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
```

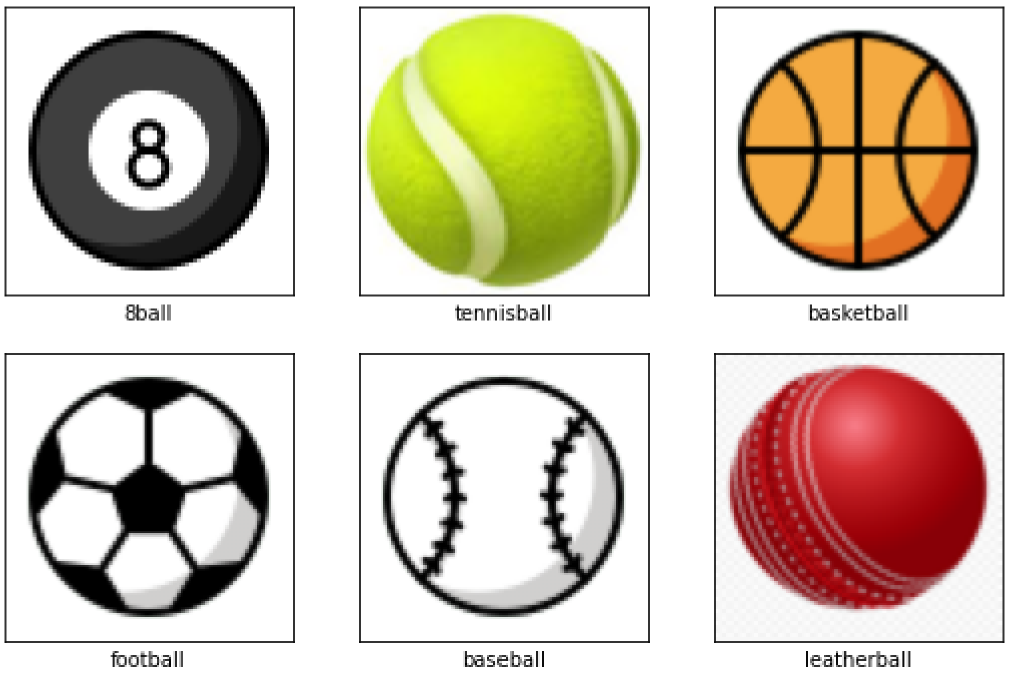
```
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.4->tensorflow
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.4->tensor
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorbo
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=2.4->tensorflow==
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard~=
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: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6.3->
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: 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->tenso
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: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorb
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: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensor
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: 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=68718 sha256=08aa31f12f9f954dd51a777fb6c25
  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
    Found existing installation: tensorflow 2.7.0
    Uninstalling tensorflow-2.7.0:
      Successfully uninstalled tensorflow-2.7.0
Successfully installed flatbuffers-1.12 gast-0.3.3 grpcio-1.32.0 h5py-2.10.0 tensorflow-2.4.0 tensorflow-estimator-2.4.0 typi
```

```
# 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
```

```
# emojis actually using in this project
emojis = {
    0: {'name': '8ball', 'file': '1F3B1.png'},
    1: {'name': 'tennisball', 'file': 'tennisball.png'},
    2: {'name': 'basketball', 'file': '1F3C0.png'},
    3: {'name': 'football', 'file': '26BD.png'},
    4: {'name': 'baseball', 'file': '26BE.png'},
    5: {'name': 'leatherball', 'file': '192950.png'}
}

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

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



Create Examples

```
# loads emojis and assign a key - image for each class in emojis dictonary
for class_id, values in emojis.items():
    png_file = Image.open(os.path.join('emojis', 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])
    emojis[class_id]['image'] = new_file
```

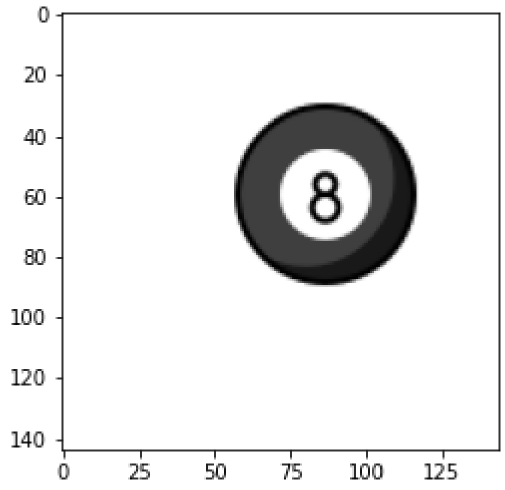
```
emojis
```

```
{0: {'file': '1F3B1.png',
  'image': <PIL.Image.Image image mode=RGB size=72x72 at 0x7F96EE4CF190>,
  'name': '8ball'},
 1: {'file': 'tennisball.png',
  'image': <PIL.Image.Image image mode=RGB size=72x72 at 0x7F96EE4CF090>,
  'name': 'tennisball'},
 2: {'file': '1F3C0.png',
  'image': <PIL.Image.Image image mode=RGB size=72x72 at 0x7F96EE4CF4D0>,
  'name': 'basketball'},
 3: {'file': '26BD.png',
  'image': <PIL.Image.Image image mode=RGB size=72x72 at 0x7F96EE4CFD50>,
  'name': 'football'},
 4: {'file': '26BE.png',
  'image': <PIL.Image.Image image mode=RGB size=72x72 at 0x7F96EE4CF610>,
  'name': 'baseball'},
 5: {'file': '192950.png',
  'image': <PIL.Image.Image image mode=RGB size=72x72 at 0x7F96EE4CFDD0>,
  '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(emojis[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 0x7f96ed038890>
```



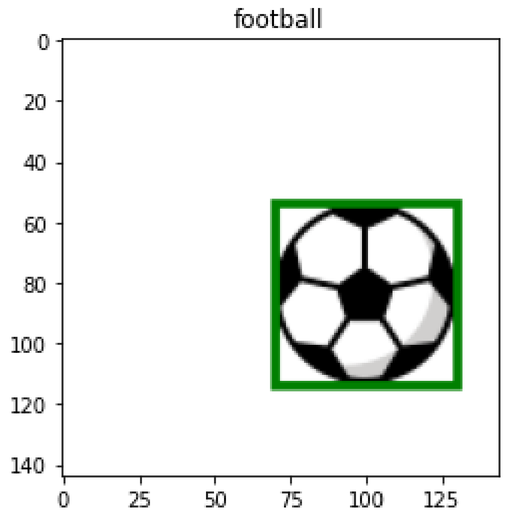
Plot Bounding Boxes

```
#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(emojis[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(emojis[class_id]['name'])
plt.show()
```



Model

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

x = input_

# we will have totoal 5 convolutional blocks
for i in range(0,5):
  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_6"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
image (InputLayer)              [(None, 144, 144, 3) 0
__________________________________________________________________________________________________
conv2d_30 (Conv2D)              (None, 142, 142, 16) 448         image[0][0]
__________________________________________________________________________________________________
batch_normalization_30 (BatchNo (None, 142, 142, 16) 64          conv2d_30[0][0]
__________________________________________________________________________________________________
max_pooling2d_30 (MaxPooling2D) (None, 71, 71, 16)   0           batch_normalization_30[0][0]
__________________________________________________________________________________________________
conv2d_31 (Conv2D)              (None, 69, 69, 32)   4640        max_pooling2d_30[0][0]
__________________________________________________________________________________________________
batch_normalization_31 (BatchNo (None, 69, 69, 32)   128         conv2d_31[0][0]
__________________________________________________________________________________________________
max_pooling2d_31 (MaxPooling2D) (None, 34, 34, 32)   0           batch_normalization_31[0][0]
__________________________________________________________________________________________________
conv2d_32 (Conv2D)              (None, 32, 32, 64)   18496       max_pooling2d_31[0][0]
__________________________________________________________________________________________________
batch_normalization_32 (BatchNo (None, 32, 32, 64)   256         conv2d_32[0][0]
__________________________________________________________________________________________________
max_pooling2d_32 (MaxPooling2D) (None, 16, 16, 64)   0           batch_normalization_32[0][0]
__________________________________________________________________________________________________
conv2d_33 (Conv2D)              (None, 14, 14, 128)  73856       max_pooling2d_32[0][0]
__________________________________________________________________________________________________
batch_normalization_33 (BatchNo (None, 14, 14, 128)  512         conv2d_33[0][0]
__________________________________________________________________________________________________
max_pooling2d_33 (MaxPooling2D) (None, 7, 7, 128)    0           batch_normalization_33[0][0]
__________________________________________________________________________________________________
conv2d_34 (Conv2D)              (None, 5, 5, 256)    295168      max_pooling2d_33[0][0]
__________________________________________________________________________________________________
batch_normalization_34 (BatchNo (None, 5, 5, 256)    1024        conv2d_34[0][0]
__________________________________________________________________________________________________
max_pooling2d_34 (MaxPooling2D) (None, 2, 2, 256)    0           batch_normalization_34[0][0]
__________________________________________________________________________________________________
flatten_6 (Flatten)             (None, 1024)         0           max_pooling2d_34[0][0]
__________________________________________________________________________________________________
dense_6 (Dense)                 (None, 256)          262400      flatten_6[0][0]
__________________________________________________________________________________________________
class_out (Dense)               (None, 6)            1542        dense_6[0][0]
__________________________________________________________________________________________________
box_out (Dense)                 (None, 2)            514         dense_6[0][0]
==================================================================================================
Total params: 659,048
Trainable params: 658,056
Non-trainable params: 992
```

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

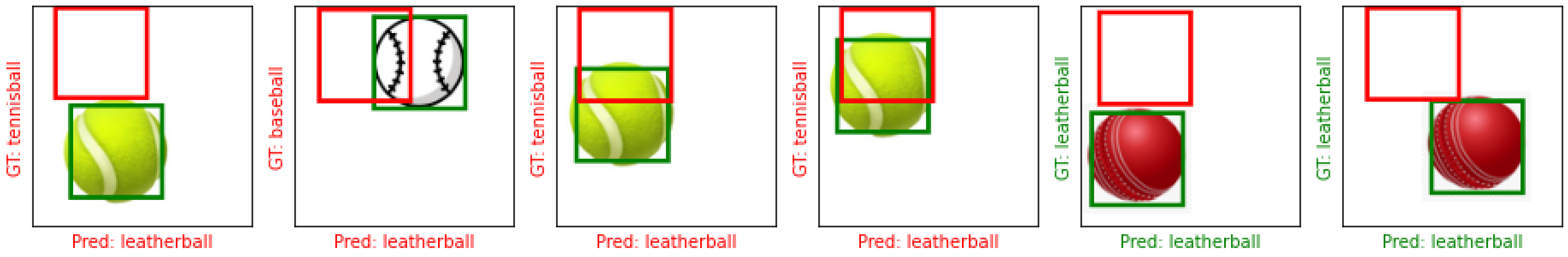
```
model.compile(
    # specify the loss for fifferent outputs
    loss={
        'class_out': 'categorical_crossentropy', # or classifaction output
        'box_out': 'mse' #for regression output
    },
    optimizer = tf.keras.optimizers.Adam(learning_rate=1e-3),

    #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 = emojis[np.argmax(y[0])]['name']
  pred_class_name = emojis[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)
```

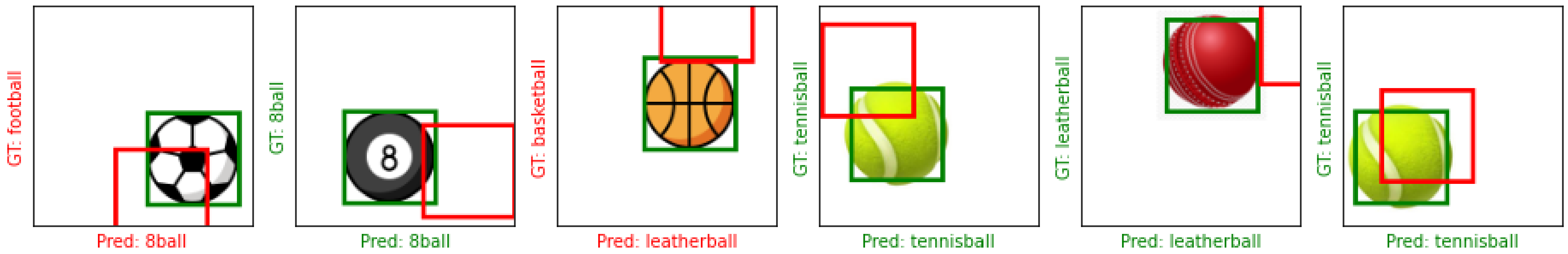


```
# create a custom call back
class ShowTestImages(tf.keras.callbacks.Callback): # to customize Callback class
  def on_epoch_end(self, epoch, logs=None): # on epoch end during the training, run test funtion
    test(self.model)
```

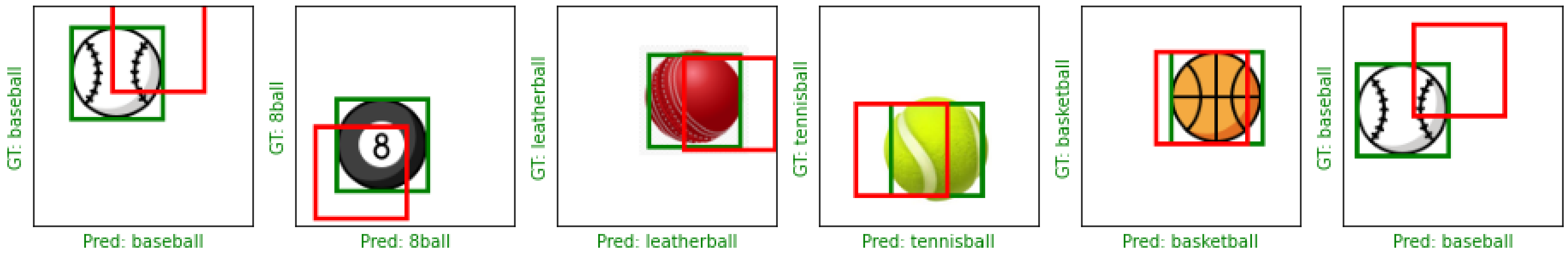
Model Training

```
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)
    ]
)
```

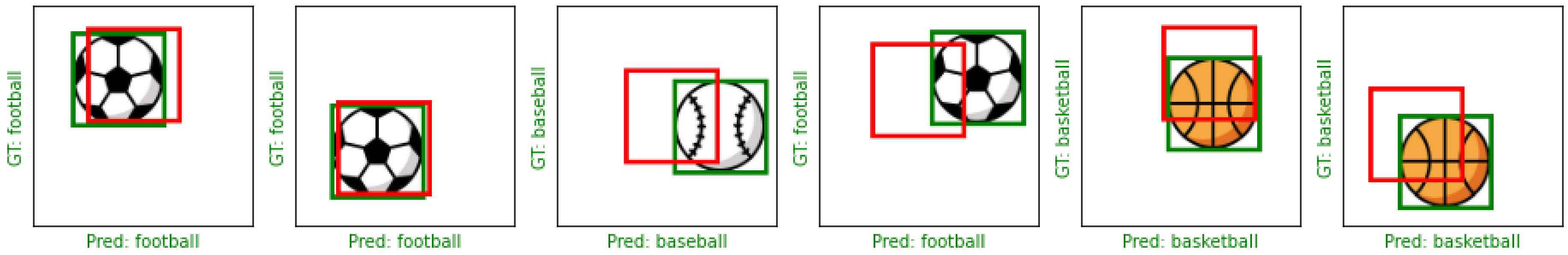
```
Epoch 1/100
500/500 [==============================] - 14s 24ms/step - loss: 1.4389 - class_out_loss: 0.3602 - box_out_loss: 1.0787 - class_
```



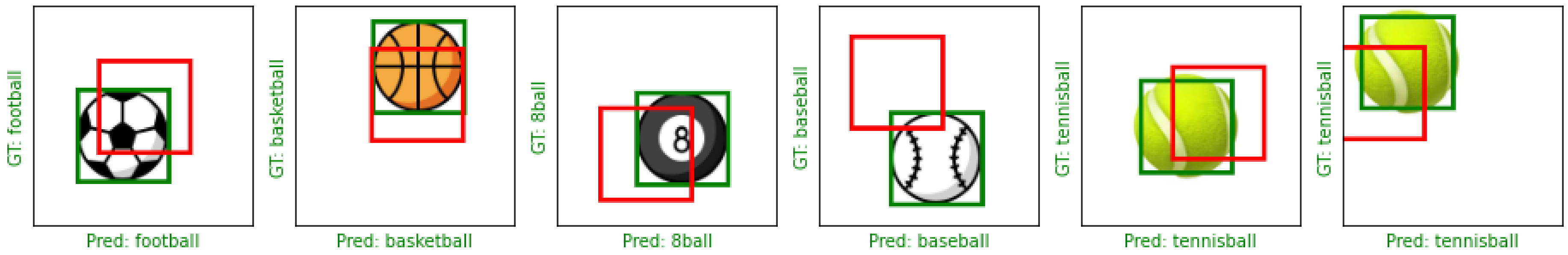
```
Epoch 2/100
500/500 [==============================] - 13s 25ms/step - loss: 0.0239 - class_out_loss: 0.0036 - box_out_loss: 0.0203 - class_
```



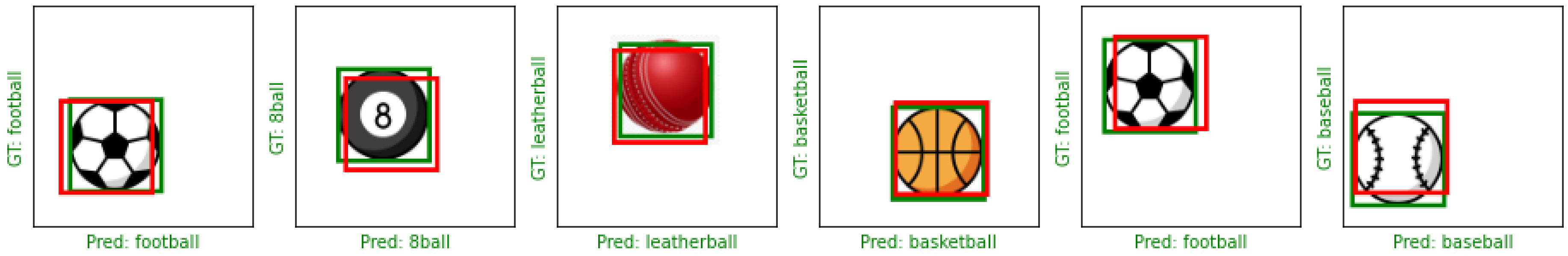
```
Epoch 3/100
500/500 [==============================] - 12s 25ms/step - loss: 0.0133 - class_out_loss: 0.0036 - box_out_loss: 0.0097 - class_
```



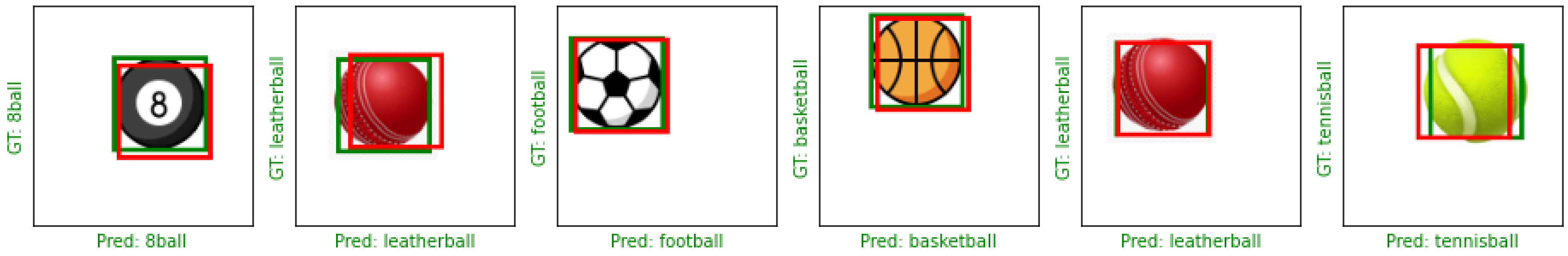
```
Epoch 4/100
500/500 [==============================] - 12s 24ms/step - loss: 0.0376 - class_out_loss: 0.0233 - box_out_loss: 0.0143 - class_
```



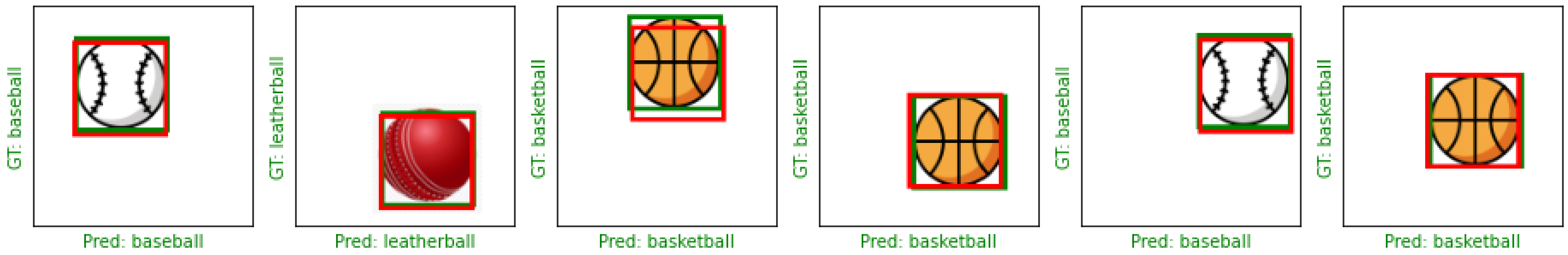
```
Epoch 5/100
500/500 [==============================] - 13s 25ms/step - loss: 0.0049 - class_out_loss: 8.1411e-04 - box_out_loss: 0.0041 - cl
```



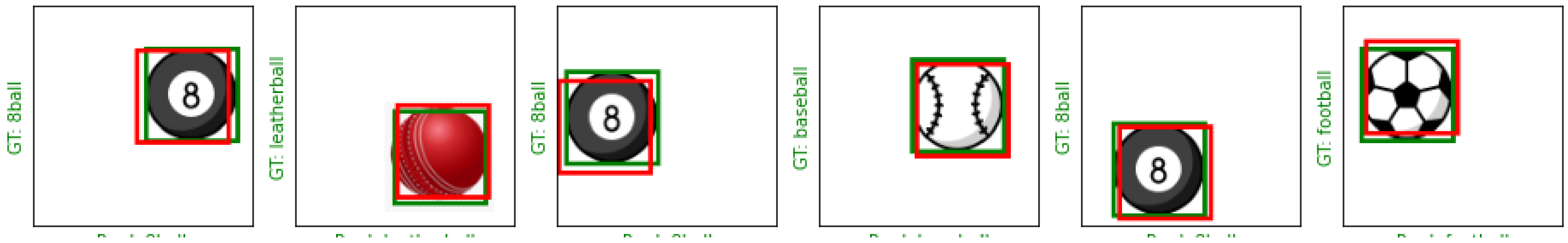
```
Epoch 6/100
500/500 [==============================] - 12s 25ms/step - loss: 0.0026 - class_out_loss: 3.2887e-04 - box_out_loss: 0.0023 - cl
```



```
Epoch 7/100
500/500 [==============================] - 13s 25ms/step - loss: 0.0236 - class_out_loss: 0.0210 - box_out_loss: 0.0026 - class_
```



```
Epoch 8/100
500/500 [==============================] - 13s 25ms/step - loss: 0.0021 - class_out_loss: 3.7915e-04 - box_out_loss: 0.0017 - cl
```


GT: 8ball
GT: leatherball
GT: 8ball
GT: baseball
GT: 8ball
GT: football