Download and Visualize Data

```
# import dataset

# !wget https://github.com/hfg-gmuend/openmoji/releases/latest/download/openmoji-72x72-color.zip
# !mkdir emojis
# !unzip -q openmoji-72x72-color.zip -d ./emojis
!pip install tensorflow==2.4
```

```
Requirement already satisfied: google-pasta~=0.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4) (0.2.0)
Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.4) (3.17.3)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.4
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.4->tensor
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.4->tensorfl
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.4->tensorflow
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.4->tensorflow==
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorbo
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.4->tensorflo
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3->t
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3->
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3->tensorboa
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dist-packages (from google-auth-oauthlib<
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.7/dist-packages (from markdown>=2.6.8->tensor
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata>=4.4->markdown>=2
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1->go
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard~
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorb
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from reques
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensor
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from requests-oauthlib>=0.7.0->goog
Building wheels for collected packages: wrapt
  Building wheel for wrapt (setup.py) ... done
  Created wheel for wrapt: filename=wrapt-1.12.1-cp37-cp37m-linux_x86_64.whl size=68717 sha256=69fdfe127412e9121b22eec09bac9a
  Stored in directory: /root/.cache/pip/wheels/62/76/4c/aa25851149f3f6d9785f6c869387ad82b3fd37582fa8147ac6
Successfully built wrapt
Installing collected packages: typing-extensions, grpcio, wrapt, tensorflow-estimator, h5py, gast, flatbuffers, tensorflow
  Attempting uninstall: typing-extensions
    Found existing installation: typing-extensions 3.10.0.2
    Uninstalling typing-extensions-3.10.0.2:
      Successfully uninstalled typing-extensions-3.10.0.2
  Attempting uninstall: grpcio
    Found existing installation: grpcio 1.42.0
    Uninstalling grpcio-1.42.0:
      Successfully uninstalled grpcio-1.42.0
  Attempting uninstall: wrapt
    Found existing installation: wrapt 1.13.3
    Uninstalling wrapt-1.13.3:
      Successfully uninstalled wrapt-1.13.3
  Attempting uninstall: tensorflow-estimator
    Found existing installation: tensorflow-estimator 2.7.0
    Uninstalling tensorflow-estimator-2.7.0:
      Successfully uninstalled tensorflow-estimator-2.7.0
  Attempting uninstall: h5py
    Found existing installation: h5py 3.1.0
    Uninstalling h5py-3.1.0:
      Successfully uninstalled h5py-3.1.0
  Attempting uninstall: gast
    Found existing installation: gast 0.4.0
    Uninstalling gast-0.4.0:
      Successfully uninstalled gast-0.4.0
  Attempting uninstall: flatbuffers
    Found existing installation: flatbuffers 2.0
    Uninstalling flatbuffers-2.0:
      Successfully uninstalled flatbuffers-2.0
  Attempting uninstall: tensorflow
```

```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

```
Choose Files 6 files
```

- **8BALL.png**(image/png) - 2232 bytes, last modified: 4/30/2020 - 100% done
- **BASEBALL.png**(image/png) - 2387 bytes, last modified: 6/11/2020 - 100% done
- **BASKETBALL.png**(image/png) - 2132 bytes, last modified: 4/30/2020 - 100% done
- **FOOTBALL.png**(image/png) - 2193 bytes, last modified: 4/29/2020 - 100% done
- **LEATHERBALL.png**(image/png) - 7495 bytes, last modified: 12/12/2021 - 100% done
- **TENNISBALL.png**(image/png) - 9556 bytes, last modified: 12/12/2021 - 100% done

```
Saving 8BALL.png to 8BALL.png
Saving BASEBALL.png to BASEBALL.png
Saving BASKETBALL.png to BASKETBALL.png
Saving FOOTBALL.png to FOOTBALL.png
Saving LEATHERBALL.png to LEATHERBALL.png
Saving TENNISBALL.png to TENNISBALL.png
User uploaded file "8BALL.png" with length 2232 bytes
User uploaded file "BASEBALL.png" with length 2387 bytes
```

```
# import libraries
%matplotlib inline

import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import os

from PIL import Image, ImageDraw
from tensorflow.keras.layers import Input, Dense, Flatten, Conv2D, MaxPool2D, BatchNormalization, Dropout
print('Check if we are using TensorFlow 2.4')
print('Using TensorFlow version', tf.__version__)
```

```
Check if we are using TensorFlow 2.4
Using TensorFlow version 2.4.0
```

```
# balls actually using in this project
balls = {
    0: {'name': '8ball', 'file': '8BALL.png'},
    1: {'name': 'tennisball', 'file': 'TENNISBALL.png'},
    2: {'name': 'basketball', 'file': 'BASKETBALL.png'},
    3: {'name': 'football', 'file': 'FOOTBALL.png'},
    4: {'name': 'baseball', 'file': 'BASEBALL.png'},
    5: {'name': 'leatherball', 'file': 'LEATHERBALL.png'}
}

# place images in larger images - help synthesis data for localization
plt.figure(figsize=(9, 9))

for i, (j, e) in enumerate(balls.items()):
    plt.subplot(3, 3, i + 1)
    plt.imshow(plt.imread(e['file']))
    plt.xlabel(e['name'])
    plt.xticks([])
    plt.yticks([])
plt.show()
```

Create Examples

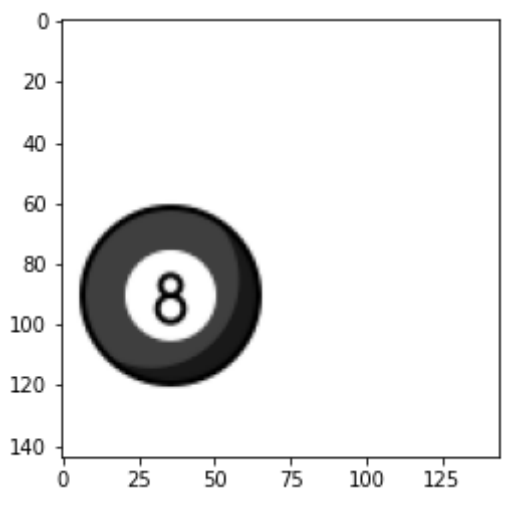
```
# loads balls and assign a key - image for each class in balls dictonary
for class_id, values in balls.items():
```

```
    png_file = Image.open(values['file']).convert('RGBA')
    png_file.load()
    new_file = Image.new("RGB", png_file.size, (255, 255, 255))
    new_file.paste(png_file, mask=png_file.split()[3])
    balls[class_id]['image'] = new_file


balls
```

```
 {0: {'file': '8BALL.png',
   'image': <PIL.Image.Image image mode=RGB size=72x72 at 0x7FA7C2D9D950>,
   'name': '8ball'},
  1: {'file': 'TENNISBALL.png',
   'image': <PIL.Image.Image image mode=RGB size=72x72 at 0x7FA7C45EB910>,
   'name': 'tennisball'},
  2: {'file': 'BASKETBALL.png',
   'image': <PIL.Image.Image image mode=RGB size=72x72 at 0x7FA7C2D9D990>,
   'name': 'basketball'},
  3: {'file': 'FOOTBALL.png',
   'image': <PIL.Image.Image image mode=RGB size=72x72 at 0x7FA7C45C5410>,
   'name': 'football'},
  4: {'file': 'BASEBALL.png',
   'image': <PIL.Image.Image image mode=RGB size=72x72 at 0x7FA7C45D0690>,
   'name': 'baseball'},
  5: {'file': 'LEATHERBALL.png',
   'image': <PIL.Image.Image image mode=RGB size=72x72 at 0x7FA7C2DAAD90>,
   'name': 'leatherball'}}
```

```
def create_example():
  class_id = np.random.randint(0,6) # randomly choose a emoji
  image = np.ones((144,144,3)) * 255 # create a white image
  row = np.random.randint(0,72) # place image randomly
  col = np.random.randint(0,72) #place image ramdomly
  # place emoji in blank emoji
  image[row: row+72, col: col+72, :]= np.array(balls[class_id]['image'])
  # return synthesize image
  return image.astype('uint8'), class_id, (row+5)/144 , (col+5)/144      # +10 becuase there is a white space of around 10 pixels in
  # /144 is normalization of image


image, class_id, row, col = create_example()
plt.imshow(image)
```

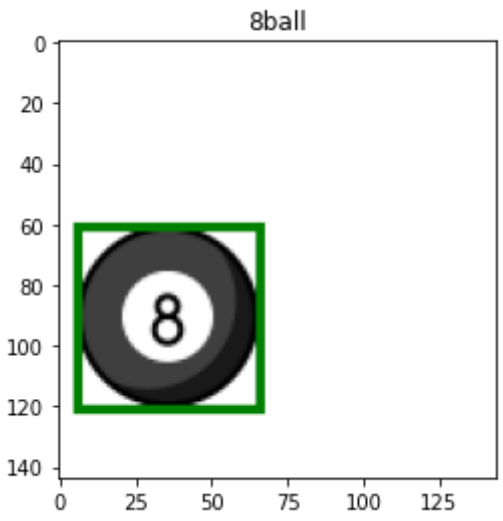
```
<matplotlib.image.AxesImage at 0x7fa7c4633a50>
```



Plot Bounding Boxes

```
#plotting bounding boxes
def plot_bounding_box(image, gt_coords, pred_coords=[], norm=False): # pass image, ground truth row col coordinates, predicted coordi
  if norm:
    image *=255 # if norm is true, we will denormalize it
    image=image.astype('uint8')
  image = Image.fromarray(image) #to convert image array to pil image
  draw = ImageDraw.Draw(image)

  # extrating row and col from groud truth values
  row, col = gt_coords

  # denormalizing coords
  row *= 144
  col *= 144
  draw.rectangle((col, row, col+62, row+62), outline = 'green', width=3)  #+52 becuase iage is of 72 pixels becuase image has buffer

  # now same for pred coords
```

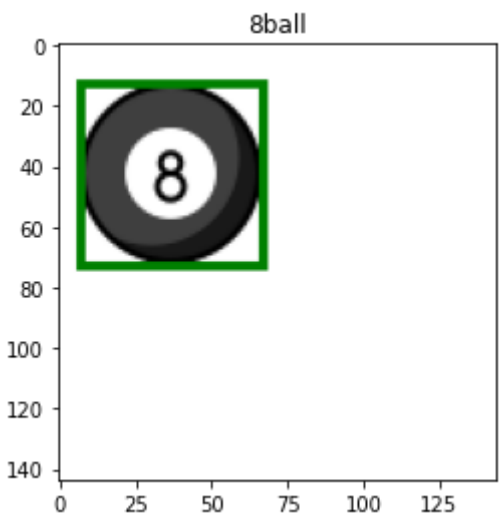
```
    if len(pred_coords)==2:
      # extrating row and col from groud truth values
      row, col = pred_coords

      # denormalizing coords
      row *= 144
      col *= 144
      draw.rectangle((col, row, col+62, row+62), outline = 'red', width=3)

    return image


image = plot_bounding_box(image, gt_coords = [row, col])
plt.imshow(image)
plt.title(balls[class_id]['name'])
plt.show()
```



Data Generator

```
# create endless stream of these randomly generated eg which we will use in our model
def data_generator(batch_size=16):
  #run in endless loop and create example and labels of batch size
  while True:
    x_batch = np.zeros((batch_size, 144, 144, 3)) # 144 = size of image
    y_batch = np.zeros((batch_size, 6)) #9= no of class ids
    bbox_batch = np.zeros((batch_size, 2)) #2 = for row and col values

    # create examples of no of batch size
    for i in range(0, batch_size):
      image, class_id, row, col = create_example()
      x_batch[i] = image/255   # normalize image and 255 because they are pixel values
      y_batch[i, class_id] = 1.0
      bbox_batch[i] = np.array([row,col])
    yield {'image':x_batch} ,{'class_out': y_batch, 'box_out': bbox_batch}
```

```
example, label = next(data_generator(1))
image = example['image'][0]
class_id = np.argmax(label['class_out'][0])  # to get the actual class id
coords = label['box_out'][0]

image = plot_bounding_box(image, coords, norm=True)
plt.imshow(image)
plt.title(balls[class_id]['name'])
plt.show()
```



Model

```
# CNN model
input_ = Input(shape=(144,144,3), name='image')

x = input_

# we will have totoal 4 convolutional blocks
for i in range(0,4):
  n_filters = 2**(4+i)
  x = Conv2D(n_filters, 3, activation='relu')(x) # (x) - input is x
  x = BatchNormalization()(x)
  x = MaxPool2D(2)(x) #pool size od 2x2

x = Flatten()(x)
x = Dense(256, activation = 'relu')(x)

# now connect fully conected layer to our 2 output
class_out = Dense(6, activation='softmax', name = 'class_out')(x)   # 9= outputs as we have 9 clasess# for classifaction out, we use
box_out = Dense(2, name = 'box_out')(x) # we dont specify any activation as it is regression output and it is linear by defualt

# now construct the model
model = tf.keras.models.Model(input_, [class_out, box_out])
model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
image (InputLayer)              [(None, 144, 144, 3) 0
__________________________________________________________________________________________________
conv2d (Conv2D)                 (None, 142, 142, 16) 448         image[0][0]
__________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 142, 142, 16) 64          conv2d[0][0]
__________________________________________________________________________________________________
max_pooling2d (MaxPooling2D)    (None, 71, 71, 16)   0           batch_normalization[0][0]
__________________________________________________________________________________________________
conv2d_1 (Conv2D)               (None, 69, 69, 32)   4640        max_pooling2d[0][0]
__________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 69, 69, 32)   128         conv2d_1[0][0]
__________________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 34, 34, 32)   0           batch_normalization_1[0][0]
__________________________________________________________________________________________________
conv2d_2 (Conv2D)               (None, 32, 32, 64)   18496       max_pooling2d_1[0][0]
__________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 32, 32, 64)   256         conv2d_2[0][0]
__________________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 16, 16, 64)   0           batch_normalization_2[0][0]
__________________________________________________________________________________________________
conv2d_3 (Conv2D)               (None, 14, 14, 128)  73856       max_pooling2d_2[0][0]
__________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 14, 14, 128)  512         conv2d_3[0][0]
__________________________________________________________________________________________________
max_pooling2d_3 (MaxPooling2D)  (None, 7, 7, 128)    0           batch_normalization_3[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 6272)         0           max_pooling2d_3[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 256)          1605888     flatten[0][0]
__________________________________________________________________________________________________
class_out (Dense)               (None, 6)            1542        dense[0][0]
__________________________________________________________________________________________________
box_out (Dense)                 (None, 2)            514         dense[0][0]
==================================================================================================
Total params: 1,706,344
Trainable params: 1,705,864
Non-trainable params: 480
__________________________________________________________________________________________________
```

Custom Metric: IoU

```
# intersection over union is the evaluation metric
# to measure the performance of the model - common in finding accuracy in object detector and object localizers
# iou is area of overlap [intesection of 2 boxes] between the predicted bounding box and actual values and
```

```
# combining the areas of both minus intersection will give area of union
# divide area of overlap by area of union   IoU values  if 1   prediction is accurate
class IoU(tf.keras.metrics.Metric):
  def __init__(self, **kwargs):
    super(IoU, self).__init__(**kwargs)

    self.iou = self.add_weight(name='iou', initializer='zeros')
    self.total_iou = self.add_weight(name='total_iou', initializer='zeros')
    self.num_ex = self.add_weight(name='num_ex', initializer='zeros')

  def update_state(self, y_true, y_pred, sample_weight=None):
    def get_box(y):
      rows, cols = y[:, 0], y[:, 1]
      rows, cols = rows * 144, cols * 144
      y1, y2 = rows, rows + 62
      x1, x2 = cols, cols + 62
      return x1, y1, x2, y2

    def get_area(x1, y1, x2, y2):
      return tf.math.abs(x2 - x1) * tf.math.abs(y2 - y1)

    gt_x1, gt_y1, gt_x2, gt_y2 = get_box(y_true)
    p_x1, p_y1, p_x2, p_y2 = get_box(y_pred)

    i_x1 = tf.maximum(gt_x1, p_x1)
    i_y1 = tf.maximum(gt_y1, p_y1)
    i_x2 = tf.minimum(gt_x2, p_x2)
    i_y2 = tf.minimum(gt_y2, p_y2)

    i_area = get_area(i_x1, i_y1, i_x2, i_y2)
    u_area = get_area(gt_x1, gt_y1, gt_x2, gt_y2) + get_area(p_x1, p_y1, p_x2, p_y2) - i_area

    iou = tf.math.divide(i_area, u_area)
    self.num_ex.assign_add(1)
    self.total_iou.assign_add(tf.reduce_mean(iou))
    self.iou = tf.math.divide(self.total_iou, self.num_ex)

  def result(self):
    return self.iou

  def reset_state(self):
    self.iou = self.add_weight(name='iou', initializer='zeros')
    self.total_iou = self.add_weight(name='total_iou', initializer='zeros')
    self.num_ex = self.add_weight(name='num_ex', initializer='zeros')
```

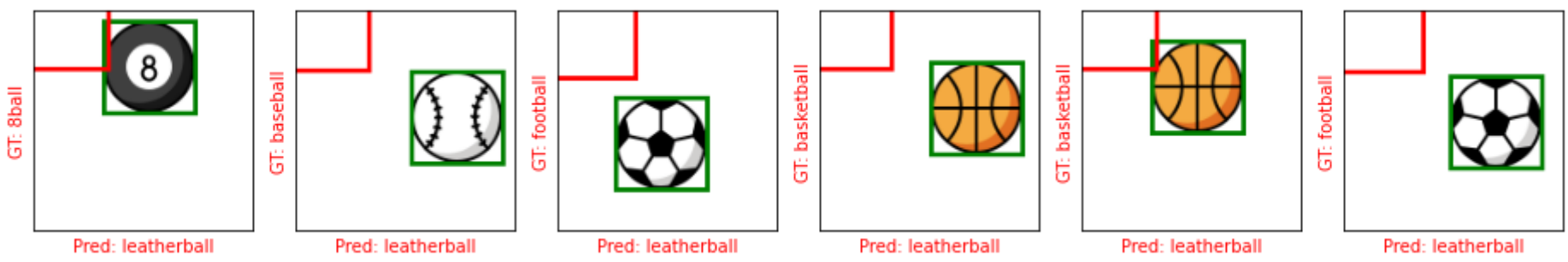
Task 8: Compile the Model

```
model.compile(
    # specify the loss for fifferent outputs
    loss={
        'class_out': 'categorical_crossentropy', # or classifaction output
        'box_out': 'mse' #for regression output
    },
    optimizer = tf.keras.optimizers.Adam(learning_rate=1e-3),

    #set differnet metric for different output
    metrics={
        'class_out': 'accuracy',
        'box_out': IoU(name='iou')
    }
)
```

Custom Callback: Model Testing

```
def test_model(model, test_datagen):
  example, label = next(test_datagen)
  x = example['image']
  y = label['class_out']
  box = label['box_out']

  pred_y, pred_box = model.predict(x)

  pred_coords = pred_box[0]  # [0] beacuse we want only one example
  gt_coords = box[0]
```

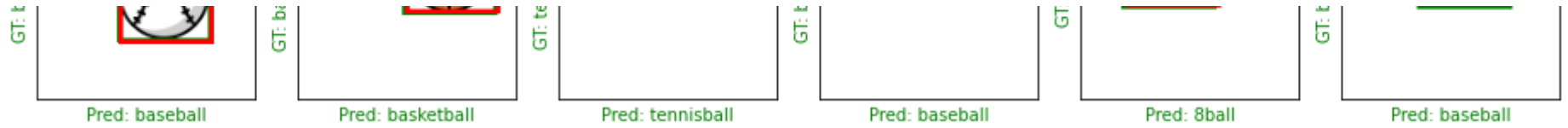
```
    pred_class = np.argmax(pred_y[0])
    image = x[0]

    gt = balls[np.argmax(y[0])]['name']
    pred_class_name = balls[pred_class]['name']

    image = plot_bounding_box(image, gt_coords, pred_coords, norm=True)

    # set text colors of labels
    color = 'green' if gt == pred_class_name else 'red'

    plt.imshow(image)
    plt.xlabel(f'Pred: {pred_class_name}', color=color)
    plt.ylabel(f'GT: {gt}', color=color)
    plt.xticks([])
    plt.yticks([])


  def test(model):
    test_datagen = data_generator(1) #1 = batch size is 1


    plt.figure(figsize=(16, 4))

    # plot 6 images
    for i in range(0, 6):
      plt.subplot(1, 6, i + 1)
      test_model(model, test_datagen)
    plt.show()


test(model)
```



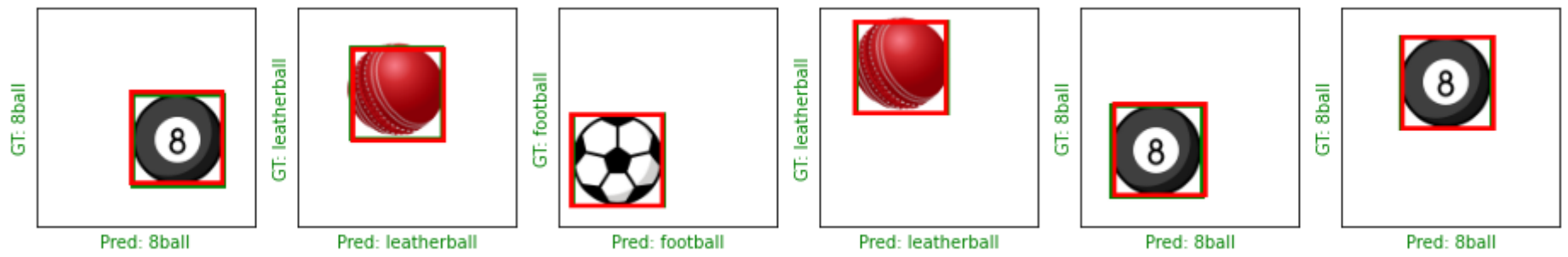

```
# create a custom call back
class ShowTestImages(tf.keras.callbacks.Callback): # to customize Callback class
  def on_epoch_end(self, epoch, logs=None): # on epoch end during the training, run test funtion
    test(self.model)
```

Model Training

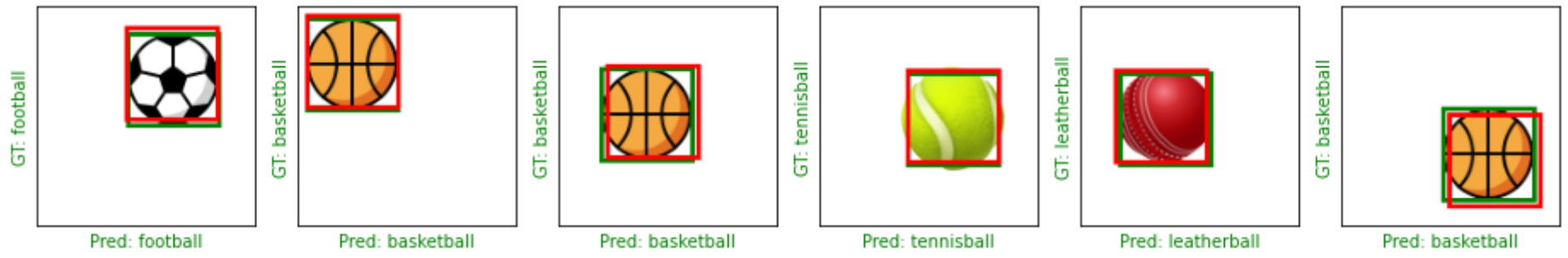
```
def lr_schedule(epoch, lr):
  if (epoch+1)%5 == 0:
    lr *= 0.2
  return max(lr, 3e-7)

_ = model.fit(
    data_generator(),
    epochs = 100,
    steps_per_epoch = 500,
    callbacks=[
              ShowTestImages(),
              tf.keras.callbacks.EarlyStopping(monitor='box_out_iou', patience=3, mode='max'),
              #stop is iou values does not incraese for 4 consective epochs

              tf.keras.callbacks.LearningRateScheduler(lr_schedule)
    ]
)
```



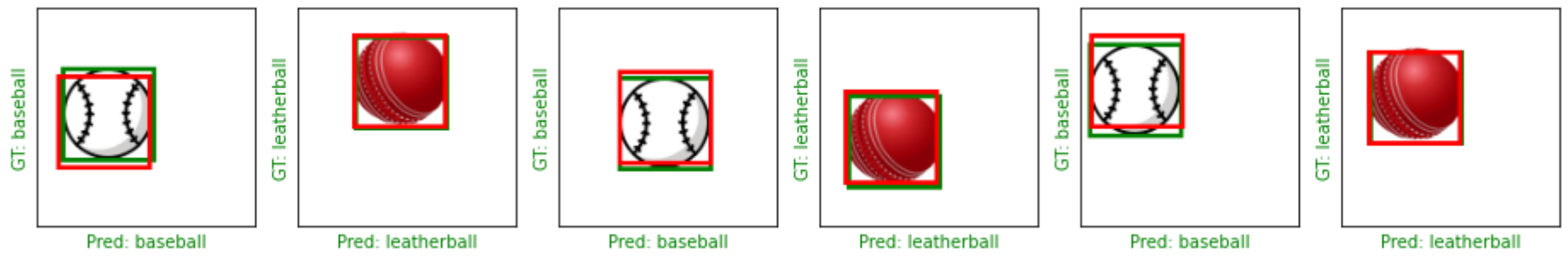

```
Epoch 24/100
500/500 [==============================] - 14s 28ms/step - loss: 4.2631e-04 - class_out_loss: 2.7818e-05 - box_out_loss: 3.9849
```



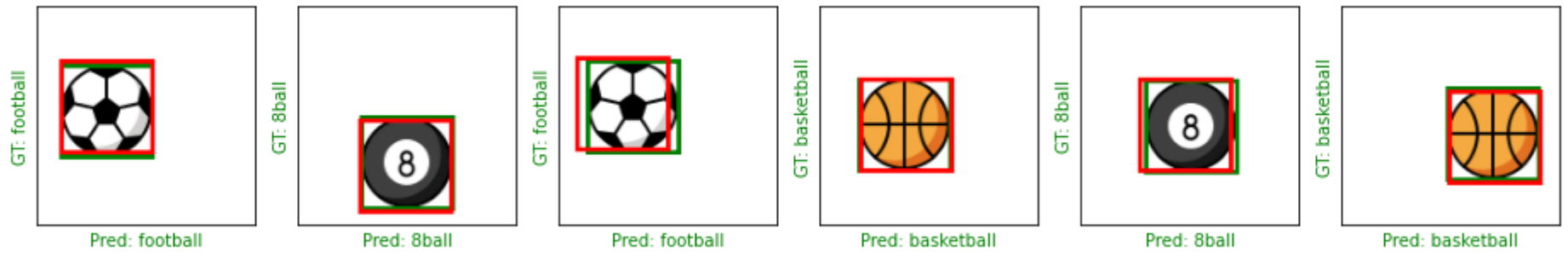

```
Epoch 25/100
500/500 [==============================] - 14s 28ms/step - loss: 4.3857e-04 - class_out_loss: 2.7475e-05 - box_out_loss: 4.1110
```




```
Epoch 26/100
500/500 [==============================] - 14s 28ms/step - loss: 6.2530e-04 - class_out_loss: 2.5180e-04 - box_out_loss: 3.7350
```




```
Epoch 27/100
500/500 [==============================] - 14s 28ms/step - loss: 4.0534e-04 - class_out_loss: 2.3729e-05 - box_out_loss: 3.8162
```




```
Epoch 28/100
500/500 [==============================] - 14s 28ms/step - loss: 4.3162e-04 - class_out_loss: 4.7983e-05 - box_out_loss: 3.8364
```