Segmentation of Indian Traffic

In [1]:

```
from google.colab import drive
drive.mount('/gdrive')
%cd /gdrive
```

```
Mounted at /gdrive
/gdrive
```

In [2]:

```
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 urllib.request
```

```
1. You can download the data from this link, and extract it

2. All your data will be in the folder "data"

3. Inside the data you will be having two folders

|--- data
|-----| ---- images
|-----| ------|----- Scene 1
|-----| ------|--------| ----- Frame 1 (image 1)
|-----| ------|--------| ----- Frame 2 (image 2)
|-----| ------|--------| ----- ...
```

```
|-----| ------|----- Scene 2
|-----| ------|--------| ----- Frame 1 (image 1)
|-----| ------|--------| ----- Frame 2 (image 2)
|-----| ------|--------| ----- ...
|-----| ------|----- .....
|-----| ---- masks
|-----| ------|----- Scene 1
|-----| ------|--------| ----- json 1 (labeled objects in image 1)
|-----| ------|--------| ----- json 2 (labeled objects in image 1)
|-----| ------|--------| ----- ...
|-----| ------|----- Scene 2
|-----| ------|--------| ----- json 1 (labeled objects in image 1)
|-----| ------|--------| ----- json 2 (labeled objects in image 1)
|-----| ------|--------| ----- ...
|-----| ------|----- .....
```

Task 1: Preprocessing

1. Get all the file name and corresponding json files

In [3]:

```
os.chdir('/gdrive/My Drive/Image_Segmentation/segmentation')
os.listdir()
```

Out[3]:

```
['data',
 'Preprocessing.csv',
 'logs',
 'Model_save',
 'Segmentation_Assignment.ipynb',
 'tf_ckpts',
 'Copy of Segmentation_Assignment.ipynb',
 'test_image.png',
 'Preprocessing_2.csv',
 'preprocessed_data.csv',
 'Reference_Preptrained_Unet.ipynb',
 'model4.png',
 'model.png']
```

In [ ]:

```
# First check both image and Mask folder contains same number of sub-folder with same name respectively
image_sub_folder = sorted(os.listdir('data/images'))
mask_sub_folder = sorted(os.listdir('data/mask'))
print('Length of image folder',len(image_sub_folder))
print('Length of image folder',len(mask_sub_folder))
print('Both Image and Mask contains same folder names - ',image_sub_folder == (mask_sub_folder))
```

```
Length of image folder 143
Length of image folder 143
Both Image and Mask contains same folder names -  True
```

In [ ]:

```
def return_file_names_df():
    # write the code that will create a dataframe with two columns ['images', 'json']
    # the column 'image' will have path to images
    # the column 'json' will have path to json files
    img_path = []
    mask_path = []

    for i in tqdm(image_sub_folder):
      img_loc = sorted(os.listdir('data/images/'+str(i)))
      mask_loc = sorted(os.listdir('data/mask/'+str(i)))

      for file_I,file_M in zip(img_loc,mask_loc):
        img_pa = os.path.join('data/images/'+str(i),file_I)
        mask_pa = os.path.join('data/mask/'+str(i),file_M)
        img_path.append(img_pa)
        mask_path.append(mask_pa)

    data_df = pd.DataFrame({'image': img_path,'json': mask_path})


    return data_df
```

In [ ]:

```
data_df = return_file_names_df()
data_df.head()
```

```
100%|██████████| 143/143 [00:09<00:00, 15.24it/s]
```

Out[64]:

	image	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

If you observe the dataframe, we can consider each row as single data point, where first feature is image and the second feature is corresponding json file

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[66]:

```
True
```

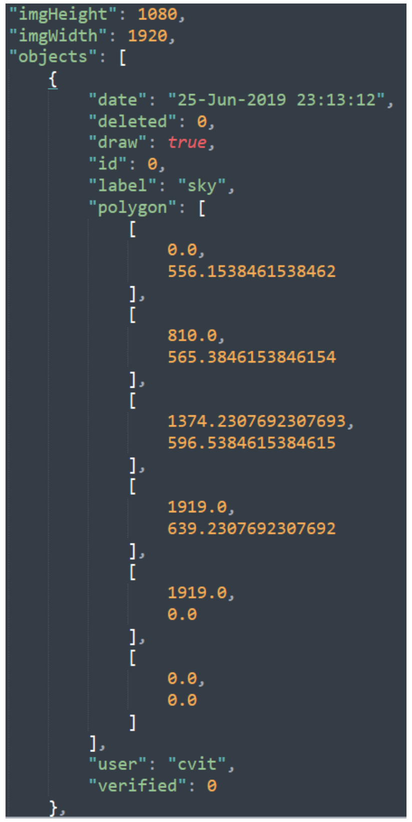
In [ ]:

```
data_df.shape
```

Out[67]:

```
(4008, 2)
```

2. Structure of sample Json file



```
"imgHeight": 1080,
"imgWidth": 1920,
"objects": [
    {
        "date": "25-Jun-2019 23:13:12",
        "deleted": 0,
        "draw": true,
        "id": 0,
        "label": "sky",
        "polygon": [
            [
                0.0,
                556.1538461538462
            ],
            [
                810.0,
                565.3846153846154
            ],
            [
                1374.2307692307693,
                596.5384615384615
            ],
            [
                1919.0,
                639.2307692307692
            ],
            [
                1919.0,
                0.0
            ],
            [
                0.0,
                0.0
            ]
        ],
        "user": "cvit",
        "verified": 0
    },
```

- Each File will have 3 attributes
    - imgHeight: which tells the height of the image
    - imgWidth: which tells the width of the image
    - objects: it is a list of objects, each object will have multiple attributes,
        - label: the type of the object
        - polygon: a list of two element lists, representing the coordinates of the polygon

Compute the unique labels

Let's see how many unique objects are there in the json file. to see how to get the object from the json file please check this blog (https://www.geeksforgeeks.org/read-js python/)

In [ ]:

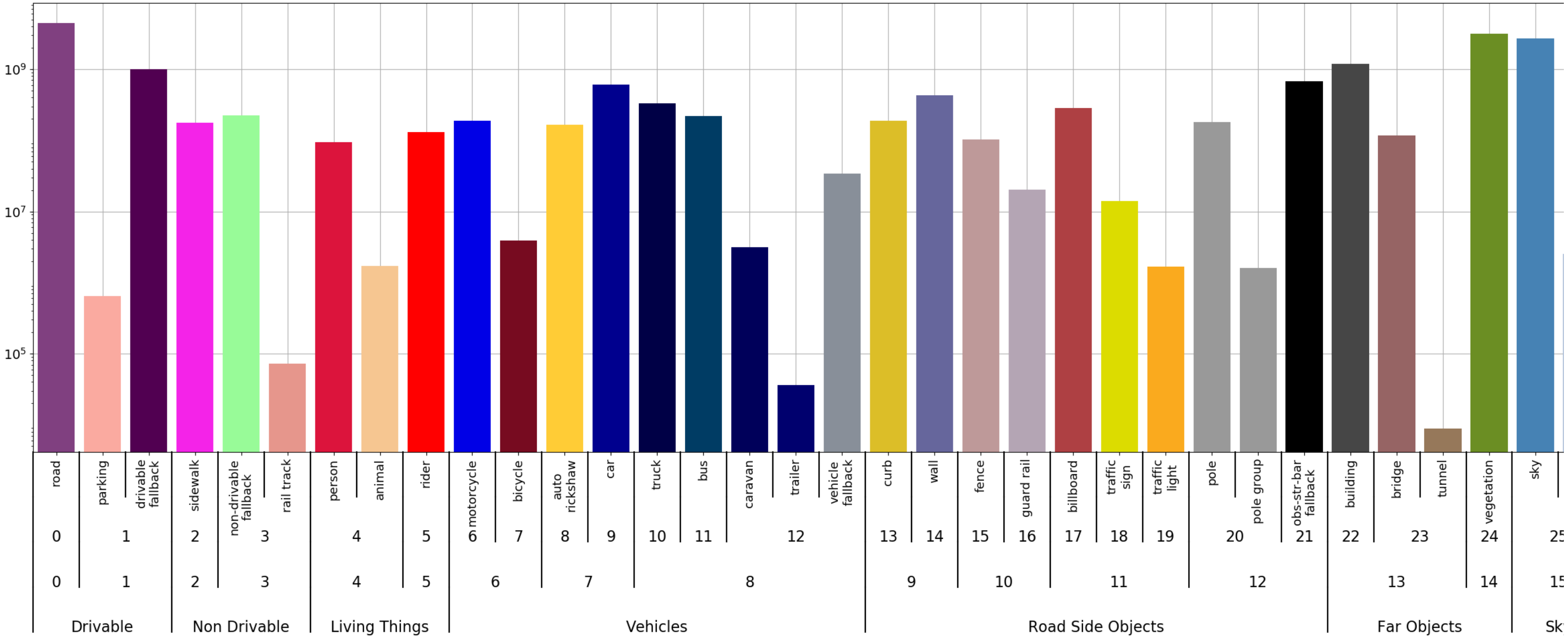
```
def return_unique_labels(data_df):
    # for each file in the column json
    #       read and store all the objects present in that file
    # compute the unique objects and retrun them
    # if open any json file using any editor you will get better sense of it
    all_attributes = [] # storing all attributes
    all_labels = [] # stroing all label values of each row

    for i in tqdm(range(data_df.shape[0])):
      f = open(data_df.json[i],)
      data = json.load(f)
      for j in data['objects']:
        all_attributes.append(j)
      f.close()

    # to get unique label count
    for k in tqdm(range(len(all_attributes))):
     all_labels.append( all_attributes[k]['label'])


    # get unique
    unique_labels = list(set(all_labels))
    print('Number of unique labels ',len(unique_labels))

    return unique_labels
```

In [ ]:

```
unique_labels = return_unique_labels(data_df)
```

```
100%|██████████| 4008/4008 [22:43<00:00,  2.94it/s]
100%|██████████| 434321/434321 [00:00<00:00, 1934991.27it/s]

Number of unique labels  40
```

In [ ]:



In [4]:
```
label_clr = {'road':10, 'parking':20, 'drivable fallback':20,'sidewalk':30,'non-drivable fallback':40,'rail track':40,\
                        'person':50, 'animal':50, 'rider':60, 'motorcycle':70, 'bicycle':70, 'autorickshaw':80,\
                        'car':80, 'truck':90, 'bus':90, 'vehicle fallback':90, 'trailer':90, 'caravan':90,\
                        'curb':100, 'wall':100, 'fence':110,'guard rail':110, 'billboard':120,'traffic sign':120,\
                        'traffic light':120, 'pole':130, 'polegroup':130, 'obs-str-bar-fallback':130,'building':140,\
                        'bridge':140,'tunnel':140, 'vegetation':150, 'sky':160, 'fallback background':160,'unlabeled':0,\
                        'out of roi':0, 'ego vehicle':170, 'ground':180,'rectification border':190,\
                   'train':200}
```

In [5]:
```
class_values = sorted(list(set(label_clr.values())))
print('Class labels', class_values)
class_values = [int(x / 10 )for x in class_values]
print('Class labels', class_values)
print('Number of unique class labels',len(set(label_clr.values())))
```

```
Class labels [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200]
Class labels [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
Number of unique class labels 21
```

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
```

```
* here we have given a number for each of object types, if you see we are having 21 different set of objects
* Note that we have multiplies each object's number with 10, that is just to make different objects look differently in the segmenta
* Before you pass it to the models, you might need to devide the image array /10.
```

3. Extracting the polygons from the json files

In [ ]:

```
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['polygon']])
    w= data['imgWidth']
    h=data['imgHeight']

    return w, h, label, vertexlist
```

In [ ]:

```
w, h, labels, vertexlist = get_poly('data/mask/201/frame0029_gtFine_polygons.json')
```

In [ ]:

```
def grader_3(file):

  w, h, labels, vertexlist = get_poly(file)
  print(len((set(labels)))==18 and len(vertexlist)==227 and w==1920 and 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
```

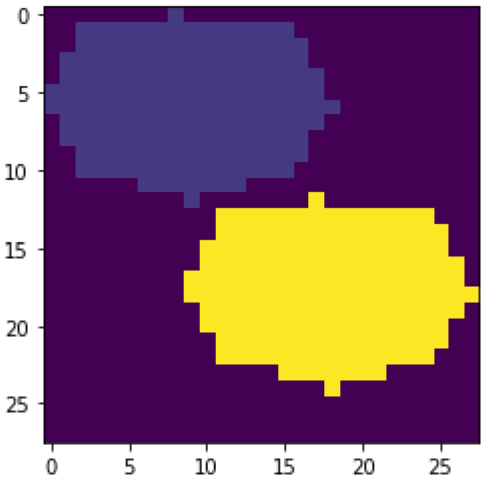
4. Creating Image segmentations by drawing set of polygons

Example

In [ ]:

```
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 [i * (2 * math.pi) / side for i in range(side)] ]
x2 = [ ((math.cos(th) + 2) *9, (math.sin(th) + 3) *6) for th in [i * (2 * math.pi) / side for i in range(side)] ]

img = Image.new("RGB", (28,28))
img1 = ImageDraw.Draw(img)
print('Before',img1)
# please play with the fill value
# writing the first polygon
img1.polygon(x1, fill =10)
# writing the second polygon
img1.polygon(x2, fill =60)
print('After',img1)

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

```
Before <PIL.ImageDraw.ImageDraw object at 0x7fc5eb03f5c0>
After <PIL.ImageDraw.ImageDraw object at 0x7fc5eb03f5c0>
(28, 28, 3)
[[0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0]
 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0]
 [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0]
 [0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0]
 [0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 6 6 6 6 6 6 6 6 6 6 6 6 6 6 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 0 0]
 [0 0 0 0 0 0 0 0 0 0 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 0 0]
 [0 0 0 0 0 0 0 0 0 0 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 0]
 [0 0 0 0 0 0 0 0 0 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 0]
 [0 0 0 0 0 0 0 0 0 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6]
 [0 0 0 0 0 0 0 0 0 0 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 0]
 [0 0 0 0 0 0 0 0 0 0 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 6 6 6 6 6 6 6 6 6 6 6 6 6 6 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 6 6 6 6 6 6 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 6 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]]
```



In [ ]:
```
#os.makedirs('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[labels[i]])
        img=np.array(img)
        im = Image.fromarray(img[:,:,0])
        new_file=file.replace('mask','output')
        new_file=new_file.replace('json','png')
        os.makedirs('data/output/'+file.split('/')[2],exist_ok=True)
        im.save(new_file)
        mask.append(new_file)
    data_df['mask']=mask

    return data_df
```

In [ ]:
```
data_df = compute_masks(data_df)
```

```
100%|██████████| 4008/4008 [05:14<00:00, 12.75it/s]
```

In [ ]:
```
data_df.head(5)
```

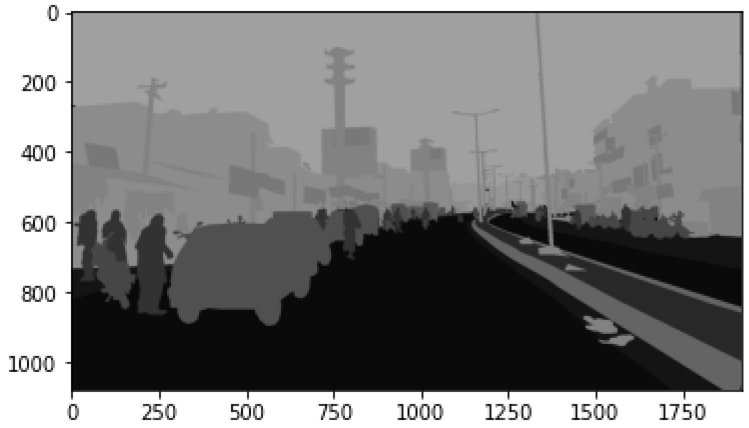
Out[81]:

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

In [ ]:
```
data_df.to_csv('Preprocessing_2.csv',index=False)
```

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]))
    #print(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 [6]:
```
data_df = pd.read_csv('preprocessed_data.csv')
data_df.drop(['Unnamed: 0','json'],inplace=True,axis=1)
data_df.head(2)
```

Out[6]:

	image	mask
0	data/images/201/frame0029_leftImg8bit.jpg	data/output/201/frame0029_gtFine_polygons.png
1	data/images/201/frame0299_leftImg8bit.jpg	data/output/201/frame0299_gtFine_polygons.png

Task 2: Applying Unet to segment the images

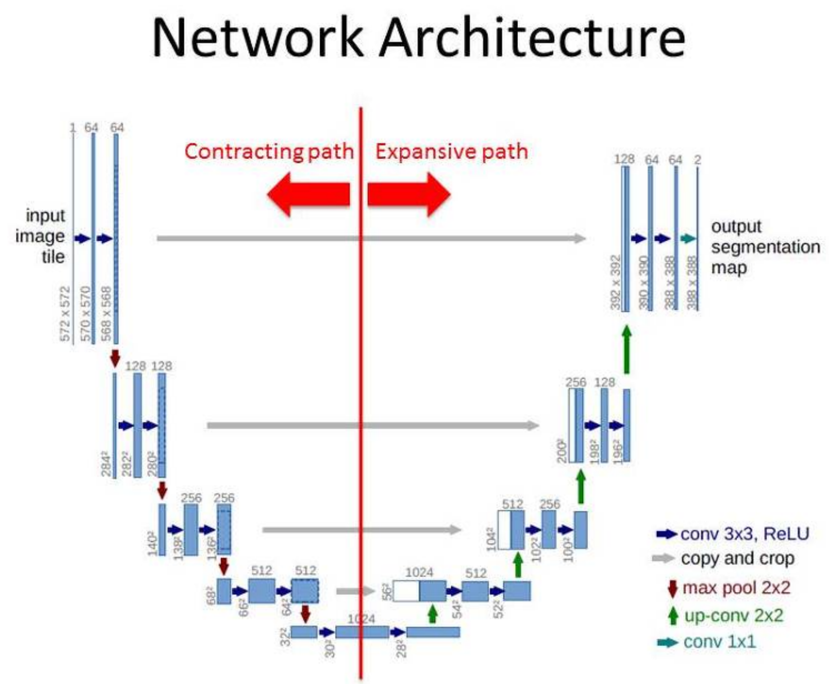
Channels Last

```
. Image data is represented in a three-dimensional array where the last channel represents the color channels, e.g. [rows][cols][cha
```

Channels First

```
Image data is represented in a three-dimensional array where the first channel represents the color channels, e.g. [channels][rows][
```

```
* please check the paper: https://arxiv.org/abs/1505.04597

*
```



```
* As a part of this assignment we won't writingt this whole architecture, rather we will be doing transfer learning
```

* please check the library https://github.com/qubvel/segmentation_models

* You can install it like this "pip install -U segmentation-models==0.2.1", even in google colab you can install the   same with "!
all -U segmentation-models==0.2.1"

* Check the reference notebook in which we have solved one end to end case study of image forgery detection using same unet

* The number of channels in the output will depend on the number of classes in your data, since we know that we are having 21 classe
umber of channels in the output will also be 21

* **This is where we want you to explore, how do you featurize your created segmentation map note that the original map will be of (w,
nd the output will be (w, h, 21) how will you calculate the loss**, you can check the examples in segmentation github

* please use the loss function that is used in the refence notebooks

In [7]:
```
!pip install tensorflow==2.2.0
```

```
Collecting tensorflow==2.2.0
  Downloading https://files.pythonhosted.org/packages/3d/be/679ce5254a8c8d07470efb4a4c00345fae91f766e64f1c2aece8796d7218/tensorflow-2.2
-manylinux2010_x86_64.whl (https://files.pythonhosted.org/packages/3d/be/679ce5254a8c8d07470efb4a4c00345fae91f766e64f1c2aece8796d7218/te
2.0-cp36-cp36m-manylinux2010_x86_64.whl) (516.2MB)
     |████████████████████████████████| 516.2MB 32kB/s
Requirement already satisfied: google-pasta>=0.1.8 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.2.0) (0.2.0)
Requirement already satisfied: h5py<2.11.0,>=2.10.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.2.0) (2.10.0)
Collecting tensorflow-estimator<2.3.0,>=2.2.0
  Downloading https://files.pythonhosted.org/packages/a4/f5/926ae53d6a226ec0fda5208e0e581cffed895ccc89e36ba76a8e60895b78/tensorflow_est
py2.py3-none-any.whl (https://files.pythonhosted.org/packages/a4/f5/926ae53d6a226ec0fda5208e0e581cffed895ccc89e36ba76a8e60895b78/tensor
r-2.2.0-py2.py3-none-any.whl) (454kB)
     |████████████████████████████████| 460kB 47.5MB/s
Requirement already satisfied: absl-py>=0.7.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.2.0) (0.10.0)
Requirement already satisfied: gast==0.3.3 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.2.0) (0.3.3)
Requirement already satisfied: protobuf>=3.8.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.2.0) (3.12.4)
Requirement already satisfied: grpcio>=1.8.6 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.2.0) (1.33.1)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.2.0) (1.1.0)
Requirement already satisfied: keras-preprocessing>=1.1.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.2.0) (1.1.2)
Requirement already satisfied: scipy==1.4.1; python_version >= "3" in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.2.0) (
Collecting tensorboard<2.3.0,>=2.2.0
  Downloading https://files.pythonhosted.org/packages/1d/74/0a6fcb206dcc72a6da9a62dd81784bfdbff5fedb099982861dc2219014fb/tensorboard-2.
any.whl (https://files.pythonhosted.org/packages/1d/74/0a6fcb206dcc72a6da9a62dd81784bfdbff5fedb099982861dc2219014fb/tensorboard-2.2.2-py
hl) (3.0MB)
     |████████████████████████████████| 3.0MB 50.8MB/s
Requirement already satisfied: astunparse==1.6.3 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.2.0) (1.6.3)
Requirement already satisfied: wrapt>=1.11.1 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.2.0) (1.12.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.2.0) (3.3.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.2.0) (1.15.0)
Requirement already satisfied: numpy<2.0,>=1.16.0 in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.2.0) (1.18.5)
Requirement already satisfied: wheel>=0.26; python_version >= "3" in /usr/local/lib/python3.6/dist-packages (from tensorflow==2.2.0) (0
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from protobuf>=3.8.0->tensorflow==2.2.0) (50.3.2)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.6/dist-packages (from tensorboard<2.3.0,>=2.2.0->tensorflow==2
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.6/dist-packages (from tensorboard<2.3.0,>=2.2.0->tensorflow
23.0)
Requirement already satisfied: google-auth<2,>=1.6.3 in /usr/local/lib/python3.6/dist-packages (from tensorboard<2.3.0,>=2.2.0->tensorf
 (1.17.2)
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.6/dist-packages (from tensorboard<2.3.0,>=2.2.0->tensorflow=
1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.6/dist-packages (from tensorboard<2.3.0,>=2.2
w==2.2.0) (0.4.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.6/dist-packages (from tensorboard<2.3.0,>=2.2.0-
2.2.0) (1.7.0)
Requirement already satisfied: importlib-metadata; python_version < "3.8" in /usr/local/lib/python3.6/dist-packages (from markdown>=2.6
rd<2.3.0,>=2.2.0->tensorflow==2.2.0) (2.0.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests<3,>=2.21.0->tensorboard<2.3.
nsorflow==2.2.0) (2020.6.20)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dist-packages (from requests<3,>=2.2
ard<2.3.0,>=2.2.0->tensorflow==2.2.0) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests<3,>=2.21.0->tensorboard<2.3.0,>=2.
ow==2.2.0) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests<3,>=2.21.0->tensorboard<2.3.0
sorflow==2.2.0) (3.0.4)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.6/dist-packages (from google-auth<2,>=1.6.3->tensorboard
0->tensorflow==2.2.0) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4; python_version >= "3" in /usr/local/lib/python3.6/dist-packages (from google-auth<2,>=1.6
rd<2.3.0,>=2.2.0->tensorflow==2.2.0) (4.6)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.6/dist-packages (from google-auth<2,>=1.6.3->tensorboar
2.0->tensorflow==2.2.0) (4.1.1)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.6/dist-packages (from google-auth-oauthlib<0.5,>=0.4.
d<2.3.0,>=2.2.0->tensorflow==2.2.0) (1.3.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.6/dist-packages (from importlib-metadata; python_version < "3.8"->ma
->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (3.3.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.6/dist-packages (from pyasn1-modules>=0.2.1->google-auth<
nsorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.6/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oa
0.4.1->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (3.1.0)
Installing collected packages: tensorflow-estimator, tensorboard, tensorflow
  Found existing installation: tensorflow-estimator 2.3.0
    Uninstalling tensorflow-estimator-2.3.0:
      Successfully uninstalled tensorflow-estimator-2.3.0
  Found existing installation: tensorboard 2.3.0
    Uninstalling tensorboard-2.3.0:
      Successfully uninstalled tensorboard-2.3.0
  Found existing installation: tensorflow 2.3.0
    Uninstalling tensorflow-2.3.0:
      Successfully uninstalled tensorflow-2.3.0
Successfully installed tensorboard-2.2.2 tensorflow-2.2.0 tensorflow-estimator-2.2.0
```

```
In [8]: !pip install keras==2.3.1
```

```
Collecting keras==2.3.1
  Downloading https://files.pythonhosted.org/packages/ad/fd/6bfe87920d7f4fd475acd28500a42482b6b84479832bdc0fe9e589a60ceb/Keras-2.3.1-py2
y.whl (https://files.pythonhosted.org/packages/ad/fd/6bfe87920d7f4fd475acd28500a42482b6b84479832bdc0fe9e589a60ceb/Keras-2.3.1-py2.py3-n
 (377kB)
     |████████████████████████████████| 378kB 10.3MB/s eta 0:00:01
Collecting keras-applications>=1.0.6
  Downloading https://files.pythonhosted.org/packages/71/e3/19762fdfc62877ae9102edf6342d71b28fbfd9dea3d2f96a882ce099b03f/Keras_Applicat
3-none-any.whl (https://files.pythonhosted.org/packages/71/e3/19762fdfc62877ae9102edf6342d71b28fbfd9dea3d2f96a882ce099b03f/Keras_Applica
py3-none-any.whl) (50kB)
     |████████████████████████████████| 51kB 7.5MB/s  eta 0:00:01
Requirement already satisfied: scipy>=0.14 in /usr/local/lib/python3.6/dist-packages (from keras==2.3.1) (1.4.1)
Requirement already satisfied: h5py in /usr/local/lib/python3.6/dist-packages (from keras==2.3.1) (2.10.0)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.6/dist-packages (from keras==2.3.1) (3.13)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.6/dist-packages (from keras==2.3.1) (1.18.5)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from keras==2.3.1) (1.15.0)
Requirement already satisfied: keras-preprocessing>=1.0.5 in /usr/local/lib/python3.6/dist-packages (from keras==2.3.1) (1.1.2)
Installing collected packages: keras-applications, keras
  Found existing installation: Keras 2.4.3
    Uninstalling Keras-2.4.3:
      Successfully uninstalled Keras-2.4.3
Successfully installed keras-2.3.1 keras-applications-1.0.8
```

In [9]:
```
!pip install -U segmentation-models==0.2.1
```

```
Collecting segmentation-models==0.2.1
  Downloading https://files.pythonhosted.org/packages/10/bf/253c8834014a834cacf2384c72872167fb30ccae7a56c6ce46285b03245c/segmentation_mo
y2.py3-none-any.whl (https://files.pythonhosted.org/packages/10/bf/253c8834014a834cacf2384c72872167fb30ccae7a56c6ce46285b03245c/segment
0.2.1-py2.py3-none-any.whl) (44kB)
     |████████████████████████████████| 51kB 5.5MB/s  eta 0:00:01
Collecting image-classifiers==0.2.0
  Downloading https://files.pythonhosted.org/packages/de/32/a1e74e03f74506d1e4b46bb2732ca5a7b18ac52a36b5e3547e63537ce74c/image_classifi
2.py3-none-any.whl (https://files.pythonhosted.org/packages/de/32/a1e74e03f74506d1e4b46bb2732ca5a7b18ac52a36b5e3547e63537ce74c/image_cl
2.0-py2.py3-none-any.whl) (76kB)
     |████████████████████████████████| 81kB 7.5MB/s  eta 0:00:01
Requirement already satisfied, skipping upgrade: scikit-image in /usr/local/lib/python3.6/dist-packages (from segmentation-models==0.2.
Requirement already satisfied, skipping upgrade: keras>=2.2.0 in /usr/local/lib/python3.6/dist-packages (from segmentation-models==0.2.
Requirement already satisfied, skipping upgrade: keras-applications>=1.0.7 in /usr/local/lib/python3.6/dist-packages (from segmentation
1) (1.0.8)
Requirement already satisfied, skipping upgrade: scipy>=0.19.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image->segmentati
2.1) (1.4.1)
Requirement already satisfied, skipping upgrade: networkx>=2.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image->segmentati
2.1) (2.5)
Requirement already satisfied, skipping upgrade: pillow>=4.3.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image->segmentati
2.1) (7.0.0)
Requirement already satisfied, skipping upgrade: PyWavelets>=0.4.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image->segment
==0.2.1) (1.1.1)
Requirement already satisfied, skipping upgrade: matplotlib!=3.0.0,>=2.0.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image
n-models==0.2.1) (3.2.2)
Requirement already satisfied, skipping upgrade: imageio>=2.3.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image->segmentat
0.2.1) (2.4.1)
Requirement already satisfied, skipping upgrade: h5py in /usr/local/lib/python3.6/dist-packages (from keras>=2.2.0->segmentation-models=
0.0)
Requirement already satisfied, skipping upgrade: keras-preprocessing>=1.0.5 in /usr/local/lib/python3.6/dist-packages (from keras>=2.2.
on-models==0.2.1) (1.1.2)
Requirement already satisfied, skipping upgrade: pyyaml in /usr/local/lib/python3.6/dist-packages (from keras>=2.2.0->segmentation-mode
 (3.13)
Requirement already satisfied, skipping upgrade: six>=1.9.0 in /usr/local/lib/python3.6/dist-packages (from keras>=2.2.0->segmentation-
1) (1.15.0)
Requirement already satisfied, skipping upgrade: numpy>=1.9.1 in /usr/local/lib/python3.6/dist-packages (from keras>=2.2.0->segmentatio
2.1) (1.18.5)
Requirement already satisfied, skipping upgrade: decorator>=4.3.0 in /usr/local/lib/python3.6/dist-packages (from networkx>=2.0->scikit
ntation-models==0.2.1) (4.4.2)
Requirement already satisfied, skipping upgrade: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib!=3.0.0,>=2.0.0
e->segmentation-models==0.2.1) (0.10.0)
Requirement already satisfied, skipping upgrade: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib!=3.0.0
kit-image->segmentation-models==0.2.1) (2.8.1)
Requirement already satisfied, skipping upgrade: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib!=3.0.0,>=2
-image->segmentation-models==0.2.1) (1.2.0)
Requirement already satisfied, skipping upgrade: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packages (fro
b!=3.0.0,>=2.0.0->scikit-image->segmentation-models==0.2.1) (2.4.7)
Installing collected packages: image-classifiers, segmentation-models
Successfully installed image-classifiers-0.2.0 segmentation-models-0.2.1
```

In [10]:

```python
# install required Package
import tensorflow as tf
# tf.enable_eager_execution()
import os
import numpy as np
import pandas as pd
import cv2
import matplotlib.pyplot as plt
# from hilbert import hilbertCurve
import imgaug.augmenters as iaa
import numpy as np
# import albumentations as A
os.environ['TF_FORCE_GPU_ALLOW_GROWTH'] = 'true'
from tensorflow.keras import layers,Model
from tensorflow.keras.layers import Dense,Input,Conv2D,MaxPool2D,Activation,Dropout,Flatten, BatchNormalization, ReLU, Reshape,Flatten
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, LearningRateScheduler, ReduceLROnPlateau, TensorBoard

from tensorflow.keras.models import Model
import random as rn
```

In [11]:

```
# here dir_path is the route directory where all the images and segmentation maps are there
dir_path = "data/images/"
dir_path_output = "data/output/"
file_names = set()
file_names_output = set()
for folder in tqdm(os.listdir(dir_path)):

    dir_paths = "data/images/" +str(folder)
    for i in os.listdir(dir_paths):
      path= (i.split('.')[0].split('_')[0])
      file_names.add(str(folder) +str('/')+path)



for folder in tqdm(os.listdir(dir_path_output)):
    dir_paths = "data/output/" +str(folder)
    for i in os.listdir(dir_paths):
      path= (i.split('.')[0].split('_')[0])
      file_names_output.add(str(folder) +str('/')+path)
```

```
100%|██████████| 143/143 [00:13<00:00, 10.53it/s]
100%|██████████| 143/143 [00:11<00:00, 12.53it/s]
```

In [12]:

```
print('Total_number of unique files', len(file_names))
print('Total_number of unique files- Output Mask folder', len(file_names_output))
```

```
Total_number of unique files 4008
Total_number of unique files- Output Mask folder 4008
```

In [13]:

```
from sklearn.model_selection import train_test_split
X_train, X_test = train_test_split(list(file_names), test_size=0.20, random_state=42)
```

In [14]:

```
X_train[:5]
```

Out[14]:

```
['280/frame0574',
 '283/frame3574',
 '252/frame1536',
 '338/frame61726',
 '231/frame3047']
```

In [15]:

```
# we are importing the pretrained unet from the segmentation models
# https://github.com/qubvel/segmentation_models
import segmentation_models as sm
from segmentation_models import Unet
# sm.set_framework('tf.keras')
tf.keras.backend.set_image_data_format('channels_last')
```

```
Using TensorFlow backend.
/usr/local/lib/python3.6/dist-packages/classification_models/resnext/__init__.py:4: UserWarning: Current ResNext models are deprecated,
plications ResNeXt models
  warnings.warn('Current ResNext models are deprecated, '
```

In [16]:

```
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense,Input,Conv2D,MaxPool2D,Activation,Dropout,Flatten,BatchNormalization, ReLU, Reshape,Flatten
from tensorflow.keras.models import Model
import random as rn
import keras
```

In [17]:

```
# loading the unet model and using the resnet 34 and initilized weights with imagenet weights
# "classes" :different types of classes in the dataset
# Create Model
os.environ['PYTHONHASHSEED'] = '0'

##https://keras.io/getting-started/faq/#how-can-i-obtain-reproducible-results-using-keras-during-development
## Have to clear the session. If you are not clearing, Graph will create again and again and graph size will increses.
## Varibles will also set to some value from before session
tf.keras.backend.clear_session()

## Set the random seed values to regenerate the model.
np.random.seed(0)
rn.seed(0)

model = Unet('resnet34', encoder_weights='imagenet', classes=21, activation='softmax',encoder_freeze=True, input_shape=(224,224,3))
```

```
Downloading data from https://github.com/qubvel/classification_models/releases/download/0.0.1/resnet34_imagenet_1000_no_top.h5 (https:/
ubvel/classification_models/releases/download/0.0.1/resnet34_imagenet_1000_no_top.h5)
85524480/85521592 [==============================] - 1s 0us/step
```

In [ ]:

```
model.summary()
```

```
zero_padding2d_9 (ZeroPadding2D (None, 58, 58, 64)   0           stage2_unit1_relu1[0][0]

stage2_unit1_conv1 (Conv2D)     (None, 28, 28, 128)  73728       zero_padding2d_9[0][0]

stage2_unit1_bn2 (BatchNormaliz (None, 28, 28, 128)  512         stage2_unit1_conv1[0][0]

stage2_unit1_relu2 (Activation) (None, 28, 28, 128)  0           stage2_unit1_bn2[0][0]

zero_padding2d_10 (ZeroPadding2 (None, 30, 30, 128)  0           stage2_unit1_relu2[0][0]

stage2_unit1_conv2 (Conv2D)     (None, 28, 28, 128)  147456      zero_padding2d_10[0][0]

stage2_unit1_sc (Conv2D)        (None, 28, 28, 128)  8192        stage2_unit1_relu1[0][0]

add_4 (Add)                     (None, 28, 28, 128)  0           stage2_unit1_conv2[0][0]
                                                                 stage2_unit1_sc[0][0]

stage2_unit2_bn1 (BatchNormaliz (None, 28, 28, 128)  512         add_4[0][0]
```

In [18]:

```
# 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))
```

In [19]:

```
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:
    # we will be modifying this CLASSES according to your data/problems
    #CLASSES = class_values
    CLASSES = list(np.unique(list(label_clr.values())))
    #classes=CLASSES

    # the parameters needs to changed based on your requirements
    # here we are collecting the file_names because in our dataset, both our images and maks will have same file name
    # ex: fil_name.jpg   file_name.mask.jpg
    def __init__(self, images_dir,images_dir_mask ,file_names,classes, isTest):
        print(classes)

        self.ids = file_names
        # the paths of images
        self.images_fps   = [os.path.join(images_dir, image_id+'_leftImg8bit.jpg') for image_id in self.ids]
        # the paths of segmentation images
        self.masks_fps    = [os.path.join(images_dir_mask, image_id+"_gtFine_polygons.png") for image_id in self.ids]
        # giving labels for each class
        #self.class_values = [self.CLASSES.index(cls) for cls in classes]
        self.class_values = CLASSES
        print(self.class_values)
        # As per Hint - Augumentation not required for Validation data
        self.isTest = isTest
    def __getitem__(self, i):

        # read data
        #print('Reading a data')

        image = cv2.imread(self.images_fps[i], cv2.IMREAD_UNCHANGED)
        image = cv2.resize(image, (224, 224),interpolation=cv2.INTER_AREA)
        #image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        mask  = cv2.imread(self.masks_fps[i], cv2.IMREAD_UNCHANGED)
        mask = cv2.resize(mask, (224, 224),interpolation=cv2.INTER_AREA)

        image_mask = mask

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

        #Augumentation only for train
        if self.isTest == False:
            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)
            else:
                image = aug5.augment_image(image)
                #image_mask = image_mask


        return image, image_mask

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


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)]
        #print(type(batch))

        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 [19]:

In [20]:

```
# Dataset for train images
CLASSES = list(np.unique(list(label_clr.values())))
train_dataset = Dataset(dir_path,dir_path_output,X_train, classes=CLASSES,isTest=False)
test_dataset  = Dataset(dir_path,dir_path_output,X_test, classes=CLASSES,isTest=True)
```

```
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200]
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200]
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200]
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200]
```

In [21]:

```
#UNET
train_dataloader = Dataloder(train_dataset, batch_size=32, shuffle=True)
test_dataloader = Dataloder(test_dataset, batch_size=32, shuffle=True)

print(train_dataloader[0][0].shape)
assert train_dataloader[0][0].shape == (32, 224, 224, 3)
assert train_dataloader[0][1].shape == (32, 224, 224, 21)
```

```
(32, 224, 224, 3)
```

In [ ]:

In [36]:

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, LearningRateScheduler, ReduceLROnPlateau, TensorBoard
```

In [37]:

```
# TensorBoard Creation

ACCURACY_THRESHOLD_test = 0.5
class myCallback(tf.keras.callbacks.Callback):

    def on_epoch_end(self, epoch, logs={}):

      if(logs.get('val_iou_score') >= ACCURACY_THRESHOLD_test and logs.get('iou_score') >= ACCURACY_THRESHOLD_test):
        print("\nReached %2.2f%% accuracy, so stopping training!!" %(ACCURACY_THRESHOLD_test*100))
        self.model.stop_training = True



early_stop_iou_scores = myCallback()


%load_ext tensorboard
import datetime
folder_name = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

# Create log folder - TensorBoard
log_dir="/gdrive/My Drive/Image_Segmentation/segmentation/logs/fit/" + folder_name
tensorboard_callback =TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True)

print('Folder_name', folder_name)


early_stop = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss', min_delta=0, patience=20, verbose=0, mode='auto',
    baseline=None, restore_best_weights=False
)

red_lr = tf.keras.callbacks.ReduceLROnPlateau(
    monitor="val_loss",
    factor=0.1,
    patience=5,
    verbose=0,
    mode="auto",
    min_delta=0.0001,
    cooldown=0,
    min_lr=0
)
```

```
filepath="/gdrive/My Drive/Image_Segmentation/segmentation/Model_save/better_model_updated-{epoch:02d}.h5"
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_iou_score',  verbose=1, save_best_only=True, mode='max')
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
Folder_name 20201103-013529
```

In [40]:

```
# TensorBoard Creation

ACCURACY_THRESHOLD_test = 0.5
class myCallback(tf.keras.callbacks.Callback):

    def on_epoch_end(self, epoch, logs={}):

      if(logs.get('val_iou_score') >= ACCURACY_THRESHOLD_test and logs.get('iou_score') >= ACCURACY_THRESHOLD_test):
        print("\nReached %2.2f%% accuracy, so stopping training!!" %(ACCURACY_THRESHOLD_test*100))
        self.model.stop_training = True



early_stop_iou_scores = myCallback()


%load_ext tensorboard
import datetime
folder_name = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")

# Create log folder - TensorBoard
log_dir="/gdrive/My Drive/Image_Segmentation/segmentation/logs/fit/" + folder_name
tensorboard_callback =keras.callbacks.TensorBoard(log_dir=log_dir,histogram_freq=1, write_graph=True)

print('Folder_name', folder_name)


early_stop = keras.callbacks.EarlyStopping(
    monitor='val_loss', min_delta=0, patience=20, verbose=0, mode='auto',
    baseline=None, restore_best_weights=False
)

red_lr = keras.callbacks.ReduceLROnPlateau(
    monitor="val_loss",
    factor=0.1,
    patience=5,
    verbose=0,
    mode="auto",
    min_delta=0.0001,
    cooldown=0,
    min_lr=0
)
```

```
filepath="/gdrive/My Drive/Image_Segmentation/segmentation/Model_save/better_model_updated-{epoch:02d}.h5"
checkpoint = keras.callbacks.ModelCheckpoint(filepath=filepath, monitor='val_iou_score',  verbose=1, save_best_only=True, mode='max')
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
Folder_name 20201103-013833
```

In [33]:

```
# https://github.com/qubvel/segmentation_models
import segmentation_models as sm
from segmentation_models.metrics import iou_score
from segmentation_models import Unet
import tensorflow as tf
import keras
optim = keras.optimizers.Adam(learning_rate=0.001)

focal_loss = sm.losses.cce_dice_loss
```

In [ ]:

```


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

focal_loss = sm.losses.cce_dice_loss

# actulally total_loss can be imported directly from library, above example just show you how to manipulate with losses
# total_loss = sm.losses.binary_focal_dice_loss
# or total_loss = sm.losses.categorical_focal_dice_loss

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

In [ ]:

```
#UNET and Res34 step per epoch 100 -  Batch size 32
history = model.fit_generator(train_dataloader, epochs=150,
                              validation_data=test_dataloader ,
                              callbacks = [early_stop_iou_scores,checkpoint,red_lr,tensorboard_callback,early_stop ])
```

```
del_news-06.h5
Epoch 7/150
100/100 [==============================] - 293s 3s/step - loss: 1.2464 - iou_score: 0.3759 - val_loss: 1.2391 - val_iou_score: 0.3812

Epoch 00007: val_iou_score improved from 0.37684 to 0.38124, saving model to /gdrive/My Drive/Image_Segmentation/segmentation/Model_save
del_news-07.h5
Epoch 8/150
100/100 [==============================] - 299s 3s/step - loss: 1.2178 - iou_score: 0.3815 - val_loss: 1.3417 - val_iou_score: 0.3857

Epoch 00008: val_iou_score improved from 0.38124 to 0.38570, saving model to /gdrive/My Drive/Image_Segmentation/segmentation/Model_save
del_news-08.h5
Epoch 9/150
100/100 [==============================] - 280s 3s/step - loss: 1.2146 - iou_score: 0.3832 - val_loss: 1.2445 - val_iou_score: 0.3847

Epoch 00009: val_iou_score did not improve from 0.38570
Epoch 10/150
100/100 [==============================] - 292s 3s/step - loss: 1.1794 - iou_score: 0.3887 - val_loss: 1.1154 - val_iou_score: 0.3870

Epoch 00010: val_iou_score improved from 0.38570 to 0.38698, saving model to /gdrive/My Drive/Image_Segmentation/segmentation/Model_save
del news-10.h5
```

In [3]:

```
#reconstruction 1 - Above training stopped unfortunately, so using best model weight to continue the processing
import keras
model = keras.models.load_model("/gdrive/My Drive/Image_Segmentation/segmentation/Model_save/better_model_news-30.h5")
history = model.fit_generator(train_dataloader, epochs=150,
                              validation_data=test_dataloader ,
                              callbacks = [early_stop_iou_scores,checkpoint,red_lr,tensorboard_callback,early_stop ])
```

```
Epoch 42/150
100/100 [==============================] - 319s 3s/step - loss: 0.8007 - iou_score: 0.4978 - val_loss: 0.7816 - val_iou_score: 0.4980

Epoch 00042: val_iou_score improved from 0.49699 to 0.49801,  saving model to /gdrive/My Drive/Image_Segmentation/segmentation/Model_sav
odel_updated-42.h5
Epoch 43/150
100/100 [==============================] - 293s 3s/step - loss: 0.7938 - iou_score: 0.4977 - val_loss: 0.7826 - val_iou_score: 0.4948

Epoch 00043: val_iou_score did not improve from 0.49801
Epoch 44/150
100/100 [==============================] - 283s 3s/step - loss: 0.7988 - iou_score: 0.4974 - val_loss: 0.7805 - val_iou_score: 0.4936

Epoch 00044: val_iou_score did not improve from 0.49801
Epoch 45/150
100/100 [==============================] - 291s 3s/step - loss: 0.7956 - iou_score: 0.4977 - val_loss: 0.7810 - val_iou_score: 0.4966


Epoch 00045: val_iou_score did not improve from 0.49801
```

In [2]:

```
#reconstruction 2 -  Above training stopped due to exceed RAM usage in colab, so using best model weight to continue the processing

import keras
model = keras.models.load_model("/gdrive/My Drive/Image_Segmentation/segmentation/Model_save/better_model_news-42.h5")
history = model.fit_generator(train_dataloader, epochs=150,
                              validation_data=test_dataloader ,
                              callbacks = [early_stop_iou_scores,checkpoint,red_lr,tensorboard_callback,early_stop ])
```

```
Epoch 1/150
100/100 [==============================] - 2844s 25s/step - loss: 0.7917 - iou_score: 0.5001 - val_loss: 0.7816 - val_iou_score: 0.5003

Epoch 00001: val_iou_score improved from -inf to 0.50034, saving model to /gdrive/My Drive/Image_Segmentation/segmentation/Model_save/be
ews-01.h5

Reached 50.00% accuracy, so stopping training!!
```

In [ ]:

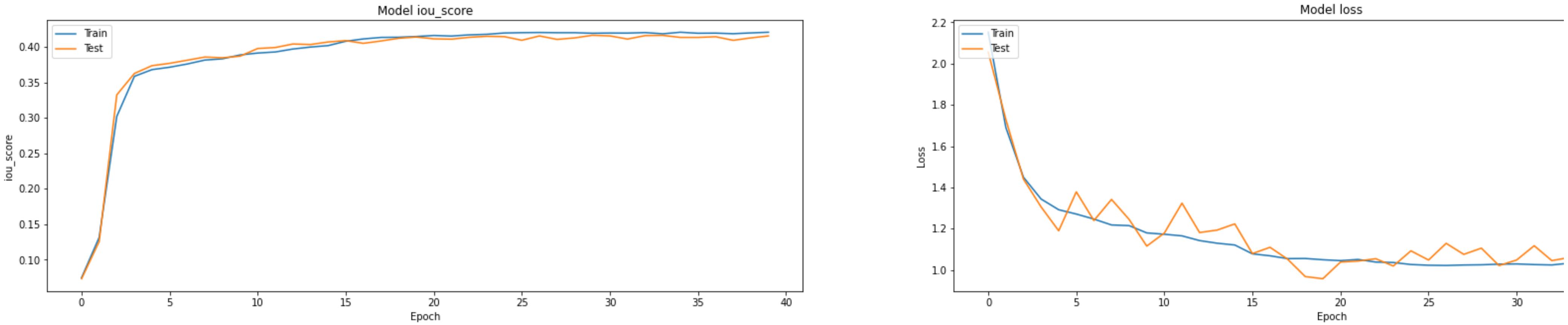
In [ ]:

```
# /gdrive/My Drive/Image_Segmentation/segmentation/Model_save/best_model_news-17.h5 - Best weight location
# best - Epoch 00060: val_iou_score improved from 0.44134 to 0.44197, saving model to /gdrive/My Drive/Image_Segmentation/segmentation/M
# best - /gdrive/My Drive/Image_Segmentation/segmentation/Model_save/best_model_news-01.h5  = 0.472
```

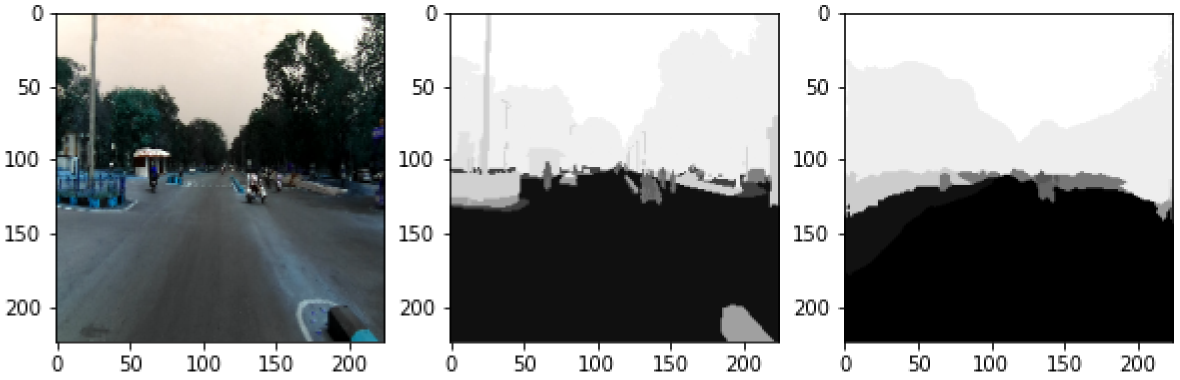
In [ ]:

```
# The below grapgh is only from Epoch 1 to Epoch 40
# Recondtsruction 1-  Stopped unfortunately due to RAM limitage reached -  unable to draw the grap
# Recondtsruction 2 -  Achieved expected result in first epoch itself -  So graph not required.
```

In [ ]:
```
# 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()
```



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

            #image = cv2.imread(list(X_test['image'])[p], cv2.IMREAD_UNCHANGED)
            image = cv2.imread(os.path.join(dir_path, i+'_leftImg8bit.jpg'), cv2.IMREAD_UNCHANGED)
            image = cv2.resize(image, (224,224),interpolation = cv2.INTER_NEAREST)

            #predicted segmentation map
            #print(np.newaxis)
            pred_mask  = model.predict(image[np.newaxis,:,:,:])
            pred_mask = tf.argmax(pred_mask, axis=-1)

            #original segmentation map
            image_mask = cv2.imread(os.path.join(dir_path_output, i+'_gtFine_polygons.png'), cv2.IMREAD_UNCHANGED)
            image_mask = cv2.resize(image_mask, (224,224),interpolation = cv2.INTER_NEAREST)


            plt.figure(figsize=(10,6))
            plt.subplot(131)
            plt.imshow(image)
            plt.subplot(132)
            plt.imshow(image_mask, cmap='gray')
            plt.subplot(133)
            plt.imshow(pred_mask[0], cmap='gray')
            plt.show()

            if p == 20:
              break
```



0
50
0
50
0
50