```
!pip install segmentation-models-pytorch
!pip install -U git+https://github.com/albumentations-team/albumentations
!pip
```

Locate in Drive

Open in playground mode

New notebook

Open notebook ⌘/Ctrl+O

Upload notebook

Rename

Move

Move to trash

Save a copy in Drive

Save a copy as a GitHub Gist

Save a copy in GitHub

Save ⌘/Ctrl+S

Save and pin revision ⌘/Ctrl+M S

Revision history

Download ▶

Print ⌘/Ctrl+P

```
on

e-any.whl (10 kB)
-extensions in /usr/local/lib/python3.
in /usr/local/lib/python3.7/dist-packa
ts in /usr/local/lib/python3.7/dist-pa
 /usr/local/lib/python3.7/dist-package
,>=2.5 in /usr/local/lib/python3.7/dis
i>=2017.4.17 in /usr/local/lib/python3
3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /
t<4,>=3.0.2 in /usr/local/lib/python3.
s: efficientnet-pytorch, pretrainedmod
orch (setup.py) ... done
rch: filename=efficientnet_pytorch-0.7
ip/wheels/0e/cc/b2/49e74588263573ff778
 (setup.py) ... done
 filename=pretrainedmodels-0.7.4-py3-n
ip/wheels/ed/27/e8/9543d42de2740d3544d
ch pretrainedmodels
 timm, pretrainedmodels, efficientnet-
ytorch-0.7.1 munch-2.5.0 pretrainedmod
simple, https://us-python.pkg.dev/cola
umentations-team/albumentations
ations-team/albumentations to /tmp/pip
//github.com/albumentations-team/album
=1.11.1 in /usr/local/lib/python3.7/di
=1.1.0 in /usr/local/lib/python3.7/dis
-image>=0.16.1 in /usr/local/lib/pytho
 in /usr/local/lib/python3.7/dist-pack
>=0.0.4 in /usr/local/lib/python3.7/di
  Requirement already satisfied: opencv-python>=4.1.1 in /usr/local/lib/pytho
  Requirement already satisfied: scikit-learn>=0.19.1 in /usr/local/lib/pytho
  Requirement already satisfied: opencv-python-headless>=4.0.1 in /usr/local/
  Requirement already satisfied: typing-extensions in /usr/local/lib/python3.
  Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.7/d
  Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.
  Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python
  Requirement already satisfied: matplotlib!=3.0.0,>=2.0.0 in /usr/local/lib/
  Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.7/di
  Requirement already satisfied: pillow!=7.1.0,!=7.1.1,>=4.3.0 in /usr/local/
  Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dis
  Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/pytho
  Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
  Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.
  Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-pa
  Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/pytho
  Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dis
  Building wheels for collected packages: albumentations
    Building wheel for albumentations (setup.py) ... done
    Created wheel for albumentations: filename=albumentations-1.3.0-py3-none-
    Stored in directory: /tmp/pip-ephem-wheel-cache-u1q2uunw/wheels/3a/25/ed/
  Successfully built albumentations
  Installing collected packages: albumentations
    Attempting uninstall: albumentations
      Found existing installation: albumentations 1.2.1
      Uninstalling albumentations-1.2.1:
        Successfully uninstalled albumentations-1.2.1
  Successfully installed albumentations-1.3.0
  Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/cola
  Requirement already satisfied: opencv-contrib-python in /usr/local/lib/pyth
  Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/di
```

```
!git clone https://github.com/parth1620/Road_seg_dataset.git
```

```
  Cloning into 'Road_seg_dataset'...
  remote: Enumerating objects: 411, done.
  remote: Total 411 (delta 0), reused 0 (delta 0), pack-reused 411
  Receiving objects: 100% (411/411), 851.74 MiB | 37.44 MiB/s, done.
  Resolving deltas: 100% (2/2), done.
  Checking out files: 100% (401/401), done.
```

```
import sys
sys.path.append('/content/Road_seg_dataset')
```

```
import torch
import cv2

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from tqdm import tqdm

import helper
```

## Configurations

```
CSV_FILE = '/content/Road_seg_dataset/train.csv'
DATA_DIR = '/content/Road_seg_dataset/'
DEVICE = 'cuda'
EPOCHS = 25
LR = 0.003
BATCH_SIZE = 5
IMG_SIZE = 512
ENCODER = 'timm-efficientnet-b0'
WEIGHTS = 'imagenet'
```

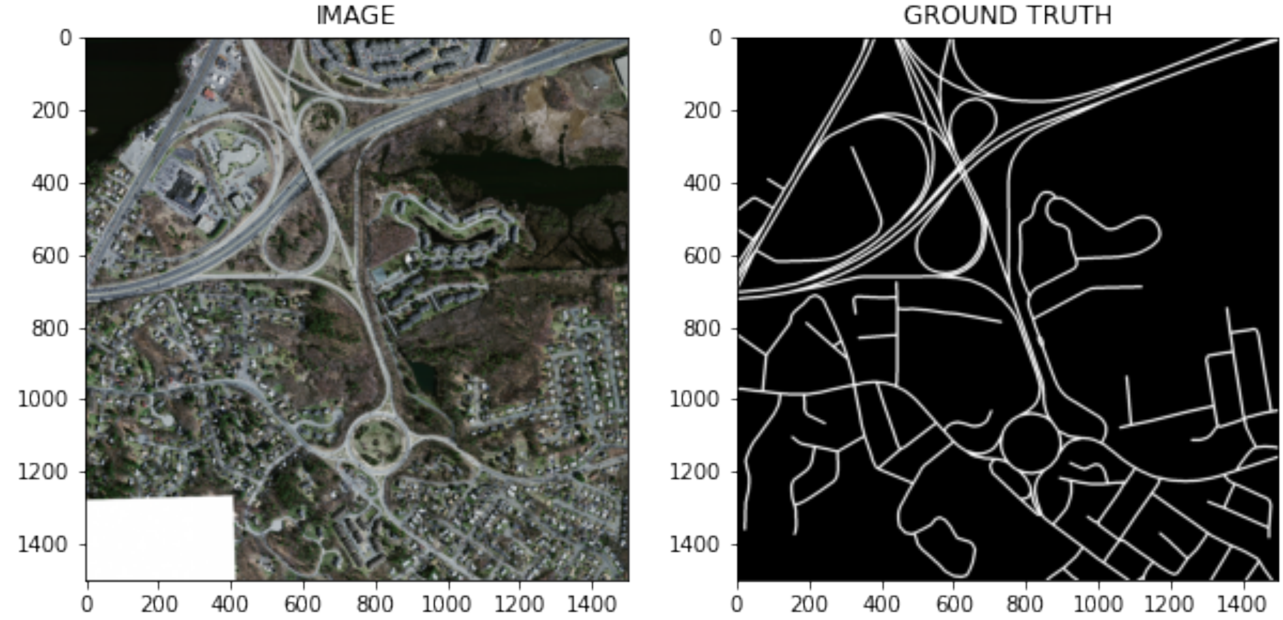
```
df =pd.read_csv(CSV_FILE)
df.head()
```

| | images | masks |
|---|---|---|
| 0 | images/17428750_15.png | masks/17428750_15.png |
| 1 | images/23279080_15.png | masks/23279080_15.png |
| 2 | images/24179185_15.png | masks/24179185_15.png |
| 3 | images/24179035_15.png | masks/24179035_15.png |
| 4 | images/11128810_15.png | masks/11128810_15.png |

```
idx = 2
row = df.iloc [idx]
image_path = DATA_DIR + row.images
mask_path = DATA_DIR + row. masks
image = cv2.imread(image_path)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
mask = cv2.imread (mask_path, cv2.IMREAD_GRAYSCALE) / 255
```

```
f, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,5))

ax1.set_title('IMAGE')
ax1.imshow(image)

ax2.set_title('GROUND TRUTH')
ax2.imshow(mask,cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x7ff219610f50>
```



```
train_df, valid_df = train_test_split(df, test_size=0.20, random_state=42)
len(train_df)
```

```
159
```

## Augmentation Functions

```
import albumentations as A
```

```
def get_train_augs():
  return A.Compose([
      A.Resize( IMG_SIZE, IMG_SIZE),
      A.HorizontalFlip (p = 0.5),
      A.VerticalFlip (p = 0.5)])
def get_valid_augs():
  return A.Compose([
      A.Resize(IMG_SIZE, IMG_SIZE)
])
```

## Creation of Custom Dataset

```
from torch.utils.data import Dataset
```

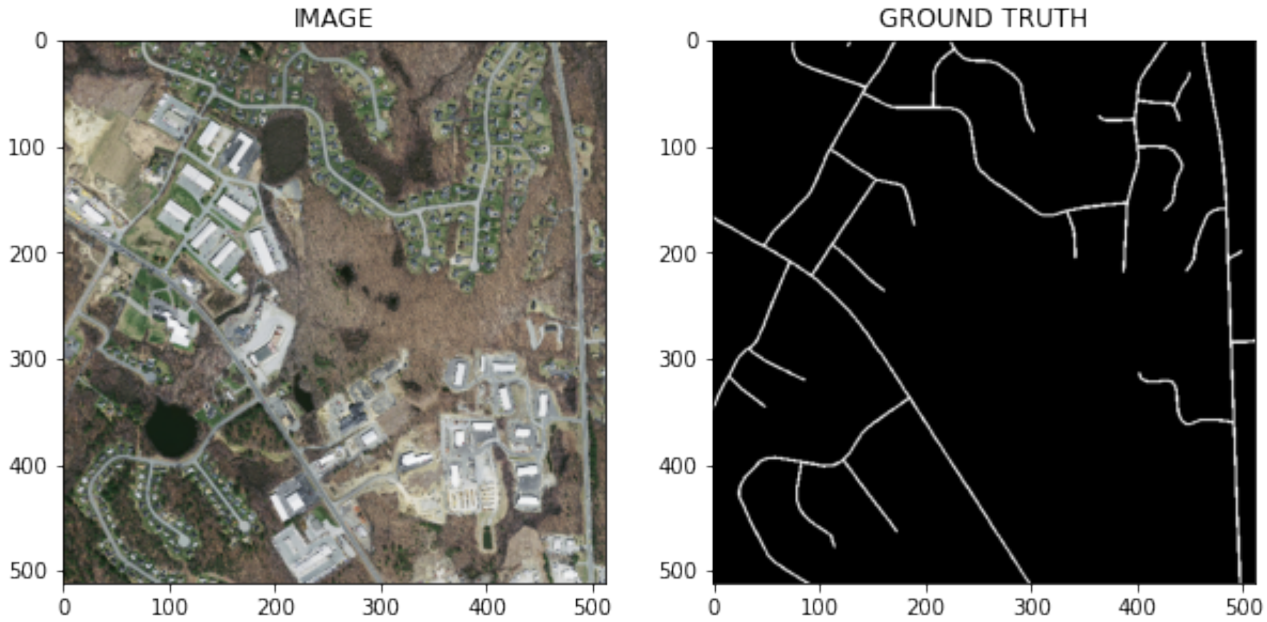
```
class SegmentationDataset(Dataset):

    def __init__(self, df, augmentations):
        self.df = df
        self.augmentations = augmentations

    def __len__(self):
        return len(self.df)
    def __getitem__(self, idx):
        row= df.iloc [idx]
        image_path = DATA_DIR+ row.images
        mask_path = DATA_DIR+ row.masks

        image = cv2.imread(image_path)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) #{h, W, c)

        mask = cv2.imread(mask_path, cv2.IMREAD_GRAYSCALE)#(h, w)
        mask = np.expand_dims(mask, axis = -1) #(h, W, c)

        if self.augmentations:
            data = self.augmentations(image= image, mask= mask)
            image = data['image'] #(h, w, C)
            mask = data['mask']

        image = np.transpose(image, (2, 0, 1)).astype(np.float32) #(c, h, w)
        mask = np.transpose(mask, (2, 0, 1)).astype(np.float32) #(c, h, w)

        image=torch.Tensor(image) / 255.0
        mask=torch.round(torch.Tensor(mask) / 255.0)

        return image, mask
```

```
trainset = SegmentationDataset(train_df, get_train_augs())
validset = SegmentationDataset(valid_df, get_valid_augs())
```

```
print(f'Size of trainset: {len(trainset)}')
print(f'Size of validset: {len(validset)}')
```

```
Size of trainset: 159
Size of validset: 40
```

```
idx = 69
image, mask = trainset[idx]
helper.show_image(image, mask)
```



## Loading dataset into batches

```
from torch.utils.data import DataLoader
```

```
trainloader = DataLoader(trainset, batch_size=BATCH_SIZE, shuffle = True)
validloader = DataLoader(validset, batch_size=BATCH_SIZE)
```

```
print(f'Total no. of batched in trainloader : {len(trainloader)}')
print(f'Total no. of batched in validloader : {len(validloader)}')
```

```
Total no. of batched in trainloader : 32
Total no. of batched in validloader : 8
```

```
for images, masks in trainloader:
    print(f"One batch image shape : {images.shape}")
    print(f"One batch mask shape : {masks.shape}")
    break;
```

```
One batch image shape : torch.Size([5, 3, 512, 512])
One batch mask shape : torch.Size([5, 1, 512, 512])
```

# Create Segmentation Model

```
import segmentation_models_pytorch as smp
from segmentation_models_pytorch.losses import DiceLoss
from torch import nn
```

```
import gc

gc.collect()

torch.cuda.empty_cache()
```

# MODEL 1: UNet with EfficientNetb0 Encoder

```
class SegmentationModel1 (nn.Module) :
    def __init__(self):
        super(SegmentationModel1,self).__init__()

        self.backbone = smp.Unet(
            encoder_name = 'timm-efficientnet-b0',
            encoder_weights = WEIGHTS,
            in_channels = 3,
            classes = 1,
        )
    def forward (self, images, masks = None):
      logits = self. backbone (images)
      if masks != None:
          return logits, DiceLoss (mode = 'binary')(logits, masks) + nn.BCEWith
      return logits
```

# MODEL 2: UNet with EfficientNetb3 Encoder

```
class SegmentationModel2 (nn.Module) :
    def __init__(self):
        super(SegmentationModel2,self).__init__()

        self.backbone = smp.Unet(
            encoder_name = 'timm-efficientnet-b3',
            encoder_weights = WEIGHTS,
            in_channels = 3,
            classes = 1,
        )
    def forward (self, images, masks = None):
      logits = self. backbone (images)
      if masks != None:
          return logits, DiceLoss (mode = 'binary')(logits, masks) + nn.BCEWith
      return logits
```

## MODEL 3: UNet++ with EfficientNetb0 Encoder

```
class SegmentationModel3 (nn.Module) :
    def __init__(self):
        super(SegmentationModel3,self).__init__()

        self.backbone = smp.UnetPlusPlus(
            encoder_name = 'timm-efficientnet-b3',
            encoder_weights = WEIGHTS,
            in_channels = 3,
            classes = 1,
        )
    def forward (self, images, masks = None):
      logits = self. backbone (images)
      if masks != None:
          return logits, DiceLoss (mode = 'binary')(logits, masks) + nn.BCEWith
      return logits
```

```
model1 = SegmentationModel1()
model1.to(DEVICE);

model2 = SegmentationModel2()
model2.to(DEVICE);

model3 = SegmentationModel3()
model3.to(DEVICE);
```

```
Downloading: "https://github.com/rwightman/pytorch-image-models/releases/do
100%                                        20.4M/20.4M [00:00<00:00, 38.5MB/s]
Downloading: "https://github.com/rwightman/pytorch-image-models/releases/do
100%                                        47.1M/47.1M [00:00<00:00, 72.5MB/s]
```

## Training and Validation Functions

```
def train_fn(dataloader, model, optimizer) :
    model.train() # Turn ON d ropout, batchnorm, etc..
    total_loss = 0.0
    for images, masks in tqdm (dataloader):
        images = images.to (DEVICE)
        masks = masks.to (DEVICE)
        optimizer.zero_grad ()
        logits, loss = model (images, masks)
        loss.backward ()
        optimizer. step()
        total_loss += loss.item()
    return total_loss / len (dataloader)


def eval_fn(dataloader, model):
    model.eval() # Turn OFF dropout, batchno rm, etc.
    total_loss = 0.0
    with torch.no_grad():
        for images, masks in tqdm (dataloader):
            images = images.to(DEVICE)
            masks = masks.to(DEVICE)
            logits, loss = model(images, masks)
            total_loss += loss.item()
        return total_loss / len(dataloader)
```

## Training the Model

```
optimizer1 = torch.optim.Adam(model1.parameters(), lr= LR)
optimizer2 = torch.optim.Adam(model2.parameters(), lr= LR)
optimizer3 = torch.optim.Adam(model3.parameters(), lr= LR)


EPOCHS=30
best_loss1 = np.Inf
best_loss2 = np.Inf
best_loss3 = np.Inf

for i in range(EPOCHS):
    train_loss1 = train_fn(trainloader, model1, optimizer1)
    valid_loss1 = eval_fn(validloader, model1)
    if valid_loss1 < best_loss1:
        torch.save(model1.state_dict(), "best-model1.pt")
        print("SAVED-MODEL")
        best_loss1 = valid_loss1
    print(f"Epoch : {i+1} Train Loss : {train_loss1} Valid Loss : {valid_loss1}

    train_loss2 = train_fn(trainloader, model2, optimizer2)
    valid_loss2 = eval_fn(validloader, model2)
    if valid_loss2 < best_loss2:
        torch.save(model2.state_dict(), "best-model2.pt")
        print("SAVED-MODEL")
        best_loss2 = valid_loss2
    print(f"Epoch : {i+1} Train Loss : {train_loss2} Valid Loss : {valid_loss2}

    train_loss3 = train_fn(trainloader, model3, optimizer3)
    valid_loss3 = eval_fn(validloader, model3)
    if valid_loss3 < best_loss3:
        torch.save(model3.state_dict(), "best-model3.pt")
        print("SAVED-MODEL")
        best_loss3 = valid_loss3
    print(f"Epoch : {i+1} Train Loss : {train_loss3} Valid Loss : {valid_loss3}
```

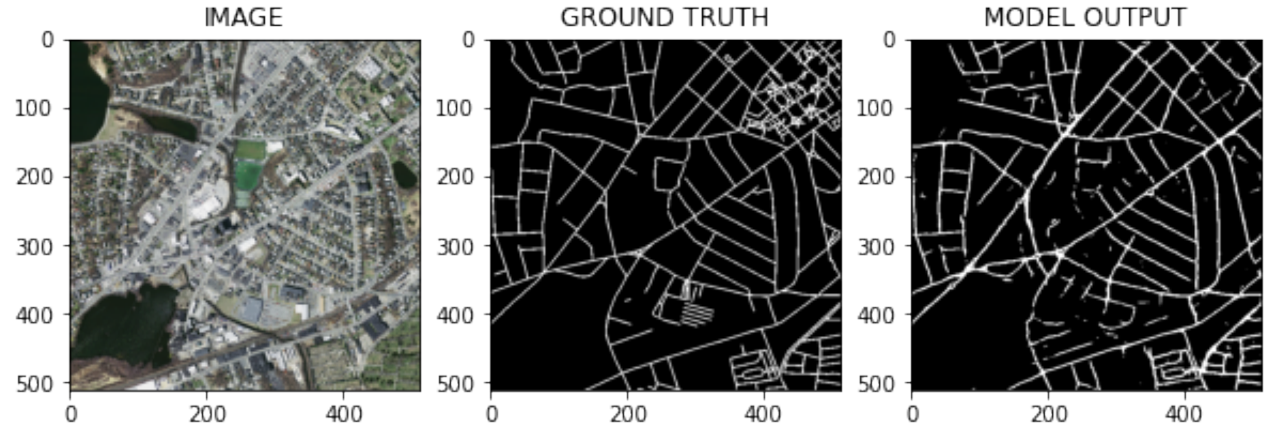
```
100%|██████████| 8/8 [00:04<00:00,  1.78it/s]
SAVED-MODEL
Epoch : 24 Train Loss : 0.5556764397770166 Valid Loss : 0.5246622487902641
100%|██████████| 32/32 [00:21<00:00,  1.47it/s]
100%|██████████| 8/8 [00:03<00:00,  2.12it/s]
SAVED-MODEL
Epoch : 25 Train Loss : 0.5815475732088089 Valid Loss : 0.5648483261466026
100%|██████████| 32/32 [00:27<00:00,  1.18it/s]
100%|██████████| 8/8 [00:04<00:00,  1.97it/s]
```

```
Epoch : 25 Train Loss : 0.5667834188789129 Valid Loss : 0.5883173123002052
100%|██████████| 32/32 [00:32<00:00,  1.01s/it]
100%|██████████| 8/8 [00:04<00:00,  1.80it/s]
Epoch : 25 Train Loss : 0.5638497276231647 Valid Loss : 0.580753531306982
100%|██████████| 32/32 [00:21<00:00,  1.50it/s]
100%|██████████| 8/8 [00:03<00:00,  2.11it/s]
Epoch : 26 Train Loss : 0.5920161530375481 Valid Loss : 0.6015725657343864
100%|██████████| 32/32 [00:26<00:00,  1.19it/s]
100%|██████████| 8/8 [00:04<00:00,  1.98it/s]
Epoch : 26 Train Loss : 0.5625287089496851 Valid Loss : 0.5780405476689339
100%|██████████| 32/32 [00:31<00:00,  1.00it/s]
100%|██████████| 8/8 [00:04<00:00,  1.79it/s]
Epoch : 26 Train Loss : 0.559606340713799 Valid Loss : 0.5456802845001221
100%|██████████| 32/32 [00:21<00:00,  1.47it/s]
100%|██████████| 8/8 [00:03<00:00,  2.13it/s]
SAVED-MODEL
Epoch : 27 Train Loss : 0.5731905614957213 Valid Loss : 0.5243817865848541
100%|██████████| 32/32 [00:26<00:00,  1.20it/s]
100%|██████████| 8/8 [00:04<00:00,  1.98it/s]
SAVED-MODEL
Epoch : 27 Train Loss : 0.5568776084110141 Valid Loss : 0.5387633070349693
100%|██████████| 32/32 [00:32<00:00,  1.00s/it]
100%|██████████| 8/8 [00:04<00:00,  1.77it/s]
Epoch : 27 Train Loss : 0.5523277018219233 Valid Loss : 0.5448591336607933
100%|██████████| 32/32 [00:21<00:00,  1.47it/s]
100%|██████████| 8/8 [00:03<00:00,  2.11it/s]
Epoch : 28 Train Loss : 0.5802625641226768 Valid Loss : 0.5963739082217216
100%|██████████| 32/32 [00:27<00:00,  1.18it/s]
100%|██████████| 8/8 [00:04<00:00,  1.98it/s]
Epoch : 28 Train Loss : 0.562823121166229 Valid Loss : 0.5891459099948406
100%|██████████| 32/32 [00:32<00:00,  1.01s/it]
100%|██████████| 8/8 [00:04<00:00,  1.75it/s]
Epoch : 28 Train Loss : 0.5525912512093782 Valid Loss : 0.6006879806518555
100%|██████████| 32/32 [00:21<00:00,  1.49it/s]
100%|██████████| 8/8 [00:03<00:00,  2.14it/s]
Epoch : 29 Train Loss : 0.5834842585027218 Valid Loss : 0.5809385180473328
100%|██████████| 32/32 [00:26<00:00,  1.20it/s]
100%|██████████| 8/8 [00:04<00:00,  1.81it/s]
Epoch : 29 Train Loss : 0.5659398334100842 Valid Loss : 0.5410635136067867
100%|██████████| 32/32 [00:31<00:00,  1.00it/s]
100%|██████████| 8/8 [00:04<00:00,  1.79it/s]
SAVED-MODEL
Epoch : 29 Train Loss : 0.5455511370673776 Valid Loss : 0.5026553012430668
100%|██████████| 32/32 [00:21<00:00,  1.49it/s]
100%|██████████| 8/8 [00:04<00:00,  1.91it/s]
Epoch : 30 Train Loss : 0.5821127658709884 Valid Loss : 0.5855755917727947
100%|██████████| 32/32 [00:26<00:00,  1.20it/s]
100%|██████████| 8/8 [00:04<00:00,  1.81it/s]
SAVED-MODEL
Epoch : 30 Train Loss : 0.5407118182629347 Valid Loss : 0.5286730341613293
```

## Outputs

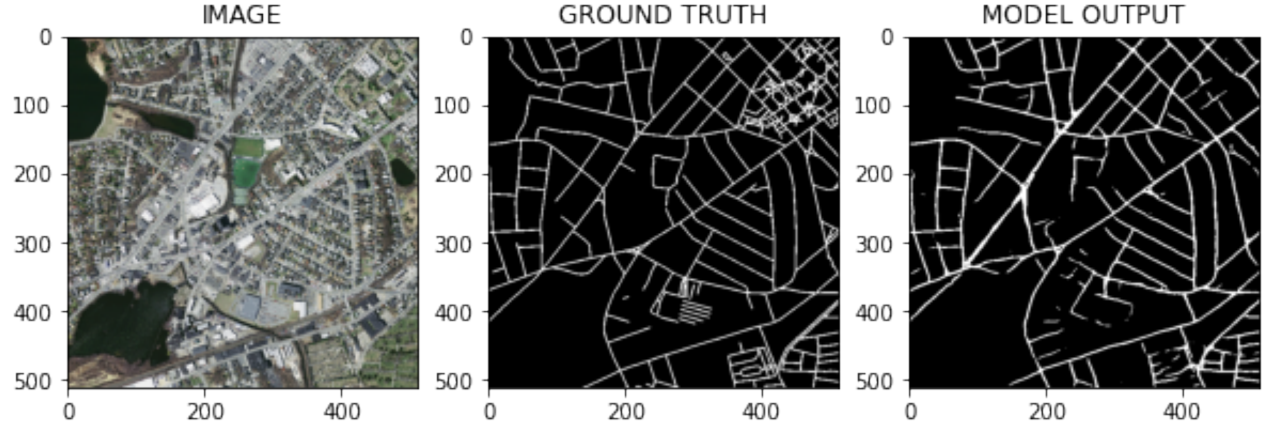
```
def displaymodel1(idx):
  model1.load_state_dict(torch.load('/content/best-model1.pt'))
  image, mask = validset[idx]
  logits_mask1 = model1(image.to(DEVICE).unsqueeze(0)) #(c, h, w) -> (b, c, h,
  global pred_mask1
  pred_mask1 = torch.sigmoid(logits_mask1)
  pred_mask1 = (pred_mask1 > 0.5)*1.0
  print('MODEL 1 OUTPUT')
  helper.show_image(image, mask, pred_mask1.detach().cpu().squeeze(0))

idx= 9
displaymodel1(idx)
```

```
MODEL 1 OUTPUT
```



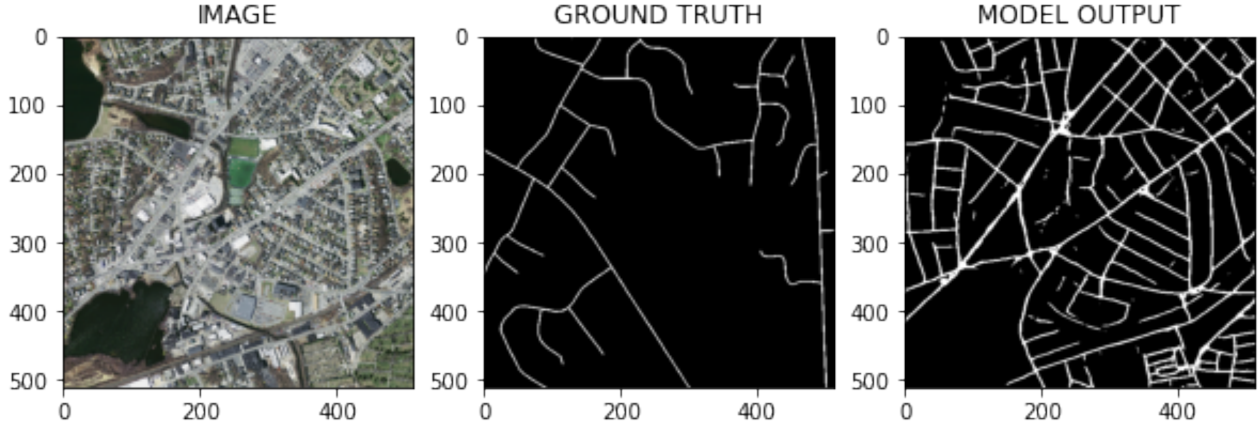
```
def displaymodel2(idx):
  model2.load_state_dict(torch.load('/content/best-model2.pt'))
  image, mask = validset[idx]
  logits_mask2 = model2(image.to(DEVICE).unsqueeze(0)) #(c, h, w) -> (b, c, h,
  global pred_mask2
  pred_mask2 = torch.sigmoid(logits_mask2)
  pred_mask2 = (pred_mask2 > 0.5)*1.0
  print('MODEL 2 OUTPUT')
  helper.show_image(image, mask, pred_mask2.detach().cpu().squeeze(0))

idx= 9
displaymodel2(idx)
```

```
MODEL 2 OUTPUT
```

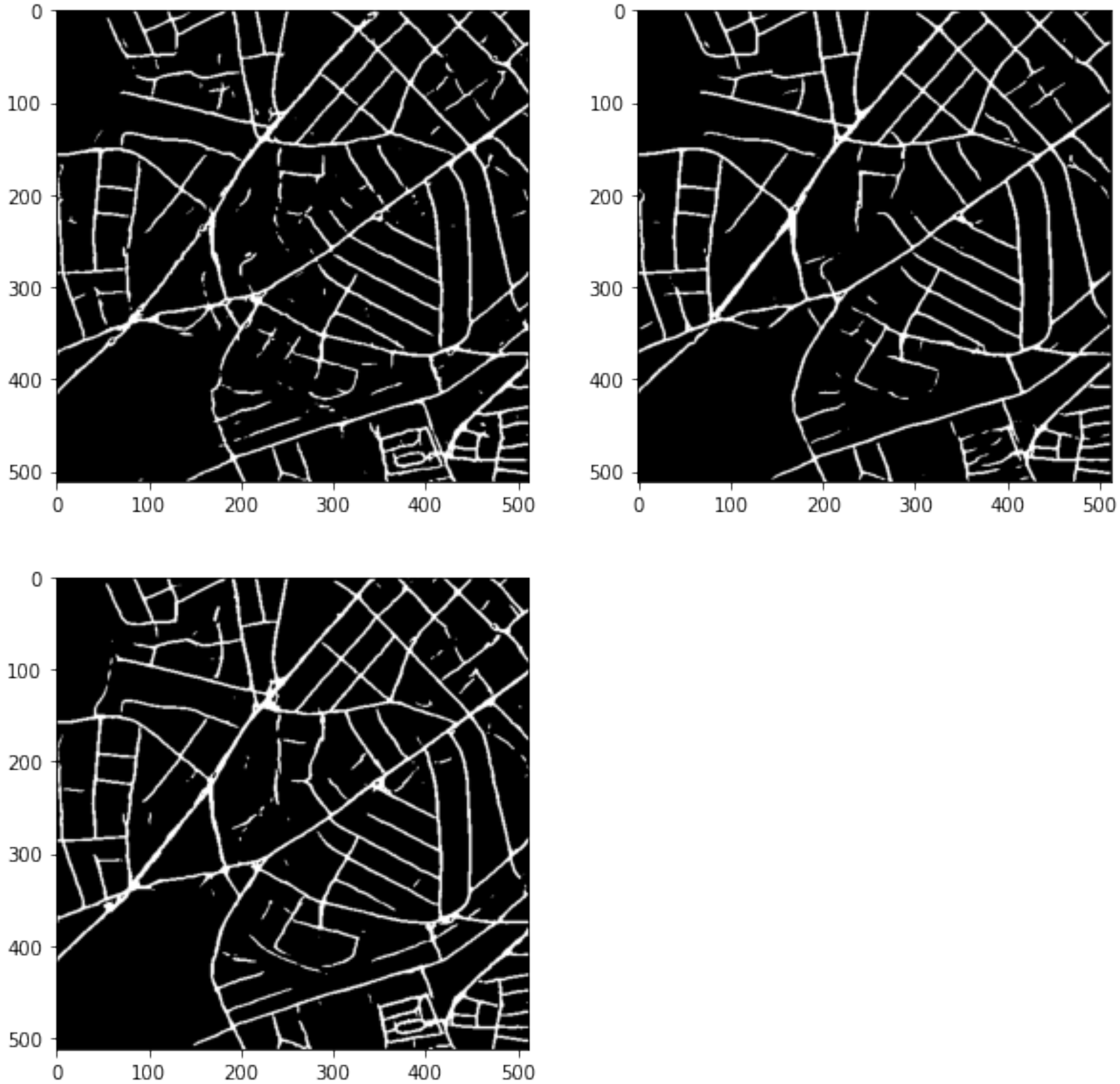


```
def displaymodel3(idx):
  model3.load_state_dict(torch.load('/content/best-model3.pt'))
  image, mask3 = validset[idx]
  logits_mask3 = model3(image.to(DEVICE).unsqueeze(0)) #(c, h, w) -> (b, c, h,
  global pred_mask3
  pred_mask3 = torch.sigmoid(logits_mask3)
  pred_mask3 = (pred_mask3 > 0.5)*1.0
  print('MODEL 3 OUTPUT')
  helper.show_image(image, mask, pred_mask3.detach().cpu().squeeze(0))

idx= 9
displaymodel3(idx)
```

```
MODEL 3 OUTPUT
```



```
idx=69

def displayAllModels():
  fig = plt.figure(figsize=(10,10))
  ax1 = fig.add_subplot(2,2,1)
  ax1.imshow(pred_mask1.detach().cpu().squeeze(0).squeeze(),cmap='gray')
  ax2 = fig.add_subplot(2,2,2)
  ax2.imshow(pred_mask2.detach().cpu().squeeze(0).squeeze(),cmap='gray')
  ax3 = fig.add_subplot(2,2,3)
  ax3.imshow(pred_mask3.detach().cpu().squeeze(0).squeeze(),cmap='gray')

displayAllModels()
```



## JACCARD SCORE

```
from sklearn.metrics import jaccard_score
def calc_jaccard(model,i):

  image, mask = validset[i]
  logits_mask = model(image.to(DEVICE).unsqueeze(0)) #(c, h, w) -> (b, c, h, w)
  pred_mask = torch.sigmoid(logits_mask)
  pred_mask = (pred_mask > 0.5)*1.0
  mask=np.squeeze(np.array(mask))
  pred=np.squeeze(np.array(pred_mask.detach().cpu().squeeze(0)))

  y_true = mask
  y_pred = pred

  labels = [0, 1]
  jaccards = []
  for label in labels:
    jaccard = jaccard_score(y_pred.flatten(),y_true.flatten(), pos_label=label)
    jaccards.append(jaccard)
  return(np.mean(jaccards))


sum1=0
sum2=0
sum3=0
bestj=0

for i in range(len(validset)):
  j1=calc_jaccard(model1,i)
  j2=calc_jaccard(model2,i)
  j3=calc_jaccard(model3,i)
  maxj=max(j1,j2,j3)
  sum1=sum1+j1
  sum2=sum2+j2
  sum3=sum3+j3
  bestj=bestj+maxj



print('MEAN JACCARD SCORE OF MODEL1 = ', sum1/len(validset))
print('MEAN JACCARD SCORE OF MODEL2 = ', sum2/len(validset))
print('MEAN JACCARD SCORE OF MODEL3 = ', sum3/len(validset))
print('JACCARD SCORE OF THE COMBINED MODEL = ', bestj/len(validset))
```

```
MEAN JACCARD SCORE OF MODEL1 =  0.7200820217263841
MEAN JACCARD SCORE OF MODEL2 =  0.7144829456864012
MEAN JACCARD SCORE OF MODEL3 =  0.726446186840874
JACCARD SCORE OF THE COMBINED MODEL =  0.7290105266211586
```

# COMBINED MODEL

```
#PROPOSED MODEL
def combinedmodel(idx):
  j1=calc_jaccard(model1,idx)
  j2=calc_jaccard(model2,idx)
  j3=calc_jaccard(model3,idx)
  max=pd.Series([j1,j2,j3]).idxmax()
  #TO KNOW WHICH ONE IS PERFORMING BETTER
  #print(j1,j2,j3)
  #print(MODEL)

  image, mask = validset[idx]

  if (max==0):
    logits_mask1 = model1(image.to(DEVICE).unsqueeze(0))
    pred_mask1 = torch.sigmoid(logits_mask1)
    pred_mask1 = (pred_mask1 > 0.5)*1.0
    helper.show_image(image, mask, pred_mask1.detach().cpu().squeeze(0))

  elif(max==1):
    logits_mask2 = model2(image.to(DEVICE).unsqueeze(0))
    pred_mask2 = torch.sigmoid(logits_mask2)
    pred_mask2 = (pred_mask1 > 0.5)*1.0
    helper.show_image(image, mask, pred_mask2.detach().cpu().squeeze(0))

  else:
    logits_mask3 = model3(image.to(DEVICE).unsqueeze(0))
    pred_mask3 = torch.sigmoid(logits_mask3)
    pred_mask3 = (pred_mask3 > 0.5)*1.0
    helper.show_image(image, mask, pred_mask3.detach().cpu().squeeze(0))
```

```
combinedmodel(21)
combinedmodel(34)
combinedmodel(44)
combinedmodel(22)
```

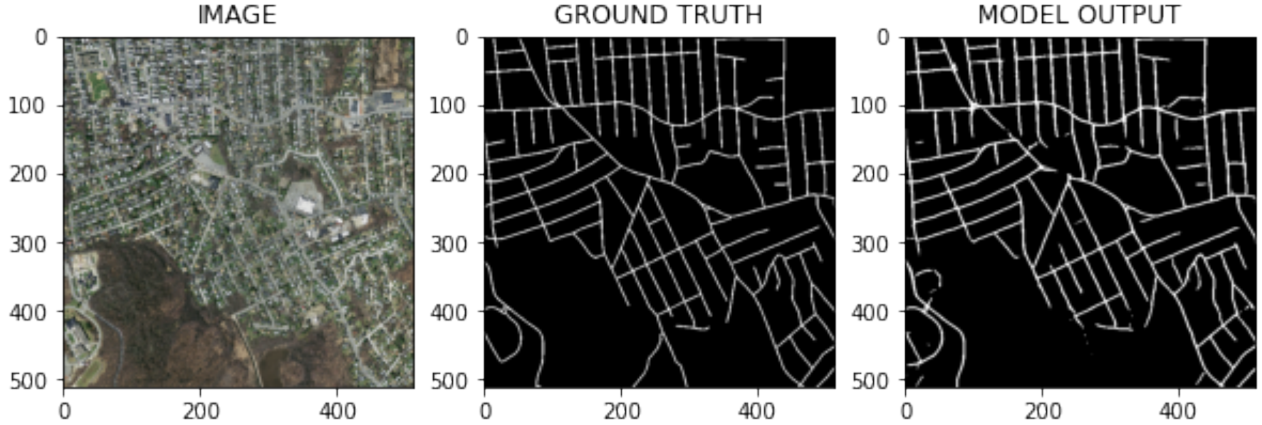


IMAGE
GROUND TRUTH
MODEL OUTPUT
0
100
200
300
400
500
0
200
400

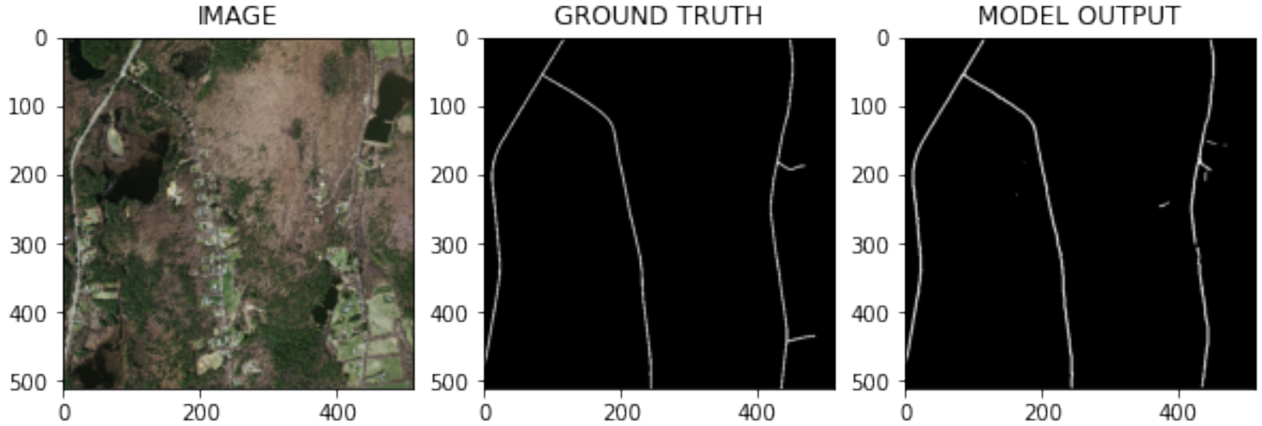


IMAGE
GROUND TRUTH
MODEL OUTPUT
0
100
200
300
400
500
0
200
400

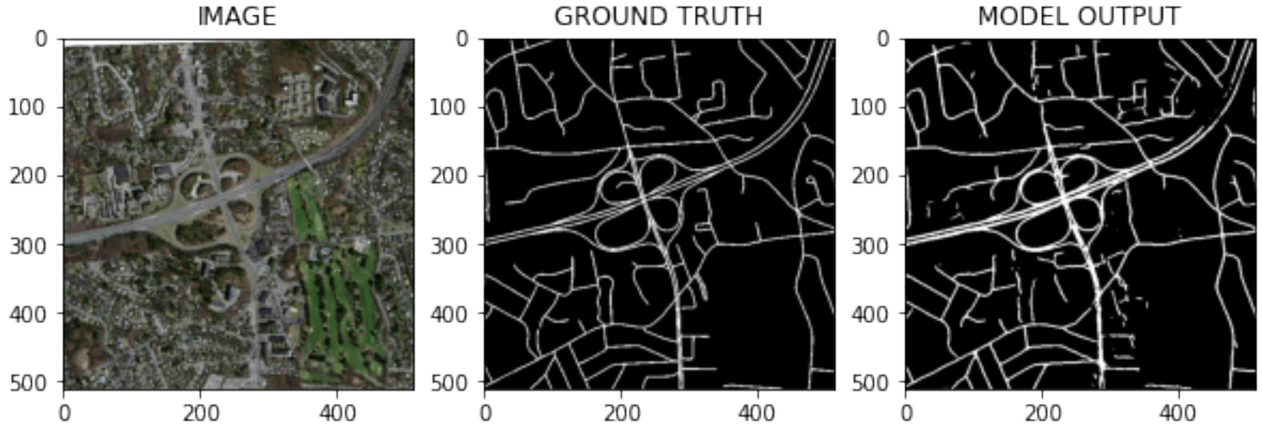


IMAGE
GROUND TRUTH
MODEL OUTPUT
0
100
200
300
400
500
0
200
400

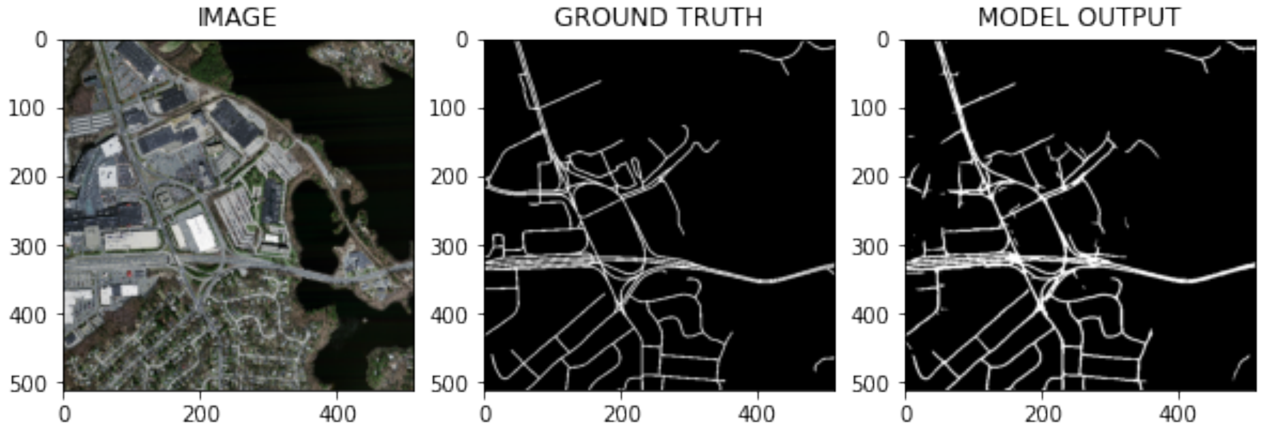


IMAGE
GROUND TRUTH
MODEL OUTPUT
0
100
200
300
400
500
0
200
400