New Section

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import zipfile
import cv2
from skimage import io

import tensorflow as tf
from tensorflow.python.keras import Sequential
from tensorflow.keras import layers, optimizers
from tensorflow.keras.layers import *
from tensorflow.keras.models import Model
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.utils import plot_model
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlySt
import tensorflow.keras.backend as K

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

import random
import glob
from sklearn.preprocessing import StandardScaler, normalize
from IPython.display import display

!gdown --id 1iQ93IWVdR6dZ6W7RahbLq166u-6ADelJ


!unzip -qn data.zip
```

```
Downloading...
From: https://drive.google.com/uc?id=1iQ93IWVdR6dZ6W7RahbLq166u-6ADelJ
To: /content/data.zip
2.34GB [00:23, 98.0MB/s]
```

```
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense,Input,Conv2D,MaxPool2D,
from tensorflow.keras.models import Model
import random as rn

!pip install imgaug
```

```
Requirement already satisfied: imgaug in /usr/local/lib/python3.6/dist-packages (0.2.9)
Requirement already satisfied: Shapely in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from imga
Requirement already satisfied: opencv-python in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: Pillow in /usr/local/lib/python3.6/dist-packages (from i
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: scikit-image>=0.11.0 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from im
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: imageio in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/l
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.6/dist-packag
```

```
#!rm -r '/content/data'
```

```
rm: cannot remove '/content/data': No such file or directory
```

```
#!cp -R "/content/drive/MyDrive/ImageSegmentation/data" "/content

from tensorflow.keras.layers import Flatten

def get_poly(file):

    f = open(file,)
    data = json.load(f)
    label,vertexlist=[],[]
    for obj in data['objects']:
        label.append(obj['label'])
        vertexlist.append([tuple(vertex) for vertex in obj['polyg
    w= data['imgWidth']
    h=data['imgHeight']

    return w, h, label, vertexlist



os.makedirs('/content/data/output')
def compute_masks(data_df):
    mask=[]
    for file in tqdm(data_df['json']):
        w, h, labels, vertexlist = get_poly(file)

        img = Image.new("RGB",(w,h))
        img1 = ImageDraw.Draw(img)
        for i in range(len(labels)):
            if(len(vertexlist[i])>1):
                img1.polygon(vertexlist[i], fill = label_clr[labe
        img=np.array(img)
        im = Image.fromarray(img[:,:,0])
        new_file=file.replace('mask','output')
        new_file=new_file.replace('json','png')
        os.makedirs('/content/data/output/'+file.split('/')[4],ex
        im.save(new_file)
        mask.append(new_file)
    data_df['mask']=mask
```

```
    data_df['mask']=mask
    return data_df


root_dir = '/content/data'

def return_file_names_df(root_dir):
    # write the code that will create a dataframe with two column
    # the column 'image' will have path to images
    # the column 'json' will have path to json files

    img_dir = root_dir + '/images'
    mask_dir = root_dir + '/mask'
    frames = []
    masks = []
    c = set()
    d = set()
    count = 0
    for img in os.listdir(img_dir):
      c.add(int(img))
    #print(c)
    for i in sorted(list(c)):
      for info in os.listdir(os.path.join(img_dir + '/',str(i))):
            frames.append(os.path.join(img_dir + '/'+str(i)+'/',i
            count+=1
    print(count)
    count = 0
    for img in os.listdir(mask_dir):
      d.add(img)
    for i in sorted(list(d)):
      for info in os.listdir(os.path.join(mask_dir + '/',str(i)))
            masks.append(os.path.join(mask_dir + '/'+str(i)+'/',i
            count += 1
    print(count)
    data_df = pd.DataFrame({'image':sorted(frames) , 'json':sorte
    #print(data_df.head())

    return data_df
```

```
data_df = return_file_names_df(root_dir)
```

```
4008
4008
```

```
data_df.head()
```

	image	jso
0	/content/data/images/201/frame0029_leftImg8bit...	/content/data/mask/201/frame0029_gtFine_polygo
1	/content/data/images/201/frame0299_leftImg8bit...	/content/data/mask/201/frame0299_gtFine_polygo
2	/content/data/images/201/frame0779_leftImg8bit...	/content/data/mask/201/frame0779_gtFine_polygo
3	/content/data/images/201/frame1019_leftImg8bit...	/content/data/mask/201/frame1019_gtFine_polygo
4	/content/data/images/201/frame1469_leftImg8bit...	/content/data/mask/201/frame1469_gtFine_polygo

```
label_clr = {'road':10, 'parking':20, 'drivable fallback':20,'sid
                        'person':50, 'animal':50, 'rider':60, 'mo
                        'car':80, 'truck':90, 'bus':90, 'vehicle
                        'curb':100, 'wall':100, 'fence':110,'guar
                        'traffic light':120, 'pole':130, 'polegro
                        'bridge':140,'tunnel':140, 'vegetation':1
                        'out of roi':0, 'ego vehicle':170, 'groun
                    'train':200}
```

```
data_df_ = compute_masks(data_df)
```

```
100%|██████████| 4008/4008 [04:06<00:00, 16.25it/s]
```

```
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][
            return False
    return True
```

```
grader_1(data_df)
```

```
True
```

```
def create_image_label_df(data_df):
```

```
def return_unique_labels(data_df):
    from tqdm import tqdm
    # for each file in the column json
    #       read and store all the objects present in that file
    # compute the unique objects and retrun them
    # if open any json file using any editor you will get better
    unique_labels = []
    for f in tqdm(data_df['json'].values):
      obj = json.load(open(f,'rb'))
      for lab in obj['objects']:
        unique_labels.append(lab['label'])
    return set(unique_labels)


unique_labels = return_unique_labels(data_df)
```

```
100%|██████████| 4008/4008 [00:29<00:00, 136.70it/s]
```

```
def grader_2(unique_labels):
    if (not (set(label_clr.keys())-set(unique_labels))) and len(u
        print("True")
    else:
        print("Flase")

grader_2(unique_labels)
```

```
True
```

```
def grader_3(file):
    w, h, labels, vertexlist = get_poly(file)
    print(len((set(labels)))==18 and len(vertexlist)==227 and w==
          and isinstance(vertexlist,list) and isinstance(vertexli

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

```
True
```

```
import math
from PIL import Image, ImageDraw
from PIL import ImagePath
side=8
```

```
x1 = [ ((math.cos(th) + 1) *9, (math.sin(th) + 1) * 6) for th in
x2 = [ ((math.cos(th) + 2) *9, (math.sin(th) + 3) *6) for th in [

img = Image.new("RGB", (28,28))
img1 = ImageDraw.Draw(img)
# please play with the fill value
# writing the first polygon
img1.polygon(x1, fill =20)
# writing the second polygon
img1.polygon(x2, fill =30)

img=np.array(img)
# note that the filling of the values happens at the channel 1, s
plt.imshow(img[:,:,0])
print(img.shape)
print(img[:,:,0]//10)
im = Image.fromarray(img[:,:,0])
im.save("test_image.png")
```

```
(28, 28, 3)
[[0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0]
 [0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0]
 [0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0]
 [2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0]
 [2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0]
 [0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0]
 [0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 3 3 3 3 3 3 3 3 3 3 3 3 0 0 0]
```

```
data_df_.head()
```

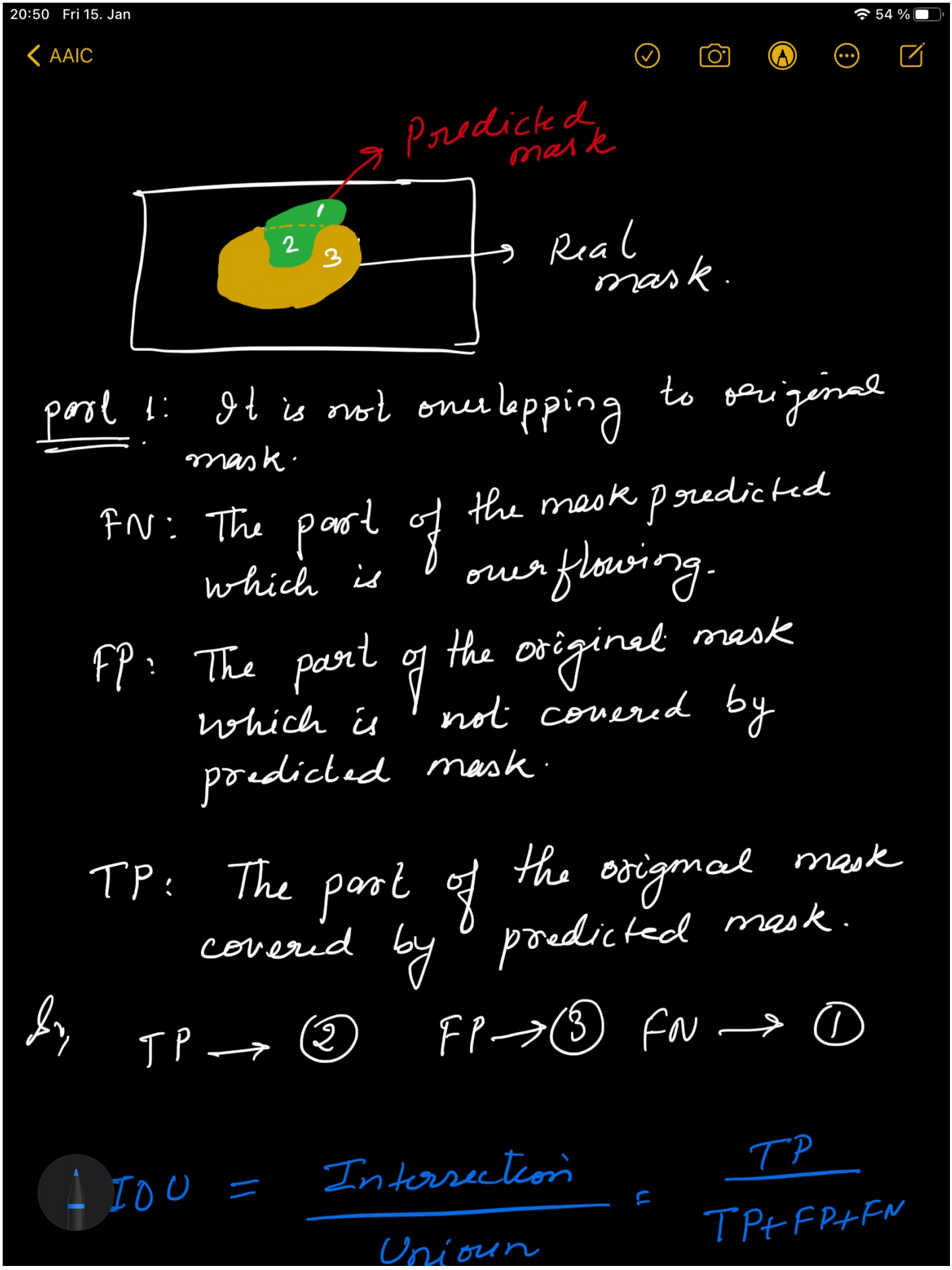
	image	jsc
0	/content/data/images/201/frame0029_leftImg8bit...	/content/data/mask/201/frame0029_gtFine_polygo
1	/content/data/images/201/frame0299_leftImg8bit...	/content/data/mask/201/frame0299_gtFine_polygo
2	/content/data/images/201/frame0779_leftImg8bit...	/content/data/mask/201/frame0779_gtFine_polygo
3	/content/data/images/201/frame1019_leftImg8bit...	/content/data/mask/201/frame1019_gtFine_polygo
4	/content/data/images/201/frame1469_leftImg8bit...	/content/data/mask/201/frame1469_gtFine_polygo

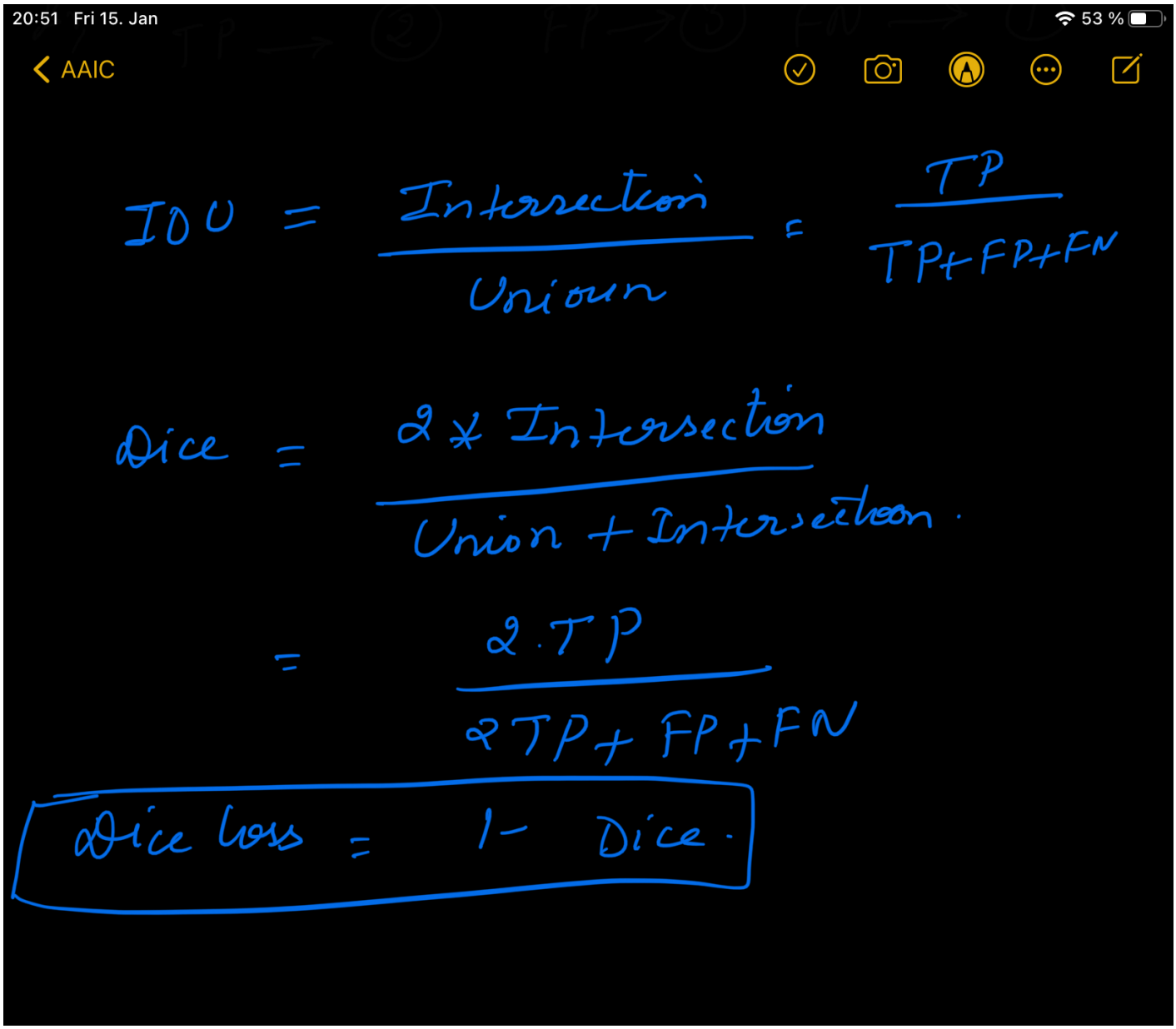

```
def grader_3():
    url = "https://i.imgur.com/4XSUlHk.png"
    url_response = urllib.request.urlopen(url)
    img_array = np.array(bytearray(url_response.read()), dtype=np
    img = cv2.imdecode(img_array, -1)
    my_img = cv2.imread('/content/data/output/201/frame0029_gtFin
    plt.imshow(my_img)
    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]
```

Task:1

```
* Explain the Dice loss
* 1. Write the formualtion
* 2. Range of the loss function
* 3. Interpretation of loss function
* 4. Write your understanding of the loss function, how does it helps in segmentation
```


20:50 Fri 15. Jan
54 %
AAIC
Predicted mask
1
2
3
Real mask.
part 1: It is not overlapping to original mask.
FN: The part of the mask predicted which is overflowing.
FP: The part of the original mask which is not covered by predicted mask.
TP: The part of the original mask covered by predicted mask.
So, TP → (2) FP → (3) FN → (1)
IOU = Intersection / Union = TP / TP+FP+FN



Task 2.2: Training Unet

* Split the data into 80:20.
* Train the UNET on the given dataset and plot the train and validation loss.
* As shown in the reference notebook plot 20 images from the test data along with its segme

```
!pip install git+https://github.com/qubvel/segmentation_models #t
```

```
Collecting git+https://github.com/qubvel/segmentation_models
  Cloning https://github.com/qubvel/segmentation_models to /tmp/pip-req-build-_0cjabdl
```

```
  Running command git clone -q https://github.com/qubvel/segmentation_models /tmp/pip-r
  Running command git submodule update --init --recursive -q
Collecting keras_applications<=1.0.8,>=1.0.7
  Downloading https://files.pythonhosted.org/packages/71/e3/19762fdfc62877ae9102edf6342
     |████████████████████████████████| 51kB 4.5MB/s
Collecting image-classifiers==1.0.0
  Downloading https://files.pythonhosted.org/packages/81/98/6f84720e299a4942ab80df5f76a
Collecting efficientnet==1.0.0
  Downloading https://files.pythonhosted.org/packages/97/82/f3ae07316f0461417dc54affab6
Requirement already satisfied: h5py in /usr/local/lib/python3.6/dist-packages (from ker
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: scikit-image in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from h5py
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: scipy>=0.19.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: pillow>=4.3.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.0 in /usr/local/lib/python3.6/di
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/l
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packa
Building wheels for collected packages: segmentation-models
  Building wheel for segmentation-models (setup.py) ... done
  Created wheel for segmentation-models: filename=segmentation_models-1.0.1-cp36-none-a
  Stored in directory: /tmp/pip-ephem-wheel-cache-vpswbpu7/wheels/49/cf/46/cbb4bb64518c
Successfully built segmentation-models
Installing collected packages: keras-applications, image-classifiers, efficientnet, seg
Successfully installed efficientnet-1.0.0 image-classifiers-1.0.0 keras-applications-1.
```

```
# here dir_path is the route directory where all the images and s
from tqdm import tqdm
dir_path = "/content/data/images"
file_names = set()
for i in tqdm(os.listdir(dir_path)):
  for j in os.listdir(os.path.join(dir_path+'/',i)):
    file_names.add(j.split('.')[0])

file_names = list(file_names)
print(len(file_names))
file_names[0:5]
```

```
100%|██████████| 143/143 [00:00<00:00, 17577.68it/s]3440

['frame43344_leftImg8bit',
 'frame9874_leftImg8bit',
 'frame8069_leftImg8bit',
```

```
 'frame21354_leftImg8bit',
 'frame2095_leftImg8bit']
```

```
# we are importing the pretrained unet from the segmentation mode
# https://github.com/qubvel/segmentation_models
%env SM_FRAMEWORK=tf.keras
import segmentation_models as sm
from segmentation_models import Unet
# sm.set_framework('tf.keras')
tf.keras.backend.set_image_data_format('channels_last')
iou_score = sm.metrics.IOUScore(threshold=0.5)
```

```
env: SM_FRAMEWORK=tf.keras
Segmentation Models: using `tf.keras` framework.
```

```
import imgaug.augmenters as iaa
# For the assignment choose any 4 augumentation techniques
# check the imgaug documentations for more augmentations
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))
```

```
data = data_df_.drop('json', axis=1)
data.head()
```

	image	mask
0	/content/data/images/201/frame0029_leftImg8bit...	/content/data/output/201/frame0029_gtFine_poly...
1	/content/data/images/201/frame0299_leftImg8bit...	/content/data/output/201/frame0299_gtFine_poly...
2	/content/data/images/201/frame0779_leftImg8bit...	/content/data/output/201/frame0779_gtFine_poly...
3	/content/data/images/201/frame1019_leftImg8bit...	/content/data/output/201/frame1019_gtFine_poly...
4	/content/data/images/201/frame1469_leftImg8bit...	/content/data/output/201/frame1469_gtFine_poly...

```
X_test = data.iloc[0:round(0.2*data.shape[0]),:]
X_train = data.iloc[round(0.2*data.shape[0]):,:]
print(X_train.shape,X_test.shape)
```

```
(3206, 2) (802, 2)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test = train_test_split(data,test_size=0.13,random_sta

print(X_train.shape,X_test.shape)
```

```
(3486, 2) (522, 2)
```

```
CLASSES = list(np.unique(list(label_clr.values())))
class Dataset:

    def __init__(self,data):

        #self.ids = file_names
        # the paths of images
        self.images_fps   = data['image'].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_UNCHANG
        image = cv2.resize(image,(256,256),interpolation = cv2.IN
        mask  = cv2.imread(self.masks_fps[i], cv2.IMREAD_UNCHANGE
        mask = cv2.resize(mask,(256,256),interpolation = cv2.INTE
        #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):
        #print(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(*da

        return tuple(batch)

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

    def on_epoch_end(self):
```

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


train_dataset = Dataset(X_train)

class Dataset:

    def __init__(self,data):

        #self.ids = file_names
        # the paths of images
        self.images_fps   = data['image'].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_UNCHANG
        image = cv2.resize(image,(256,256),interpolation = cv2.IN
        mask  = cv2.imread(self.masks_fps[i], cv2.IMREAD_UNCHANGE
        mask = cv2.resize(mask,(256,256),interpolation = cv2.INTE
        #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

        return image, image_mask

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



test_dataset  = Dataset(X_test)
```

```
test_dataset = Dataset(x_test)
```

```
# loading the unet model and using the resnet 34 and initilized w
# "classes" :different types of classes in the dataset
model_unet = Unet('resnet34', encoder_weights='imagenet', classes
```

```
Downloading data from https://github.com/qubvel/classification_models/releases/download
85524480/85521592 [==============================] - 4s 0us/step
```

```
model_unet.summary()
```

```
decoder_stage1b_relu (Activatio (None, 32, 32, 128)  0           decoder_stage1b_bn[

decoder_stage2_upsampling (UpSa (None, 64, 64, 128)  0           decoder_stage1b_rel

decoder_stage2_concat (Concaten (None, 64, 64, 192)  0           decoder_stage2_upsa
                                                                 stage2_unit1_relu1[

decoder_stage2a_conv (Conv2D)   (None, 64, 64, 64)   110592      decoder_stage2_conc

decoder_stage2a_bn (BatchNormal (None, 64, 64, 64)   256         decoder_stage2a_con

decoder_stage2a_relu (Activatio (None, 64, 64, 64)   0           decoder_stage2a_bn[

decoder_stage2b_conv (Conv2D)   (None, 64, 64, 64)   36864       decoder_stage2a_rel

decoder_stage2b_bn (BatchNormal (None, 64, 64, 64)   256         decoder_stage2b_con

decoder_stage2b_relu (Activatio (None, 64, 64, 64)   0           decoder_stage2b_bn[

decoder_stage3_upsampling (UpSa (None, 128, 128, 64) 0           decoder_stage2b_rel

decoder_stage3_concat (Concaten (None, 128, 128, 128 0           decoder_stage3_upsa
                                                                 relu0[0][0]

decoder_stage3a_conv (Conv2D)   (None, 128, 128, 32) 36864       decoder_stage3_conc

decoder_stage3a_bn (BatchNormal (None, 128, 128, 32) 128         decoder_stage3a_con

decoder_stage3a_relu (Activatio (None, 128, 128, 32) 0           decoder_stage3a_bn[

decoder_stage3b_conv (Conv2D)   (None, 128, 128, 32) 9216        decoder_stage3a_rel

decoder_stage3b_bn (BatchNormal (None, 128, 128, 32) 128         decoder_stage3b_con

decoder_stage3b_relu (Activatio (None, 128, 128, 32) 0           decoder_stage3b_bn[

decoder_stage4_upsampling (UpSa (None, 256, 256, 32) 0           decoder_stage3b_rel

decoder_stage4a_conv (Conv2D)   (None, 256, 256, 16) 4608        decoder_stage4_upsa
```

```
decoder_stage4a_bn (BatchNormal (None, 256, 256, 16) 64          decoder_stage4a_con
decoder_stage4a_relu (Activatio (None, 256, 256, 16) 0           decoder_stage4a_bn[
decoder_stage4b_conv (Conv2D)   (None, 256, 256, 16) 2304        decoder_stage4a_rel
decoder_stage4b_bn (BatchNormal (None, 256, 256, 16) 64          decoder_stage4b_con
decoder_stage4b_relu (Activatio (None, 256, 256, 16) 0           decoder_stage4b_bn[
final_conv (Conv2D)             (None, 256, 256, 21) 3045        decoder_stage4b_rel
softmax (Activation)            (None, 256, 256, 21) 0           final_conv[0][0]
==================================================================================
Total params: 24,459,054
Trainable params: 3,169,960
Non-trainable params: 21,289,094
```

```
# https://github.com/qubvel/segmentation_models
import segmentation_models as sm
from segmentation_models.metrics import iou_score
from segmentation_models import Unet

optim = tf.keras.optimizers.Adam(0.001)

focal_loss = sm.losses.cce_dice_loss

# actulally total_loss can be imported directly from library, abo
# total_loss = sm.losses.binary_focal_dice_loss
# or total_loss = sm.losses.categorical_focal_dice_loss

model_unet.compile(optim, focal_loss, metrics=[iou_score])


# Dataset for train images
#dir_path = '/content/drive/MyDrive/ImageSegmentation/data/images
#CLASSES = ['edited']

train_dataloader = Dataloder(train_dataset, batch_size=1, shuffle
test_dataloader = Dataloder(test_dataset, batch_size=1, shuffle=T
BATCH_SIZE = 1
print(train_dataloader[0][0].shape)
assert train_dataloader[0][0].shape == (BATCH_SIZE, 256, 256, 3)
```

```
assert train_dataloader[0][1].shape == (BATCH_SIZE, 256, 256, 21)

# define callbacks for learning rate scheduling and best checkpoi
callbacks = [
    tf.keras.callbacks.ModelCheckpoint('./best_model.h5', save_we
                                       mode='min', monitor='val_i
    tf.keras.callbacks.ReduceLROnPlateau(monitor='val_iou_score',
]
```

```
(1, 256, 256, 3)
```

```
history = model_unet.fit_generator(train_dataloader, steps_per_ep
                                   validation_data=test_dataloader,cal
```

```
/usr/local/lib/python3.6/dist-packages/tensorflow/python/keras/engine/training.py:1844:
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/120
3486/3486 [==============================] - 235s 64ms/step - loss: 0.8979 - iou_score:
Epoch 2/120
3486/3486 [==============================] - 223s 64ms/step - loss: 0.6629 - iou_score:
Epoch 3/120
3486/3486 [==============================] - 222s 64ms/step - loss: 0.5628 - iou_score:
Epoch 4/120
3486/3486 [==============================] - 220s 63ms/step - loss: 0.5419 - iou_score:
Epoch 5/120
3486/3486 [==============================] - 219s 63ms/step - loss: 0.5250 - iou_score:
Epoch 6/120
3486/3486 [==============================] - 219s 63ms/step - loss: 0.5115 - iou_score:
Epoch 7/120
3486/3486 [==============================] - 218s 63ms/step - loss: 0.5063 - iou_score:
Epoch 8/120
3486/3486 [==============================] - 217s 62ms/step - loss: 0.4998 - iou_score:
Epoch 9/120
3486/3486 [==============================] - 216s 62ms/step - loss: 0.5105 - iou_score:
Epoch 10/120
3486/3486 [==============================] - 217s 62ms/step - loss: 0.4884 - iou_score:
Epoch 11/120
3486/3486 [==============================] - 222s 64ms/step - loss: 0.4794 - iou_score:
Epoch 12/120
3486/3486 [==============================] - 219s 63ms/step - loss: 0.4695 - iou_score:
Epoch 13/120
3486/3486 [==============================] - 217s 62ms/step - loss: 0.4636 - iou_score:
Epoch 14/120
3486/3486 [==============================] - 217s 62ms/step - loss: 0.4676 - iou_score:
Epoch 15/120
3486/3486 [==============================] - 216s 62ms/step - loss: 0.4581 - iou_score:
Epoch 16/120
3486/3486 [==============================] - 216s 62ms/step - loss: 0.4602 - iou_score:
Epoch 17/120
3486/3486 [==============================] - 217s 62ms/step - loss: 0.4369 - iou_score:
Epoch 18/120
3486/3486 [==============================] - 217s 62ms/step - loss: 0.4279 - iou_score:
Epoch 19/120
3486/3486 [==============================] - 217s 62ms/step - loss: 0.4301 - iou_score:
Epoch 20/120
3486/3486 [==============================] - 218s 63ms/step - loss: 0.4221 - iou_score:
Epoch 21/120
3486/3486 [==============================] - 218s 62ms/step - loss: 0.4152 - iou_score:
Epoch 22/120
3486/3486 [==============================] - 216s 62ms/step - loss: 0.4236 - iou_score:
Epoch 23/120
3486/3486 [==============================] - 218s 63ms/step - loss: 0.4226 - iou_score:
Epoch 24/120
3486/3486 [==============================] - 217s 62ms/step - loss: 0.4114 - iou_score:
Epoch 25/120
3486/3486 [==============================] - 217s 62ms/step - loss: 0.3803 - iou_score:
Epoch 26/120
3486/3486 [==============================] - 218s 63ms/step - loss: 0.3738 - iou_score:
Epoch 27/120
3486/3486 [==============================] - 216s 62ms/step - loss: 0.3740 - iou_score:
Epoch 28/120
```

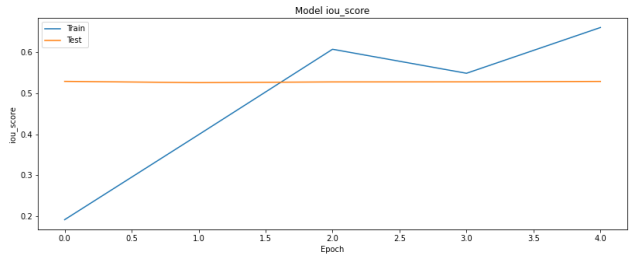
```
3486/3486 [==============================] - 216s 62ms/step - loss: 0.3715 - iou_score:
Epoch 29/120
3486/3486 [==============================] - 217s 62ms/step - loss: 0.3636 - iou_score:
Epoch 30/120
3486/3486 [==============================] - 216s 62ms/step - loss: 0.3713 - iou_score:
Epoch 31/120
3486/3486 [==============================] - 222s 64ms/step - loss: 0.3624 - iou_score:
Epoch 32/120
3486/3486 [==============================] - 220s 63ms/step - loss: 0.3552 - iou_score:
Epoch 33/120
3486/3486 [==============================] - 221s 63ms/step - loss: 0.3696 - iou_score:
Epoch 34/120
3486/3486 [==============================] - 218s 63ms/step - loss: 0.3608 - iou_score:
Epoch 35/120
3486/3486 [==============================] - 219s 63ms/step - loss: 0.3586 - iou_score:
Epoch 36/120
3486/3486 [==============================] - 218s 63ms/step - loss: 0.3536 - iou_score:
Epoch 37/120
3486/3486 [==============================] - 218s 63ms/step - loss: 0.3672 - iou_score:
Epoch 38/120
3486/3486 [==============================] - 219s 63ms/step - loss: 0.3525 - iou_score:
Epoch 39/120
3486/3486 [==============================] - 219s 63ms/step - loss: 0.3519 - iou_score:
Epoch 40/120
```

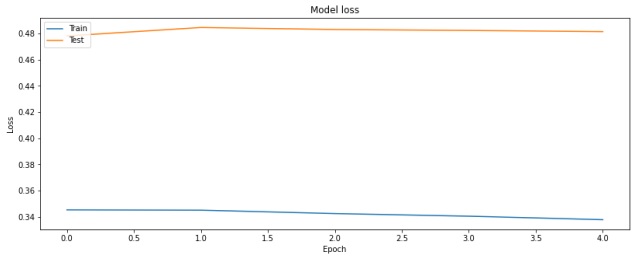
```
history = model_unet.fit_generator(train_dataloader, steps_per_ep
                                   validation_data=test_dataloader,cal
```

```
Epoch 1/5
   1/3486 [..............................] - ETA: 4:26 - loss: 0.4863 - iou_score: 0.50
  warnings.warn('`Model.fit_generator` is deprecated and '
3486/3486 [==============================] - 217s 62ms/step - loss: 0.3454 - iou_score:
Epoch 2/5
3486/3486 [==============================] - 217s 62ms/step - loss: 0.3452 - iou_score:
Epoch 3/5
3486/3486 [==============================] - 217s 62ms/step - loss: 0.3425 - iou_score:
Epoch 4/5
3486/3486 [==============================] - 217s 62ms/step - loss: 0.3406 - iou_score:
Epoch 5/5
3486/3486 [==============================] - 217s 62ms/step - loss: 0.3379 - iou_score:
```

```
# Plot training & validation iou_score values
plt.figure(figsize=(30, 5))
plt.subplot(121)
plt.plot(history.history['iou_score'])
plt.plot(history.history['val_iou_score'])
plt.title('Model iou_score')
plt.ylabel('iou_score')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
```

```
plt.legend(['Train', 'Test'], loc='upper left')

# Plot training & validation loss values
plt.subplot(122)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```





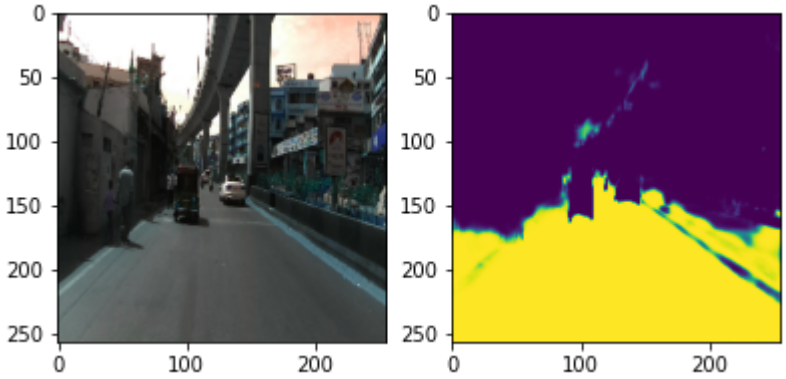

```
X_test.values[0]
```

```
array(['/content/data/images/421/0004379_leftImg8bit.jpg',
       '/content/data/output/421/0004379_gtFine_polygons.png'],
      dtype=object)
```

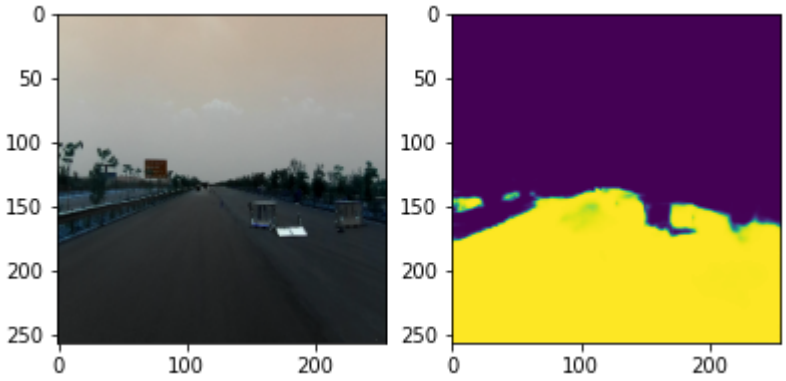
```
for p, i in enumerate(X_test.values[0:10]):
    #original image
    image = cv2.imread(i[0], cv2.IMREAD_UNCHANGED)
    #print(i[0],i[1])
    print(image.shape)
    image = cv2.resize(image, (256,256))
    #print(i[0],i[1])
    #predicted segmentation map
    predicted  = model_unet.predict(image[np.newaxis,:,:,:])

    #original segmentation map
```

```
    image_mask = cv2.imread(i[1], cv2.IMREAD_UNCHANGED)
    image_mask = cv2.resize(image_mask, (256,256))


    plt.figure(figsize=(10,6))
    plt.subplot(131)
    plt.imshow(image)
    plt.subplot(132)
    #plt.imshow(image_mask[0,:,:,1], cmap='gray', vmax=1, vmin=0)
    #plt.subplot(133)
    plt.imshow(predicted[0,:,:,1], vmax=1, vmin=0)
    plt.show()
```

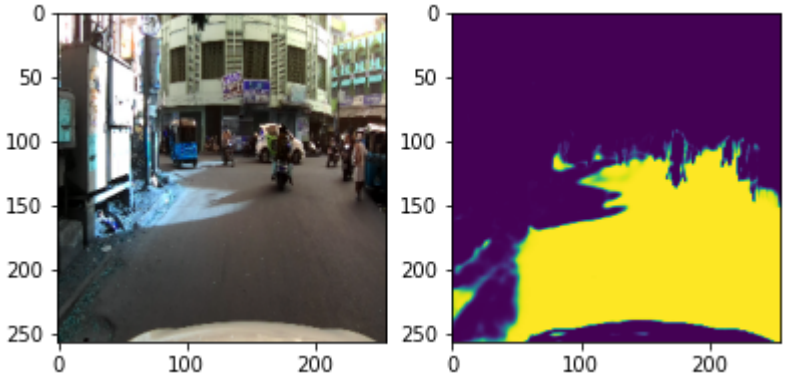
```
(1080, 1920, 3)
/usr/local/lib/python3.6/dist-packages/tensorflow/python/data/ops/dataset_ops.py:3504:
  "Even though the tf.config.experimental_run_functions_eagerly "
```



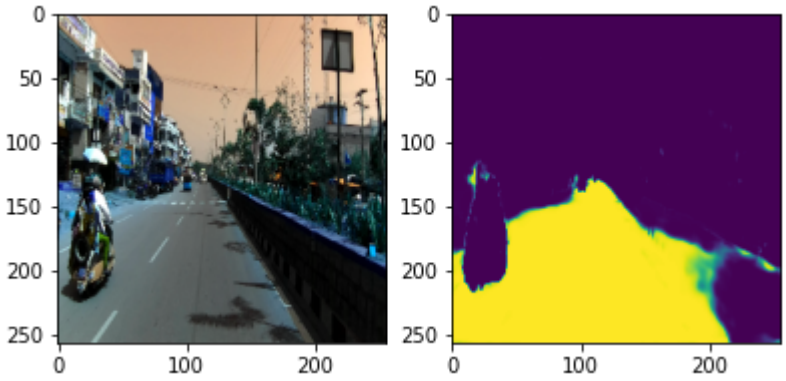
```
(1080, 1920, 3)
```



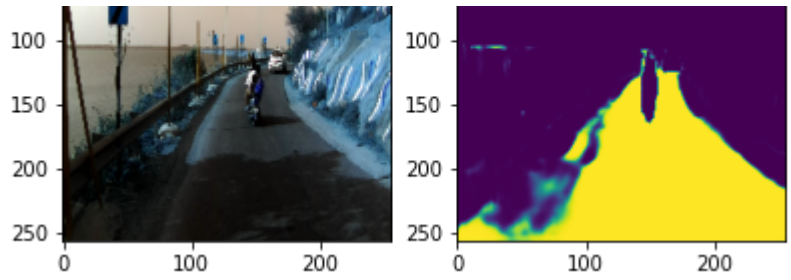
```
(1080, 1920, 3)
```



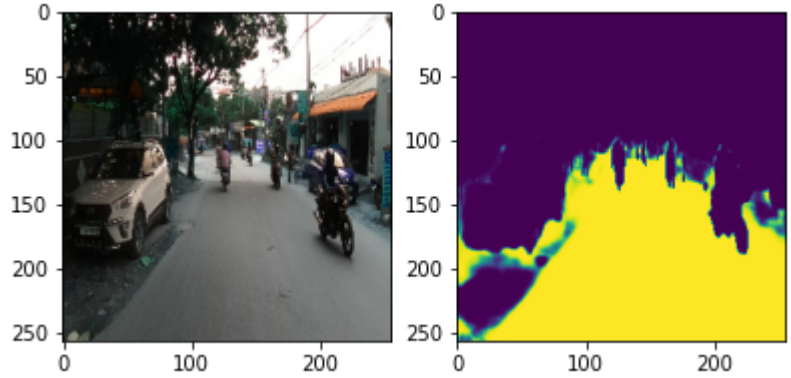
```
(1080, 1920, 3)
```



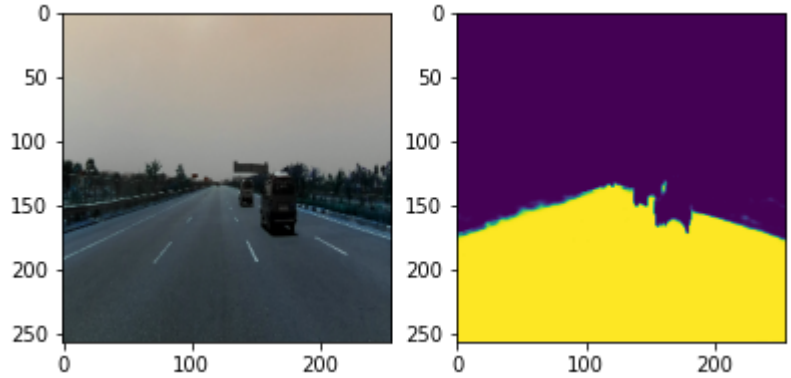
```
(1080, 1920, 3)
```



(1080, 1920, 3)



(1080, 1920, 3)



(1080, 1920, 3)

CANet model implementation

```
image = cv2.imread(X_test.values[10][0], cv2.IMREAD_UNCHANGED)
image = cv2.resize(image, (256,256))
print(X_test.values[10][0],i[1])
#predicted segmentation map
predicted  = model_unet.predict(image[np.newaxis,:,:,:])

#original segmentation map
image_mask = cv2.imread(X_test.values[10][1], cv2.IMREAD_UNCHANGE
image_mask = cv2.resize(image_mask, (256,256))
```

```
plt.figure(figsize=(10,6))
plt.subplot(131)
plt.imshow(image)
plt.subplot(132)
#print(image_mask.shape)
#plt.imshow(image_mask, cmap='gray', vmax=1, vmin=0)
#plt.subplot(133)
plt.imshow(predicted[0,:,:,1], vmax=1, vmin=0)
plt.show()
```



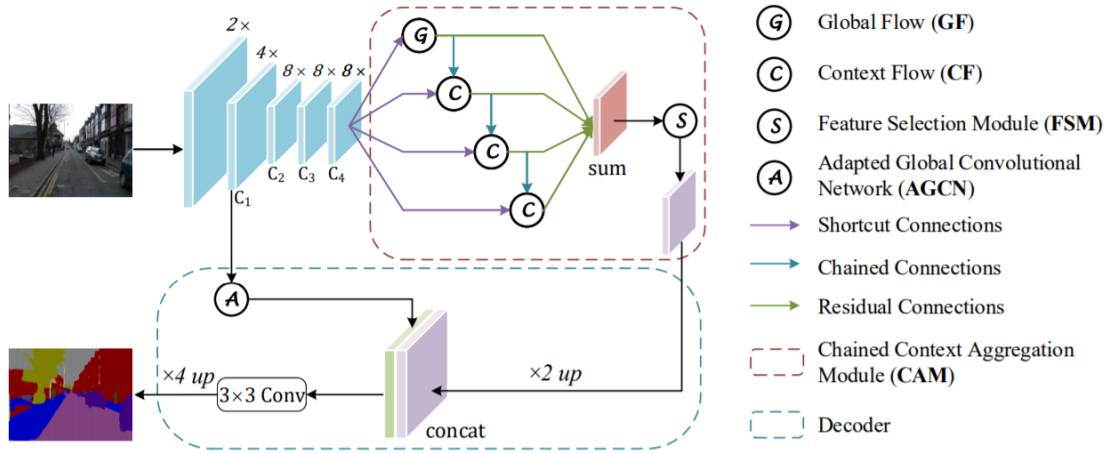
```
print(predicted.shape)
```

```
(1, 256, 256, 21)
```

```
class convolutional_block(tf.keras.layers.Layer):
    def __init__(self, kernel=3,  filters=[4,4,8], stride=1, name=
        super().__init__(name=name)
        self.F1, self.F2, self.F3 = filters
        self.kernel = kernel
        self.stride = stride
        self.conv2d_1=Conv2D(filters=self.F1,kernel_size=1,strides
        self.batch_norm1 = BatchNormalization(name=name+'_bn1')
        self.conv2d_2=Conv2D(filters=self.F2,kernel_size=self.kern
        self.batch_norm2 = BatchNormalization(name=name+'_bn2')
        self.conv2d_3=Conv2D(filters=self.F3,kernel_size=1,strides
        self.batch_norm3 = BatchNormalization(name=name+'_bn3')
        self.conv2d_p=Conv2D(filters=self.F3,kernel_size=self.kern
        self.batch_norm_p = BatchNormalization(name=name+'_bn_para
        self.add = Add()
        self.activation = Activation('relu')

    def call(self, X):
        X_parallel=X
        X=self.conv2d_1(X)
        X=self.batch_norm1(X)
        X=self.activation(X)

        X=self.conv2d_2(X)
        X=self.batch_norm2(X)
        X=self.activation(X)
```

```
X=self.activation(X)

X=self.conv2d_3(X)
X=self.batch_norm3(X)
X=self.activation(X)

X_parallel = self.conv2d_p(X_parallel)
X_parallel = self.batch_norm_p(X_parallel)
X_parallel = self.activation (X_parallel)

X=self.add([X,X_parallel])
#X=self.activation(X)
return X
```

- as a part of this assignment we will be implementing the architecture based on this paper https://arxiv.org/pdf/2002.12041.pdf
- We will be using the custom layers concept that we used in seq-seq assignment
- You can devide the whole architecture can be devided into two parts
    1. Encoder
    2. Decoder



- Encoder:
    - The first step of the encoder is to create the channel maps [C_1, C_2, C_3, C_4]
    - C_1 width and heigths are 4x times less than the original image

- C_2 width and heigths are 8x times less than the original image
- C_3 width and heigths are 8x times less than the original image
- C_4 width and heigths are 8x times less than the original image
- *you can reduce the dimensions by using stride parameter*.
- [C_1, C_2, C_3, C_4] are formed by applying a "conv block" followed by k number of "identity block". i.e the C_k feature map will single "conv block" followed by k number of "identity blocks".

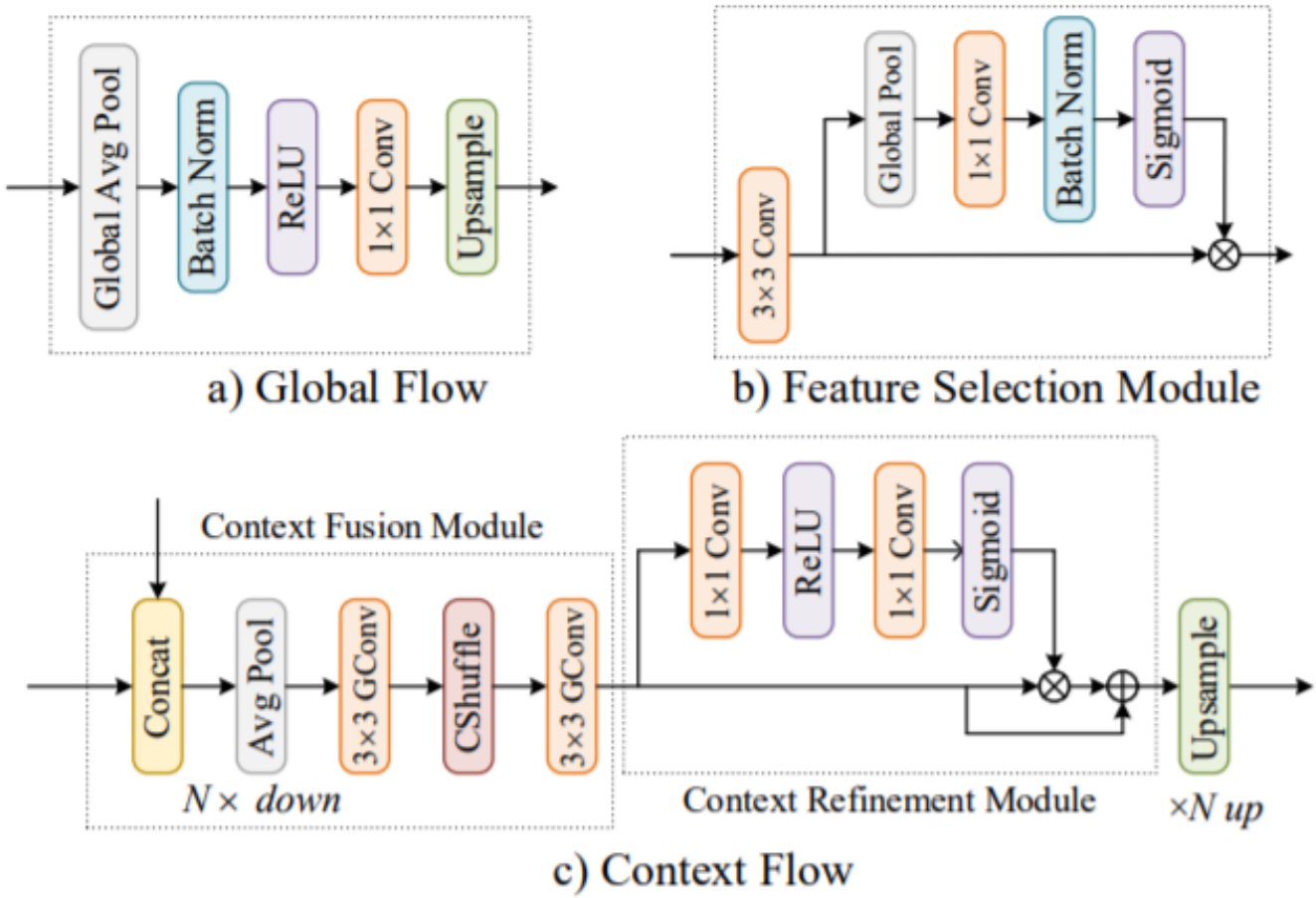
- **The conv block and identity block of C_1**: the number filters in the covolutional layers will be $[4,4,8]$ and the number of filters in the parallel conv layer will also be 8.
- **The conv block and identity block of C_2**: the number filters in the covolutional layers will be $[8,8,16]$ and the number of filters in the parallel conv layer will also be 16.
- **The conv block and identity block of C_3**: the number filters in the covolutional layers will be $[16,16,32]$ and the number of filters in the parallel conv layer will also be 32.
- **The conv block and identity block of C_4**: the number filters in the covolutional layers will be $[32,32,64]$ and the number of filters in the parallel conv layer will also be 64.
- Here $\oplus$ represents the elementwise sum

NOTE: these filters are of your choice, you can explore more options also

- Example: if your image is of size $(512, 512, 3)$
  - the output after C_1 will be $128*128*8$
  - the output after C_2 will be $64*64*16$
  - the output after C_3 will be $64*64*32$
  - the output after C_4 will be $64*64*64$

```
class identity_block(tf.keras.layers.Layer):
    def __init__(self, kernel=3,  filters=[4,4,8], name="identity
        super().__init__(name=name)
        self.F1, self.F2, self.F3 = filters
        self.kernel = kernel
        self.F1, self.F2, self.F3 = filters
        self.kernel = kernel
        self.conv2d_1=Conv2D(filters=self.F1,kernel_size=1,stride
        self.batch_norm1 = BatchNormalization(axis = 3,name=name+
        self.conv2d_2=Conv2D(filters=self.F2,kernel_size=self.ker
```

```
        self.conv2d_2=Conv2D(filters=self.F2,kernel_size=self.ke
        self.batch_norm2 = BatchNormalization(axis = 3,name=name+
        self.conv2d_3=Conv2D(filters=self.F3,kernel_size=1,stride
        self.batch_norm3 = BatchNormalization(axis = 3,name=name+
        self.add = Add()
        self.activation = Activation('relu')
    def call(self, X):
        x = X
        X=self.conv2d_1(X)
        X=self.batch_norm1(X)
        X=self.activation(X)

        X=self.conv2d_2(X)
        X=self.batch_norm2(X)
        X=self.activation(X)

        X=self.conv2d_3(X)
        X=self.batch_norm3(X)

        X=self.add([x,X])
        #return X
        # write the architecutre that was mentioned above
        #x = X
        #print('*')
        #X = Conv2D(self.F1, (1,1), activation='relu')(X)
        #print('**')
        #X = BatchNormalization()(X)
        #X = ReLU()(X)
        #print('**')
        #X = Conv2D(self.F2, (3,3), activation='relu',padding='sa
        #X = BatchNormalization()(X)
        #X = ReLU()(X)
        #print('***')
        #X = Conv2D(self.F3, (1,1))(X)
        #X = BatchNormalization()(X)
        return X
```

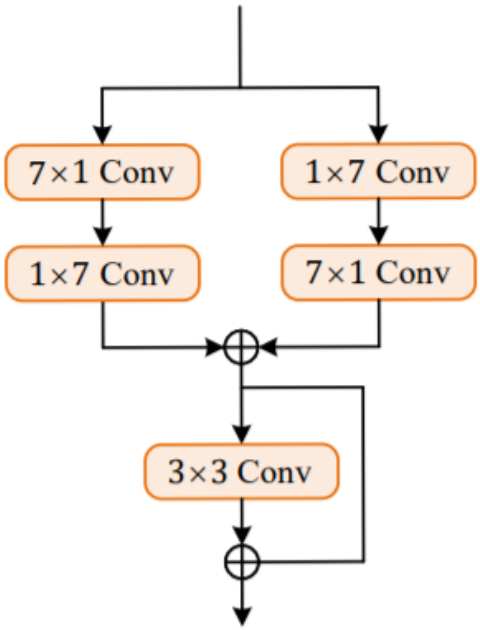



```
class global_flow(tf.keras.layers.Layer):
    def __init__(self,name="global_flow"):
        super().__init__(name=name)
        s = (X.shape[1], X.shape[2])
        self.conv2d_1=Conv2D(64,(1,1),strides=1,padding='same',na
        self.batch_norm1 = BatchNormalization(name=name+'_bn1')
        self.batch_norm2 = BatchNormalization(name=name+'_bn2')
        self.global_avg = GlobalAveragePooling2D()
        self.add = Add()
        self.upsample = UpSampling2D(size=32, interpolation='near
        self.activation = Activation('relu')
    def call(self, X):
        # implement the global flow operatiom
        s = (X.shape[1], X.shape[2])
        #print('in global flow: ', X.shape)
```

```
        X = self.global_avg(X)
        #print('in global flow: ', X.shape)
        X = self.batch_norm1(X)
        #print('in global flow: ', X.shape)
        X = self.activation(X)
        #print('2_in global flow: ', X.shape)
        X = tf.reshape(X,(-1,1,1,X.shape[-1]))
        X = self.conv2d_1(X)
        X = self.batch_norm2(X)
        #print('3_in global flow: ', X.shape)
        #X = UpSampling2D(s,interpolation='nearest')(X)
        #print('4_in global flow: ', X.shape)
        return self.upsample(X)


class context_flow(tf.keras.layers.Layer):
    def __init__(self, name="context_flow"):
        super().__init__(name=name)
        self.conv2d_1 = Conv2D(64, (3,3),activation='relu',paddin
        self.conv2d_2 = Conv2D(64,(1,1),name=name+'_c2')
        self.conv2d_3 = Conv2D(64,(1,1),name=name+'_c3')
        self.concat = Concatenate()
        self.avg_pool = AveragePooling2D()
        self.add = Add()
        self.activation = Activation('relu')
        self.upsample = UpSampling2D((2,2),interpolation='nearest
        self.bn = BatchNormalization()

    def call(self, X):
        #print('1_in context flow: ',X[0].shape, X[1].shape)
        # here X will a list of two elements
        INP, FLOW = X[0], X[1]
        #print(INP.shape, FLOW.shape)
        f = INP.shape[-1]
        # implement the context flow as mentioned in the above ce

        x = self.concat([INP, FLOW])
        #print('1_in context flow: ',x.shape)
        x = self.avg_pool(x)
        #print('2_in context flow: ',x.shape)
```

```
        #print('2_in context flow: ',x.shape)
        x = self.conv2d_1(x)
        #print('3_in context flow: ',x.shape)
        #x = UpSampling2D(interpolation='nearest')(x)
        #print('4_in context flow: ',x.shape)
        x_1 = self.conv2d_2(x)
        #print('5_in context flow: ',x_1.shape)
        x_1 = self.activation(x_1)
        x_1 = self.conv2d_3(x_1)
        x_1 = self.bn(x_1)
        #print('6_in context flow: ',x_1.shape)
        x_1 = tf.keras.activations.sigmoid(x_1)
        #print('7_in context flow: ',x_1.shape)
        x_2 = x_1 * x
        #print('8_in context flow: ',x_2.shape)
        x = self.add([x_2 , x])
        #print('9_in context flow: ',x.shape)
        #x = UpSampling2D((2,2),interpolation='nearest')(x)
        #print('10_in context flow: ',x.shape)
        return self.upsample(x)


class fsm(tf.keras.layers.Layer):
    def __init__(self, name="feature_selection"):
        super().__init__(name=name)

        self.conv2d_1 = Conv2D(32,(3,3),padding='same',name=name+
        self.global_avg = GlobalMaxPool2D()
        self.conv2d_2 = Conv2D(32,(1,1),name=name+'_c2')
        self.bn = BatchNormalization()

    def call(self, X):
        # implement the FSM modules based on image in the above c
        #print('1_in fsm: ',X.shape)
        X_2 = self.conv2d_1(X)
        X_1 = X
        #print('2_in fsm: ',X_1.shape)
        X_1 = self.global_avg(X_1)
        #print('3_in fsm: ',X_1.shape)
        X_1 = tf.reshape(X_1,(-1,1,1,X_1.shape[-1]))
        X_1 = self.conv2d_2(X_1)
```

```
        #print('4_in fsm: ',X_1.shape)
        X_1 = self.bn(X_1)
        X_1 = tf.keras.activations.sigmoid(X_1)
        FSM_Conv_T = X_1 * X_2
        #print('5_in fsm: ',FSM_Conv_T.shape)
        return FSM_Conv_T
```



b) AGCN

```
class agcn(tf.keras.layers.Layer):
    def __init__(self, name="global_conv_net"):
        super().__init__(name=name)

        self.conv2d_1 = Conv2D(32,(7,1),padding='same',name=name+
        self.conv2d_2 = Conv2D(32,(1,7),padding='same',name=name+
        self.conv2d_3 = Conv2D(32,(1,7),padding='same',name=name+
        self.conv2d_4 = Conv2D(32,(7,1),padding='same',name=name+
        self.conv2d_5 = Conv2D(32,(3,3),padding='same',name=name+
        self.bn1 = BatchNormalization()
        self.bn2 = BatchNormalization()
        self.bn3 = BatchNormalization()
        self.add = Add()
```

```
    def call(self, X_C1):
        # please implement the above mentioned architecture
        #print('1_in agcn: ',X_C1.shape)
        X_1 = self.conv2d_1(X_C1)
        #print('2_in agcn: ',X_1.shape)
        X_1 = self.conv2d_2(X_1)
        X_1 = self.bn1(X_1)
        #print('3_in agcn: ',X_1.shape)
        X_2 = self.conv2d_3(X_C1)
        #print('4_in agcn: ',X_2.shape)
        X_2 = self.conv2d_4(X_2)
        X_2 = self.bn2(X_2)
        #print('5_in agcn: ',X_2.shape)

        X_3 = self.add([X_1 , X_2])
        #print('6_in agcn: ',X_3.shape)
        X_4 = self.conv2d_5(X_3)
        X_4 = self.bn3(X_4)
        #print('7_in agcn: ',X_4.shape)
        #X = self.add2([X_3 , X_4])
        #print('8_in agcn: ',X.shape)
        return self.add([X_3 , X_4])


tf.config.run_functions_eagerly(True)
X_input = Input(shape=(256,256,3))

# Stage 1
X = Conv2D(64, (3, 3), name='conv1', padding="same", kernel_initi
X = BatchNormalization(axis=3, name='bn_conv1')(X)
X = Activation('relu')(X)
X = MaxPooling2D((2, 2), strides=(2,2))(X)
#print(X.shape)

C_1 = convolutional_block(stride=2,name='conv_block_1')(X)
#print('1_C_1 shape : ',C_1.shape)
agcn_C1 = C_1
C_1 = identity_block(name='i_1')(C_1)
#print('2_C_1 shape : ',C_1.shape)
```

```
C_2 = convolutional_block(3,[8,8,16],stride=2,name='conv_block_2'
#print('1_C_2 shape: ', C_2.shape)
C_2 = identity_block(3,[8,8,16],name='i_2')(C_2)
C_2 = identity_block(3,[8,8,16,],name='i_3')(C_2)
#print('2_C_2 shape: ', C_2.shape)


C_3 = convolutional_block(3,[16,16,32],name='conv_block_3')(C_2)
#print('1_C_3 shape: ', C_3.shape)
C_3 = identity_block(3,[16,16,32],name='i_4')(C_3)
C_3 = identity_block(3,[16,16,32],name='i_5')(C_3)
C_3 = identity_block(3,[16,16,32],name='i_6')(C_3)
#print('2_C_3 shape: ', C_3.shape)


C_4 = convolutional_block(3,[32,32,64],name='conv_block_4')(C_3)
#print('1_C_4 shape: ', C_4.shape)
C_4 = identity_block(3,[32,32,64],name='i_7')(C_4)
C_4 = identity_block(3,[32,32,64],name='i_8')(C_4)
C_4 = identity_block(3,[32,32,64],name='i_9')(C_4)
C_4 = identity_block(3,[32,32,64],name='i_10')(C_4)
print('1_C_4 shape: ', C_4.shape)


G = global_flow()(C_4)
print(C_4.shape,G.shape)
con_1 = context_flow(name='con_1')([C_4,G])
con_2 = context_flow(name='con_2')([C_4,con_1])
con_3 = context_flow(name='con_3')([C_4,con_2])


con = Add()([G,con_1,con_2,con_3])
print(con.shape)
FSM = fsm()(con)
print(FSM.shape)
FSM = UpSampling2D(interpolation='nearest')(FSM)
print(FSM.shape)
```

```
A = agcn()(agcn_C1)
print(agcn_C1.shape)
con_A_S = Concatenate()([A,FSM])
print(A.shape,FSM.shape)
conv_out = Conv2D(21, (3,3),padding='same')(con_A_S)
up_conv = UpSampling2D(size=(4,4), interpolation='nearest')(conv_
print(up_conv.shape)

final_out = Activation('softmax')(up_conv)
print(final_out.shape)
model = Model(inputs = X_input, outputs = final_out)
```

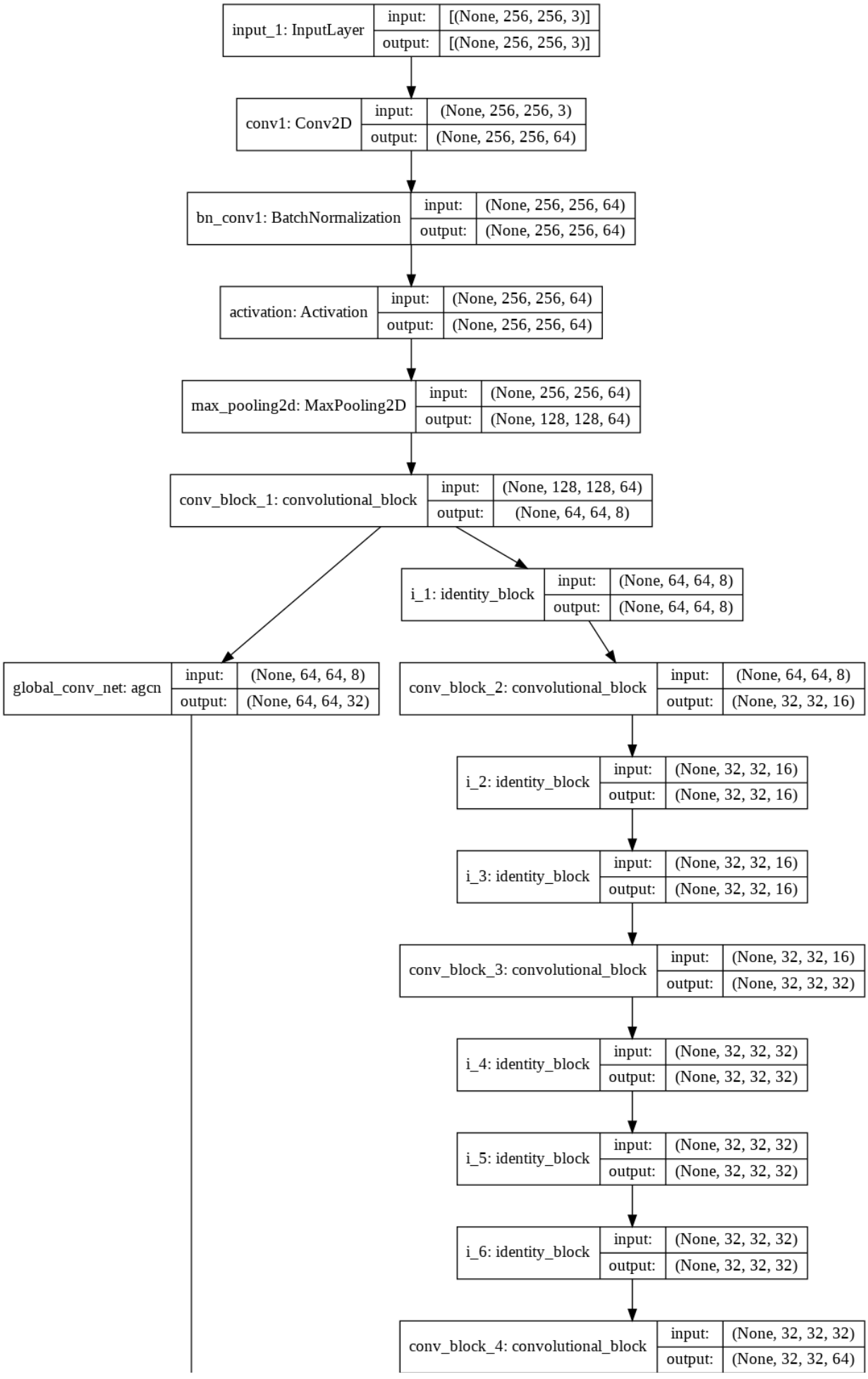
```
1_C_4 shape:  (None, 32, 32, 64)
(None, 32, 32, 64) (None, 32, 32, 64)
(None, 32, 32, 64)
(None, 32, 32, 32)
(None, 64, 64, 32)
(None, 64, 64, 8)
(None, 64, 64, 32) (None, 64, 64, 32)
(None, 256, 256, 21)
(None, 256, 256, 21)
```

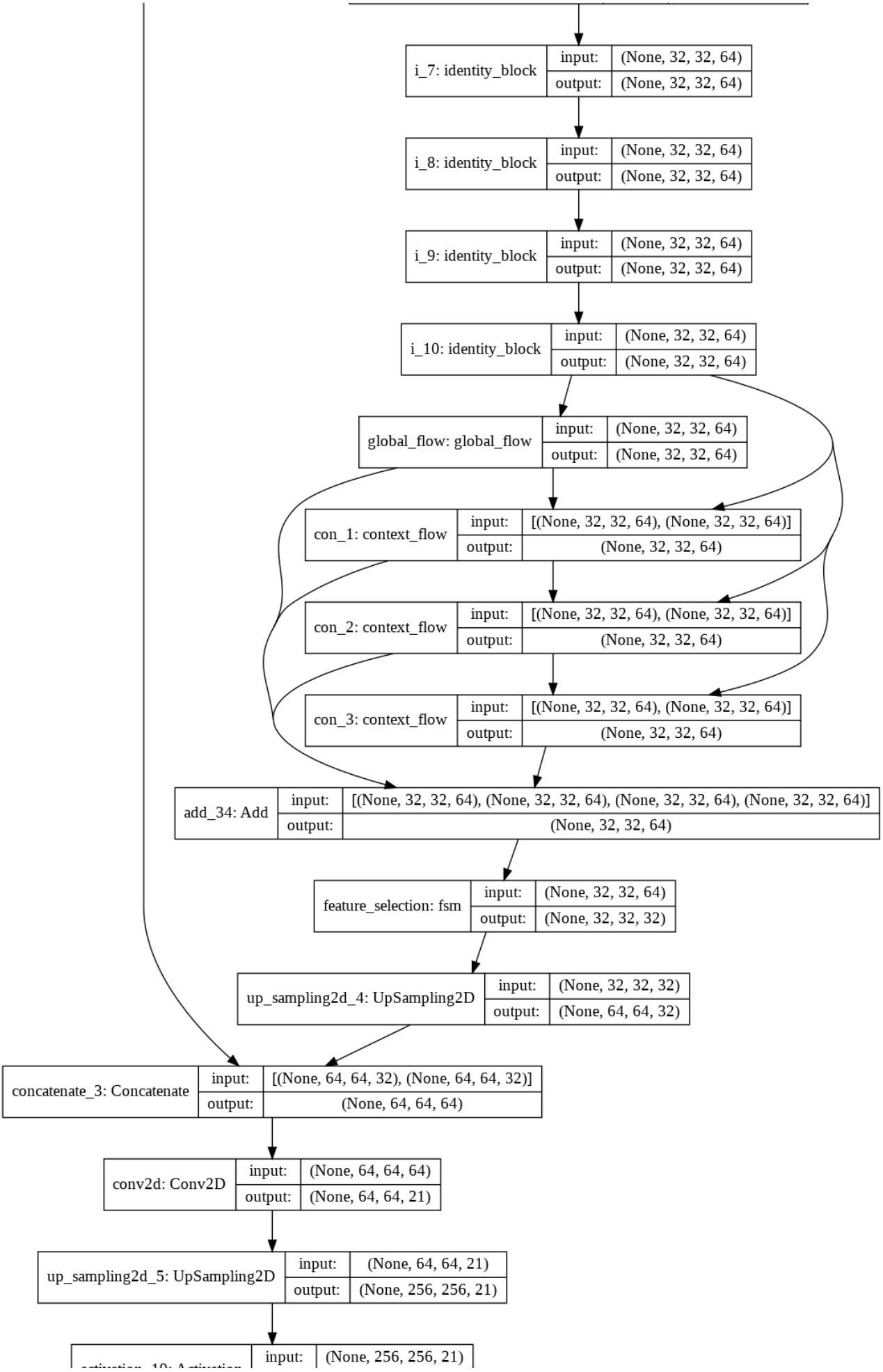
```
model.summary()
```

```
i_2 (identity_block)            (None, 32, 32, 16)   992         conv_block_2[0][0]
i_3 (identity_block)            (None, 32, 32, 16)   992         i_2[0][0]
conv_block_3 (convolutional_blo (None, 32, 32, 32)   8160        i_3[0][0]
i_4 (identity_block)            (None, 32, 32, 32)   3648        conv_block_3[0][0]
i_5 (identity_block)            (None, 32, 32, 32)   3648        i_4[0][0]
i_6 (identity_block)            (None, 32, 32, 32)   3648        i_5[0][0]
conv_block_4 (convolutional_blo (None, 32, 32, 64)   31680       i_6[0][0]
i_7 (identity_block)            (None, 32, 32, 64)   13952       conv_block_4[0][0]
i_8 (identity_block)            (None, 32, 32, 64)   13952       i_7[0][0]
i_9 (identity_block)            (None, 32, 32, 64)   13952       i_8[0][0]
i_10 (identity_block)           (None, 32, 32, 64)   13952       i_9[0][0]
global_flow (global_flow)       (None, 32, 32, 64)   4672        i_10[0][0]
con_1 (context_flow)            (None, 32, 32, 64)   82368       i_10[0][0]
```

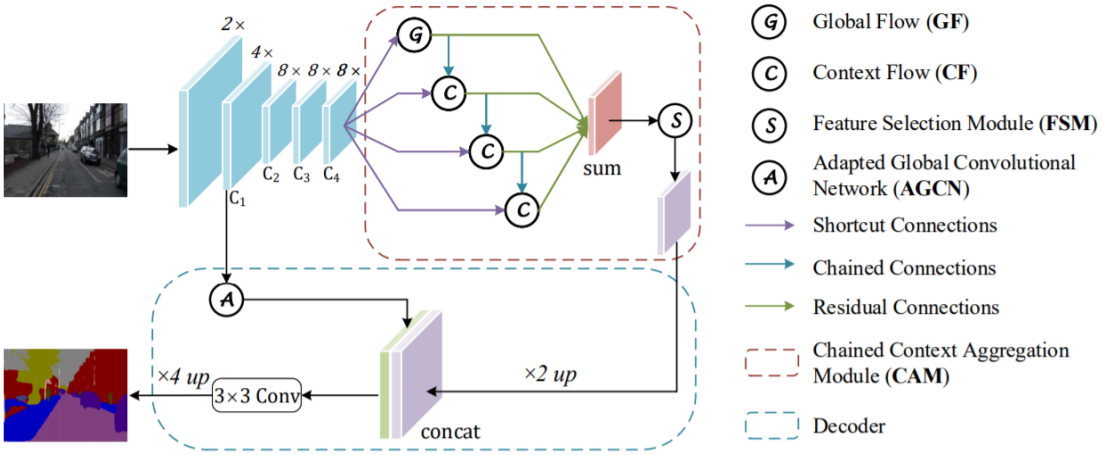
```
                                                                 global_flow[0][0]
__________________________________________________________________________________________________
con_2 (context_flow)            (None, 32, 32, 64)   82368       i_10[0][0]
                                                                 con_1[0][0]
__________________________________________________________________________________________________
con_3 (context_flow)            (None, 32, 32, 64)   82368       i_10[0][0]
                                                                 con_2[0][0]
__________________________________________________________________________________________________
add_34 (Add)                    (None, 32, 32, 64)   0           global_flow[0][0]
                                                                 con_1[0][0]
                                                                 con_2[0][0]
                                                                 con_3[0][0]
__________________________________________________________________________________________________
feature_selection (fsm)         (None, 32, 32, 32)   20672       add_34[0][0]
__________________________________________________________________________________________________
global_conv_net (agcn)          (None, 64, 64, 32)   27680       conv_block_1[0][0]
__________________________________________________________________________________________________
up_sampling2d_4 (UpSampling2D)  (None, 64, 64, 32)   0           feature_selection[0
__________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 64, 64, 64)   0           global_conv_net[0][
                                                                 up_sampling2d_4[0][
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 64, 64, 21)   12117       concatenate_3[0][0]
__________________________________________________________________________________________________
up_sampling2d_5 (UpSampling2D)  (None, 256, 256, 21) 0           conv2d[0][0]
__________________________________________________________________________________________________
activation_19 (Activation)      (None, 256, 256, 21) 0           up_sampling2d_5[0][
==================================================================================================
Total params: 430,477
Trainable params: 427,165
Non-trainable params: 3,312
__________________________________________________________________________________________________
```

```
tf.keras.utils.plot_model(model,show_shapes=True)
```


input_1: InputLayer
input: [(None, 256, 256, 3)]
output: [(None, 256, 256, 3)]
conv1: Conv2D
input: (None, 256, 256, 3)
output: (None, 256, 256, 64)
bn_conv1: BatchNormalization
input: (None, 256, 256, 64)
output: (None, 256, 256, 64)
activation: Activation
input: (None, 256, 256, 64)
output: (None, 256, 256, 64)
max_pooling2d: MaxPooling2D
input: (None, 256, 256, 64)
output: (None, 128, 128, 64)
conv_block_1: convolutional_block
input: (None, 128, 128, 64)
output: (None, 64, 64, 8)
i_1: identity_block
input: (None, 64, 64, 8)
output: (None, 64, 64, 8)
global_conv_net: agcn
input: (None, 64, 64, 8)
output: (None, 64, 64, 32)
conv_block_2: convolutional_block
input: (None, 64, 64, 8)
output: (None, 32, 32, 16)
i_2: identity_block
input: (None, 32, 32, 16)
output: (None, 32, 32, 16)
i_3: identity_block
input: (None, 32, 32, 16)
output: (None, 32, 32, 16)
conv_block_3: convolutional_block
input: (None, 32, 32, 16)
output: (None, 32, 32, 32)
i_4: identity_block
input: (None, 32, 32, 32)
output: (None, 32, 32, 32)
i_5: identity_block
input: (None, 32, 32, 32)
output: (None, 32, 32, 32)
i_6: identity_block
input: (None, 32, 32, 32)
output: (None, 32, 32, 32)
conv_block_4: convolutional_block
input: (None, 32, 32, 32)
output: (None, 32, 32, 64)


i_7: identity_block
input: (None, 32, 32, 64)
output: (None, 32, 32, 64)
i_8: identity_block
input: (None, 32, 32, 64)
output: (None, 32, 32, 64)
i_9: identity_block
input: (None, 32, 32, 64)
output: (None, 32, 32, 64)
i_10: identity_block
input: (None, 32, 32, 64)
output: (None, 32, 32, 64)
global_flow: global_flow
input: (None, 32, 32, 64)
output: (None, 32, 32, 64)
con_1: context_flow
input: [(None, 32, 32, 64), (None, 32, 32, 64)]
output: (None, 32, 32, 64)
con_2: context_flow
input: [(None, 32, 32, 64), (None, 32, 32, 64)]
output: (None, 32, 32, 64)
con_3: context_flow
input: [(None, 32, 32, 64), (None, 32, 32, 64)]
output: (None, 32, 32, 64)
add_34: Add
input: [(None, 32, 32, 64), (None, 32, 32, 64), (None, 32, 32, 64), (None, 32, 32, 64)]
output: (None, 32, 32, 64)
feature_selection: fsm
input: (None, 32, 32, 64)
output: (None, 32, 32, 32)
up_sampling2d_4: UpSampling2D
input: (None, 32, 32, 32)
output: (None, 64, 64, 32)
concatenate_3: Concatenate
input: [(None, 64, 64, 32), (None, 64, 64, 32)]
output: (None, 64, 64, 64)
conv2d: Conv2D
input: (None, 64, 64, 64)
output: (None, 64, 64, 21)
up_sampling2d_5: UpSampling2D
input: (None, 64, 64, 21)
output: (None, 256, 256, 21)
input: (None, 256, 256, 21)

activation_19: Activation	output:	(None, 256, 256, 21)




```
# Dataset for train images
#dir_path = '/content/drive/MyDrive/ImageSegmentation/data/images
#CLASSES = ['edited']

train_dataloader = Dataloder(train_dataset, batch_size=16, shuffl
test_dataloader = Dataloder(test_dataset, batch_size=16, shuffle=
BATCH_SIZE = 16
print(train_dataloader[0][0].shape)
assert train_dataloader[0][0].shape == (BATCH_SIZE, 256, 256, 3)
assert train_dataloader[0][1].shape == (BATCH_SIZE, 256, 256, 21)

# define callbacks for learning rate scheduling and best checkpoi
callbacks = [
    tf.keras.callbacks.ModelCheckpoint('./best_model.h5', save_we
                                       mode='min', monitor='val_i
    tf.keras.callbacks.ReduceLROnPlateau(monitor='val_iou_score',
]
```

```
(16, 256, 256, 3)
```

```
# https://github.com/qubvel/segmentation_models
import segmentation_models as sm
from segmentation_models.metrics import iou_score
from segmentation_models import Unet

optim = tf.keras.optimizers.Adam(0.001)
```

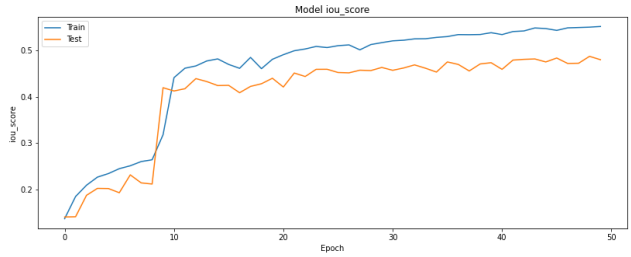
```
optim = tf.keras.optimizers.Adam(0.001)

focal_loss = sm.losses.cce_dice_loss

# actulally total_loss can be imported directly from library, abo
# total_loss = sm.losses.binary_focal_dice_loss
# or total_loss = sm.losses.categorical_focal_dice_loss

model.compile(optim, focal_loss, metrics=[iou_score])

history = model.fit_generator(train_dataloader,  steps_per_epoch=
                              validation_data=test_dataloader,val
```

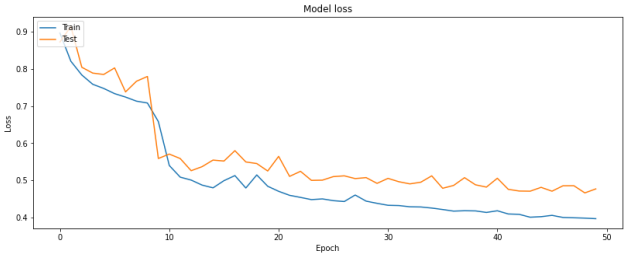
```
Epoch 22/50
217/217 [==============================] - 225s 1s/step - loss: 0.4615 - iou_score:
Epoch 23/50
217/217 [==============================] - 227s 1s/step - loss: 0.4533 - iou_score:
Epoch 24/50
217/217 [==============================] - 228s 1s/step - loss: 0.4489 - iou_score:
Epoch 25/50
217/217 [==============================] - 227s 1s/step - loss: 0.4518 - iou_score:
Epoch 26/50
217/217 [==============================] - 228s 1s/step - loss: 0.4474 - iou_score:
Epoch 27/50
217/217 [==============================] - 228s 1s/step - loss: 0.4422 - iou_score:
Epoch 28/50
217/217 [==============================] - 226s 1s/step - loss: 0.4505 - iou_score:
Epoch 29/50
217/217 [==============================] - 228s 1s/step - loss: 0.4406 - iou_score:
Epoch 30/50
217/217 [==============================] - 230s 1s/step - loss: 0.4342 - iou_score:
Epoch 31/50
217/217 [==============================] - 230s 1s/step - loss: 0.4362 - iou_score:
Epoch 32/50
217/217 [==============================] - 231s 1s/step - loss: 0.4334 - iou_score:
Epoch 33/50
217/217 [==============================] - 231s 1s/step - loss: 0.4266 - iou_score:
Epoch 34/50
217/217 [==============================] - 231s 1s/step - loss: 0.4236 - iou_score:
Epoch 35/50
217/217 [==============================] - 230s 1s/step - loss: 0.4242 - iou_score:
Epoch 36/50
217/217 [==============================] - 227s 1s/step - loss: 0.4149 - iou_score:
Epoch 37/50
217/217 [==============================] - 228s 1s/step - loss: 0.4128 - iou_score:
Epoch 38/50
217/217 [==============================] - 224s 1s/step - loss: 0.4131 - iou_score:
Epoch 39/50
217/217 [==============================] - 221s 1s/step - loss: 0.4163 - iou_score:
Epoch 40/50
217/217 [==============================] - 222s 1s/step - loss: 0.4081 - iou_score:
Epoch 41/50
217/217 [==============================] - 217s 999ms/step - loss: 0.4170 - iou_scor
```

```
Epoch 42/50
217/217 [==============================] - 217s 1000ms/step - loss: 0.4081 - iou_sco
Epoch 43/50
217/217 [==============================] - 213s 982ms/step - loss: 0.4143 - iou_scor
Epoch 44/50
217/217 [==============================] - 213s 982ms/step - loss: 0.3980 - iou_scor
Epoch 45/50
217/217 [==============================] - 210s 966ms/step - loss: 0.4041 - iou_scor
Epoch 46/50
217/217 [==============================] - 212s 976ms/step - loss: 0.4066 - iou_scor

Epoch 47/50
217/217 [==============================] - 210s 968ms/step - loss: 0.3984 - iou_scor
Epoch 48/50
217/217 [==============================] - 214s 984ms/step - loss: 0.4003 - iou_scor
Epoch 49/50
217/217 [==============================] - 218s 1s/step - loss: 0.4008 - iou_score:
Epoch 50/50
217/217 [==============================] - 220s 1s/step - loss: 0.4013 - iou_score:
```

```
model.save_weights('/content/drive/MyDrive/ImageSegmentation/CANe
```

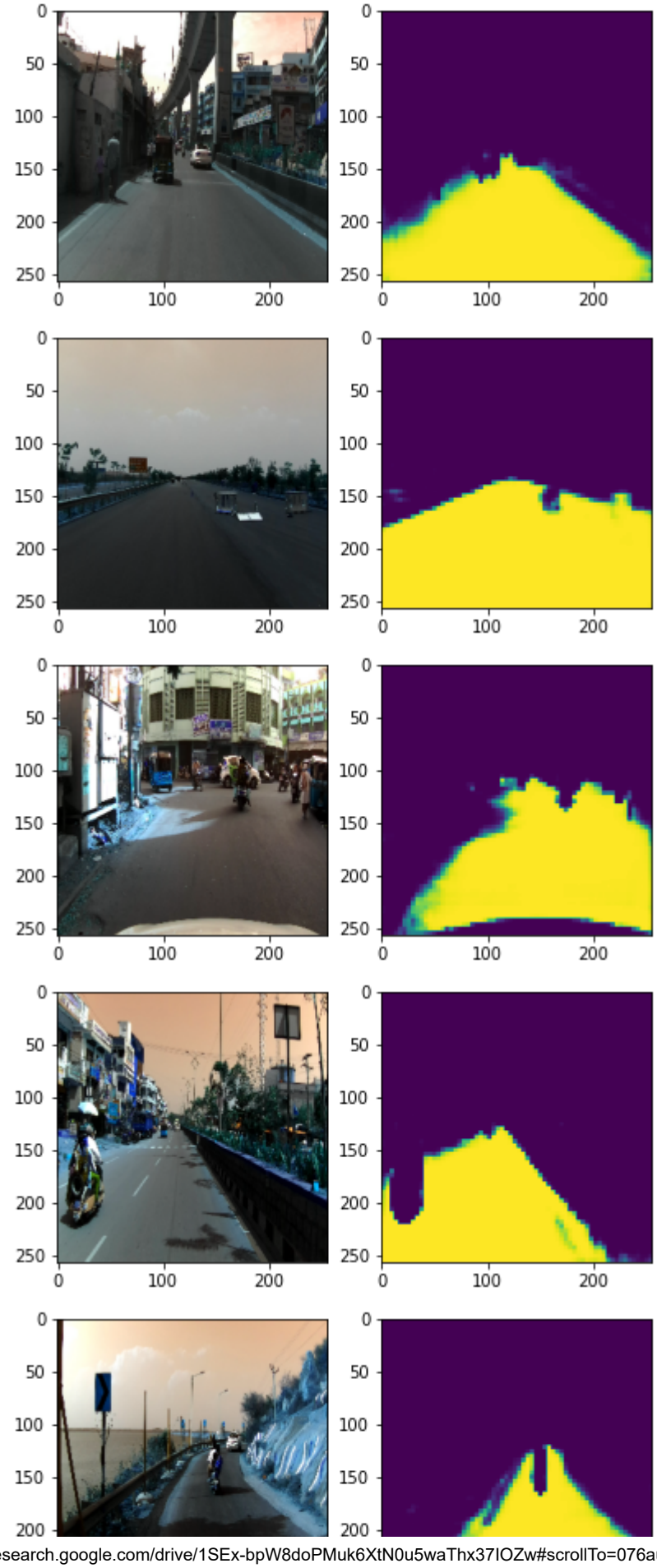
```
# Plot training & validation iou_score values
plt.figure(figsize=(30, 5))
plt.subplot(121)
plt.plot(history.history['iou_score'])
plt.plot(history.history['val_iou_score'])
plt.title('Model iou_score')
plt.ylabel('iou_score')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')

# Plot training & validation loss values
plt.subplot(122)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

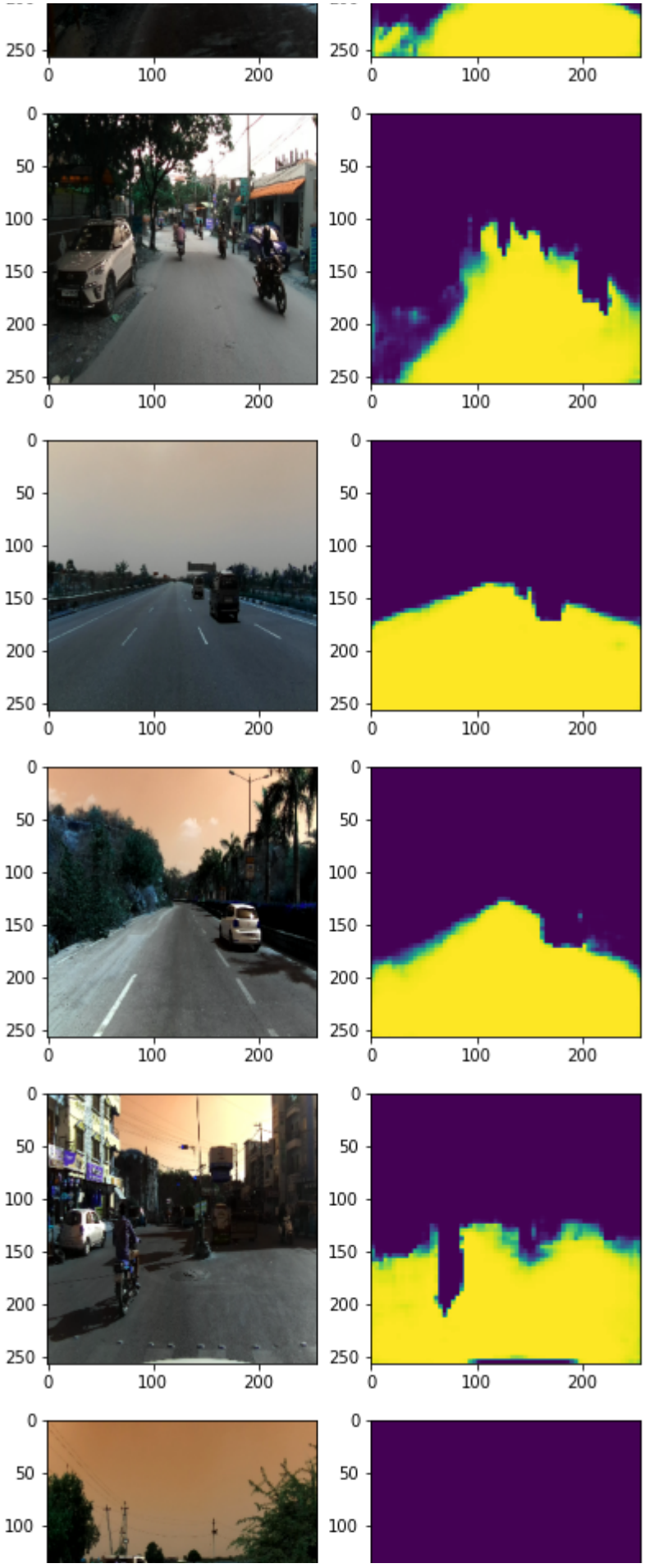





```
for p, i in enumerate(X_test.values[0:10]):
    #original image
    image = cv2.imread(i[0], cv2.IMREAD_UNCHANGED)
    image = cv2.resize(image, (256,256))

    #predicted segmentation map
    predicted  = model.predict(image[np.newaxis,:,:,:])

    #original segmentation map
    image_mask = cv2.imread(i[1], cv2.IMREAD_UNCHANGED)
    image_mask = cv2.resize(image_mask, (256,256))


    plt.figure(figsize=(10,6))
    plt.subplot(131)
    plt.imshow(image)
    plt.subplot(132)
    #plt.imshow(image_mask, cmap='gray', vmax=1, vmin=0)
    #plt.subplot(133)
    plt.imshow(predicted[0,:,:,1], vmax=1, vmin=0)
    plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/tensorflow/python/data/ops/dataset_ops.py:3504:
  "Even though the tf.config.experimental_run_functions_eagerly "
```





```
model_unet.save_weights('/content/drive/MyDrive/ImageSegmentation
```