Bounding box detection - Racoon data

Data files

- images_racoon.rar: contain images of racoons
- train_labels.cv: contains coordinates for bounding box for every image

Import the necessary libraries

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491h

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

```
%tensorflow_version 2.x
import tensorflow as tf
import numpy as np
import pandas as pd
import tensorflow.keras
import keras
from tensorflow.keras import Model
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.applications.mobilenet import MobileNet, preprocess_input
from tensorflow.keras.layers import Conv2D, Reshape
from PIL import Image
from tensorflow.keras.backend import epsilon
from tensorflow.keras.backend import epsilon
```

Change directory

```
project_path = "/content/drive/My Drive/Residency_9_Assignments/"
import os
os.chdir(project_path)
```

Load the training data from train.csv file

```
racoontraindata=pd.read_csv('train_labels.csv')
```

Print the shape of the train dataset

```
racoontraindata.shape
```

```
(173, 8)
```

Declare a variable IMAGE_SIZE = 128 as we will be using MobileNet which will be taking Input shape as 128

```
IMAGE_SIZE=128
```

```
#!pip install patool
import patoolib
patoolib.extract_archive("images_racoon.rar", outdir="train")
```

```
patool: Extracting images_racoon.rar ...
patool: running /usr/bin/unrar x -- "/content/drive/My Drive/Residency_9_Assignments/images_racoon.rar"
patool:     with cwd='train'
patool: ... images_racoon.rar extracted to `train'.
'train'
```

With the help of csv.reader write a for loop which can load the train.csv file and store the path, width, height, x0,y0,x1,y1 in induvidual variables.

1. Create a list variable known as 'path' which has all the path for all the training images
2. Create an array 'coords' which has the resized coordinates of the bounding box for the training images

Note: All the training images should be downsampled to 128 * 128 as it is the input shape of MobileNet (which we will be using for Obj detection). Hence the corresponding coordinates of the bounding boxes should be changed to match the image dimension of 128 * 12

```
import csv
with open('train_labels.csv', 'r') as csvfile:
    paths = []
    i=1
    coords = np.zeros((sum(1 for line in csvfile) -1, 4))
    reader = csv.reader(csvfile, delimiter=',')
    csvfile.seek(0)
    next(reader, None)
    for col, row in enumerate(reader):
        path, image_width, image_height, imageclass, x0, y0, x1, y1 = row
        coords[col, 0] = int(x0) * IMAGE_SIZE / int(image_width) # Normalize bounding box by image size
        coords[col, 1] = int(y0) * IMAGE_SIZE / int(image_height) # Normalize bounding box by image size
        coords[col, 2] = int((int(x1) - int(x0))) * IMAGE_SIZE / int(image_width) # Normalize bounding box by image size
        coords[col, 3] = int((int(y1) - int(y0))) * IMAGE_SIZE / int(image_height)
        paths.append(path)
```

```
batch_images.shape,coords.shape
```

```
((173, 128, 128, 3), (173, 4))
```

Write a for loop which can load all the training images into a variable 'batch_images' using the paths from th 'paths' variable

Note: Convert the image to RGB scale as the MobileNet accepts 3 channels as inputs

```
batch_images = np.zeros((len(paths), IMAGE_SIZE, IMAGE_SIZE, 3), dtype=np.float32)
for i, f in enumerate(paths):
  loc="train/images/"+f
  img = Image.open(loc) # Read image
  img = img.resize((IMAGE_SIZE, IMAGE_SIZE)) # Resize image
  img = img.convert('RGB')
  batch_images[i] = preprocess_input(np.array(img, dtype=np.float32))
```

```
coords=coords[0:173]
```

```
len(coords)
```

```
173
```

Import MobileNet and load MobileNet into a variable named 'model' which takes input shape of 128 * 128 *
Freeze all the layers. Add convolution and reshape layers at the end to ensure the output is 4 coordinates

```
ALPHA = 1.0
from tensorflow.keras.applications.mobilenet import MobileNet
model = MobileNet(input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3), include_top=False, alpha=ALPHA) # Load pre-trained mobilenet

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


x = model.layers[-1].output
x = Conv2D(4, kernel_size=4, name="coords")(x)

x = Reshape((4,))(x)

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

```
model.summary()
model.summary()
```

```
_________________________________________________________________
conv_dw_10 (DepthwiseConv2D) (None, 8, 8, 512)         4608      
_________________________________________________________________
conv_dw_10_bn (BatchNormaliz (None, 8, 8, 512)         2048      
_________________________________________________________________
conv_dw_10_relu (ReLU)       (None, 8, 8, 512)         0         
_________________________________________________________________
conv_pw_10 (Conv2D)          (None, 8, 8, 512)         262144    
_________________________________________________________________
conv_pw_10_bn (BatchNormaliz (None, 8, 8, 512)         2048      
_________________________________________________________________
conv_pw_10_relu (ReLU)       (None, 8, 8, 512)         0         
_________________________________________________________________
conv_dw_11 (DepthwiseConv2D) (None, 8, 8, 512)         4608      
_________________________________________________________________
conv_dw_11_bn (BatchNormaliz (None, 8, 8, 512)         2048      
_________________________________________________________________
conv_dw_11_relu (ReLU)       (None, 8, 8, 512)         0         
_________________________________________________________________
conv_pw_11 (Conv2D)          (None, 8, 8, 512)         262144    
_________________________________________________________________
conv_pw_11_bn (BatchNormaliz (None, 8, 8, 512)         2048      
_________________________________________________________________
conv_pw_11_relu (ReLU)       (None, 8, 8, 512)         0         
_________________________________________________________________
conv_pad_12 (ZeroPadding2D)  (None, 9, 9, 512)         0         
_________________________________________________________________
conv_dw_12 (DepthwiseConv2D) (None, 4, 4, 512)         4608      
_________________________________________________________________
conv_dw_12_bn (BatchNormaliz (None, 4, 4, 512)         2048      
_________________________________________________________________
conv_dw_12_relu (ReLU)       (None, 4, 4, 512)         0         
_________________________________________________________________
conv_pw_12 (Conv2D)          (None, 4, 4, 1024)        524288    
_________________________________________________________________
conv_pw_12_bn (BatchNormaliz (None, 4, 4, 1024)        4096      
_________________________________________________________________
conv_pw_12_relu (ReLU)       (None, 4, 4, 1024)        0         
_________________________________________________________________
conv_dw_13 (DepthwiseConv2D) (None, 4, 4, 1024)        9216      
_________________________________________________________________
conv_dw_13_bn (BatchNormaliz (None, 4, 4, 1024)        4096      
_________________________________________________________________
conv_dw_13_relu (ReLU)       (None, 4, 4, 1024)        0         
```

```
_________________________________________________________________
conv_pw_13 (Conv2D)          (None, 4, 4, 1024)        1048576   
_________________________________________________________________
conv_pw_13_bn (BatchNormaliz (None, 4, 4, 1024)        4096      
_________________________________________________________________
conv_pw_13_relu (ReLU)       (None, 4, 4, 1024)        0         
_________________________________________________________________
coords (Conv2D)              (None, 1, 1, 4)           65540     
_________________________________________________________________
reshape_3 (Reshape)          (None, 4)                 0         
=================================================================
Total params: 3,294,404
Trainable params: 65,540
Non-trainable params: 3,228,864
_________________________________________________________________
```

Define a custom loss function IoU which calculates Intersection Over Union

```
gt = coords
def loss(gt,pred):
    intersections = 0
    unions = 0
    diff_width = np.minimum(gt[:,0] + gt[:,2], pred[:,0] + pred[:,2]) - np.maximum(gt[:,0], pred[:,0])
    diff_height = np.minimum(gt[:,1] + gt[:,3], pred[:,1] + pred[:,3]) - np.maximum(gt[:,1], pred[:,1])
    intersection = diff_width * diff_height

    # Compute union
    area_gt = gt[:,2] * gt[:,3]
    area_pred = pred[:,2] * pred[:,3]
    union = area_gt + area_pred - intersection

#     Compute intersection and union over multiple boxes
    for j, _ in enumerate(union):
        if union[j] > 0 and intersection[j] > 0 and union[j] >= intersection[j]:
            intersections += intersection[j]
            unions += union[j]

    # Compute IOU. Use epsilon to prevent division by zero
    iou = np.round(intersections / (unions + epsilon()), 4)
    iou = iou.astype(np.float32)
    return iou
```

```
def IoU(y_true, y_pred):
    iou = tf.py_function(loss, [y_true, y_pred], tf.float32)
    return iou
```

Write model.compile function & model.fit function with:

1. Optimizer = Adam, Loss = 'mse' and metrics = IoU
2. Epochs = 30, batch_size = 32, verbose = 1

```
model.compile(optimizer='Adam', loss='mse', metrics=[IoU]) # Regression loss is MSE

model.fit(batch_images,gt,
            epochs=100,batch_size = 32,
            verbose=1)
```

```
173/173 [==============================] - 4s 21ms/sample - loss: 56.4457 - IoU: 0.8489
Epoch 73/100
173/173 [==============================] - 4s 21ms/sample - loss: 54.6637 - IoU: 0.8442
Epoch 74/100
173/173 [==============================] - 4s 21ms/sample - loss: 52.6785 - IoU: 0.8507
Epoch 75/100
173/173 [==============================] - 4s 21ms/sample - loss: 52.8372 - IoU: 0.8472
Epoch 76/100
173/173 [==============================] - 4s 21ms/sample - loss: 50.3953 - IoU: 0.8506
Epoch 77/100
173/173 [==============================] - 4s 21ms/sample - loss: 53.9261 - IoU: 0.8465
Epoch 78/100
173/173 [==============================] - 4s 21ms/sample - loss: 49.6683 - IoU: 0.8379
Epoch 79/100
173/173 [==============================] - 4s 20ms/sample - loss: 53.4254 - IoU: 0.8569
Epoch 80/100
173/173 [==============================] - 3s 20ms/sample - loss: 54.5921 - IoU: 0.8423
Epoch 81/100
173/173 [==============================] - 4s 20ms/sample - loss: 57.1800 - IoU: 0.8268
Epoch 82/100
173/173 [==============================] - 4s 20ms/sample - loss: 60.8048 - IoU: 0.8080
Epoch 83/100
173/173 [==============================] - 4s 20ms/sample - loss: 57.7072 - IoU: 0.8177
Epoch 84/100
173/173 [==============================] - 4s 20ms/sample - loss: 67.1617 - IoU: 0.8165
Epoch 85/100
173/173 [==============================] - 4s 23ms/sample - loss: 59.1716 - IoU: 0.8136
Epoch 86/100
173/173 [==============================] - 4s 21ms/sample - loss: 63.3584 - IoU: 0.8301
Epoch 87/100
173/173 [==============================] - 4s 21ms/sample - loss: 53.2042 - IoU: 0.8298
Epoch 88/100
173/173 [==============================] - 4s 20ms/sample - loss: 53.7854 - IoU: 0.8500
Epoch 89/100
173/173 [==============================] - 4s 21ms/sample - loss: 66.5297 - IoU: 0.8441
Epoch 90/100
173/173 [==============================] - 4s 21ms/sample - loss: 57.8389 - IoU: 0.8458
Epoch 91/100
173/173 [==============================] - 4s 20ms/sample - loss: 55.8730 - IoU: 0.8450
Epoch 92/100
173/173 [==============================] - 4s 21ms/sample - loss: 56.4150 - IoU: 0.8412
Epoch 93/100
173/173 [==============================] - 4s 21ms/sample - loss: 54.8108 - IoU: 0.8395
```

```
Epoch 94/100
173/173 [==============================] - 4s 21ms/sample - loss: 55.7613 - IoU: 0.8383
Epoch 95/100
173/173 [==============================] - 4s 21ms/sample - loss: 57.5143 - IoU: 0.8432
Epoch 96/100
173/173 [==============================] - 4s 21ms/sample - loss: 57.8352 - IoU: 0.8323
Epoch 97/100
173/173 [==============================] - 4s 20ms/sample - loss: 55.7905 - IoU: 0.8285
Epoch 98/100
173/173 [==============================] - 4s 21ms/sample - loss: 58.1668 - IoU: 0.8351
Epoch 99/100
173/173 [==============================] - 4s 21ms/sample - loss: 52.5766 - IoU: 0.8532
Epoch 100/100
173/173 [==============================] - 4s 20ms/sample - loss: 57.1602 - IoU: 0.8461
<tensorflow.python.keras.callbacks.History at 0x7f961c351278>
```

```
len(coords)
```

```
173
```

Pick a test image from the given data

```
import cv2
filename = './train/images/raccoon-146.jpg'
unscaled = cv2.imread(filename)
```

Resize the image to 128 * 128 and preprocess the image for the MobileNet model

```
image_height, image_width, _ = unscaled.shape
```

```
image = cv2.resize(unscaled, (IMAGE_SIZE, IMAGE_SIZE)) # Rescaled image to run the network
image.shape
```

```
(128, 128, 3)
```

```
feat_scaled = preprocess_input(np.array(image, dtype=np.float32))
```

Predict the coordinates of the bounding box for the given test image

```
region = model.predict(x=np.array([feat_scaled]))[0]
region
```

```
array([ 21.523046,   6.597309,  95.82405 , 124.61007 ], dtype=float32)
```

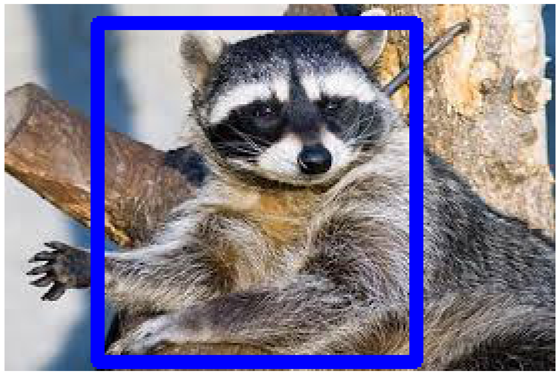
Plot the test image using .imshow and draw a boundary box around the image with the coordinates obtaine the model

```
x0 = int(region[0] * image_width / IMAGE_SIZE) # Scale the BBox
y0 = int(region[1] * image_height / IMAGE_SIZE)

x1 = int((region[2]) * image_width / IMAGE_SIZE)
y1 = int((region[3]) * image_height / IMAGE_SIZE)
print(x0,y0,x1,y1)
```

```
46 9 205 178
```

```
rt=cv2.rectangle(unscaled,(x0,y0),(x1,y1),(255,0,0),5)
```

```
from google.colab.patches import cv2_imshow

cv2_imshow(rt)
```



Time Series Prediction using LSTM

Download Data

Link: https://datamarket.com/data/set/2324/daily-minimum-temperatures-in-melbourne-australia-1981-1990#!ds=2324&display=line

Description

Daily minimum temperatures in Melbourne, Australia, 1981-1990

Units: Degrees Celcius

Steps before loading

- Rename the column name with temprature values to "Temprature"
- In the last, there is one extra row in the data, remove it by opening the file and save it again.

- There are some values in Temprature column which have a "?" before them, they will give error, remove "?" before them and save t
- If you don't want to do these steps, just load the data file given by Great Learning.

Mount google drive

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remou
```

Change your present working directory

```
project_path = "/content/drive/My Drive/Residency_9_Assignments/"
import os
os.chdir(project_path)
```

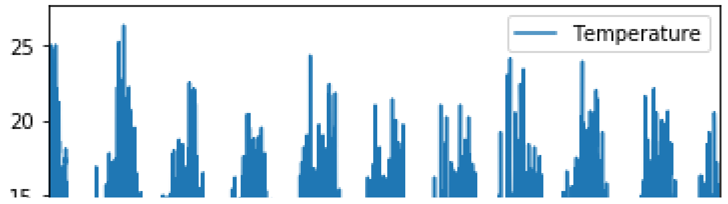
Load your data file

```
tempdata=pd.read_csv('daily-minimum-temperatures-in-me.csv')
```

Plot data

```
tempdata.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f961c8f4208>
```



Descibe your dataframe

```
tempdata.dtypes
```

```
Date              object
Temperature      float64
dtype: object
```

```
tempdata.describe()
```

	Temperature
count	3650.000000
mean	11.177753
std	4.071837
min	0.000000
25%	8.300000
50%	11.000000
75%	14.000000
max	26.300000

Check for null values

```
tempdata.isnull().sum()
```

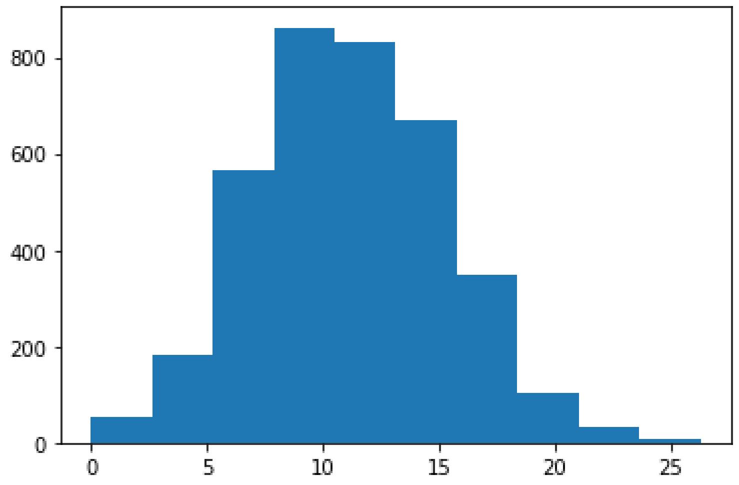
```
Date           0
Temperature    0
dtype: int64
```

Drop null values

Get the representation of the distribution of data in the form of histogram

```
plt.hist(tempdata['Temperature'])
```

```
(array([ 53., 181., 564., 862., 830., 670., 347., 102.,  32.,   9.]),
 array([ 0.  ,  2.63,  5.26,  7.89, 10.52, 13.15, 15.78, 18.41, 21.04,
        23.67, 26.3 ]),
 <a list of 10 Patch objects>)
```



Check the maximum and minimum values

```
print('Min', np.min(tempdata))
print('Max', np.max(tempdata))
```

```
Min Date           1981-01-01
Temperature             0
dtype: object
Max Date           1990-12-31
Temperature          26.3
dtype: object
```

Normalize the data

```
Temperaturedata=tempdata[['Temperature']]
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(Temperaturedata)
```

Check the maximum and minimum values of scaled data

```
print('Min', np.min(scaled_data))
print('Max', np.max(scaled_data))
```

```
Min 0.0
Max 1.0
```

Look into some of the scaled values

```
scaled_data[0:5]
```

```
array([[0.78707224],
       [0.68060837],
       [0.7148289 ],
       [0.55513308],
       [0.60076046]])
```

Split data into Training and Testing

```
train_size = int(len(scaled_data) * 0.70)
test_size = len(scaled_data - train_size)
```

Print train and test size

```
train, test = scaled_data[0:train_size, :], scaled_data[train_size: len(scaled_data), :]
print('train: {}\ntest: {}'.format(len(train), len(test)))
```

```
train: 2555
test: 1095
```

Create the sequential data

Map the temprature at a particular time t to the temprature at time t+n, where n is any number you define.

For example: to map tempratures of consecutive days, use t+1, i.e. loop_back = 1

Define your function to create dataset

```
#window - how long the sequence will be
def create_dataset(dataset, window=1):

    dataX, dataY = [], []

    for i in range(len(dataset)-window):

        a = dataset[i:(i+window), 0]
        dataX.append(a)
        dataY.append(dataset[i + window, 0])

    return np.array(dataX), np.array(dataY)
```

Use function to get training and test set

```
#Create Input and Output
window_size = 1
X_train, y_train = create_dataset(train, window_size)
X_test, y_test = create_dataset(test, window_size)
```

Transform the prepared train and test input data into the expected structure using numpy.reshape()

```
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
print(X_train.shape)
print(X_test.shape)
```

```
(2554, 1, 1)
(1094, 1, 1)
```

Define Model

Define sequntial model, add LSTM layer and compile the model

```
tempmodel = tf.keras.Sequential()
tempmodel.add(tf.keras.layers.LSTM(32, input_shape=(window_size, 1)))
tempmodel.add(tf.keras.layers.Dense(1))
tempmodel.compile(optimizer='adam', loss='mse')
```

Summarize your model

```
tempmodel.summary()
```