In [ ]:
```
import math
from PIL import Image, ImageDraw
from PIL import ImagePath
import pandas as pd
import os
from os import path
from tqdm import tqdm
import json
import cv2
import numpy as np
import matplotlib.pyplot as plt
import urllib
```

In [ ]:
```
from google.colab import drive
drive.mount('/content/drive/')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?
client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleuser
content.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_t
ype=code&scope=email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.t
est%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20https%3a%2f%2fw
ww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.go
ogleapis.com%2fauth%2fpeopleapi.readonly

Enter your authorization code:
..........
Mounted at /content/drive/
```

In [ ]:
```
from zipfile import ZipFile
file_name='/content/drive/My Drive/seg/data.zip'
with ZipFile(file_name,'r') as Zip:
  Zip.extractall()
```

In [ ]:
```
import tensorflow as tf
# tf.compat.v1.enable_eager_execution()
```

```
from tensorflow import keras
from tensorflow.keras.layers import *
from tensorflow.keras.preprocessing import image
from tensorflow.keras import backend as K
from tensorflow.keras.utils import plot_model
K.set_image_data_format('channels_last')
K.set_learning_phase(1)
```

In [ ]:
```
data_df=pd.read_csv('/content/drive/My Drive/seg/seg.csv').iloc[:,1:]
data_df.head()
```

Out[ ]:

	images	json
0	data/images/201/frame0029_leftImg8bit.jpg	data/mask/201/frame0029_gtFine_polygons.json
1	data/images/201/frame0299_leftImg8bit.jpg	data/mask/201/frame0299_gtFine_polygons.json
2	data/images/201/frame0779_leftImg8bit.jpg	data/mask/201/frame0779_gtFine_polygons.json
3	data/images/201/frame1019_leftImg8bit.jpg	data/mask/201/frame1019_gtFine_polygons.json
4	data/images/201/frame1469_leftImg8bit.jpg	data/mask/201/frame1469_gtFine_polygons.json

In [ ]:
```
def grader_1(data_df):
    for i in data_df.values:
        if not (path.isfile(i[0]) and path.isfile(i[1]) and i[0][12:i[0
].find('_')]==i[1][10:i[1].find('_')]):
            return False
    return True
```

In [ ]:
```
grader_1(data_df)
```

Out[ ]: True

In [ ]:
```
from tqdm import tqdm
```

In [ ]:
```
def return_unique_labels(data_df):

    un_lab=[]
```

```
    for i in tqdm(data_df.json):
        po=open(i)
        jp=json.load(po)
        for i in jp:
            if i not in un_lab:
                #print(type(jp[i]))
                #if type(jp[i])==int:
                 #   un_lab.append(i)
                if type(jp[i])==list:
                    for j in jp[i]:
                        if j['label'] not in un_lab:
                            un_lab.append(j['label'])


    return un_lab
```

In [ ]:
```
unique_labels = return_unique_labels(data_df)
```

```
100%|██████████| 4008/4008 [00:30<00:00, 132.62it/s]
```

In [ ]:
```
label_clr = {'road':10, 'parking':20, 'drivable fallback':20,'sidewalk'
:30,'non-drivable fallback':40,'rail track':40,\
                        'person':50, 'animal':50, 'rider':60, 'motorcyc
le':70, 'bicycle':70, 'autorickshaw':80,\
                        'car':80, 'truck':90, 'bus':90, 'vehicle fallba
ck':90, 'trailer':90, 'caravan':90,\
                        'curb':100, 'wall':100, 'fence':110,'guard rai
l':110, 'billboard':120,'traffic sign':120,\
                        'traffic light':120, 'pole':130, 'polegroup':13
0, 'obs-str-bar-fallback':130,'building':140,\
                        'bridge':140,'tunnel':140, 'vegetation':150, 's
ky':160, 'fallback background':160,'unlabeled':0,\
                        'out of roi':0, 'ego vehicle':180, 'ground':190
,'rectification border':200,\
                    'train':210}
```

In [ ]:
```
def grader_2(unique_labels):
```

```
    if (not (set(label_clr.keys())-set(unique_labels))) and len(unique_
labels) == 40:
        print("True")
    else:
        print("Flase")

grader_2(unique_labels)
```

```
True
```

In [ ]:

```
%%time
def get_poly(file):
    label=[]
    w=[]
    h=[]
    vertexlist=[]
    po=open(file)
    jp=json.load(po)
    w.append(jp['imgWidth'])
    h.append(jp['imgHeight'])
    vertex=[]
    for k in jp:
        if type(jp[k])==list:
                for j in jp[k]:
                    label.append(j['label'])
                    fir=[]
                    for f in j['polygon']:
                        fir.append(tuple(f))
                    vertex.append(fir)
    vertexlist.append(vertex)
    return w[0], h[0], label, vertexlist[0]

    # this function will take a file name as argument
```

```
CPU times: user 6 µs, sys: 0 ns, total: 6 µs
Wall time: 10 µs
```

In [ ]:

```
%%time
w=[]
```

```
h=[]
labels=[]
vertexlist=[]
for i in tqdm(data_df.json):
    a,b,c,d=get_poly(i)
    w.append(a)
    h.append(b)
    labels.append(c)
    vertexlist.append(d)
```

```
100%|██████████| 4008/4008 [00:58<00:00, 68.42it/s]
```

```
CPU times: user 56.9 s, sys: 1.71 s, total: 58.6 s
Wall time: 58.6 s
```

In [ ]:
```
def grader_3(file):
    w, h, labels, vertexlist = get_poly(file)
    print(len((set(labels)))==18 and len(vertexlist)==227 and w==1920 a
nd h==1080 \
          and isinstance(vertexlist,list) and isinstance(vertexlist[0],
list) and isinstance(vertexlist[0][0],tuple) )

grader_3('data/mask/201/frame0029_gtFine_polygons.json')
```

```
True
```

In [ ]:
```
path="data/output/"
uniq=[]
for k in data_df.json:
    if k.split('/')[-2] not in uniq:
        uniq.append(k.split('/')[-2])
        os.makedirs(path+str(k.split('/')[-2]))
```

In [ ]:
```
def compute_masks(data_df):
    # after you have computed the vertexlist plot that polygone in imag
e like this

    path="data/output/"
```

```
    mask_path=[]
    for j,k in tqdm(zip(range(data_df.shape[0]),data_df.json)):
        ver=vertexlist[j]
        lab=labels[j]
        img =Image.new('RGB',(w[j],h[j]))
        for i in range(len(ver)):
            img1 = ImageDraw.Draw(img)

            if len(ver[i])>1:
                img1.polygon(ver[i], fill = label_clr[lab[i]])
        img_m=np.array(img)
        pa=path+k.split('mask/')[-1].split('.')[0]+'.png'
        mask_path.append(pa)
        cv2.imwrite(pa,img_m[:,:,0])
    if len(mask_path)==data_df.shape[0]:
        data_df['mask']=mask_path

    return data_df
```

In [ ]:

```
%%time
data_df=compute_masks(data_df)
```

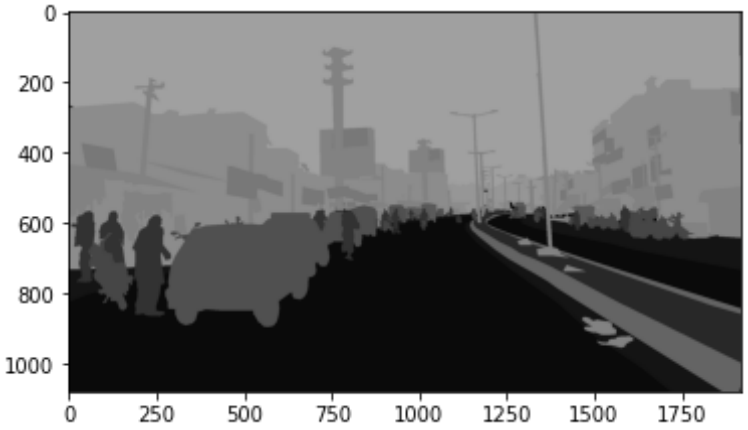
```
4008it [01:13, 54.45it/s]

CPU times: user 1min 12s, sys: 1.28 s, total: 1min 13s
Wall time: 1min 13s
```

In [ ]:

```
def grader_3():
    url = "https://i.imgur.com/4XSUlHk.png"
    url_response = urllib.request.urlopen(url)
    img_array = np.array(bytearray(url_response.read()), dtype=np.uint8
)
    img = cv2.imdecode(img_array, -1)
    my_img = cv2.imread('data/output/201/frame0029_gtFine_polygons.png'
)
    plt.imshow(my_img)
    print((my_img[:,:,0]==img).all())
    print(np.unique(img))
```

```
    print(np.unique(my_img[:,:,0]))
    data_df.to_csv('preprocessed_data.csv', index=False)
grader_3()
```

```
True
[  0  10  20  40  50  60  70  80  90 100 120 130 140 150 160]
[  0  10  20  40  50  60  70  80  90 100 120 130 140 150 160]
```



In [ ]:
```
pip install segmentation-models
```

```
Collecting segmentation-models
  Downloading https://files.pythonhosted.org/packages/da/b9/4a183518c21
689a56b834eaaa45cad242d9ec09a4360b5b10139f23c63f4/segmentation_models-
1.0.1-py3-none-any.whl
Collecting image-classifiers==1.0.0
  Downloading https://files.pythonhosted.org/packages/81/98/6f84720e299
a4942ab80df5f76ab97b7828b24d1de5e9b2cbbe6073228b7/image_classifiers-1.
0.0-py3-none-any.whl
Collecting efficientnet==1.0.0
  Downloading https://files.pythonhosted.org/packages/97/82/f3ae07316f0
461417dc54affab6e86ab188a5a22f33176d35271628b96e0/efficientnet-1.0.0-py
3-none-any.whl
Requirement already satisfied: keras-applications<=1.0.8,>=1.0.7 in /us
r/local/lib/python3.6/dist-packages (from segmentation-models) (1.0.8)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.
6/dist-packages (from efficientnet==1.0.0->segmentation-models) (0.16.
```

```
2)
Requirement already satisfied: h5py in /usr/local/lib/python3.6/dist-pa
ckages (from keras-applications<=1.0.8,>=1.0.7->segmentation-models)
(2.10.0)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.
6/dist-packages (from keras-applications<=1.0.8,>=1.0.7->segmentation-m
odels) (1.18.5)
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.
6/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation-m
odels) (2.4)
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python
3.6/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation
-models) (2.4.1)
Requirement already satisfied: scipy>=0.19.0 in /usr/local/lib/python3.
6/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation-m
odels) (1.4.1)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.0 in /usr/local/
lib/python3.6/dist-packages (from scikit-image->efficientnet==1.0.0->se
gmentation-models) (3.2.2)
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/pyth
on3.6/dist-packages (from scikit-image->efficientnet==1.0.0->segmentati
on-models) (1.1.1)
Requirement already satisfied: pillow>=4.3.0 in /usr/local/lib/python3.
6/dist-packages (from scikit-image->efficientnet==1.0.0->segmentation-m
odels) (7.0.0)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-pac
kages (from h5py->keras-applications<=1.0.8,>=1.0.7->segmentation-model
s) (1.12.0)
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/pytho
n3.6/dist-packages (from networkx>=2.0->scikit-image->efficientnet==1.
0.0->segmentation-models) (4.4.2)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/p
ython3.6/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->e
fficientnet==1.0.0->segmentation-models) (2.8.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.
6/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->efficien
tnet==1.0.0->segmentation-models) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1
in /usr/local/lib/python3.6/dist-packages (from matplotlib!=3.0.0,>=2.
```

```
0.0->scikit-image->efficientnet==1.0.0->segmentation-models) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/pyth
on3.6/dist-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image->effi
cientnet==1.0.0->segmentation-models) (1.2.0)
Installing collected packages: image-classifiers, efficientnet, segment
ation-models
Successfully installed efficientnet-1.0.0 image-classifiers-1.0.0 segme
ntation-models-1.0.1
```

In [ ]:
```
import segmentation_models as sm
from segmentation_models import Unet
from segmentation_models.metrics import iou_score
focal_loss = sm.losses.cce_dice_loss
import tensorflow as tf
from tensorflow.keras import backend as K
from PIL import Image
from sklearn.model_selection import train_test_split
```

```
Using TensorFlow backend.
```

```
Segmentation Models: using `keras` framework.
```

In [ ]:
```
```

In [ ]:
```
from sklearn.model_selection import train_test_split
```

In [ ]:
```
X_train,X_test=train_test_split(data_df,test_size=.2,random_state=42)
```

In [ ]:
```
import tensorflow as tf
from tensorflow.keras import backend as K
import segmentation_models as sm
from tensorflow.keras.layers import Input, Lambda, Flatten,Conv2D,Dense
,MaxPooling2D,Dropout,BatchNormalization,GlobalAveragePooling2D
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.callbacks import ModelCheckpoint,TensorBoard
import datetime
from tensorflow.keras.models import Model
import tensorflow
import keras
from random import random
from segmentation_models import Unet
import segmentation_models as sm
from segmentation_models.metrics import iou_score
from segmentation_models import Unet
from tensorflow.keras.initializers import glorot_uniform
%load_ext tensorboard
```

In [ ]:

```
class convolutional_block(tf.keras.layers.Layer):
    def __init__(self, kernel,stride,  filters):
        super().__init__()
        self.F1, self.F2, self.F3 = filters
        self.kernel = kernel
        self.s=stride
        self.con1=Conv2D(filters = self.F3, kernel_size = (1,1), stride
s = (self.s,self.s), padding = 'valid')
        self.bn1=BatchNormalization(axis = 3)
        self.con2=Conv2D(filters = self.F1, kernel_size = (1, 1), strid
es = (self.s,self.s), padding = 'valid')
        self.bn2=BatchNormalization(axis = 3)
        self.con3=Conv2D(filters = self.F2, kernel_size = (3, 3), strid
es = (1,1), padding = 'same')
        self.bn3=BatchNormalization(axis = 3)
        self.con4=Conv2D(filters = self.F3, kernel_size = (1, 1), strid
es = (1,1), padding = 'valid')
        self.bn4=BatchNormalization(axis = 3)
        self.add=Add()
        self.act=Activation('relu')
    def get_config(self):

        config = super().get_config().copy()
        config.update({
            'kernel': self.kernel,
            'con1': self.con1,
```

```
            'bn1': self.bn1,
            'con2': self.con2,
            'bn2': self.bn2,
            'con3': self.con3,
            'bn3':self.bn3,
            'con4':self.con4,
            'bn4':self.bn4,
            'add':self.add,
            'act':self.act,
            's':self.s,
            'F1':self.F1,
            'F2':self.F2,
            'F3':self.F3
        })
        return config
    def call(self, X):
        X_=X
        X_ = self.con1(X_)
        X_ = self.bn1(X_)
        X_ = self.act(X_)

        X = self.con2(X)
        X = self.bn2(X)
        X = self.act(X)

        X = self.con3(X)
        X = self.bn3(X)
        X = self.act(X)

        X = self.con4(X)
        X = self.bn4(X)
        X = self.add([X, X_])

        X = self.act(X)
        return X
```

In [ ]:
```
class identity_block(tf.keras.layers.Layer):
    def __init__(self, kernel,  filters):
        super().__init__()
```

```
        self.F1, self.F2, self.F3 = filters
        self.kernel = kernel
        self.con1=Conv2D(filters = self.F1, kernel_size = (1, 1), strid
es = (1,1), padding = 'valid')
        self.bn1=BatchNormalization(axis=3)
        self.con2=Conv2D(filters = self.F2, kernel_size = (3, 3), strid
es = (1,1), padding = 'same')
        self.bn2=BatchNormalization(axis = 3)
        self.con3=Conv2D(filters = self.F3, kernel_size = (1, 1), strid
es = (1,1), padding = 'valid')
        self.bn3=BatchNormalization(axis = 3)
        self.act=Activation('relu')
        self.add=Add()
    def get_config(self):

        config = super().get_config().copy()
        config.update({
            'kernel': self.kernel,
            'con1': self.con1,
            'bn1': self.bn1,
            'con2': self.con2,
            'bn2': self.bn2,
            'con3': self.con3,
            'bn3':self.bn3,
            'add':self.add,
            'act':self.act,
            'F1':self.F1,
            'F2':self.F2,
            'F3':self.F3
        })
        return config

    def call(self, X):

        X_=X

        X = self.con1(X)
        X = self.bn1(X)
        X = self.act(X)
```

```
        X = self.con2(X)
        X = self.bn2(X)
        X = self.act(X)

        X = self.con3(X)
        X = self.bn3(X)

        X = self.add([X, X_])

        X = self.act(X)

        return X
```

In [ ]:
```
class global_flow(tf.keras.layers.Layer):
    def __init__(self):
        super().__init__()
        self.gap=GlobalAveragePooling2D()
        self.batch=BatchNormalization()
        self.act=Activation('relu')
        self.con=Conv2D(64, (1, 1), activation = 'relu', padding = 'sam
e')
        self.ct=Conv2DTranspose(64 ,(32,32), use_bias = False)
    def call(self, X):
        # implement the global flow operatiom
        GAP = self.gap(X)
        BN = self.batch(tf.reshape(GAP, shape=[tf.shape(GAP)[0],1,1,GAP
.shape[1]]))
        A = self.act(BN)
        CON = self.con(A)
        S = self.ct(CON)
        X=S
        return X
```

In [ ]:
```
class context_flow(tf.keras.layers.Layer):
    def __init__(self):
        super().__init__()
        self.coc=Concatenate()
```

```
        self.ap=AveragePooling2D((2, 2), (2, 2))
        self.con1=Conv2D(32, (3, 3), activation = 'relu', padding = 'sa
me' )
        self.con2=Conv2D(32, (3, 3), activation = 'relu', padding = 'sa
me')
        self.con3=Conv2D(32, (1, 1), activation = 'relu', padding = 'sa
me' )
        self.act1= Activation('relu')
        self.con4= Conv2D(32, (1, 1), activation = 'relu', padding = 's
ame' )
        self.act2=Activation('sigmoid')
        self.mul=Multiply()
        self.add=Add()
        self.ct=Conv2DTranspose(64 ,  (17,17), use_bias = False )
    def call(self, X):
        # here X will a list of two elements
        INP, FLOW = X[0], X[1]

        # implement the context flow as mentioned in the above cell
        C = self.coc([INP,FLOW])
        AP = self.ap(C)
        CON1 = self.con1(AP)
        CON2 = self.con2(CON1)
        CON3 = self.con3(CON2)
        A = self.act1(CON3)
        CON4 = self.con4(A)
        A = self.act2(CON4)
        MUL = self.mul([CON2 , CON4])
        ADD = self.add([CON2,MUL])
        ST = self.ct(ADD)

        X=ST
        return X
```

In [ ]:

```
class fsm(tf.keras.layers.Layer):
    def __init__(self):
        super().__init__()
        self.add=Add()
        self.con1=Conv2D(64,(3,3), activation = 'relu', padding = 'sam
```

```
e')
        self.gap=GlobalAveragePooling2D()
        self.con2=Conv2D(64,(1,1), activation = 'relu', padding = 'sam
e')
        self.bn=BatchNormalization()
        self.act=Activation('sigmoid')
        self.mul=Multiply()
        self.ups=UpSampling2D((2,2), interpolation='bilinear')
    def call(self, X):
        GF, CF1, CF2, CF3=X
        ADD = self.add([GF, CF1, CF2, CF3])
        C1 = self.con1(ADD)
        AP = self.gap(C1)
        C2 = self.con2((tf.reshape(AP, shape=[tf.shape(AP)[0],1,1,AP.sh
ape[1]])))
        BN = self.bn(C2)
        A = self.act(BN)
        M = self.mul([C1,A])
        US = self.ups(M)
        X=US
        # implement the FSM modules based on image in the above cells
        return X
```

In [ ]:

```
class agcn(tf.keras.layers.Layer):
    def __init__(self):
        super().__init__()
        self.con1=Conv2D(64,(7,1),strides=(1,1), padding = 'same')
        self.con2=Conv2D(64,(1,7),strides=(1,1) , padding = 'same')
        self.con3=Conv2D(64,(7,1),strides=(1,1) , padding = 'same')
        self.con4=Conv2D(64,(1,7),strides=(1,1) , padding = 'same')
        self.add=Add()
        self.con=Conv2D(64,(1,7),strides=(1,1) , padding = 'same')

    def call(self, X):
        C1 = self.con1(X)
        C2 = self.con2(C1)
        C1_ = self.con3(X)
        C2_ = self.con4(C1_)
        AD=self.add([C2,C2_])
```

```
        C3 = self.con(AD)
        Ad=self.add([AD,C3])
        X=Ad
        return X
```

In [ ]:

In [ ]:

In [ ]:
```
import imgaug.augmenters as iaa
aug2 = iaa.Fliplr(1)
aug3 = iaa.Flipud(1)
aug4 = iaa.Emboss(alpha=(1), strength=1)
aug5 = iaa.DirectedEdgeDetect(alpha=(0.8), direction=(1.0))
aug6 = iaa.Sharpen(alpha=(1.0), lightness=(1.5))
```

In [ ]:
```
X_train.head()
```

Out[ ]:

	images	json	
2473	data/images/338/frame55835_leftImg8bit.jpg	data/mask/338/frame55835_gtFine_polygons.json	data/ou
1338	data/images/275/frame17353_leftImg8bit.jpg	data/mask/275/frame17353_gtFine_polygons.json	data/ou
1613	data/images/285/frame1566_leftImg8bit.jpg	data/mask/285/frame1566_gtFine_polygons.json	data/
1610	data/images/285/frame0336_leftImg8bit.jpg	data/mask/285/frame0336_gtFine_polygons.json	data/
2600	data/images/348/frame7104_leftImg8bit.jpg	data/mask/348/frame7104_gtFine_polygons.json	data/

In [ ]:
```
def visualize(**images):
    n = len(images)
    plt.figure(figsize=(16, 5))
    for i, (name, image) in enumerate(images.items()):
        plt.subplot(1, n, i + 1)
        plt.xticks([])
        plt.yticks([])
        plt.title(' '.join(name.split('_')).title())
```

```
        if i==1:
            plt.imshow(image, cmap='gray', vmax=1, vmin=0)
        else:
            plt.imshow(image)
    plt.show()

def normalize_image(mask):
    mask = mask/255
    return mask

class Dataset:

    def __init__(self,data):

        #self.ids = file_names
        # the paths of images
        self.images_fps   = data['images'].tolist()
        # the paths of segmentation images
        self.masks_fps    = data['mask'].tolist()
        # giving labels for each class
        self.class_values = CLASSES

    def __getitem__(self, i):

        # read data
        image = cv2.imread(self.images_fps[i], cv2.IMREAD_UNCHANGED)
        image = cv2.resize(image,(512,512),interpolation = cv2.INTER_NE
AREST)
        mask  = cv2.imread(self.masks_fps[i], cv2.IMREAD_UNCHANGED)
        mask = cv2.resize(mask,(512,512),interpolation = cv2.INTER_NEAR
EST)
        #image_mask = normalize_image(mask)


        image_masks = [(mask == v) for v in self.class_values]
        image_mask = np.stack(image_masks, axis=-1).astype('float')

        a = np.random.uniform()
        if a<0.2:
```

```
            image = aug2.augment_image(image)
            image_mask = aug2.augment_image(image_mask)
        elif a<0.4:
            image = aug3.augment_image(image)
            image_mask = aug3.augment_image(image_mask)
        elif a<0.6:
            image = aug4.augment_image(image)
            image_mask = aug4.augment_image(image_mask)
        elif a<0.8:
            image = aug5.augment_image(image)
            image_mask = image_mask
        else:
            image = aug6.augment_image(image)
            image_mask = aug6.augment_image(image_mask)

        return image, image_mask

    def __len__(self):
        return len(self.images_fps)


class Dataloder(tf.keras.utils.Sequence):
    def __init__(self, dataset, batch_size=1, shuffle=False):
        self.dataset = dataset
        self.batch_size = batch_size
        self.shuffle = shuffle
        self.indexes = np.arange(len(dataset))

    def __getitem__(self, i):

        # collect batch data
        start = i * self.batch_size
        stop = (i + 1) * self.batch_size
        data = []
        for j in range(start, stop):
            data.append(self.dataset[j])

        batch = [np.stack(samples, axis=0) for samples in zip(*data)]
```

```
        return tuple(batch)

    def __len__(self):
        return len(self.indexes) // self.batch_size

    def on_epoch_end(self):
        if self.shuffle:
            self.indexes = np.random.permutation(self.indexes)
```

In [ ]:
```
# Dataset for train images
import keras
CLASSES = list(np.unique(list(label_clr.values())))
train_dataset = Dataset(X_train)
test_dataset  = Dataset(X_test)

BATCH_SIZE=12

train_dataloader = Dataloder(train_dataset, batch_size=12, shuffle=True
)
test_dataloader = Dataloder(test_dataset, batch_size=12, shuffle=True)

print(train_dataloader[0][0].shape)
assert train_dataloader[0][0].shape == (BATCH_SIZE, 512, 512, 3)
assert train_dataloader[0][1].shape == (BATCH_SIZE, 512, 512, 21)
```

```
(12, 512, 512, 3)
```

In [ ]:
```
X_input = Input(shape=(512,512,3))

# Stage 1
X = Conv2D(64, (3, 3), name='conv1', padding="same", kernel_initializer
=glorot_uniform(seed=0))(X_input)
X = BatchNormalization(axis=3, name='bn_conv1')(X)
X = Activation('relu')(X)
X = MaxPooling2D((2, 2), strides=(2, 2))(X)

X1=convolutional_block(kernel=3, filters=[4, 4, 8],stride=4)(X)
X1=identity_block(kernel=3,filters=[4,4,8])(X1)
```

```
X=convolutional_block(kernel=3, filters=[8, 8, 16],stride=2)(X1)
X=identity_block(kernel=3,filters=[8, 8, 16])(X)
X=identity_block(kernel=3,filters=[8,8,16])(X)

X=convolutional_block(kernel=3, filters=[16, 16, 32],stride=1)(X)
X=identity_block(kernel=3,filters=[16, 16, 32])(X)
X=identity_block(kernel=3,filters=[16, 16, 32])(X)
X=identity_block(kernel=3,filters=[16, 16, 32])(X)

X=convolutional_block(kernel=3, filters=[32,32, 64],stride=1)(X)
X=identity_block(kernel=3,filters=[32,32, 64])(X)
X=identity_block(kernel=3,filters=[32,32, 64])(X)
X=identity_block(kernel=3,filters=[32,32, 64])(X)
X=identity_block(kernel=3,filters=[32,32, 64])(X)

GF=global_flow()(X)

CF1=context_flow()([X,GF])
CF2=context_flow()([X,CF1])
CF3=context_flow()([X,CF2])

FSM=fsm()([GF, CF1, CF2, CF3])

AGCN=agcn()(X1)

AF=Concatenate()([AGCN,FSM])
CON=Conv2D(21,(1,1),padding='same')(AF)
US = UpSampling2D((8,8), interpolation='bilinear')(CON)
out=keras.activations.sigmoid(US)
```

In [ ]:
```
model = Model(inputs = X_input, outputs = out)
model.summary()
```

```
Model: "model_1"
__________________________________________________________________________
_______________________________
Layer (type)                    Output Shape         Param #     Connec
ted to
==========================================================================
```

```
==========================
input_2 (InputLayer)            [(None, 512, 512, 3) 0

__________________________________________________________________________
_____________________________
conv1 (Conv2D)                  (None, 512, 512, 64) 1792        input_
2[0][0]
__________________________________________________________________________
_____________________________
bn_conv1 (BatchNormalization)   (None, 512, 512, 64) 256         conv1
[0][0]
__________________________________________________________________________
_____________________________
activation_23 (Activation)      (None, 512, 512, 64) 0           bn_con
v1[0][0]
__________________________________________________________________________
_____________________________
max_pooling2d_1 (MaxPooling2D)  (None, 256, 256, 64) 0           activa
tion_23[0][0]
__________________________________________________________________________
_____________________________
convolutional_block_4 (convolut (None, 64, 64, 8)    1064        max_po
oling2d_1[0][0]
__________________________________________________________________________
_____________________________
identity_block_10 (identity_blo (None, 64, 64, 8)    288         convol
utional_block_4[0][0]
__________________________________________________________________________
_____________________________
convolutional_block_5 (convolut (None, 32, 32, 16)   1136        identi
ty_block_10[0][0]
__________________________________________________________________________
_____________________________
identity_block_11 (identity_blo (None, 32, 32, 16)   992         convol
utional_block_5[0][0]
__________________________________________________________________________
_____________________________
identity_block_12 (identity_blo (None, 32, 32, 16)   992         identi
ty_block_11[0][0]
```

```
____________________________________________________________________________
____________________________
convolutional_block_6 (convolut (None, 32, 32, 32)   4064        identi
ty_block_12[0][0]
____________________________________________________________________________
____________________________
identity_block_13 (identity_blo (None, 32, 32, 32)   3648        convol
utional_block_6[0][0]
____________________________________________________________________________
____________________________
identity_block_14 (identity_blo (None, 32, 32, 32)   3648        identi
ty_block_13[0][0]
____________________________________________________________________________
____________________________
identity_block_15 (identity_blo (None, 32, 32, 32)   3648        identi
ty_block_14[0][0]
____________________________________________________________________________
____________________________
convolutional_block_7 (convolut (None, 32, 32, 64)   15296       identi
ty_block_15[0][0]
____________________________________________________________________________
____________________________
identity_block_16 (identity_blo (None, 32, 32, 64)   13952       convol
utional_block_7[0][0]
____________________________________________________________________________
____________________________
identity_block_17 (identity_blo (None, 32, 32, 64)   13952       identi
ty_block_16[0][0]
____________________________________________________________________________
____________________________
identity_block_18 (identity_blo (None, 32, 32, 64)   13952       identi
ty_block_17[0][0]
____________________________________________________________________________
____________________________
identity_block_19 (identity_blo (None, 32, 32, 64)   13952       identi
ty_block_18[0][0]
____________________________________________________________________________
____________________________
global_flow_1 (global_flow)     (None, 32, 32, 64)   4198720     identi
```

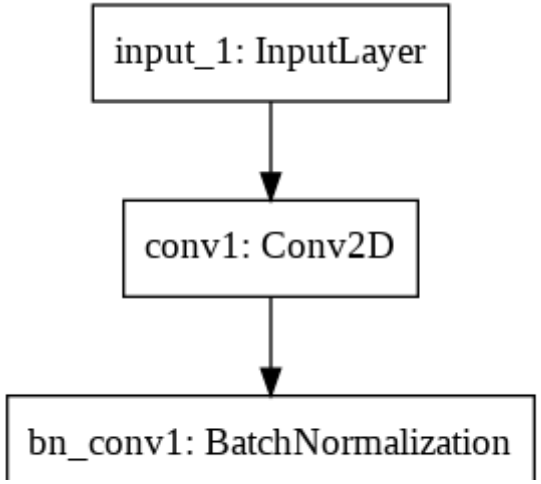
```
ty_block_19[0][0]
__________________________________________________________________________
__________________________
context_flow_3 (context_flow)   (None, 32, 32, 64)   640128      identi
ty_block_19[0][0]
                                                                 global
_flow_1[0][0]
__________________________________________________________________________
__________________________
context_flow_4 (context_flow)   (None, 32, 32, 64)   640128      identi
ty_block_19[0][0]
                                                                 contex
t_flow_3[0][0]
__________________________________________________________________________
__________________________
context_flow_5 (context_flow)   (None, 32, 32, 64)   640128      identi
ty_block_19[0][0]
                                                                 contex
t_flow_4[0][0]
__________________________________________________________________________
__________________________
agcn_1 (agcn)                   (None, 64, 64, 64)   93504       identi
ty_block_10[0][0]
__________________________________________________________________________
__________________________
fsm_1 (fsm)                     (None, 64, 64, 64)   41344       global
_flow_1[0][0]
                                                                 contex
t_flow_3[0][0]
                                                                 contex
t_flow_4[0][0]
                                                                 contex
t_flow_5[0][0]
__________________________________________________________________________
__________________________
concatenate_7 (Concatenate)     (None, 64, 64, 128)  0           agcn_1
[0][0]
                                                                 fsm_1
[0][0]
```

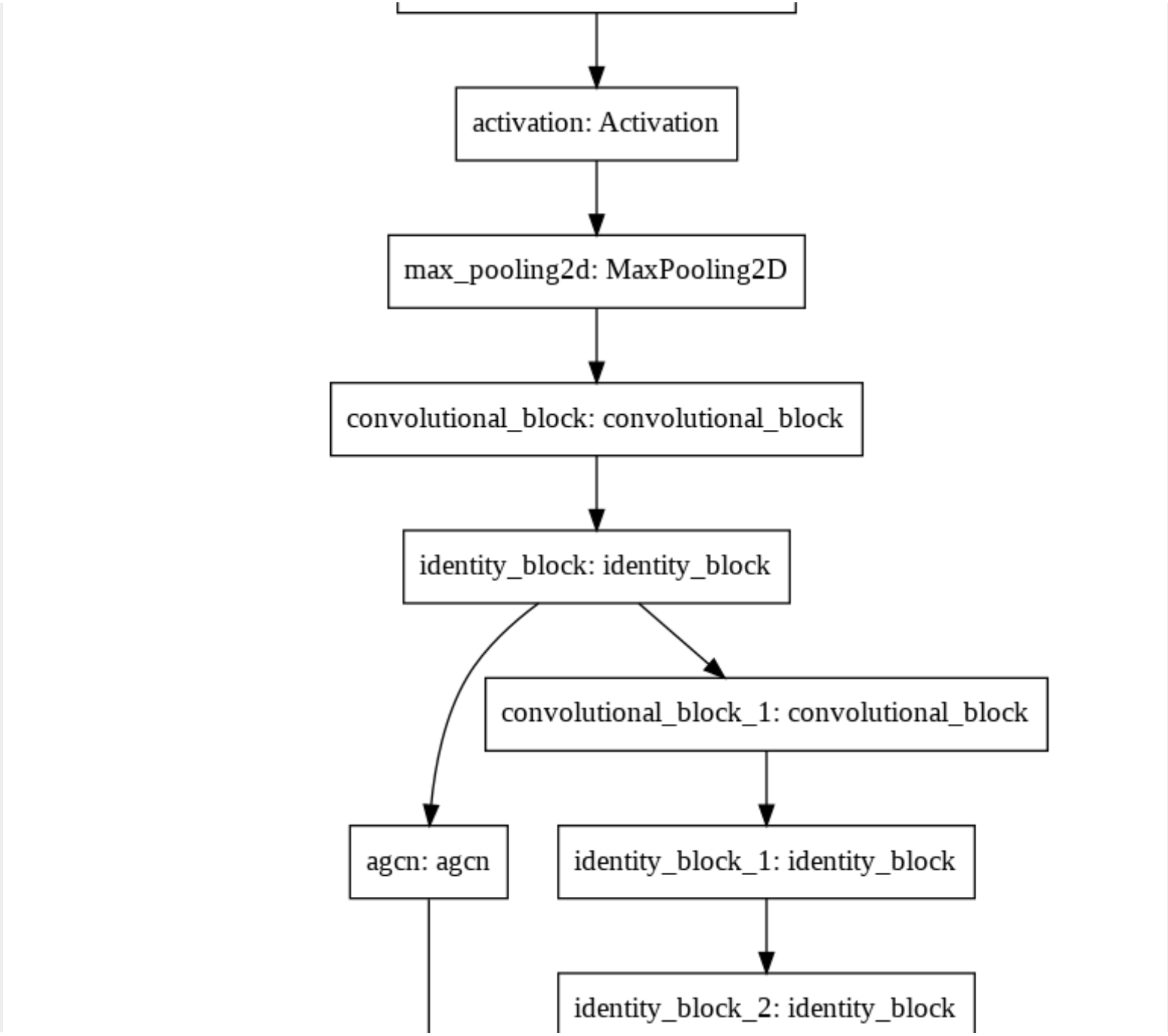
```
__________________________________________________________________________
_____________________________
conv2d_133 (Conv2D)             (None, 64, 64, 21)   2709        concat
enate_7[0][0]
__________________________________________________________________________
_____________________________
up_sampling2d_3 (UpSampling2D)  (None, 512, 512, 21) 0           conv2d
_133[0][0]
__________________________________________________________________________
_____________________________
tf_op_layer_Sigmoid_1 (TensorFl [(None, 512, 512, 21 0           up_sam
pling2d_3[0][0]
==========================================================================
============================
Total params: 6,349,293
Trainable params: 6,346,621
Non-trainable params: 2,672
__________________________________________________________________________
_____________________________
```

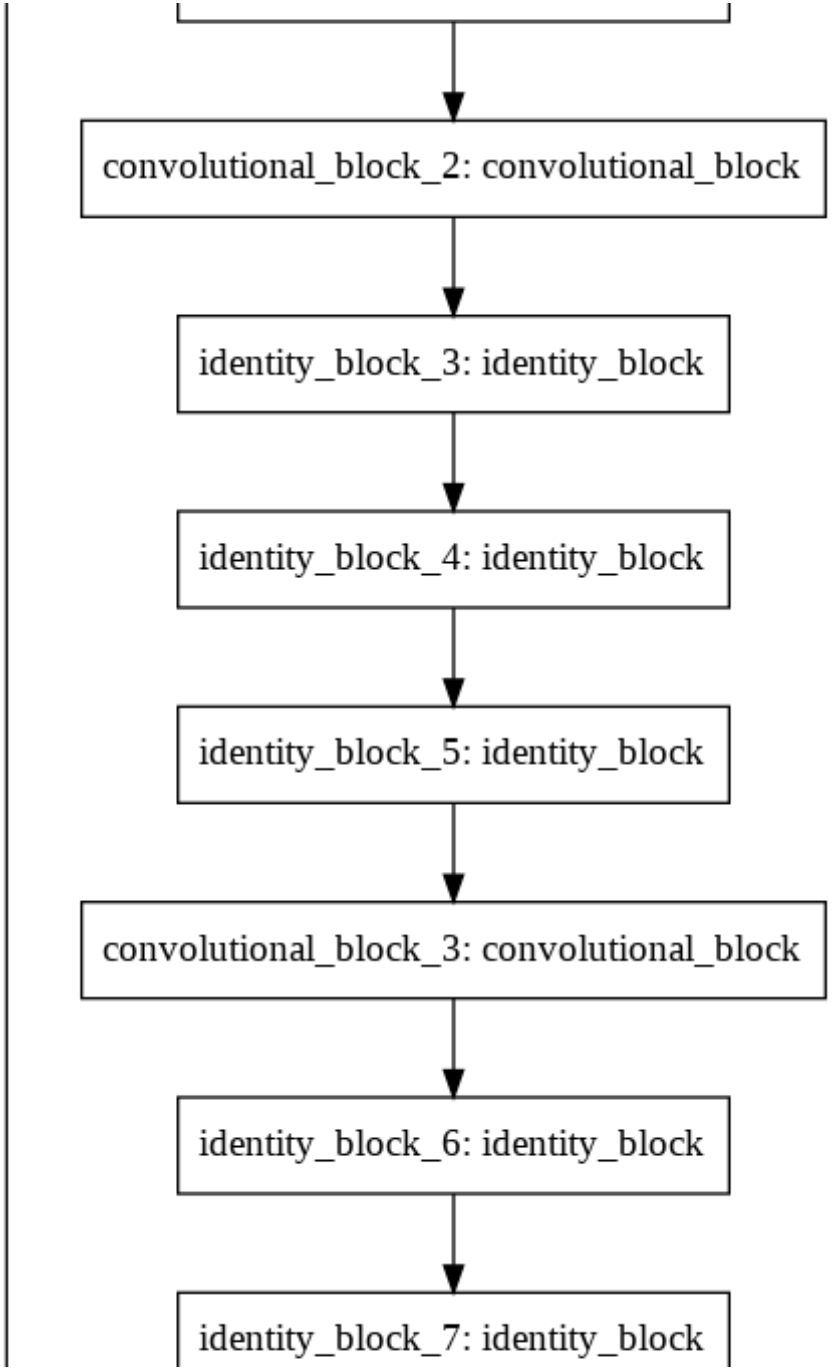
In [ ]:
```
#tf.keras.utils.plot_model(model, to_file='model.png', show_shapes=Fals
e, show_layer_names=True,rankdir='TB', expand_nested=False, dpi=96)
```

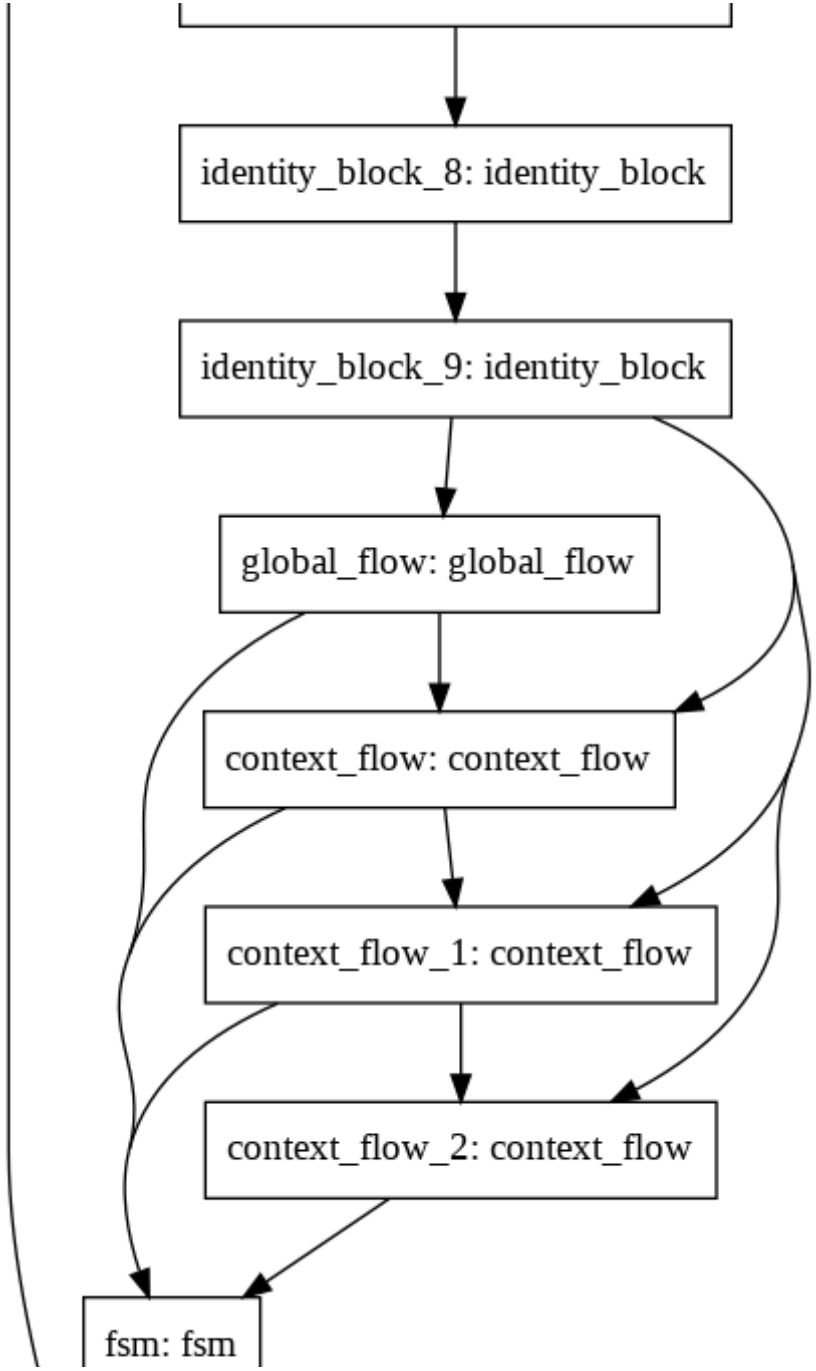
In [ ]:
```
Image.open('model.png')
```

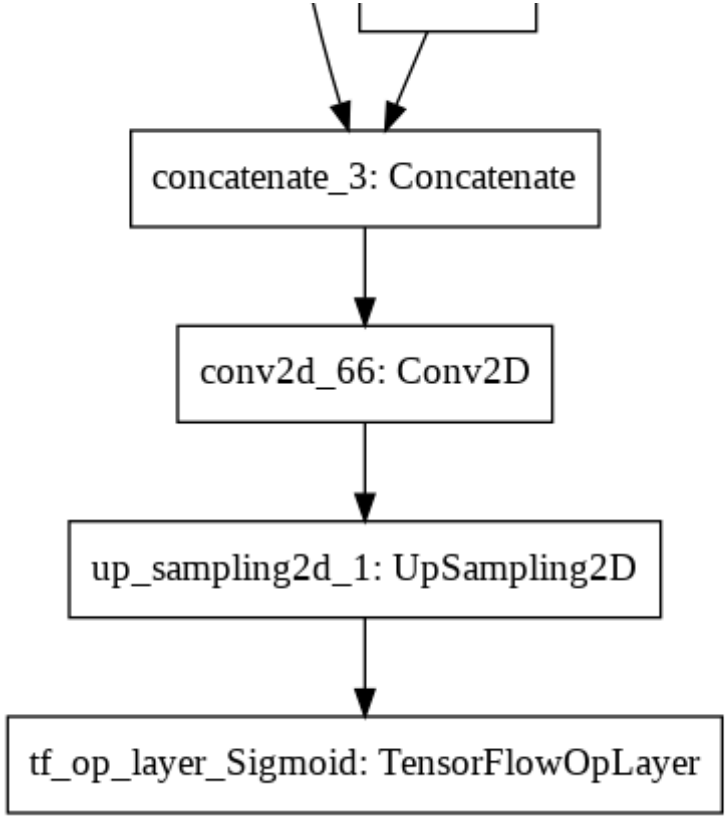
Out[ ]:











In [ ]:
```
from segmentation_models.metrics import iou_score
```

In [ ]:
```
focal_loss = sm.losses.cce_dice_loss
dice_loss=sm.losses.dice_loss
RMS = tf.keras.optimizers.RMSprop()
model.compile(RMS, focal_loss, metrics=[iou_score])
```

In [ ]:
```
filepath='/content/drive/My Drive/seg_model2/'+"weights-{epoch:02d}-{val_iou_score:.4f}.hdf5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_iou_score',verbose=1, save_best_only=True, mode='max')
log_dir="/content/drive/My Drive/seg_model2/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
```

```
tensorboard_callback = TensorBoard(log_dir=log_dir,histogram_freq=0, wr
ite_graph=True,write_grads=False)
po=[tensorboard_callback,checkpoint]
```

In [ ]:
```
X_test.shape[0]/10,len(train_dataloader)
```

Out[ ]:
```
(80.2, 267)
```

In [ ]:
```
history = model.fit_generator(train_dataloader, epochs=14, validation_d
ata=test_dataloader,callbacks=po)
#By mistake i ran this code after my whole model got train thatwhy my a
ll training result gone.
```

In [ ]:
```
po='/content/drive/My Drive/seg_model2/weights-11-0.4424.hdf5'
model.load_weights(po)
```

evaluate all the image

In [ ]:
```
pd.DataFrame([model.evaluate(test_dataloader)],columns=['test_loss','te
st_iou_score'])
```

```
66/66 [==============================] - 106s 2s/step - loss: 0.5374 -
iou_score: 0.4376
```

Out[ ]:

	test_loss	test_iou_score
0	0.537449	0.437567

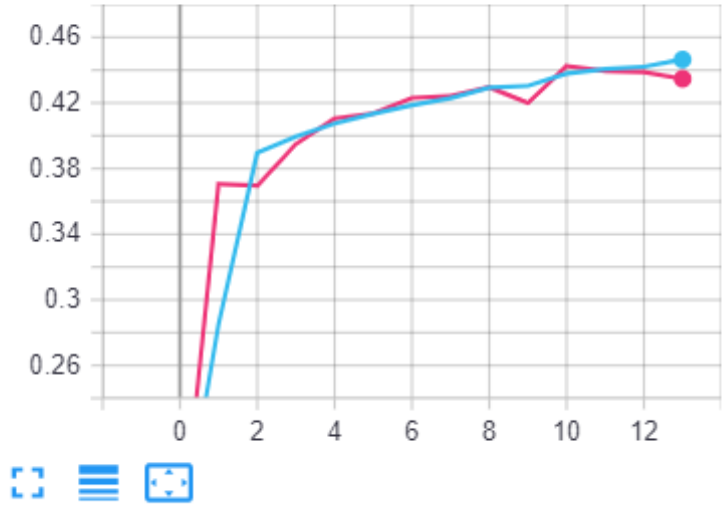
In [ ]:
```
%tensorboard --logdir .
#!kill 404
```

In [ ]:
```
from PIL import Image
```

In [ ]:
```
Image.open('sm2.png')
```

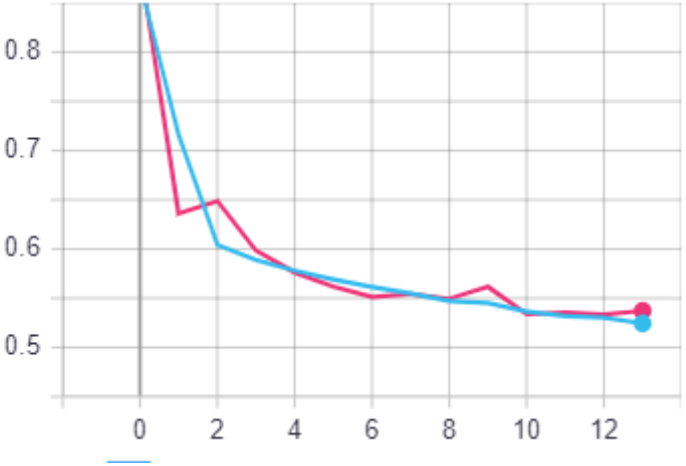
Out[ ]:

epoch_iou_score



epoch_loss

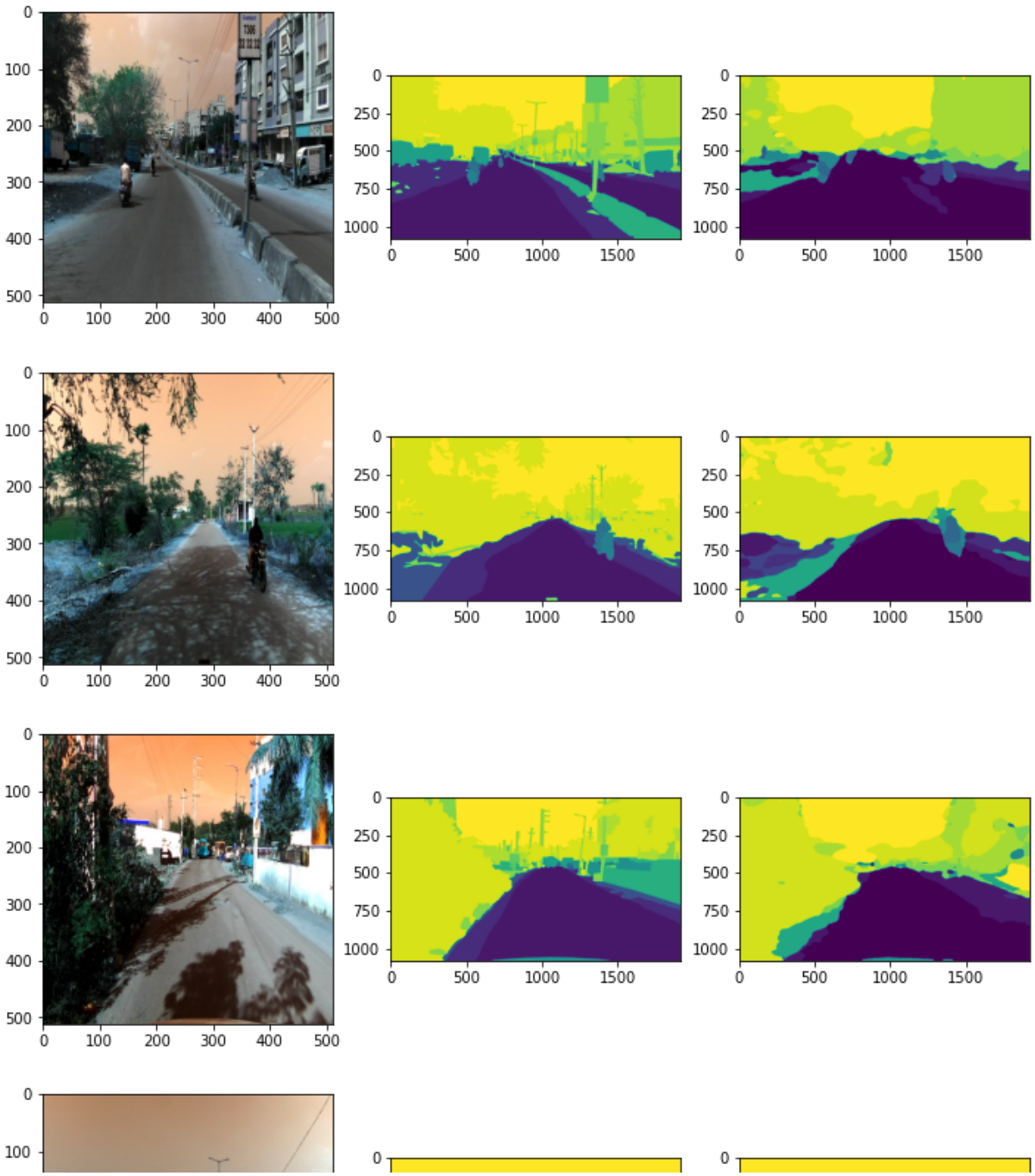
epoch_loss

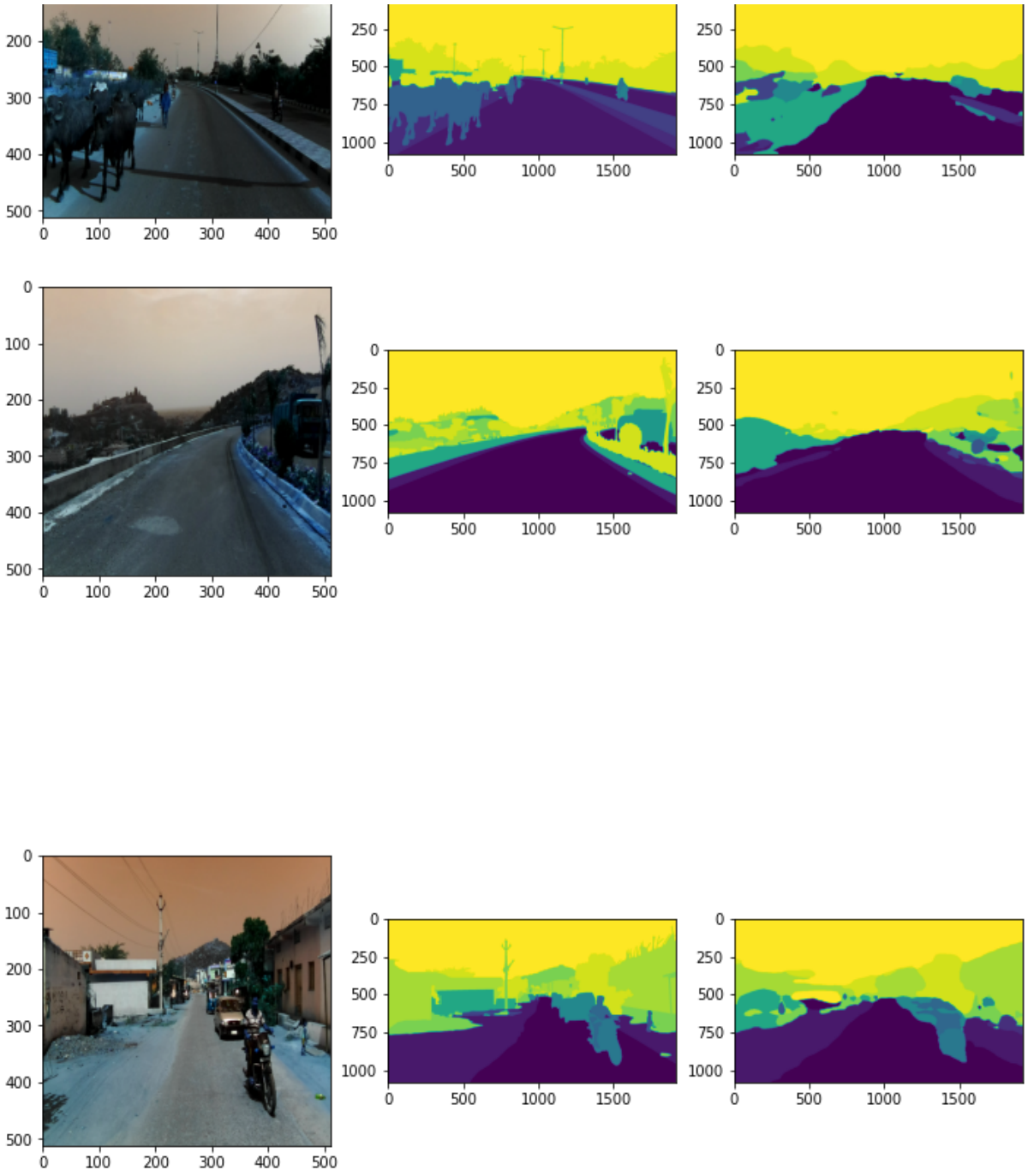


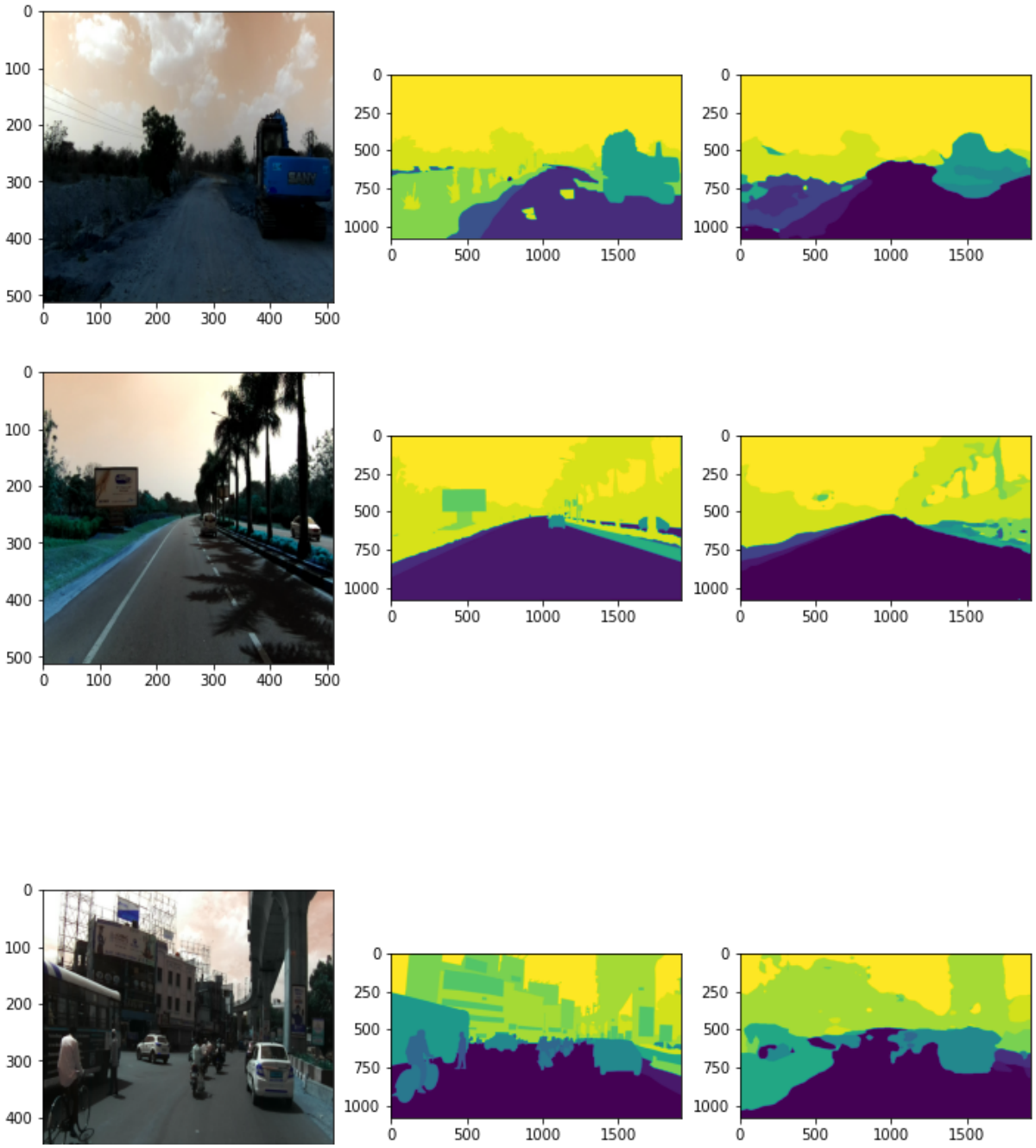
In [ ]:

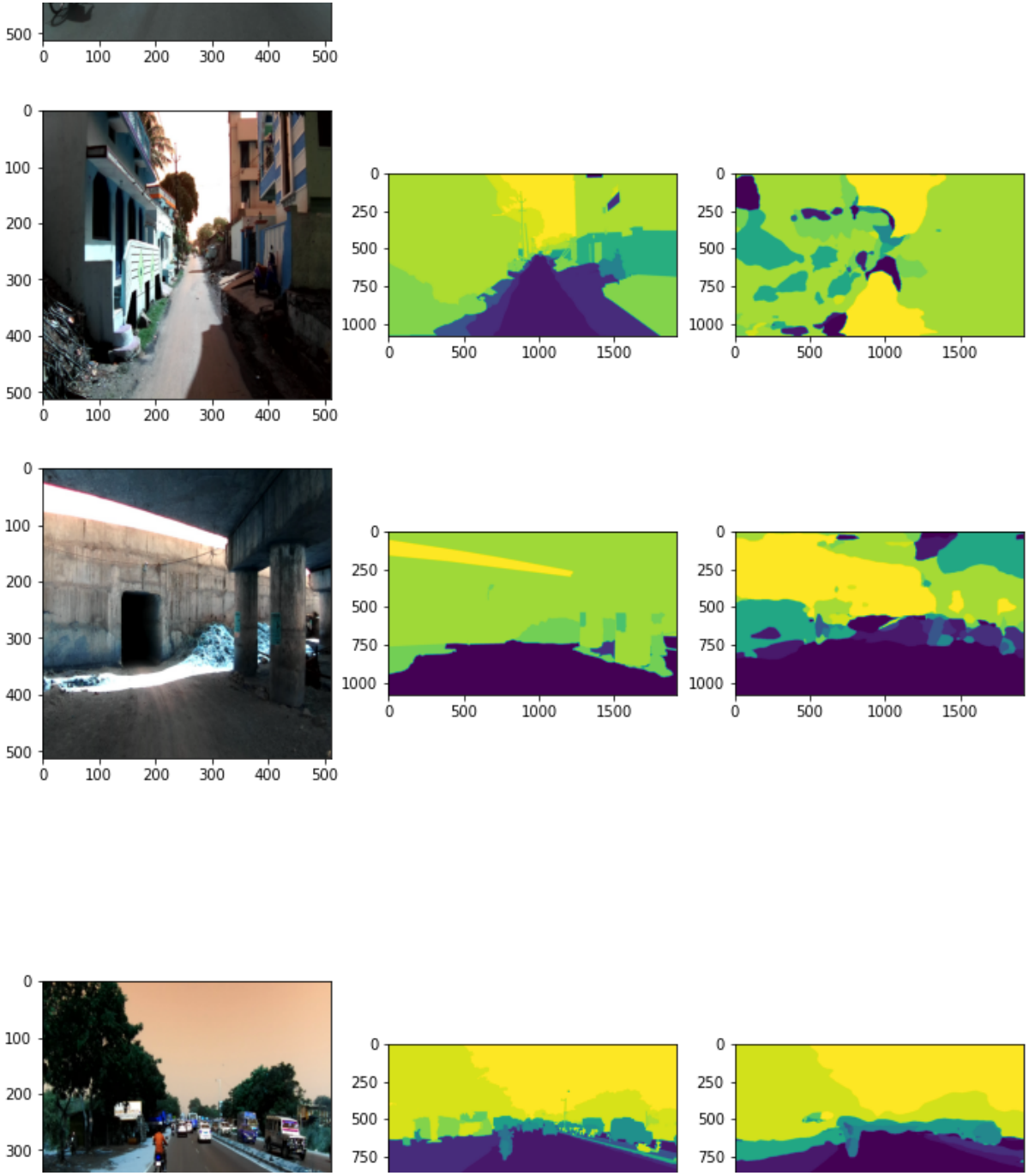
Visualisation

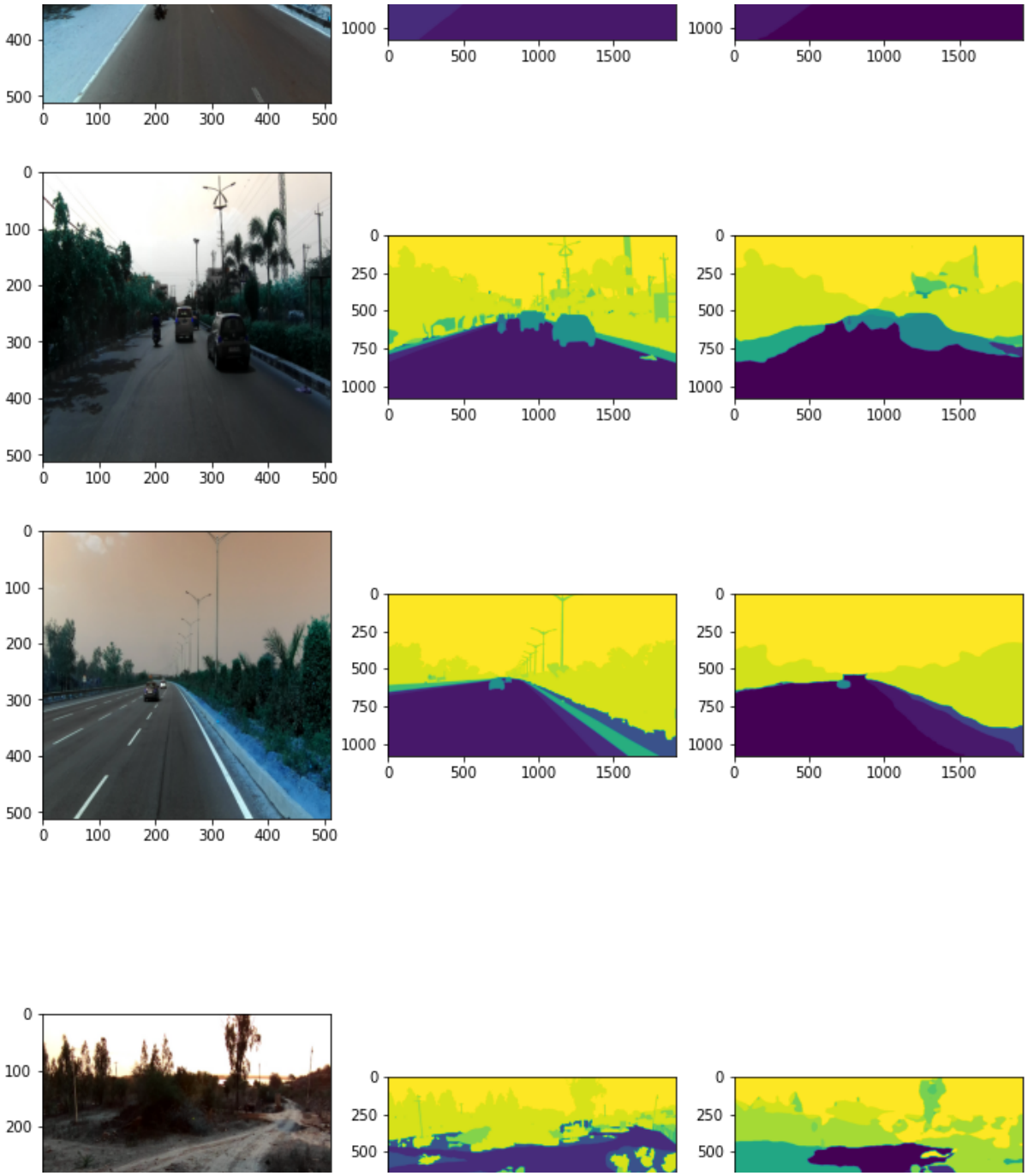
In [ ]:
```
for p, i in enumerate(X_test.values):
    #original image

    image = cv2.imread(i[0], cv2.IMREAD_UNCHANGED)
    image = cv2.resize(image, (512,512))

    #predicted segmentation map
    pre=model.predict(tf.expand_dims(image,0))
    predict = tf.argmax(pre, axis=-1)
    predict = tf.expand_dims(predict, axis=-1)
    predict=tf.image.resize(predict,size=(1080, 1920))
    predict=tf.reshape(predict,(1080, 1920))
    #original segmentation map
    image_mask = cv2.imread(i[2], cv2.IMREAD_UNCHANGED)


    plt.figure(figsize=(12,8))
    plt.subplot(131)
    plt.imshow(image)
    plt.subplot(132)
    plt.imshow(image_mask)
    plt.subplot(133)
    plt.imshow(predict)
    plt.show()
    if p==20:
      break
```

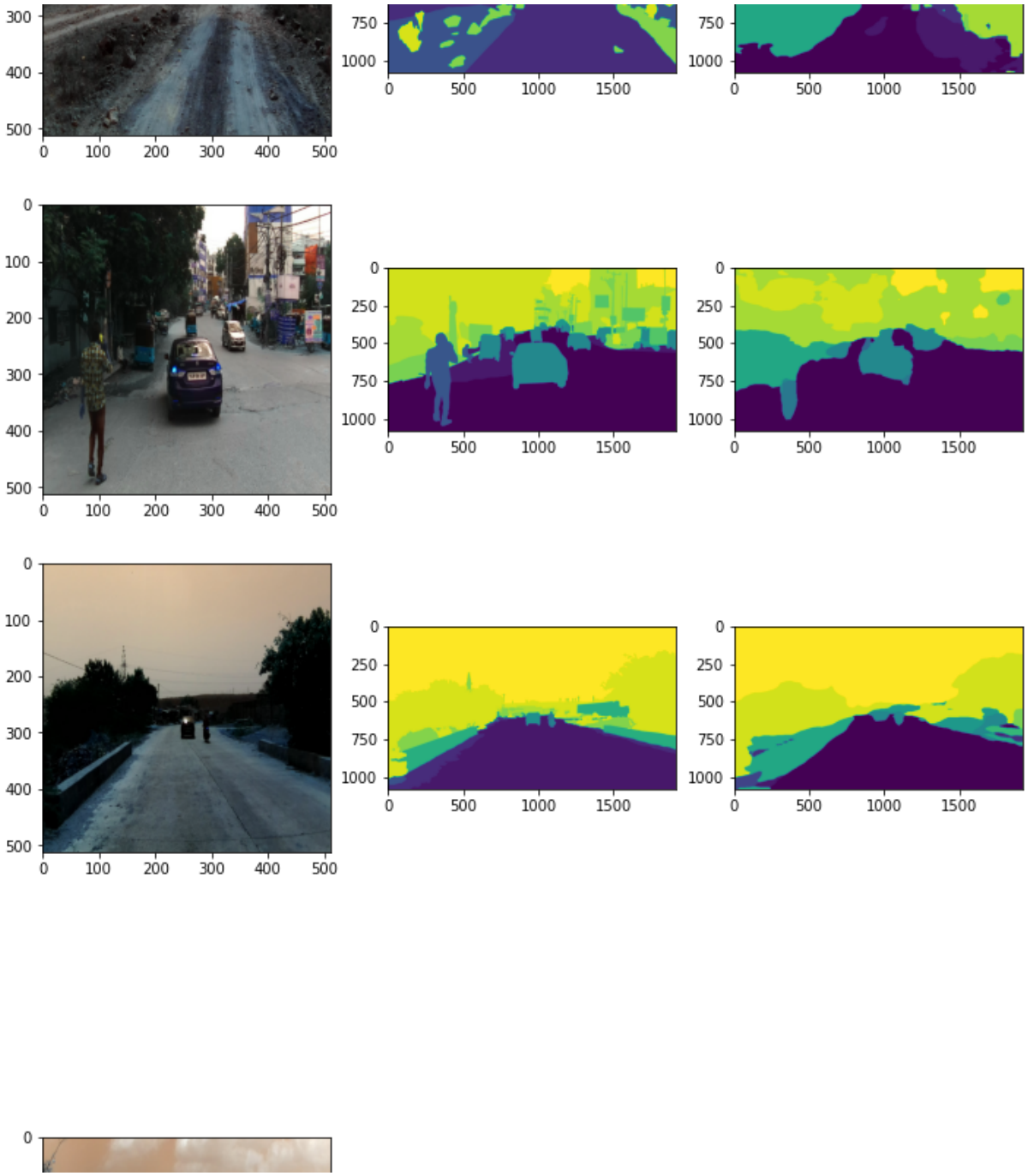


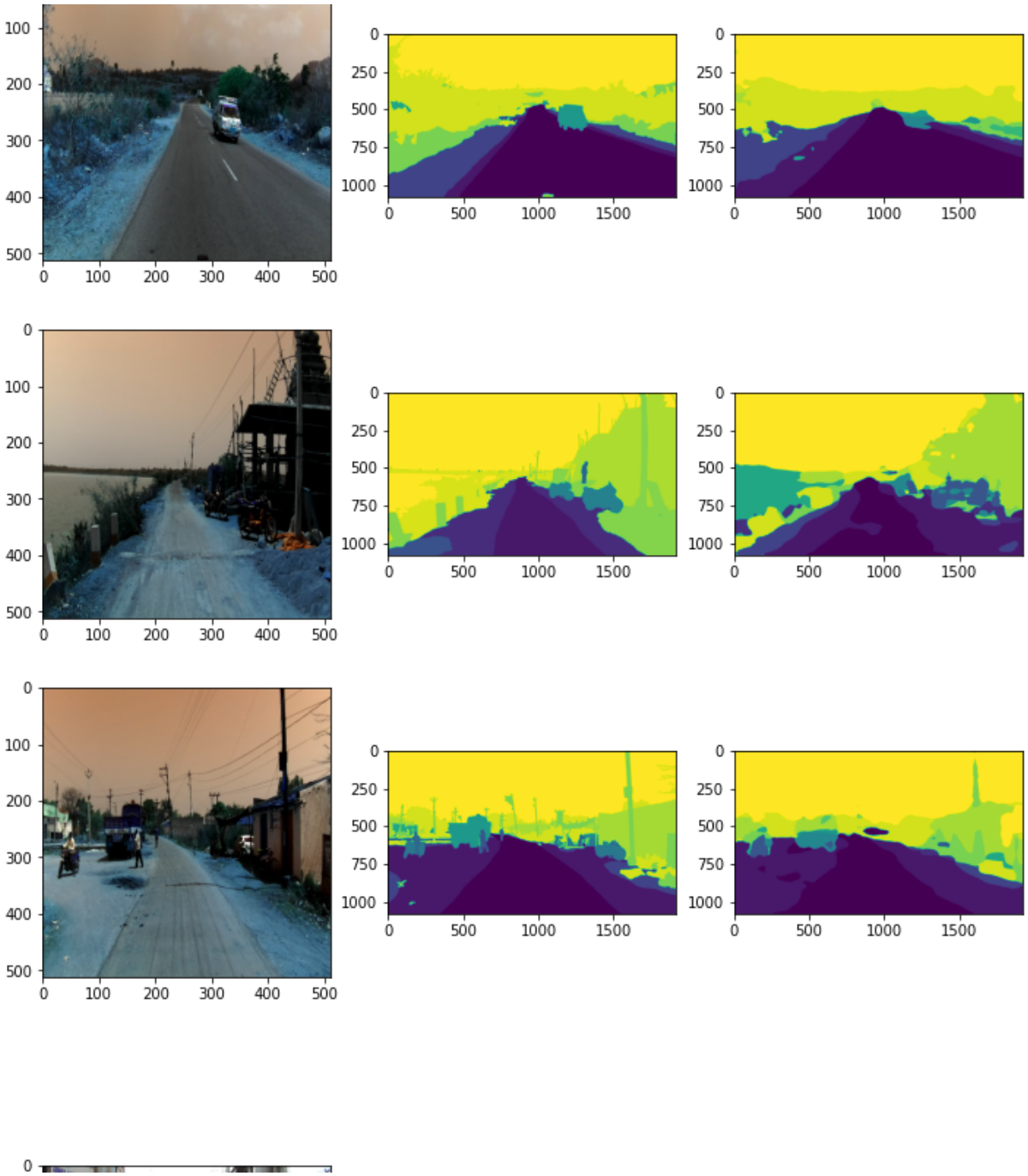


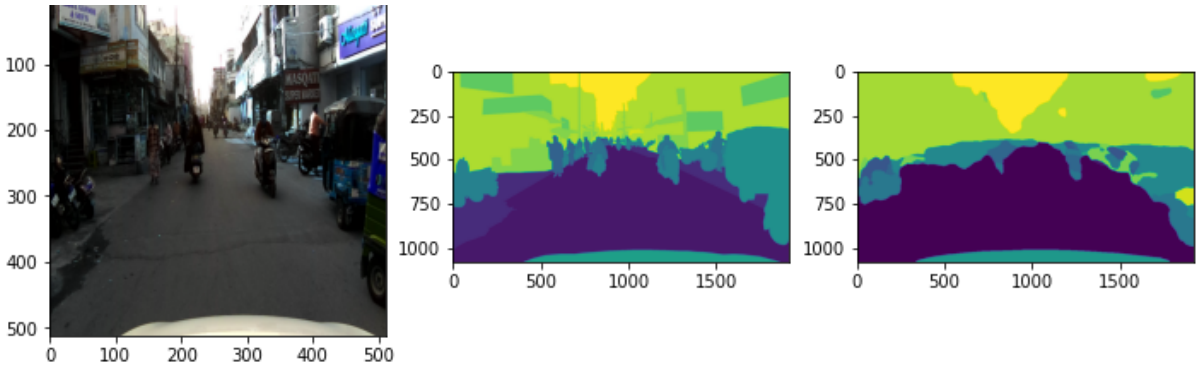












obseravtion

1. Unet is better than Canet because in unet light, bike,person,sky,roads are much clear.

In [1]:
```
from prettytable import PrettyTable
posa = PrettyTable()
posa.field_names = ["model", "IOU score"]
posa.add_row(['Unet',0.57])
posa.add_row(["Canet", 0.43])
print(posa)
```

```
+-------+-----------+
| model | IOU score |
+-------+-----------+
|  Unet |    0.57   |
| Canet |    0.43   |
+-------+-----------+
```

In [ ]:
```

```