Instructions

- Some parts of the code are already done for you
- You need to execute all the cells
- You need to add the code where ever you see `"#### Add your code here ####"`
- Marks are mentioned along with the cells

Face detection

Task is to predict the boundaries(mask) around the face in a given image.

Dataset

Faces in images marked with bounding boxes. Have around 500 images with around 1100 faces n

Mount Google drive if you are using google colab

- We recommend using Google Colab as you can face memory issues and longer runtimes wh

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=9473

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

Change current working directory to project folder (1 mark)

```
import os
PATH = "/content/drive/My Drive/Colab Notebooks/"
os.chdir(PATH)
```

Load the "images.npy" file (2 marks)

- This file contains images with details of bounding boxes

```
# IMPORT LIBRARIES AND PACKAGES

import csv
import math
```

```
import math
import cv2

import numpy as np
import tensorflow as tf
from PIL import Image
from tensorflow.keras import Model
from tensorflow.keras.applications.mobilenet import MobileNet, preprocess_input
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.layers import Concatenate, Conv2D, UpSampling2D, Reshape
from tensorflow.keras.utils import Sequence
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import binary_crossentropy
data = np.load('./images_small.npy',allow_pickle=True)
```

Check one sample from the loaded "images.npy" file (2 marks)

Hint - print data[10][1]

```
for key in data:
    print (key)
    print (data[10][1])
    break
```

```
[array([[[42, 37, 34],
         [56, 51, 48],
         [71, 66, 63],
         ...,
         [23, 33, 34],
         [26, 36, 37],
         [28, 38, 39]],

        [[40, 35, 32],
         [51, 46, 43],
         [64, 59, 56],
         ...,
         [27, 36, 35],
         [24, 33, 32],
         [26, 35, 34]],

        [[43, 38, 35],
         [51, 46, 43],
         [61, 56, 53],
         ...,
         [28, 30, 27],
         [33, 35, 32],
         [35, 37, 34]],

        ...,

        [[56, 47, 40],
         [57, 48, 41],
         [61, 52, 45],
         ...,
         [67, 48, 42],
         [55, 35, 28],
         [60, 40, 33]],

        [[53, 44, 37],
         [54, 45, 38],
         [57, 48, 41],
         ...,
         [59, 40, 34],
         [60, 40, 33],
         [54, 34, 27]],

        [[53, 44, 37],
         [54, 45, 38],
         [57, 48, 41],
         ...,
         [59, 40, 34],
         [70, 50, 43],
         [64, 44, 37]]], dtype=uint8)
 list([{'label': ['Face'], 'notes': '', 'points': [{'x': 0.08615384615384615, 'y': 0.
[{'label': ['Face'], 'notes': '', 'points': [{'x': 0.48, 'y': 0.10385756676557864}, {
```

Set image dimensions (1 mark)

- Initialize image height, image width with value: 224

```
# SETTINGS

ALPHA = 1 # Width hyper parameter for MobileNet (0.25, 0.5, 0.75, 1.0). Higher width means

IMAGE_HEIGHT = 224
IMAGE_WIDTH = 224

HEIGHT_CELLS = 28
WIDTH_CELLS = 28

CELL_WIDTH = IMAGE_WIDTH / WIDTH_CELLS
CELL_HEIGHT = IMAGE_HEIGHT / HEIGHT_CELLS

EPOCHS = 1
BATCH_SIZE = 4
PATIENCE = 10

THREADS = 1

PATH = "/content/drive/My Drive/Colab Notebooks/"

TRAIN_CSV = PATH+"train.csv"
VALIDATION_CSV = PATH+"validation.csv"
```

Create features and labels

- Here feature is the image
- The label is the mask
- Images will be stored in "X_train" array
- Masks will be stored in "masks" array

```
import cv2
from tensorflow.keras.applications.mobilenet import preprocess_input

masks = np.zeros((int(data.shape[0]), IMAGE_HEIGHT, IMAGE_WIDTH))
X_train = np.zeros((int(data.shape[0]), IMAGE_HEIGHT, IMAGE_WIDTH, 3))
for index in range(data.shape[0]):
    img = data[index][0]
    img = cv2.resize(img, dsize=(IMAGE_HEIGHT, IMAGE_WIDTH), interpolation=cv2.INTER_CUBIC
    try:
      img = img[:, :, :3]
    except:
      continue
    X_train[index] = preprocess_input(np.array(img, dtype=np.float32))
    for i in data[index][1]:
        x1 = int(i["points"][0]['x'] * IMAGE_WIDTH)
        x2 = int(i["points"][1]['x'] * IMAGE_WIDTH)
        y1 = int(i["points"][0]['y'] * IMAGE_HEIGHT)
        y2 = int(i["points"][1]['y'] * IMAGE_HEIGHT)
```

```
        masks[index][y1:y2, x1:x2] = 1
```

Print the shape of X_train and mask array (1 mark)

```
X_train.shape
```

```
(200, 224, 224, 3)
```

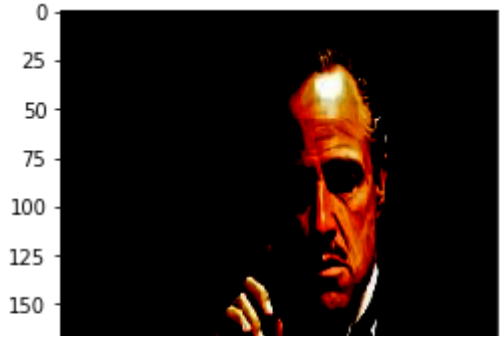
```
masks.shape
```

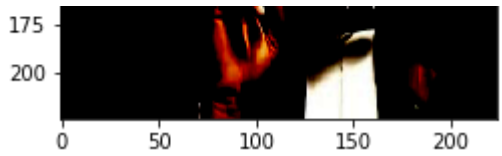
```
(200, 224, 224)
```

Print a sample image and image array

```
from matplotlib import pyplot
n = 10
print(X_train[n])
pyplot.imshow(X_train[n])
```

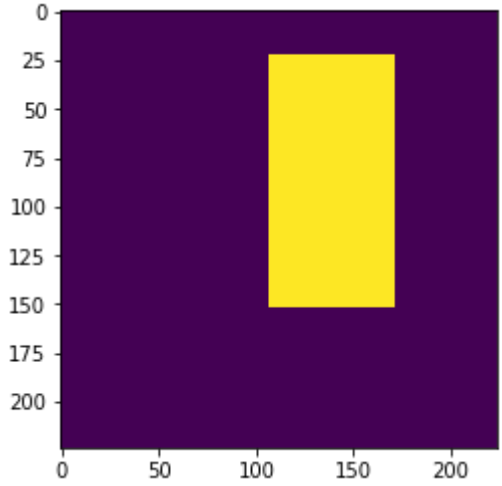
```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
[[[-0.98431373 -0.98431373 -0.98431373]
  [-0.98431373 -0.98431373 -0.98431373]
  [-0.98431373 -0.98431373 -0.98431373]
  ...
  [-1.         -1.         -1.        ]
  [-1.         -1.         -1.        ]
  [-1.         -1.         -1.        ]]

 [[-0.98431373 -0.98431373 -0.98431373]
  [-0.98431373 -0.98431373 -0.98431373]
  [-0.98431373 -0.98431373 -0.98431373]
  ...
  [-1.         -1.         -1.        ]
  [-1.         -1.         -1.        ]
  [-1.         -1.         -1.        ]]

 [[-0.98431373 -0.98431373 -0.98431373]
  [-0.98431373 -0.98431373 -0.98431373]
  [-0.98431373 -0.98431373 -0.98431373]
  ...
  [-1.         -1.         -1.        ]
  [-1.         -1.         -1.        ]
  [-1.         -1.         -1.        ]]

 ...

 [[-1.         -1.         -1.        ]
  [-1.         -1.         -1.        ]
  [-1.         -1.         -1.        ]
  ...
  [-0.96862745 -0.96862745 -0.96862745]
  [-0.96078432 -0.96078432 -0.96078432]
  [-0.96078432 -0.96078432 -0.96078432]]

 [[-1.         -1.         -1.        ]
  [-1.         -1.         -1.        ]
  [-1.         -1.         -1.        ]
  ...
  [-0.96862745 -0.96862745 -0.96862745]
  [-0.96078432 -0.96078432 -0.96078432]
  [-0.95294118 -0.95294118 -0.95294118]]

 [[-1.         -1.         -1.        ]
  [-1.         -1.         -1.        ]
  [-1.         -1.         -1.        ]
  ...
  [-0.97647059 -0.97647059 -0.97647059]
  [-0.96862745 -0.96862745 -0.96862745]
  [-0.96078432 -0.96078432 -0.96078432]]]
<matplotlib.image.AxesImage at 0x7feb01c7d908>
```






```
pyplot.imshow(masks[n])
```

```
<matplotlib.image.AxesImage at 0x7feb01748898>
```



Create the model (10 marks)

- Add MobileNet as model with below parameter values
  - input_shape: IMAGE_HEIGHT, IMAGE_WIDTH, 3
  - include_top: False
  - alpha: 1.0
  - weights: "imagenet"
- Add UNET architecture layers
  - This is the trickiest part of the project, you need to research and implement it correctly

```
from tensorflow.keras.applications.mobilenet import MobileNet
from tensorflow.keras.layers import Concatenate, UpSampling2D, Conv2D, Reshape
from tensorflow.keras.models import Model


def create_model(trainable=True):
    model = #### Add your code here ####
    for layer in model.layers:
        layer.trainable = trainable

    # Add all the UNET layers here
    #### Add your code here ####
```

```
    return #### Add your code here ####


class DataGenerator(Sequence):

    def __init__(self, csv_file):
        self.paths = []

        with open(csv_file, "r") as file:
            self.mask = np.zeros((sum(1 for line in file), HEIGHT_CELLS, WIDTH_CELLS))
            file.seek(0)

            reader = csv.reader(file, delimiter=",")

            for index, row in enumerate(reader):
                for i, r in enumerate(row[1:7]):
                    row[i+1] = int(r)

                path, image_height, image_width, x0, y0, x1, y1, _, _ = row

                path_mask = path.replace('.jpg','.png')

                mask_img = cv2.imread(path_mask)
                mask_img = (mask_img!=2)*1.0
                mask_img = cv2.resize(mask_img, (28, 28))
                mask_img = 1.0*(mask_img[:,:,0]>0.2)
                self.mask[index,:,:] = np.squeeze(mask_img)

                self.paths.append(path)

    def __len__(self):
        return math.ceil(len(self.mask) / BATCH_SIZE)

    def __getitem__(self, idx):
        batch_paths = self.paths[idx * BATCH_SIZE:(idx + 1) * BATCH_SIZE]
        batch_masks = self.mask[idx * BATCH_SIZE:(idx + 1) * BATCH_SIZE]

        batch_images = np.zeros((len(batch_paths), IMAGE_HEIGHT, IMAGE_WIDTH, 3), dtype=np
        for i, f in enumerate(batch_paths):
            img = Image.open(f)
            img = img.resize((IMAGE_WIDTH, IMAGE_HEIGHT))
            img = img.convert('RGB')

            batch_images[i] = preprocess_input(np.array(img, dtype=np.float32))
            img.close()

        return batch_images, batch_masks
```

Call the create_model function

```
# Give trainable=False as argument, if you want to freeze lower layers for fast training (
model = create_model()
```

```
# Print summary
model.summary()


def create_model(trainable=True):
    model = MobileNet(input_shape=(IMAGE_HEIGHT, IMAGE_WIDTH, 3), include_top=False, alpha

    for layer in model.layers:
        layer.trainable = trainable

    block1 = model.get_layer("conv_pw_5_relu").output
    block2 = model.get_layer("conv_pw_11_relu").output
    block3 = model.get_layer("conv_pw_13_relu").output

    x = Concatenate()([UpSampling2D()(block3), block2])
    x = Concatenate()([UpSampling2D()(x), block1])

    x = Conv2D(1, kernel_size=1, activation="sigmoid")(x)
    x = Reshape((HEIGHT_CELLS, WIDTH_CELLS))(x)

    return Model(inputs=model.input, outputs=x)
```

Define dice coefficient function (5 marks)

- Create a function to calculate dice coefficient

```
def dice_coefficient(y_true, y_pred):
    numerator = 2 * tf.reduce_sum(y_true * y_pred)
    denominator = tf.reduce_sum(y_true + y_pred)

    return numerator / (denominator + tf.keras.backend.epsilon())
```

Define loss

```
from tensorflow.keras.losses import binary_crossentropy
from tensorflow.keras.backend import log, epsilon
def loss(y_true, y_pred):
    return binary_crossentropy(y_true, y_pred) - log(dice_coefficient(y_true, y_pred) + ep


def loss(y_true, y_pred):
    return binary_crossentropy(y_true, y_pred) - tf.log(dice_coefficient(y_true, y_pred) +
```

Compile the model (2 marks)

- Complie the model using below parameters
    - loss: use the loss function defined above
    - optimizers: use Adam optimizer

- metrics: use dice_coefficient function defined above

```
model = create_model(False)
model.summary()

train_datagen = DataGenerator(TRAIN_CSV)
validation_datagen = DataGenerator(VALIDATION_CSV)

optimizer = Adam(lr=1e-4, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False
model.compile(loss=loss, optimizer=optimizer, metrics=[dice_coefficient])

checkpoint = ModelCheckpoint("model-{val_loss:.2f}.h5", monitor="val_loss", verbose=1, sav
                             save_weights_only=True, mode="auto", period=1)
stop = EarlyStopping(monitor="val_loss", patience=PATIENCE, mode="auto")
reduce_lr = ReduceLROnPlateau(monitor="val_loss", factor=0.2, patience=5, min_lr=1e-6, ver
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow_core/python
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
Downloading data from https://github.com/fchollet/deep-learning-models/releases/downl
17227776/17225924 [==============================] - 1s 0us/step
Model: "model"
```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)	0	
conv1_pad (ZeroPadding2D)	(None, 225, 225, 3)	0	input_1[0][0]
conv1 (Conv2D)	(None, 112, 112, 32)	864	conv1_pad[0][0]
conv1_bn (BatchNormalization)	(None, 112, 112, 32)	128	conv1[0][0]
conv1_relu (ReLU)	(None, 112, 112, 32)	0	conv1_bn[0][0]
conv_dw_1 (DepthwiseConv2D)	(None, 112, 112, 32)	288	conv1_relu[0][0]
conv_dw_1_bn (BatchNormalizatio	(None, 112, 112, 32)	128	conv_dw_1[0][0]
conv_dw_1_relu (ReLU)	(None, 112, 112, 32)	0	conv_dw_1_bn[0][0]
conv_pw_1 (Conv2D)	(None, 112, 112, 64)	2048	conv_dw_1_relu[0][0]
conv_pw_1_bn (BatchNormalizatio	(None, 112, 112, 64)	256	conv_pw_1[0][0]
conv_pw_1_relu (ReLU)	(None, 112, 112, 64)	0	conv_pw_1_bn[0][0]
conv_pad_2 (ZeroPadding2D)	(None, 113, 113, 64)	0	conv_pw_1_relu[0][0]
conv_dw_2 (DepthwiseConv2D)	(None, 56, 56, 64)	576	conv_pad_2[0][0]
conv_dw_2_bn (BatchNormalizatio	(None, 56, 56, 64)	256	conv_dw_2[0][0]
conv_dw_2_relu (ReLU)	(None, 56, 56, 64)	0	conv_dw_2_bn[0][0]
conv_pw_2 (Conv2D)	(None, 56, 56, 128)	8192	conv_dw_2_relu[0][0]
conv_pw_2_bn (BatchNormalizatio	(None, 56, 56, 128)	512	conv_pw_2[0][0]
conv_pw_2_relu (ReLU)	(None, 56, 56, 128)	0	conv_pw_2_bn[0][0]
conv_dw_3 (DepthwiseConv2D)	(None, 56, 56, 128)	1152	conv_pw_2_relu[0][0]
conv_dw_3_bn (BatchNormalizatio	(None, 56, 56, 128)	512	conv_dw_3[0][0]
conv_dw_3_relu (ReLU)	(None, 56, 56, 128)	0	conv_dw_3_bn[0][0]
conv_pw_3 (Conv2D)	(None, 56, 56, 128)	16384	conv_dw_3_relu[0][0]
conv_pw_3_bn (BatchNormalizatio	(None, 56, 56, 128)	512	conv_pw_3[0][0]
conv_pw_3_relu (ReLU)	(None, 56, 56, 128)	0	conv_pw_3_bn[0][0]
conv_pad_4 (ZeroPadding2D)	(None, 57, 57, 128)	0	conv_pw_3_relu[0][0]
conv_dw_4 (DepthwiseConv2D)	(None, 28, 28, 128)	1152	conv_pad_4[0][0]

conv_dw_4_bn (BatchNormalizatio	(None, 28, 28, 128)	512	conv_dw_4[0][0]
conv_dw_4_relu (ReLU)	(None, 28, 28, 128)	0	conv_dw_4_bn[0][0]
conv_pw_4 (Conv2D)	(None, 28, 28, 256)	32768	conv_dw_4_relu[0][0]
conv_pw_4_bn (BatchNormalizatio	(None, 28, 28, 256)	1024	conv_pw_4[0][0]
conv_pw_4_relu (ReLU)	(None, 28, 28, 256)	0	conv_pw_4_bn[0][0]
conv_dw_5 (DepthwiseConv2D)	(None, 28, 28, 256)	2304	conv_pw_4_relu[0][0]
conv_dw_5_bn (BatchNormalizatio	(None, 28, 28, 256)	1024	conv_dw_5[0][0]
conv_dw_5_relu (ReLU)	(None, 28, 28, 256)	0	conv_dw_5_bn[0][0]
conv_pw_5 (Conv2D)	(None, 28, 28, 256)	65536	conv_dw_5_relu[0][0]
conv_pw_5_bn (BatchNormalizatio	(None, 28, 28, 256)	1024	conv_pw_5[0][0]
conv_pw_5_relu (ReLU)	(None, 28, 28, 256)	0	conv_pw_5_bn[0][0]
conv_pad_6 (ZeroPadding2D)	(None, 29, 29, 256)	0	conv_pw_5_relu[0][0]
conv_dw_6 (DepthwiseConv2D)	(None, 14, 14, 256)	2304	conv_pad_6[0][0]
conv_dw_6_bn (BatchNormalizatio	(None, 14, 14, 256)	1024	conv_dw_6[0][0]
conv_dw_6_relu (ReLU)	(None, 14, 14, 256)	0	conv_dw_6_bn[0][0]
conv_pw_6 (Conv2D)	(None, 14, 14, 512)	131072	conv_dw_6_relu[0][0]
conv_pw_6_bn (BatchNormalizatio	(None, 14, 14, 512)	2048	conv_pw_6[0][0]
conv_pw_6_relu (ReLU)	(None, 14, 14, 512)	0	conv_pw_6_bn[0][0]
conv_dw_7 (DepthwiseConv2D)	(None, 14, 14, 512)	4608	conv_pw_6_relu[0][0]
conv_dw_7_bn (BatchNormalizatio	(None, 14, 14, 512)	2048	conv_dw_7[0][0]
conv_dw_7_relu (ReLU)	(None, 14, 14, 512)	0	conv_dw_7_bn[0][0]
conv_pw_7 (Conv2D)	(None, 14, 14, 512)	262144	conv_dw_7_relu[0][0]
conv_pw_7_bn (BatchNormalizatio	(None, 14, 14, 512)	2048	conv_pw_7[0][0]
conv_pw_7_relu (ReLU)	(None, 14, 14, 512)	0	conv_pw_7_bn[0][0]
conv_dw_8 (DepthwiseConv2D)	(None, 14, 14, 512)	4608	conv_pw_7_relu[0][0]
conv_dw_8_bn (BatchNormalizatio	(None, 14, 14, 512)	2048	conv_dw_8[0][0]
conv_dw_8_relu (ReLU)	(None, 14, 14, 512)	0	conv_dw_8_bn[0][0]
conv_pw_8 (Conv2D)	(None, 14, 14, 512)	262144	conv_dw_8_relu[0][0]
conv_pw_8_bn (BatchNormalizatio	(None, 14, 14, 512)	2048	conv_pw_8[0][0]
conv_pw_8_relu (ReLU)	(None, 14, 14, 512)	0	conv_pw_8_bn[0][0]
conv_dw_9 (DepthwiseConv2D)	(None, 14, 14, 512)	4608	conv_pw_8_relu[0][0]

conv_dw_9_bn (BatchNormalizatio	(None, 14, 14, 512)	2048	conv_dw_9[0][0]
conv_dw_9_relu (ReLU)	(None, 14, 14, 512)	0	conv_dw_9_bn[0][0]
conv_pw_9 (Conv2D)	(None, 14, 14, 512)	262144	conv_dw_9_relu[0][0]
conv_pw_9_bn (BatchNormalizatio	(None, 14, 14, 512)	2048	conv_pw_9[0][0]
conv_pw_9_relu (ReLU)	(None, 14, 14, 512)	0	conv_pw_9_bn[0][0]
conv_dw_10 (DepthwiseConv2D)	(None, 14, 14, 512)	4608	conv_pw_9_relu[0][0]
conv_dw_10_bn (BatchNormalizati	(None, 14, 14, 512)	2048	conv_dw_10[0][0]
conv_dw_10_relu (ReLU)	(None, 14, 14, 512)	0	conv_dw_10_bn[0][0]
conv_pw_10 (Conv2D)	(None, 14, 14, 512)	262144	conv_dw_10_relu[0][0
conv_pw_10_bn (BatchNormalizati	(None, 14, 14, 512)	2048	conv_pw_10[0][0]
conv_pw_10_relu (ReLU)	(None, 14, 14, 512)	0	conv_pw_10_bn[0][0]
conv_dw_11 (DepthwiseConv2D)	(None, 14, 14, 512)	4608	conv_pw_10_relu[0][0
conv_dw_11_bn (BatchNormalizati	(None, 14, 14, 512)	2048	conv_dw_11[0][0]
conv_dw_11_relu (ReLU)	(None, 14, 14, 512)	0	conv_dw_11_bn[0][0]
conv_pw_11 (Conv2D)	(None, 14, 14, 512)	262144	conv_dw_11_relu[0][0
conv_pw_11_bn (BatchNormalizati	(None, 14, 14, 512)	2048	conv_pw_11[0][0]
conv_pw_11_relu (ReLU)	(None, 14, 14, 512)	0	conv_pw_11_bn[0][0]
conv_pad_12 (ZeroPadding2D)	(None, 15, 15, 512)	0	conv_pw_11_relu[0][0
conv_dw_12 (DepthwiseConv2D)	(None, 7, 7, 512)	4608	conv_pad_12[0][0]
conv_dw_12_bn (BatchNormalizati	(None, 7, 7, 512)	2048	conv_dw_12[0][0]
conv_dw_12_relu (ReLU)	(None, 7, 7, 512)	0	conv_dw_12_bn[0][0]
conv_pw_12 (Conv2D)	(None, 7, 7, 1024)	524288	conv_dw_12_relu[0][0
conv_pw_12_bn (BatchNormalizati	(None, 7, 7, 1024)	4096	conv_pw_12[0][0]
conv_pw_12_relu (ReLU)	(None, 7, 7, 1024)	0	conv_pw_12_bn[0][0]
conv_dw_13 (DepthwiseConv2D)	(None, 7, 7, 1024)	9216	conv_pw_12_relu[0][0
conv_dw_13_bn (BatchNormalizati	(None, 7, 7, 1024)	4096	conv_dw_13[0][0]
conv_dw_13_relu (ReLU)	(None, 7, 7, 1024)	0	conv_dw_13_bn[0][0]
conv_pw_13 (Conv2D)	(None, 7, 7, 1024)	1048576	conv_dw_13_relu[0][0
conv_pw_13_bn (BatchNormalizati	(None, 7, 7, 1024)	4096	conv_pw_13[0][0]
conv_pw_13_relu (ReLU)	(None, 7, 7, 1024)	0	conv_pw_13_bn[0][0]
up_sampling2d (UpSampling2D)	(None, 14, 14, 1024)	0	conv_pw_13_relu[0][0

```
concatenate (Concatenate)       (None, 14, 14, 1536) 0           up_sampling2d[0][0]
                                                                 conv_pw_11_relu[0][0
__________________________________________________________________________________________
up_sampling2d_1 (UpSampling2D)  (None, 28, 28, 1536) 0           concatenate[0][0]
__________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 28, 28, 1792) 0           up_sampling2d_1[0][0
                                                                 conv_pw_5_relu[0][0]
__________________________________________________________________________________________
conv2d (Conv2D)                 (None, 28, 28, 1)    1793        concatenate_1[0][0]
__________________________________________________________________________________________
reshape (Reshape)               (None, 28, 28)       0           conv2d[0][0]
==========================================================================================
Total params: 3,230,657
Trainable params: 1,793
Non-trainable params: 3,228,864
__________________________________________________________________________________________
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-30-88545abbee4c> in <module>()
      2 model.summary()
      3
----> 4 train_datagen = DataGenerator(TRAIN_CSV)
      5 validation_datagen = DataGenerator(VALIDATION_CSV)
      6

<ipython-input-26-57ac1072ace7> in __init__(self, csv_file)
      4         self.paths = []
      5
----> 6         with open(csv_file, "r") as file:
      7             self.mask = np.zeros((sum(1 for line in file), HEIGHT_CELLS, WIDT
      8             file.seek(0)

FileNotFoundError: [Errno 2] No such file or directory: '/content/drive/My Drive/Cola
```

SEARCH STACK OVERFLOW

Define checkpoint and earlystopping

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
checkpoint = ModelCheckpoint("model-{loss:.2f}.h5", monitor="loss", verbose=1, save_best_o
```

```
                                save_weights_only=True, mode="min", period=1)
stop = EarlyStopping(monitor="loss", patience=5, mode="min")
reduce_lr = ReduceLROnPlateau(monitor="loss", factor=0.2, patience=5, min_lr=1e-6, verbose
```

```
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` to specify
```

Fit the model (2 marks)

- Fit the model using below parameters
  - epochs: you can decide
  - batch_size: 1
  - callbacks: checkpoint, reduce_lr, stop

```
model.fit_generator(generator=train_datagen,
                    epochs=EPOCHS,
                    validation_data=validation_datagen,
                    callbacks=[checkpoint, reduce_lr, stop],
                    workers=THREADS,
                    use_multiprocessing=False,
                    shuffle=True,
                    verbose=1)
```

Get the predicted mask for a sample image (3 marks)

```
n = 10
sample_image = X_train[n]
```

```
import cv2
import numpy as np

from keras.applications.mobilenet import preprocess_input
```

```
Using TensorFlow backend.
```

Impose the mask on the image (3 marks)

```
WEIGHTS_FILE = "model-0.92.h5"
THRESHOLD = 0.8
EPSILON = 0.02

model = create_model()
model.load_weights(WEIGHTS_FILE)
```