Segmentation of Indian Traffic

```
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
```

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

     Mounted at /gdrive
     /gdrive
```

```
from google.colab import drive
```

Saving...

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

```
!pip install pyunpack
!pip install patool

     Collecting pyunpack
       Downloading pyunpack-0.2.2-py2.py3-none-any.whl (3.8 kB)
     Collecting entrypoint2
       Downloading entrypoint2-1.0-py3-none-any.whl (9.8 kB)
     Collecting easyprocess
       Downloading EasyProcess-1.1-py3-none-any.whl (8.7 kB)
     Installing collected packages: entrypoint2, easyprocess, pyunpack
     Successfully installed easyprocess-1.1 entrypoint2-1.0 pyunpack-0.2.2
     Collecting patool
       Downloading patool-1.12-py2.py3-none-any.whl (77 kB)
          |████████████████████████████████| 77 kB 2.8 MB/s
     Installing collected packages: patool
     Successfully installed patool-1.12
```

```
!pip install pyunpack
```

```
!pip install patool

from pyunpack import Archive
```

```
Requirement already satisfied: pyunpack in /usr/local/lib/python3.7/dist-packages (0
Requirement already satisfied: easyprocess in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: entrypoint2 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: patool in /usr/local/lib/python3.7/dist-packages (1.12
```

```
!pip install -U segmentation-models
```

```
Collecting segmentation-models
  Downloading segmentation_models-1.0.1-py3-none-any.whl (33 kB)
Collecting keras-applications<=1.0.8,>=1.0.7
  Downloading Keras_Applications-1.0.8-py3-none-any.whl (50 kB)
     |████████████████████████████████| 50 kB 5.5 MB/s
Collecting image-classifiers==1.0.0
  Downloading image_classifiers-1.0.0-py3-none-any.whl (19 kB)
Collecting efficientnet==1.0.0
  Downloading efficientnet-1.0.0-py3-none-any.whl (17 kB)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages (from k
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,>=4.3.0 in /usr/local/lib/python
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.0 in /usr/local/lib/python3.7/
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: scipy>=1.0.1 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.7/dist-packag
 Saving...                            pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local
                                      python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (fr
Installing collected packages: keras-applications, image-classifiers, efficientnet, s
Successfully installed efficientnet-1.0.0 image-classifiers-1.0.0 keras-applications-
```

```
from zipfile import ZipFile
```

```
#Reference: https://thispointer.com/python-how-to-unzip-a-file-extract-single-multiple-or-

with ZipFile('/content/drive/MyDrive/Segmentation/data-002.zip', 'r') as zipObj:
   # Extract all the contents of zip file in current directory
   zipObj.extractall()
```

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

```
Collecting git+https://github.com/qubvel/segmentation_models
  Cloning https://github.com/qubvel/segmentation_models to /tmp/pip-req-build-unj6dac
  Running command git clone -q https://github.com/qubvel/segmentation_models /tmp/pip
  Running command git submodule update --init --recursive -q
Requirement already satisfied: keras_applications<=1.0.8,>=1.0.7 in /usr/local/lib/py
Requirement already satisfied: image-classifiers==1.0.0 in /usr/local/lib/python3.7/d
Requirement already satisfied: efficientnet==1.0.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: scikit-image in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages (from k
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: scipy>=1.0.1 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.0 in /usr/local/lib/python3.7/
Requirement already satisfied: pillow!=7.1.0,!=7.1.1,>=4.3.0 in /usr/local/lib/python
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (fr
```

```
import warnings
warnings.filterwarnings('ignore')

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
```

Saving...

```
import matplotlib.pyplot as plt
import seaborn as sns
import urllib
from sklearn.model_selection import train_test_split
import imgaug.augmenters as iaa
import gc
import tensorflow as tf
import math
from PIL import Image, ImageDraw
from PIL import ImagePath
import segmentation_models as sm
from segmentation_models.metrics import iou_score
from segmentation_models import Unet
```

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

Saving... ×

Task 1: Preprocessing

1. Get all the file name and corresponding json files

```
def return_file_names_df():
    directory_images = 'data/images'
    directory_mask = 'data/mask'
    image_folders = sorted(os.listdir('data/images'))
    mask_folders = sorted(os.listdir('data/mask'))
    all_image_files = []
    folder_number_image = []
    for i in image_folders:
        image_files = sorted(os.listdir(directory_images + '/' + i))
        length_1 = [i]*len(image_files)
        all_image_files = all_image_files + image_files
        folder_number_image = folder_number_image + length_1
    all_json_files = []
    folder_number_json = []
    for j in mask_folders:
        json_files = sorted(os.listdir(directory_mask + '/' + j))
        length_2 = [j]*len(json_files)
        all_json_files = all_json_files + json_files
        folder_number_json = folder_number_json + length_2
    all_image_paths = []
    all_json_paths = []
    for k in range(len(folder_number_image)):
        image_path = directory_images + '/' + folder_number_image[k] + '/' + all_image_fil
        json_path = directory_mask + '/' + folder_number_json[k] + '/' + all_json_files[k]
        all_image_paths.append(image_path)
        all_json_paths.append(json_path)
    data_df = pd.DataFrame({'image' : all_image_paths, 'json' : all_json_paths})
    return data_df
```

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

	image	json
0	data/images/201/frame0029_leftImg8bit.jpg	data/mask/201/frame0029_gtFine_polygons.json
	mg8bit.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

Saving... ×

> 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

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

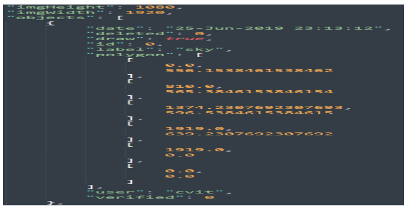
```
grader_1(data_df)
```

```
  True
```

```
data_df.shape
```

```
(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

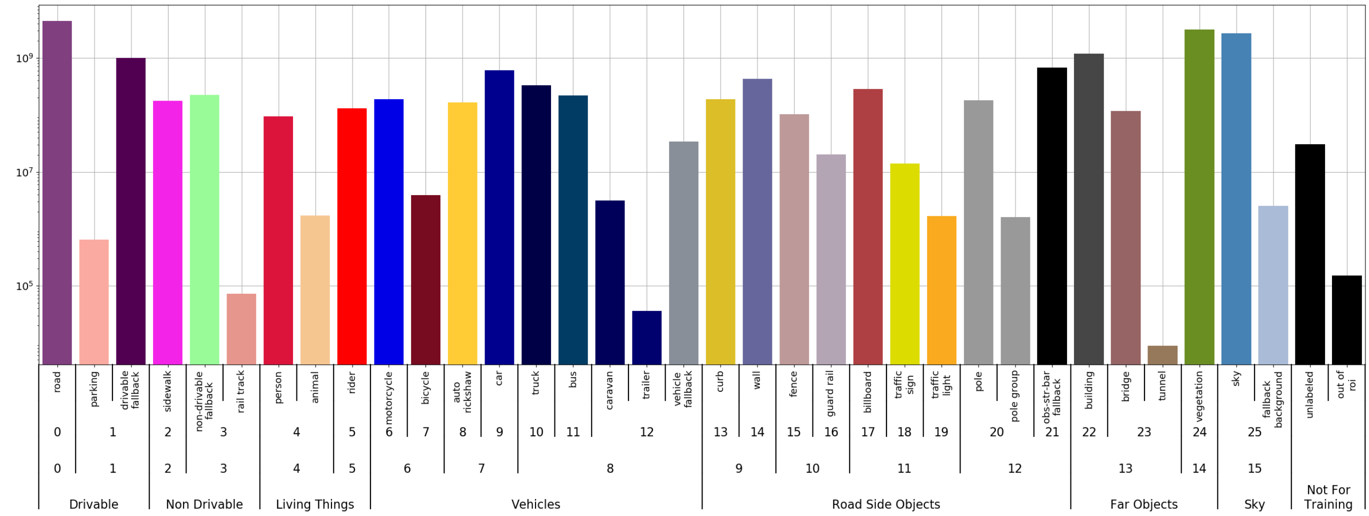
Saving... ×

there in the json file. to see how to get the object from the json file please check this blog

```
def return_unique_labels(data_df):
    labels = []
    for i in tqdm(data_df['json']):
        f0 = open(i, 'r')
        f1 = json.load(f0)
        for j in f1['objects']:
            label = j.get('label', -1)
            labels.append(label)
        f0.close()
    return set(labels)
```

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

```
100%|██████████| 4008/4008 [00:36<00:00, 108.80it/s]
```



```
label_clr = {'road':10, 'parking':20, 'drivable fallback':20,'sidewalk':30,'non-drivable f
                        'person':50, 'animal':50, 'rider':60, 'motorcycle':70, 'bicycle':7
                        'car':80, 'truck':90, 'bus':90, 'vehicle fallback':90, 'trailer':9
                        'curb':100, 'wall':100, 'fence':110,'guard rail':110, 'billboard':
                        'traffic light':120, 'pole':130, 'polegroup':130, 'obs-str-bar-fal
                        'bridge':140,'tunnel':140, 'vegetation':150, 'sky':160, 'fallback
                        'out of roi':0, 'ego vehicle':170, 'ground':180,'rectification bor
                   'train':200}
```

```
class_values = sorted(list(set(label_clr.values())))
print('Class labels', class_values)
class_values = [int(x / 10 )for x in class_values]
```

Saving...

```
',len(set(label_clr.values())))
```

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

```
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 diff
 * Note that we have multiplies each object's number with 10, that is just to make differ
 * Before you pass it to the models, you might need to devide the image array /10.
```

3. Extracting the polygons from the json files

```
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
```

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

```
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(

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

Saving... ×

4. Creating Image segmentations by drawing set of polygons

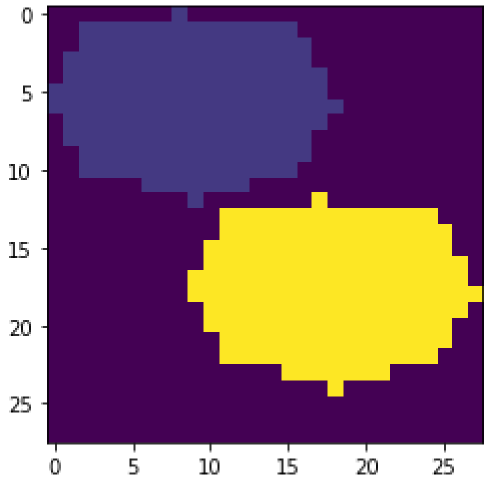
Example

```
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
x2 = [ ((math.cos(th) + 2) *9, (math.sin(th) + 3) *6) for th in [i * (2 * math.pi) / 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
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 0x7f348e63c990>
After <PIL.ImageDraw.ImageDraw object at 0x7f348e63c990>
(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]
                                6 6 6 6 6 6 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 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]]
```

Saving... ×



```
#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
```

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

```
100%|██████████| 4008/4008 [04:37<00:00, 14.44it/s]
```

```
data_df.head(5)
```

Saving... ✕

	image	json	
	mg8bit.jpg	data/mask/201/frame0029_gtFine_polygons.json	dat
1	data/images/201/frame0299_leftImg8bit.jpg	data/mask/201/frame0299_gtFine_polygons.json	dat
2	data/images/201/frame0779_leftImg8bit.jpg	data/mask/201/frame0779_gtFine_polygons.json	dat
3	data/images/201/frame1019_leftImg8bit.jpg	data/mask/201/frame1019_gtFine_polygons.json	dat
4	data/images/201/frame1469_leftImg8bit.jpg	data/mask/201/frame1469_gtFine_polygons.json	dat

```
data_df.to_csv('Preprocessing_2.csv',index=False)
```

```
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 isinstanc

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

```
True
```

```
image_meta_data = {}
for i in tqdm(data_df['json']):
    w, h, labels, vertexlist = get_poly(i)
    w, h, labels, vertexlist = get_poly(i)
    image_meta_data[i] = [w, h, labels, vertexlist]

     100%|██████████| 4008/4008 [01:02<00:00, 64.46it/s]
```

```
output_folders = data_df['json'].apply(lambda x : '/'.join(x.split('/')[:3]).replace('mask
for i in output_folders:
    os.makedirs(i, exist_ok = True)
```

Task 2: Applying Unet to segment the images

Channels Last

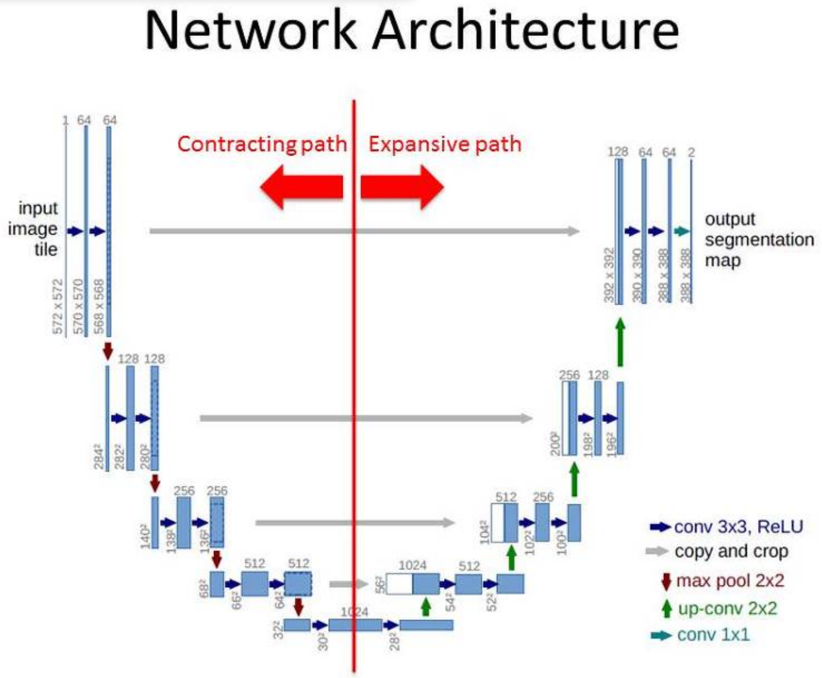
```
. Image data is represented in a three-dimensional array where the last channel represen
```

Channels First

```
Image data is represented in a three-dimensional array where the first channel represent
```

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

Saving...



```
*
```

```
* As a part of this assignment we won't writingt this whole architecture, rather we will
* 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 goog

* Check the reference notebook in which we have solved one end to end case study of imag

* The number of channels in the output will depend on the number of classes in your data

* This is where we want you to explore, how do you featurize your created segmentation m

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

```
!pip install tensorflow==2.2.0
```

```
Collecting tensorflow==2.2.0
  Downloading https://files.pythonhosted.org/packages/3d/be/679ce5254a8c8d07470efb4a4
     |████████████████████████████████| 516.2MB 32kB/s
Requirement already satisfied: google-pasta>=0.1.8 in /usr/local/lib/python3.6/dist-p
Requirement already satisfied: h5py<2.11.0,>=2.10.0 in /usr/local/lib/python3.6/dist-
Collecting tensorflow-estimator<2.3.0,>=2.2.0
  Downloading https://files.pythonhosted.org/packages/a4/f5/926ae53d6a226ec0fda5208e6
     |████████████████████████████████| 460kB 47.5MB/s
Requirement already satisfied: absl-py>=0.7.0 in /usr/local/lib/python3.6/dist-packag
Requirement already satisfied: gast==0.3.3 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: protobuf>=3.8.0 in /usr/local/lib/python3.6/dist-packa
Saving...                       grpcio>=1.8.6 in /usr/local/lib/python3.6/dist-package
                                termcolor>=1.1.0 in /usr/local/lib/python3.6/dist-pack
                                keras-preprocessing>=1.1.0 in /usr/local/lib/python3.6
Requirement already satisfied: scipy==1.4.1; python_version >= "3" in /usr/local/lib/
Collecting tensorboard<2.3.0,>=2.2.0
  Downloading https://files.pythonhosted.org/packages/1d/74/0a6fcb206dcc72a6da9a62dd8
     |████████████████████████████████| 3.0MB 50.8MB/s
Requirement already satisfied: astunparse==1.6.3 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: wrapt>=1.11.1 in /usr/local/lib/python3.6/dist-package
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: numpy<2.0,>=1.16.0 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: wheel>=0.26; python_version >= "3" in /usr/local/lib/p
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.6/dist-p
Requirement already satisfied: google-auth<2,>=1.6.3 in /usr/local/lib/python3.6/dist
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/pyt
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python
Requirement already satisfied: importlib-metadata; python_version < "3.8" in /usr/loc
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-pac
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.6/dist
Requirement already satisfied: rsa<5,>=3.1.4; python_version >= "3" in /usr/local/lib
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.6/dis
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.6/c
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.6/dist-packages (f
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.6/dist-
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.6/dist-packa
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
```

```
!pip install keras==2.3.1
```

```
Collecting keras==2.3.1
  Downloading https://files.pythonhosted.org/packages/ad/fd/6bfe87920d7f4fd475acd2856
     |████████████████████████████████| 378kB 10.3MB/s
Collecting keras-applications>=1.0.6
  Downloading https://files.pythonhosted.org/packages/71/e3/19762fdfc62877ae9102edf63
     |████████████████████████████████| 51kB 7.5MB/s
Requirement already satisfied: scipy>=0.14 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: h5py in /usr/local/lib/python3.6/dist-packages (from k
Requirement already satisfied: pyyaml in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: keras-preprocessing>=1.0.5 in /usr/local/lib/python3.6
                               keras-applications, keras
                               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
```

Saving...

```
!pip install -U segmentation-models==0.2.1
```

```
Collecting segmentation-models==0.2.1
  Downloading https://files.pythonhosted.org/packages/10/bf/253c8834014a834cacf2384c7
     |████████████████████████████████| 51kB 5.5MB/s
Collecting image-classifiers==0.2.0
  Downloading https://files.pythonhosted.org/packages/de/32/a1e74e03f74506d1e4b46bb27
     |████████████████████████████████| 81kB 7.5MB/s
Requirement already satisfied, skipping upgrade: scikit-image in /usr/local/lib/pytho
Requirement already satisfied, skipping upgrade: keras>=2.2.0 in /usr/local/lib/pytho
Requirement already satisfied, skipping upgrade: keras-applications>=1.0.7 in /usr/lo
Requirement already satisfied, skipping upgrade: scipy>=0.19.0 in /usr/local/lib/pyth
Requirement already satisfied, skipping upgrade: networkx>=2.0 in /usr/local/lib/pyth
Requirement already satisfied, skipping upgrade: pillow>=4.3.0 in /usr/local/lib/pyth
Requirement already satisfied, skipping upgrade: PyWavelets>=0.4.0 in /usr/local/lib/
Requirement already satisfied, skipping upgrade: matplotlib!=3.0.0,>=2.0.0 in /usr/lo
Requirement already satisfied, skipping upgrade: imageio>=2.3.0 in /usr/local/lib/pyt
Requirement already satisfied, skipping upgrade: h5py in /usr/local/lib/python3.6/dis
Requirement already satisfied, skipping upgrade: keras-preprocessing>=1.0.5 in /usr/l
Requirement already satisfied, skipping upgrade: pyyaml in /usr/local/lib/python3.6/d
Requirement already satisfied, skipping upgrade: six>=1.9.0 in /usr/local/lib/python3
Requirement already satisfied, skipping upgrade: numpy>=1.9.1 in /usr/local/lib/pytho
Requirement already satisfied, skipping upgrade: decorator>=4.3.0 in /usr/local/lib/p
Requirement already satisfied, skipping upgrade: cycler>=0.10 in /usr/local/lib/pytho
Requirement already satisfied, skipping upgrade: python-dateutil>=2.1 in /usr/local/l
Requirement already satisfied, skipping upgrade: kiwisolver>=1.0.1 in /usr/local/lib/
Requirement already satisfied, skipping upgrade: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2
Installing collected packages: image-classifiers, segmentation-models
Successfully installed image-classifiers-0.2.0 segmentation-models-0.2.1
```

```
# 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,Flatte
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, LearningRateSchedul

from tensorflow.keras.models import Model
```

Saving...

```
# here dir_path is the route directory where all the images and segmentation maps are ther
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]
```

```
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
```

```
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)
```

```
X_train[:5]
```

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

```
# 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: U
       warnings.warn('Current ResNext models are deprecated, '
```

Saving... ×

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

```
# loading the unet model and using the resnet 34 and initilized weights with imagenet weig
# "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-
## Have to clear the session. If you are not clearing, Graph will create again and again a
## 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',enco
     Downloading data from https://github.com/qubvel/classification_models/releases/downlc
     85524480/85521592 [==============================] - 1s 0us/step
```

```
model.summary()
```

```
stage2_unit1_conv2 (Conv2D)     (None, 28, 28, 128)  147456      zero_padding2d_10

stage2_unit1_sc (Conv2D)        (None, 28, 28, 128)  8192        stage2_unit1_relu

add_4 (Add)                     (None, 28, 28, 128)  0           stage2_unit1_conv
                                                                 stage2_unit1_sc[0

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

stage2_unit2_relu1 (Activation) (None, 28, 28, 128)  0           stage2_unit2_bn1[

zero_padding2d_11 (ZeroPadding2 (None, 30, 30, 128)  0           stage2_unit2_relu

stage2_unit2_conv1 (Conv2D)     (None, 28, 28, 128)  147456      zero_padding2d_11

stage2_unit2_bn2 (BatchNormaliz (None, 28, 28, 128)  512         stage2_unit2_conv

stage2_unit2_relu2 (Activation) (None, 28, 28, 128)  0           stage2_unit2_bn2[

zero_padding2d_12 (ZeroPadding2 (None, 30, 30, 128)  0           stage2_unit2_relu

stage2_unit2_conv2 (Conv2D)     (None, 28, 28, 128)  147456      zero_padding2d_12

add_5 (Add)                     (None, 28, 28, 128)  0           stage2_unit2_conv
                                                                 add_4[0][0]

                                (None, 28, 28, 128)  512         add_5[0][0]

stage2_unit3_relu1 (Activation) (None, 28, 28, 128)  0           stage2_unit3_bn1[

zero_padding2d_13 (ZeroPadding2 (None, 30, 30, 128)  0           stage2_unit3_relu

stage2_unit3_conv1 (Conv2D)     (None, 28, 28, 128)  147456      zero_padding2d_13

stage2_unit3_bn2 (BatchNormaliz (None, 28, 28, 128)  512         stage2_unit3_conv

stage2_unit3_relu2 (Activation) (None, 28, 28, 128)  0           stage2_unit3_bn2[

zero_padding2d_14 (ZeroPadding2 (None, 30, 30, 128)  0           stage2_unit3_relu

stage2_unit3_conv2 (Conv2D)     (None, 28, 28, 128)  147456      zero_padding2d_14

add_6 (Add)                     (None, 28, 28, 128)  0           stage2_unit3_conv
                                                                 add_5[0][0]

stage2_unit4_bn1 (BatchNormaliz (None, 28, 28, 128)  512         add_6[0][0]

stage2_unit4_relu1 (Activation) (None, 28, 28, 128)  0           stage2_unit4_bn1[

zero_padding2d_15 (ZeroPadding2 (None, 30, 30, 128)  0           stage2_unit4_relu

stage2_unit4_conv1 (Conv2D)     (None, 28, 28, 128)  147456      zero_padding2d_15
```

Saving...

```
stage2_unit4_bn2 (BatchNormaliz (None, 28, 28, 128)  512         stage2_unit4_conv
________________________________________________________________________________
stage2_unit4_relu2 (Activation) (None, 28, 28, 128)  0           stage2_unit4_bn2[
________________________________________________________________________________
```

```
# 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))


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
```

Saving...

```
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 ma
    # 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 ima
        # the paths of segmentation images
        self.masks_fps    = [os.path.join(images_dir_mask, image_id+"_gtFine_polygons.png"
        # 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(image)
                                    ugment_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
```

Saving... ×

```
        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)
```

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

Saving...

```
#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)
```

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, LearningRateSchedul
```

```
# 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') >=
        print("\nReached %2.2f%% accuracy, so stopping training!!" %(ACCURACY_THRESHOLD_te
        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",
```

Saving... ×

```
    min_lr=0
)

filepath="/gdrive/My Drive/Image_Segmentation/segmentation/Model_save/better_model_updated
checkpoint = ModelCheckpoint(filepath=filepath, monitor='val_iou_score',  verbose=1, save_
```

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

```
# 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') >=
        print("\nReached %2.2f%% accuracy, so stopping training!!" %(ACCURACY_THRESHOLD_te
        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_

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
)
```

Saving... ×

```
                                         mentation/segmentation/Model_save/better_model_updated
checkpoint = keras.callbacks.ModelCheckpoint(filepath=filepath, monitor='val_iou_score',
```

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

```
# 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
```

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

```
focal_loss = sm.losses.cce_dice_loss

# actually total_loss can be imported directly from library, above example just show you
# 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])



#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,tensorb
```

```
Epoch 00026: val_iou_score did not improve from 0.41493
Epoch 27/150
100/100 [==============================] - 272s 3s/step - loss: 1.0215 - iou_score

Epoch 00027: val_iou_score improved from 0.41493 to 0.41548, saving model to /gdri
Epoch 28/150
100/100 [==============================] - 273s 3s/step - loss: 1.0238 - iou_score

Epoch 00028: val_iou_score did not improve from 0.41548
Epoch 29/150
100/100 [==============================] - 274s 3s/step - loss: 1.0248 - iou_score

Epoch 00029: val_iou_score did not improve from 0.41548
Epoch 30/150
100/100 [==============================] - 275s 3s/step - loss: 1.0277 - iou_score

Epoch 00030: val_iou_score improved from 0.41548 to 0.41641, saving model to /gdri
Saving...
=========] - 277s 3s/step - loss: 1.0289 - iou_score

Epoch 00031: val_iou_score did not improve from 0.41641
Epoch 32/150
100/100 [==============================] - 275s 3s/step - loss: 1.0258 - iou_score

Epoch 00032: val_iou_score did not improve from 0.41641
Epoch 33/150
100/100 [==============================] - 273s 3s/step - loss: 1.0240 - iou_score

Epoch 00033: val_iou_score did not improve from 0.41641
Epoch 34/150
100/100 [==============================] - 272s 3s/step - loss: 1.0326 - iou_score

Epoch 00034: val_iou_score did not improve from 0.41641
Epoch 35/150
100/100 [==============================] - 276s 3s/step - loss: 1.0203 - iou_score

Epoch 00035: val_iou_score did not improve from 0.41641
Epoch 36/150
100/100 [==============================] - 274s 3s/step - loss: 1.0278 - iou_score

Epoch 00036: val_iou_score did not improve from 0.41641
Epoch 37/150
100/100 [==============================] - 272s 3s/step - loss: 1.0277 - iou_score
```

```
Epoch 00037: val_iou_score did not improve from 0.41641
Epoch 38/150
100/100 [==============================] - 271s 3s/step - loss: 1.0295 - iou_score

Epoch 00038: val_iou_score did not improve from 0.41641
Epoch 39/150
100/100 [==============================] - 272s 3s/step - loss: 1.0257 - iou_score

Epoch 00039: val_iou_score did not improve from 0.41641
Epoch 40/150
100/100 [==============================] - 271s 3s/step - loss: 1.0225 - iou_score

Epoch 00040: val_iou_score did not improve from 0.41641
```

```
#reconstruction 1 - Above training stopped unfortunately, so using best model weight to co
import keras
model = keras.models.load_model("/gdrive/My Drive/Image_Segmentation/segmentation/Model_sa
history = model.fit_generator(train_dataloader, epochs=150,
                              validation_data=test_dataloader ,
                              callbacks = [early_stop_iou_scores,checkpoint,red_lr,tensorb
```

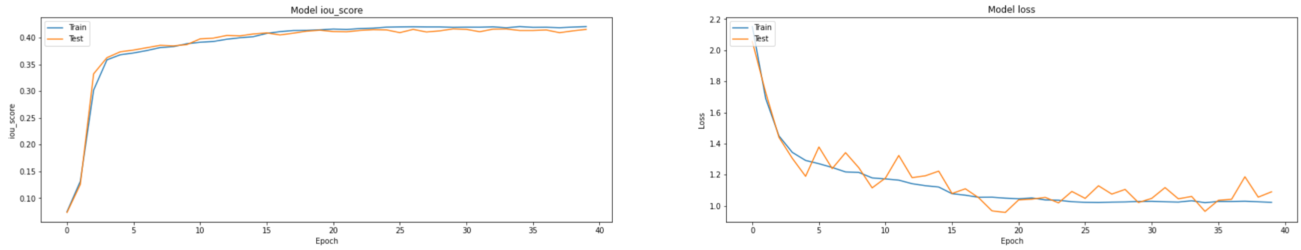
```
Epoch 1/150
  2/100 [..............................] - ETA: 21:41 - loss: 0.8241 - iou_score:
/usr/local/lib/python3.6/dist-packages/keras/callbacks/callbacks.py:95: RuntimeWar
  % (hook_name, delta_t_median), RuntimeWarning)
100/100 [==============================] - 332s 3s/step - loss: 0.8784 - iou_score

Epoch 00001: val_iou_score improved from 0.42296 to 0.46905, saving model to /gdri
Epoch 2/150
Saving...                                 =========] - 294s 3s/step - loss: 0.8825 - iou_score

                                         roved from 0.46905 to 0.47231, saving model to /gdri
Epoch 3/150
100/100 [==============================] - 292s 3s/step - loss: 0.8722 - iou_score

Epoch 00003: val_iou_score did not improve from 0.47231
Epoch 4/150
100/100 [==============================] - 291s 3s/step - loss: 0.8786 - iou_score

Epoch 00004: val_iou_score did not improve from 0.47231
Epoch 5/150
100/100 [==============================] - 289s 3s/step - loss: 0.8679 - iou_score

Epoch 00005: val_iou_score did not improve from 0.47231
Epoch 6/150
100/100 [==============================] - 288s 3s/step - loss: 0.8746 - iou_score

Epoch 00006: val_iou_score did not improve from 0.47231
Epoch 7/150
100/100 [==============================] - 289s 3s/step - loss: 0.8801 - iou_score

Epoch 00007: val_iou_score did not improve from 0.47231
Epoch 8/150
100/100 [==============================] - 289s 3s/step - loss: 0.8698 - iou_score

Epoch 00008: val_iou_score did not improve from 0.47231
Epoch 9/150
100/100 [==============================] - 289s 3s/step - loss: 0.8584 - iou_score

Epoch 00009: val_iou_score did not improve from 0.47231
Epoch 10/150
100/100 [==============================] - 286s 3s/step - loss: 0.8609 - iou_score

Epoch 00010: val_iou_score did not improve from 0.47231
Epoch 11/150
100/100 [==============================] - 285s 3s/step - loss: 0.8672 - iou_score

Epoch 00011: val_iou_score did not improve from 0.47231
Epoch 12/150
100/100 [==============================] - 283s 3s/step - loss: 0.8687 - iou_score

Epoch 00012: val_iou_score did not improve from 0.47231
Epoch 13/150
100/100 [==============================] - 284s 3s/step - loss: 0.8653 - iou_score

Epoch 00013: val_iou_score did not improve from 0.47231
```

```
#reconstruction 2 -  Above training stopped due to exceed RAM usage in colab, so using bes

import keras
model = keras.models.load_model("/gdrive/My Drive/Image_Segmentation/segmentation/Model_sa
history = model.fit_generator(train_dataloader, epochs=150,
                              validation_data=test_dataloader ,
                              callbacks = [early_stop_iou_scores,checkpoint,red_lr,tensorb
```

```
Epoch 1/150
Saving...
========] - 2844s 25s/step - loss: 0.7917 - iou_score

Epoch 00001: val_iou_score improved from -inf to 0.50034, saving model to /gdrive/My

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

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


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


# 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()
```



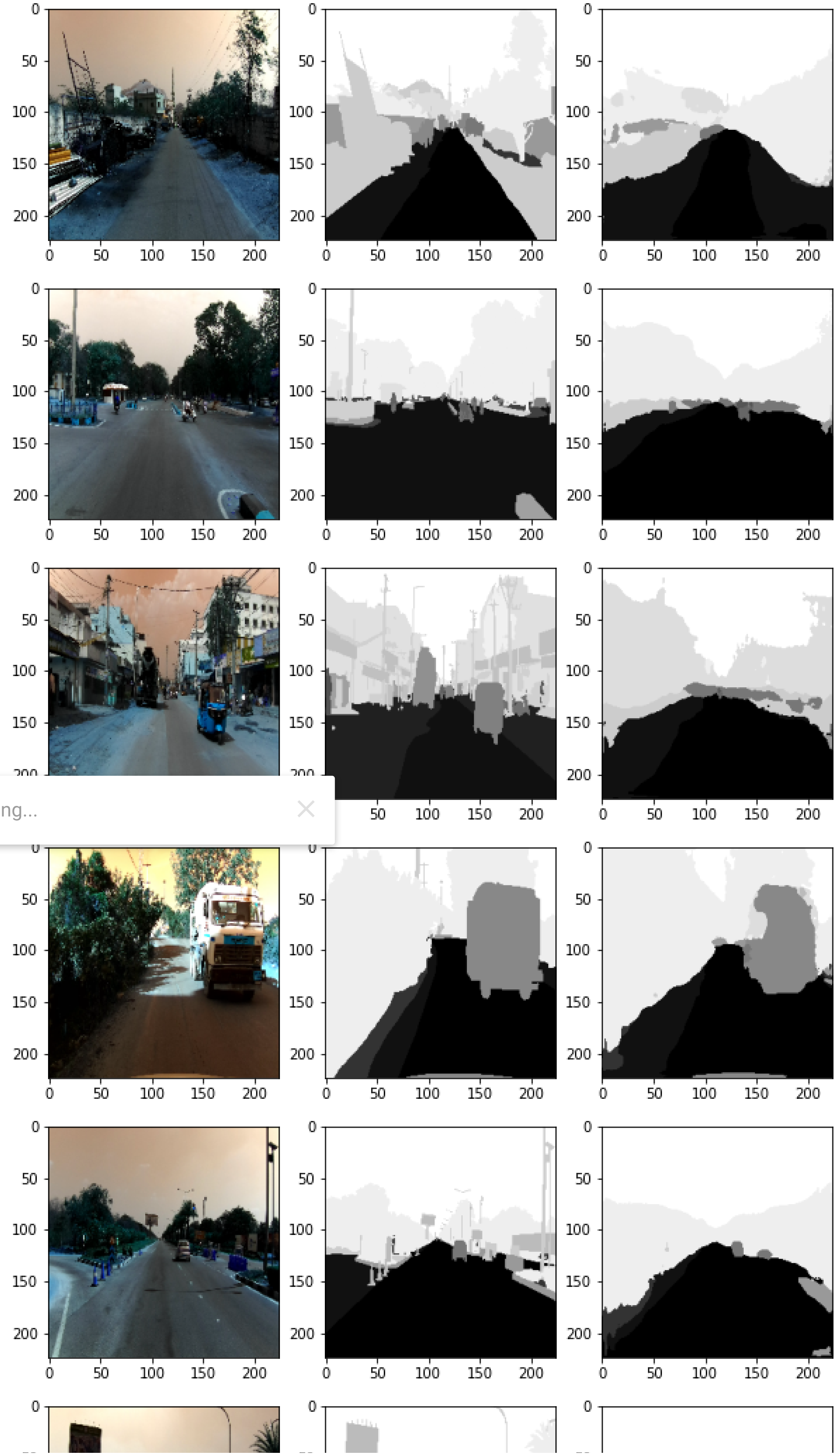
```
    for p, i in enumerate(X_test):
```

Saving...

```
        #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.I
        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
```

Saving... ×



Saving...