoch / args["max_epochs"]) ** power, 8)
    for param_group in optimizer.param_groups:
        param_group['lr'] = lr

    return lr
```

Write a trainer function for each epoch

- By Paper Author

```
def train(args, train_loader, model, criterion, optimizer, epoch):
    model.train()

    total_batches = len(train_loader)
    pbar = enumerate(train_loader)
    pbar = tqdm(pbar, total=total_batches, bar_format='{l_bar}{bar:10}{r_bar}')
    j = 0
    avg_train_loss = 0
    for i, (_, _, input, target) in pbar:
        if args["onGPU"] == True:
            input = input.cuda().float() / 255.0
        output = model(input)

        # target=target.cuda()
        optimizer.zero_grad()

        focal_loss,tversky_loss,loss = criterion(output,target)
        avg_train_loss += loss.item()

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        pbar.set_description(('%13s' * 1 + '%13.4g' * 3) %
                                    (f'{epoch}/{args["max_epochs"] - 1}', tversky_loss, focal_loss, loss.item()))
        j += 1
    return avg_train_loss/j, loss.item()
```

Train the model with custom data and also print the loss

- This loss is based on the paper

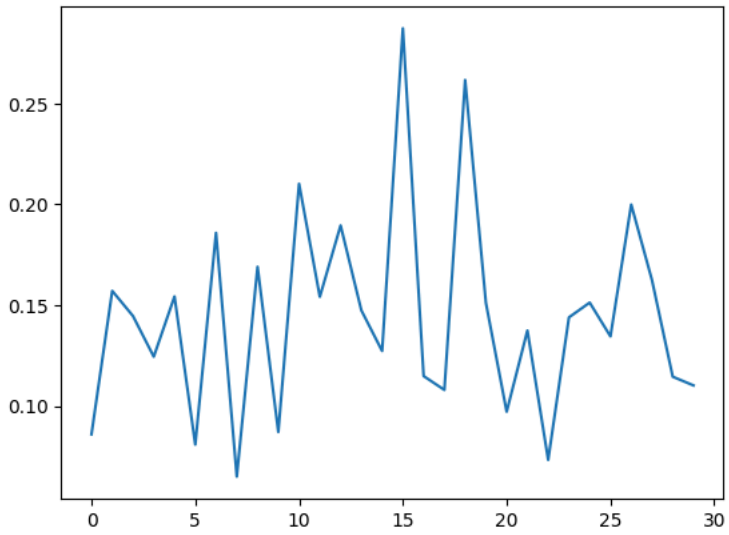
```
print("----------------------------------------")
training_loss_last_batch = []
validation_loss_last_batch = []
for epoch in range(0, args["max_epochs"]):
    print(f"Epoch: {epoch + 1}/{args['max_epochs']}")
    poly_lr_scheduler(args, optimizer, epoch)
    for param_group in optimizer.param_groups:
        lr = param_group['lr']
    print("Learning rate: " +  str(lr))
    print()

    # train for one epoch
    model.train()
    avg_train_loss, loss_for_last_batch_train = train( args, train_dataloader, model, criteria, optimizer, epoch)
    model.eval()

    avg_val_loss = 0
    i = 0
    for batch in val_dataloader:
        _, _, input, target = batch
        if args["onGPU"] == True:
            input = input.cuda().float() / 255.0
        output = model(input)
        focal_loss, tversky_loss, loss = criteria(output, target)
        avg_val_loss += loss.item()
        i += 1

    print()
    print(f"Average Training Loss: {avg_train_loss}")
    print(f"Average Validation Loss: {avg_val_loss/i}")
    print()
    print(f"Training loss for last batch: {loss_for_last_batch_train}")
    print(f"Validation loss for last batch: {loss.item()}")
    print("----------------------------------------")
    training_loss_last_batch.append(loss_for_last_batch_train)
    validation_loss_last_batch.append(loss.item())
```

```
----------------------------------------
Epoch: 1/30
Learning rate: 7.81e-06

         0/29       0.0602      0.02575      0.08595: 100%|██████████| 20/20 [00:05<00:00,  3.55it/s]

Average Training Loss: 0.16505988352000714
Average Validation Loss: 0.17195375760396323

Training loss for last batch: 0.08595025539398193
Validation loss for last batch: 0.16030383110046387
----------------------------------------
Epoch: 2/30
Learning rate: 7.3e-06

         1/29       0.1232      0.03386       0.1571: 100%|██████████| 20/20 [00:05<00:00,  3.58it/s]

Average Training Loss: 0.15430011823773385
Average Validation Loss: 0.1856925586859385

Training loss for last batch: 0.15708911418914795
Validation loss for last batch: 0.2438523769378662
----------------------------------------
Epoch: 3/30
Learning rate: 6.8e-06

         2/29      0.08851      0.05611       0.1446: 100%|██████████| 20/20 [00:05<00:00,  3.58it/s]

Average Training Loss: 0.14889373779296874
Average Validation Loss: 0.18966013938188553

Training loss for last batch: 0.14462456107139587
Validation loss for last batch: 0.29650747776031494
----------------------------------------
Epoch: 4/30
Learning rate: 6.33e-06
```

```
          3/29      0.08721      0.03725       0.1245: 100%|██████████| 20/20 [00:05<00:00,  3.63it/s]

    Average Training Loss: 0.15603391714394094
    Average Validation Loss: 0.16158105432987213

    Training loss for last batch: 0.12445536255836487
    Validation loss for last batch: 0.13932450115680695
    ----------
    Epoch: 5/30
    Learning rate: 5.87e-06

          4/29      0.08816      0.06617       0.1543: 100%|██████████| 20/20 [00:05<00:00,  3.61it/s]

    Average Training Loss: 0.15671782456338407
    Average Validation Loss: 0.1615806519985199

    Training loss for last batch: 0.15432827174663544
    Validation loss for last batch: 0.1095014363527298
    ----------
    Epoch: 6/30
```

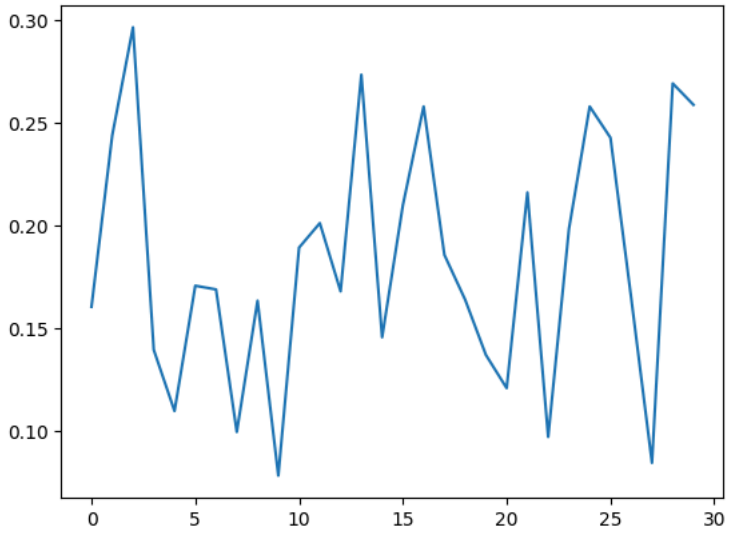
```
%matplotlib inline
import matplotlib.pyplot as plt

x = list(range(len(training_loss_last_batch)))
y = training_loss_last_batch

plt.plot(x, y)
```

```
[<matplotlib.lines.Line2D at 0x7804ac113700>]
```




```
%matplotlib inline
import matplotlib.pyplot as plt

x = list(range(len(validation_loss_last_batch)))
y = validation_loss_last_batch

plt.plot(x, y)
```

```
[<matplotlib.lines.Line2D at 0x7804a4293e80>]
```



Calculating loss on Test data

```
avg_test_loss = 0
i = 0
for batch in test_dataloader:
    _, _, input, target = batch
    if args["onGPU"] == True:
        input = input.cuda().float() / 255.0
    output = model(input)
    focal_loss, tversky_loss, loss = criteria(output, target)
    avg_test_loss += loss.item()
    i += 1

print("-------------------------------------------")
print(f"Average Testing Loss: {avg_test_loss/i}")
print(f"Testing loss for last batch: {loss.item()}")
print("-------------------------------------------")
```

```
-------------------------------------------
Average Testing Loss: 0.23437678317228952
Testing loss for last batch: 0.19679684937000275
-------------------------------------------
```

Defining functions to calculate Pixel Accuracy and Intersection of Union

- by paper author

```
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()
```