Coursera Peer-graded Assignment: Deep Learning and Reinforcement Learning

Objective: Object Classification and Localization

For an object classification and localization problem, we start off using the same network we saw in image classification.

So, we have an image as an input, which goes through a ConvNet that results in a vector of features fed to a softmax to classify the object (for example with 4 classes for pedestrians/cars/bike/background).

Now, if we want to localize those objects in the image as well, we change the neural network to have a few more output units that encompass a bounding box.

In particular, we add four more numbers, which identify the x and y coordinates of the lower left corner and upper right corner of the box (xmin, ymin, xmax, ymax).

Dataset: The Oxford-IIIT Pet Dataset

A 37 category pet dataset with roughly 200 images for each class.

The images have a large variations in scale, pose and lighting.

All images have an associated ground truth annotation of breed, head ROI, and pixel level trimap segmentation.

https://www.robots.ox.ac.uk/~vgg/data/pets/

Omkar M Parkhi and Andrea Vedaldi and Andrew Zisserman and C. V. Jawahar

In [1]:
```
#import wget
```

```
#url='https://www.robots.ox.ac.uk/~vgg/data/pets/data/images.tar.gz'
#wget.download(url,'images.tar.gz')
#url='https://www.robots.ox.ac.uk/~vgg/data/pets/data/annotations.tar.gz'
#wget.download(url,'annotations.tar.gz')
```

In [2]:
```
#images_tar=tarfile.open('images.tar.gz')
#images_tar.extractall('./Oxford_Data')
#images_tar.close()
```

In [3]:
```
#annotations_tar=tarfile.open('annotations.tar.gz')
#annotations_tar.extractall('./Oxford_Data')
#annotations_tar.close()
```

Importing necessary libraries

In [4]:
```
import os
from PIL import Image,ImageOps
import numpy as np
from collections import namedtuple
import matplotlib.pyplot as plt
from matplotlib.patches import Rectangle
import xml.etree.ElementTree as ET
import pandas as pd
import tensorflow as tf
from tensorflow.keras.applications.xception import Xception,preprocess_input
from tensorflow.keras.models import Model,Sequential
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout, Flatten
import tarfile
```

Directory Structure

In [5]:
```
os.getcwd()
```

Out[5]:
```
'C:\\Users\\Asus\\Downloads'
```

In [6]:
```
os.listdir(os.path.join(os.getcwd(),'Oxford_Data'))
```

Out[6]:
```
['annotations', 'images']
```

In [7]:
```
len(os.listdir(os.path.join(os.getcwd(),'Oxford_Data\\images')))
```

Out[7]:
```
7392
```

In [8]:
```
os.listdir(os.path.join(os.getcwd(),'Oxford_Data\\annotations'))
```

Out[8]:
```
['._trimaps',
 'list.txt',
 'README',
 'test.txt',
 'trainval.txt',
 'trimaps',
 'xmls']
```

xmls folder contains bounding box attributes corresponding to trainval data

In [9]:
```
all_images_path=os.path.join(os.getcwd(),'Oxford_Data\\images')
all_xmls_path=os.path.join(os.getcwd(),'Oxford_Data\\annotations\\xmls')
```

'process_image(args...)': to rescale the image and corresponding bounding box

In [10]:
```
Bounding_Box=namedtuple('BoundingBox',['xmin','ymin','xmax','ymax'])

def process_image(path_to_image,bounding_box=None):

    target_size=(224,224)
    img=Image.open(path_to_image)

    # Padding image to make it Square

    width,height=img.size
    h_pad=0
    w_pad=0
    bonus_h_pad=0
```

```
    bonus_w_pad=0
    pix_diff=abs(width-height)
    if width>height:
        h_pad=pix_diff//2
        bonus_h_pad=pix_diff%2
    elif height>width:
        w_pad=pix_diff//2
        bonus_w_pad=pix_diff%2
    img=ImageOps.expand(img,(w_pad,h_pad,w_pad+bonus_w_pad,h_pad+bonus_h_pad))

    # Resizing the image

    width,height=img.size
    img=img.resize((target_size[0],target_size[1]))
    img_data=np.array(img.getdata()).reshape(target_size[0],target_size[1],3)

    # Rescaling Bounding Box Dimensions

    if bounding_box is not None:

        width_scaling_factor=target_size[0]/width
        height_scaling_factor=target_size[1]/height

        new_xmin=bounding_box.xmin+w_pad
        new_xmin*=width_scaling_factor

        new_xmax=bounding_box.xmax+w_pad
        new_xmax*=width_scaling_factor

        new_ymin=bounding_box.ymin+h_pad
        new_ymin*=height_scaling_factor

        new_ymax=bounding_box.ymax+h_pad
        new_ymax*=height_scaling_factor

        return (img_data,Bounding_Box(new_xmin,new_ymin,new_xmax,new_ymax))

    return (img_data,None)
```

'get_bounding_box(args...)':to get bounding box dimensions from xml file

In [11]:

```
def get_bounding_box(path_to_xml):

    tree=ET.parse(path_to_xml)
    root=tree.getroot()
    xmin=int(root.find('./object/bndbox/xmin').text)
    ymin=int(root.find('./object/bndbox/ymin').text)
    xmax=int(root.find('./object/bndbox/xmax').text)
    ymax=int(root.find('./object/bndbox/ymax').text)
    bnd_box=Bounding_Box(xmin,ymin,xmax,ymax)

    return bnd_box
```

'image_with_bndbox(args...)':to show an image with true and/or predicted bounded box

In [12]:

```
def image_with_bndbox(image_data,true_bndbox=None,pred_bndbox=None):

    fig,ax=plt.subplots(1)
    ax.imshow(image_data)

    # Drawing True Bounding Box over image

    if true_bndbox is not None:

        anchor_x=true_bndbox.xmin
        anchor_y=true_bndbox.ymin
        r1=Rectangle((anchor_x,anchor_y),width=true_bndbox.xmax-true_bndbox.xmin,height=true_bndbox.ymax-true_bndbox.ymin,color='g
        ax.add_patch(r1)

    # Drawing Predicted Bounding Box over image

    if pred_bndbox is not None:

        r2=Rectangle((pred_bndbox.xmin,pred_bndbox.ymin),width=pred_bndbox.xmax-pred_bndbox.xmin,height=pred_bndbox.ymax-pred_bndb
        ax.add_patch(r2)

    # Calculating IOU

    if true_bndbox is not None and pred_bndbox is not None:

        xmin_inter=max(true_bndbox.xmin,pred_bndbox.xmin)
```

```
        ymin_inter=max(true_bndbox.ymin,pred_bndbox.ymin)
        xmax_inter=min(true_bndbox.xmax,pred_bndbox.xmax)
        ymax_inter=min(true_bndbox.ymax,pred_bndbox.ymax)
        true_bndbox_area=(true_bndbox.xmax-true_bndbox.xmin+1)*(true_bndbox.ymax-true_bndbox.ymin+1)
        pred_bndbox_area=(pred_bndbox.xmax-pred_bndbox.xmin+1)*(pred_bndbox.ymax-pred_bndbox.ymin+1)
        inter_area=max(0,xmax_inter-xmin_inter+1)*max(0,ymax_inter-ymin_inter+1)
        iou=(inter_area)/float(true_bndbox_area+pred_bndbox_area-inter_area)
        print('IOU:',iou)

    plt.show()
```

'final_image(args...)':acts as pipeline function to process and display image with boundng box

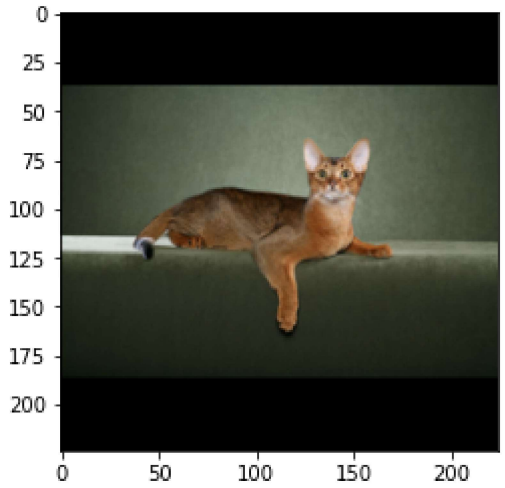
In [13]:
```
def final_image(image_name,pred_bndbox=None):

    image_path=os.path.join(all_images_path,image_name+'.jpg')
    xml_path=os.path.join(all_xmls_path,image_name+'.xml')
    true_bndbox=get_bounding_box(xml_path)
    img,true_bndbox=process_image(image_path,true_bndbox)
    image_with_bndbox(img,true_bndbox,pred_bndbox)
```

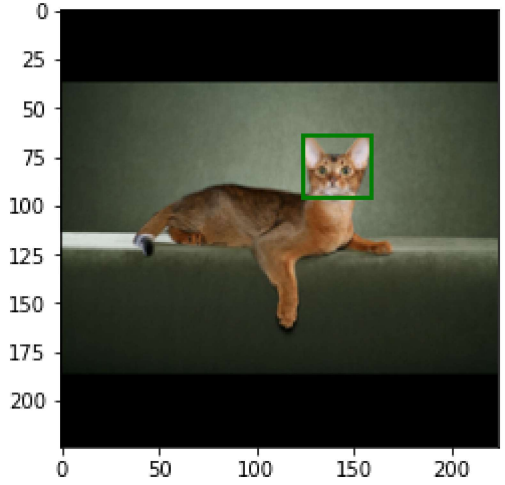
In [14]:
```
img,_=process_image('C:\\Users\\Asus\\Downloads\\Oxford_Data\\images\\Abyssinian_1.jpg')
```

In [15]:
```
plt.imshow(img)
```

Out[15]:
```
<matplotlib.image.AxesImage at 0x293db273fa0>
```



In [16]:
```
final_image('Abyssinian_1')
```



Processing Training and Testing Data

Training

In [17]:
```
df_train=pd.read_csv(os.path.join(os.getcwd(),'Oxford_Data\\annotations\\trainval.txt'),sep=' ',names=['Image','ID','Species','Bre
```

In [18]:
```
df_train.head()
```

Out[18]:

	Image	ID	Species	Breed
0	Abyssinian_100	1	1	1
1	Abyssinian_101	1	1	1
2	Abyssinian_102	1	1	1
3	Abyssinian_103	1	1	1
4	Abyssinian_104	1	1	1

In [19]:
```
df_train['Species']=df_train['Species']-1
```

In [20]:
```
print('No. of Cats:',df_train['Species'].value_counts()[0])
print('No. of Dogs:',df_train['Species'].value_counts()[1])
```

```
No. of Cats: 1188
No. of Dogs: 2492
```

In [21]:
```
print('No. of Cat Species:',len(df_train.groupby('Species')['Breed'].unique()[0]))
print('No. of Dog Species:',len(df_train.groupby('Species')['Breed'].unique()[1]))
```

```
No. of Cat Species: 12
No. of Dog Species: 25
```

In [22]:
```
train_tuple_df=np.array(df_train[['Image','Species']])
train_tuple_df
```

Out[22]:
```
array([['Abyssinian_100', 0],
       ['Abyssinian_101', 0],
       ['Abyssinian_102', 0],
       ...,
       ['yorkshire_terrier_189', 1],
       ['yorkshire_terrier_18', 1],
       ['yorkshire_terrier_190', 1]], dtype=object)
```

'process_train_data(args...)':to process and prepare training data

In [23]:
```
def process_train_data(df):

    data=list()

    for ele in df:

        image_name=ele[0]
        image_class=ele[1]

        if image_name+'.xml' not in os.listdir(all_xmls_path):

            continue

        else:

            image_path=os.path.join(all_images_path,image_name+'.jpg')
            xml_path=os.path.join(all_xmls_path,image_name+'.xml')
            true_bndbox=get_bounding_box(xml_path)
            img,true_bndbox=process_image(image_path,true_bndbox)
            data.append((image_name,img,true_bndbox,image_class))

    print('No. of Elements processed:',len(data))

    return np.array(data)
```

In [24]:
```
train_data=process_train_data(train_tuple_df)
```

```
No. of Elements processed: 3671
C:\Users\Asus\AppData\Local\Temp/ipykernel_37160/3957214975.py:24: VisibleDeprecationWarning: Creating an ndarray from ragged nest
ed sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you mean
t to do this, you must specify 'dtype=object' when creating the ndarray.
  return np.array(data)
```

train_data[0]-->image_name

train_data[1]-->img

train_data[2]-->true_bndbox

train_data[3]-->image_class

Testing

In [25]:
```
df_test=pd.read_csv(os.path.join(os.getcwd(),'Oxford_Data\\annotations\\test.txt'),sep=' ',names=['Image','ID','Species','Breed'])
```

In [26]:
```
df_test.head()
```

Out[26]:

	Image	ID	Species	Breed
0	Abyssinian_201	1	1	1
1	Abyssinian_202	1	1	1
2	Abyssinian_204	1	1	1
3	Abyssinian_205	1	1	1
4	Abyssinian_206	1	1	1

In [27]:
```
df_test['Species']=df_test['Species']-1
```

In [28]:
```
print('No. of Cats:',df_test['Species'].value_counts()[0])
print('No. of Dogs:',df_test['Species'].value_counts()[1])
```

```
No. of Cats: 1182
No. of Dogs: 2486
```

In [29]:
```
print('No. of Cat Species:',len(df_test.groupby('Species')['Breed'].unique()[0]))
print('No. of Dog Species:',len(df_test.groupby('Species')['Breed'].unique()[1]))
```

```
No. of Cat Species: 12
No. of Dog Species: 25
```

In [30]:
```
test_tuple_df=np.array(df_test[['Image','Species']])
test_tuple_df
```

```
array([['Abyssinian_201', 0],
```

Out[30]:
```
['Abyssinian_202', 0],
['Abyssinian_204', 0],
...,
['yorkshire_terrier_98', 1],
['yorkshire_terrier_99', 1],
['yorkshire_terrier_9', 1]], dtype=object)
```

'process_test_data(args...)':to process and prepare test data

In [31]:
```
def process_test_data(df):

    data=list()

    for ele in df:

        image_name=ele[0]
        image_class=ele[1]
        image_path=os.path.join(all_images_path,image_name+'.jpg')
        img,_=process_image(image_path,None)
        data.append((image_name,img,None,image_class))

    print('No. of Elements processed:',len(data))

    return np.array(data)
```

In [32]:
```
test_data=process_test_data(test_tuple_df)
```

```
No. of Elements processed: 3668
C:\Users\Asus\AppData\Local\Temp/ipykernel_37160/2614160404.py:15: VisibleDeprecationWarning: Creating an ndarray from ragged nest
ed sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you mean
t to do this, you must specify 'dtype=object' when creating the ndarray.
  return np.array(data)
```

test_data[0]-->image_name

test_data[1]-->img

test_data[2]-->None

test_data[3]-->image_class

In [33]:
```
# Shuffling Training Data
np.random.shuffle(train_data)
```

Splitting Data into Training and Validation Sets

In [34]:
```
val_data=train_data[:int(len(train_data)*0.2),:]
train_data=train_data[int(len(train_data)*0.2):,:]
```

In [35]:
```
x_train = []
y_class_train = []
y_box_train = []
x_validation = []
y_class_validation = []
y_box_validation = []
```

In [36]:
```
for img in train_data[:,1]:
    x_train.append(preprocess_input(img))

for box in train_data[:,2]:
    y_box_train.append(box)

for species in train_data[:,3]:
    y_class_train.append(species)
```

In [37]:
```
for img in val_data[:,1]:
    x_validation.append(preprocess_input(img))

for box in val_data[:,2]:
    y_box_validation.append(box)

for species in val_data[:,3]:
    y_class_validation.append(species)
```

Preparing Test Set

In [38]:
```
x_test=[]
for img in test_data[:,1]:
    x_test.append(img)
```

Casting Data to numpy array

In [39]:
```
x_train = np.array(x_train)
y_class_train = np.array(y_class_train)
y_box_train = np.array(y_box_train)
x_validation = np.array(x_validation)
y_class_validation = np.array(y_class_validation)
y_box_validation = np.array(y_box_validation)
x_test = np.array(x_test)
```

Transfer Learning using Xception Model on ImageNet weights

In [40]:
```
base_model=Xception(weights='imagenet',include_top=False,input_shape=(224,224,3))
```

In [41]:
```
# Freezing 125 Layers out of total 132 Layers
print('Total no. of layers:',len(base_model.layers))
for x in range(125):
    base_model.layers[x].trainable=False
print('No. of frozen layers:',125)
```

```
Total no. of layers: 132
No. of frozen layers: 125
```

In [42]:
```
frozen_Xception=Model(inputs=[base_model.input],outputs=[base_model.output])
```

In [43]:
```
# Classification Output
classification_output = GlobalAveragePooling2D()(frozen_Xception.output)
classification_output = Dense(units=1, activation='sigmoid')(classification_output)

# Bounding Box Params Output
```

```
localization_output = Flatten()(frozen_Xception.output)
localization_output = Dense(units=4, activation='relu')(localization_output)

model = Model(inputs=[frozen_Xception.input], outputs=[classification_output, localization_output])

model.summary()
```

```
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 block1_conv1 (Conv2D)          (None, 111, 111, 32  864         ['input_1[0][0]']
                                )

 block1_conv1_bn (BatchNormaliz  (None, 111, 111, 32  128        ['block1_conv1[0][0]']
 ation)                         )

 block1_conv1_act (Activation)  (None, 111, 111, 32  0           ['block1_conv1_bn[0][0]']
                                )

 block1_conv2 (Conv2D)          (None, 109, 109, 64  18432       ['block1_conv1_act[0][0]']
                                )

 block1_conv2_bn (BatchNormaliz  (None, 109, 109, 64  256        ['block1_conv2[0][0]']
 ation)                         )

 block1_conv2_act (Activation)  (None, 109, 109, 64  0           ['block1_conv2_bn[0][0]']
                                )

 block2_sepconv1 (SeparableConv  (None, 109, 109, 12  8768       ['block1_conv2_act[0][0]']
 2D)                            8)

 block2_sepconv1_bn (BatchNorma  (None, 109, 109, 12  512        ['block2_sepconv1[0][0]']
 lization)                      8)

 block2_sepconv2_act (Activatio  (None, 109, 109, 12  0          ['block2_sepconv1_bn[0][0]']
 n)                             8)

 block2_sepconv2 (SeparableConv  (None, 109, 109, 12  17536      ['block2_sepconv2_act[0][0]']
 2D)                            8)
```

```
block2_sepconv2_bn (BatchNorma  (None, 109, 109, 12  512        ['block2_sepconv2[0][0]']
lization)                      8)

conv2d (Conv2D)                (None, 55, 55, 128)  8192        ['block1_conv2_act[0][0]']

block2_pool (MaxPooling2D)     (None, 55, 55, 128)  0           ['block2_sepconv2_bn[0][0]']

batch_normalization (BatchNorm  (None, 55, 55, 128)  512        ['conv2d[0][0]']
alization)

add (Add)                      (None, 55, 55, 128)  0           ['block2_pool[0][0]',
                                                                 'batch_normalization[0][0]']

block3_sepconv1_act (Activatio  (None, 55, 55, 128)  0          ['add[0][0]']
n)

block3_sepconv1 (SeparableConv  (None, 55, 55, 256)  33920      ['block3_sepconv1_act[0][0]']
2D)

block3_sepconv1_bn (BatchNorma  (None, 55, 55, 256)  1024       ['block3_sepconv1[0][0]']
lization)

block3_sepconv2_act (Activatio  (None, 55, 55, 256)  0          ['block3_sepconv1_bn[0][0]']
n)

block3_sepconv2 (SeparableConv  (None, 55, 55, 256)  67840      ['block3_sepconv2_act[0][0]']
2D)

block3_sepconv2_bn (BatchNorma  (None, 55, 55, 256)  1024       ['block3_sepconv2[0][0]']
lization)

conv2d_1 (Conv2D)              (None, 28, 28, 256)  32768       ['add[0][0]']

block3_pool (MaxPooling2D)     (None, 28, 28, 256)  0           ['block3_sepconv2_bn[0][0]']

batch_normalization_1 (BatchNo  (None, 28, 28, 256)  1024       ['conv2d_1[0][0]']
rmalization)

add_1 (Add)                    (None, 28, 28, 256)  0           ['block3_pool[0][0]',
                                                                 'batch_normalization_1[0][0]']

block4_sepconv1_act (Activatio  (None, 28, 28, 256)  0          ['add_1[0][0]']
n)
```

```
block4_sepconv1 (SeparableConv  (None, 28, 28, 728)  188672     ['block4_sepconv1_act[0][0]']
2D)

block4_sepconv1_bn (BatchNorma  (None, 28, 28, 728)  2912       ['block4_sepconv1[0][0]']
lization)

block4_sepconv2_act (Activatio  (None, 28, 28, 728)  0          ['block4_sepconv1_bn[0][0]']
n)

block4_sepconv2 (SeparableConv  (None, 28, 28, 728)  536536     ['block4_sepconv2_act[0][0]']
2D)

block4_sepconv2_bn (BatchNorma  (None, 28, 28, 728)  2912       ['block4_sepconv2[0][0]']
lization)

conv2d_2 (Conv2D)               (None, 14, 14, 728)  186368     ['add_1[0][0]']

block4_pool (MaxPooling2D)      (None, 14, 14, 728)  0          ['block4_sepconv2_bn[0][0]']

batch_normalization_2 (BatchNo  (None, 14, 14, 728)  2912       ['conv2d_2[0][0]']
rmalization)

add_2 (Add)                     (None, 14, 14, 728)  0          ['block4_pool[0][0]',
                                                                 'batch_normalization_2[0][0]']

block5_sepconv1_act (Activatio  (None, 14, 14, 728)  0          ['add_2[0][0]']
n)

block5_sepconv1 (SeparableConv  (None, 14, 14, 728)  536536     ['block5_sepconv1_act[0][0]']
2D)

block5_sepconv1_bn (BatchNorma  (None, 14, 14, 728)  2912       ['block5_sepconv1[0][0]']
lization)

block5_sepconv2_act (Activatio  (None, 14, 14, 728)  0          ['block5_sepconv1_bn[0][0]']
n)

block5_sepconv2 (SeparableConv  (None, 14, 14, 728)  536536     ['block5_sepconv2_act[0][0]']
2D)

block5_sepconv2_bn (BatchNorma  (None, 14, 14, 728)  2912       ['block5_sepconv2[0][0]']
lization)
```

```
block5_sepconv3_act (Activatio  (None, 14, 14, 728)  0          ['block5_sepconv2_bn[0][0]']
n)

block5_sepconv3 (SeparableConv  (None, 14, 14, 728)  536536     ['block5_sepconv3_act[0][0]']
2D)

block5_sepconv3_bn (BatchNorma  (None, 14, 14, 728)  2912       ['block5_sepconv3[0][0]']
lization)

add_3 (Add)                    (None, 14, 14, 728)  0          ['block5_sepconv3_bn[0][0]',
                                                                'add_2[0][0]']

block6_sepconv1_act (Activatio  (None, 14, 14, 728)  0          ['add_3[0][0]']
n)

block6_sepconv1 (SeparableConv  (None, 14, 14, 728)  536536     ['block6_sepconv1_act[0][0]']
2D)

block6_sepconv1_bn (BatchNorma  (None, 14, 14, 728)  2912       ['block6_sepconv1[0][0]']
lization)

block6_sepconv2_act (Activatio  (None, 14, 14, 728)  0          ['block6_sepconv1_bn[0][0]']
n)

block6_sepconv2 (SeparableConv  (None, 14, 14, 728)  536536     ['block6_sepconv2_act[0][0]']
2D)

block6_sepconv2_bn (BatchNorma  (None, 14, 14, 728)  2912       ['block6_sepconv2[0][0]']
lization)

block6_sepconv3_act (Activatio  (None, 14, 14, 728)  0          ['block6_sepconv2_bn[0][0]']
n)

block6_sepconv3 (SeparableConv  (None, 14, 14, 728)  536536     ['block6_sepconv3_act[0][0]']
2D)

block6_sepconv3_bn (BatchNorma  (None, 14, 14, 728)  2912       ['block6_sepconv3[0][0]']
lization)

add_4 (Add)                    (None, 14, 14, 728)  0          ['block6_sepconv3_bn[0][0]',
                                                                'add_3[0][0]']

block7_sepconv1_act (Activatio  (None, 14, 14, 728)  0          ['add_4[0][0]']
n)
```

```
block7_sepconv1 (SeparableConv  (None, 14, 14, 728)  536536     ['block7_sepconv1_act[0][0]']
2D)

block7_sepconv1_bn (BatchNorma  (None, 14, 14, 728)  2912       ['block7_sepconv1[0][0]']
lization)

block7_sepconv2_act (Activatio  (None, 14, 14, 728)  0          ['block7_sepconv1_bn[0][0]']
n)

block7_sepconv2 (SeparableConv  (None, 14, 14, 728)  536536     ['block7_sepconv2_act[0][0]']
2D)

block7_sepconv2_bn (BatchNorma  (None, 14, 14, 728)  2912       ['block7_sepconv2[0][0]']
lization)

block7_sepconv3_act (Activatio  (None, 14, 14, 728)  0          ['block7_sepconv2_bn[0][0]']
n)

block7_sepconv3 (SeparableConv  (None, 14, 14, 728)  536536     ['block7_sepconv3_act[0][0]']
2D)

block7_sepconv3_bn (BatchNorma  (None, 14, 14, 728)  2912       ['block7_sepconv3[0][0]']
lization)

add_5 (Add)                    (None, 14, 14, 728)  0          ['block7_sepconv3_bn[0][0]',
                                                                'add_4[0][0]']

block8_sepconv1_act (Activatio  (None, 14, 14, 728)  0          ['add_5[0][0]']
n)

block8_sepconv1 (SeparableConv  (None, 14, 14, 728)  536536     ['block8_sepconv1_act[0][0]']
2D)

block8_sepconv1_bn (BatchNorma  (None, 14, 14, 728)  2912       ['block8_sepconv1[0][0]']
lization)

block8_sepconv2_act (Activatio  (None, 14, 14, 728)  0          ['block8_sepconv1_bn[0][0]']
n)

block8_sepconv2 (SeparableConv  (None, 14, 14, 728)  536536     ['block8_sepconv2_act[0][0]']
2D)

block8_sepconv2_bn (BatchNorma  (None, 14, 14, 728)  2912       ['block8_sepconv2[0][0]']
```

```
lization)

block8_sepconv3_act (Activatio  (None, 14, 14, 728)  0          ['block8_sepconv2_bn[0][0]']
n)

block8_sepconv3 (SeparableConv  (None, 14, 14, 728)  536536     ['block8_sepconv3_act[0][0]']
2D)

block8_sepconv3_bn (BatchNorma  (None, 14, 14, 728)  2912       ['block8_sepconv3[0][0]']
lization)

add_6 (Add)                     (None, 14, 14, 728)  0          ['block8_sepconv3_bn[0][0]',
                                                                 'add_5[0][0]']

block9_sepconv1_act (Activatio  (None, 14, 14, 728)  0          ['add_6[0][0]']
n)

block9_sepconv1 (SeparableConv  (None, 14, 14, 728)  536536     ['block9_sepconv1_act[0][0]']
2D)

block9_sepconv1_bn (BatchNorma  (None, 14, 14, 728)  2912       ['block9_sepconv1[0][0]']
lization)

block9_sepconv2_act (Activatio  (None, 14, 14, 728)  0          ['block9_sepconv1_bn[0][0]']
n)

block9_sepconv2 (SeparableConv  (None, 14, 14, 728)  536536     ['block9_sepconv2_act[0][0]']
2D)

block9_sepconv2_bn (BatchNorma  (None, 14, 14, 728)  2912       ['block9_sepconv2[0][0]']
lization)

block9_sepconv3_act (Activatio  (None, 14, 14, 728)  0          ['block9_sepconv2_bn[0][0]']
n)

block9_sepconv3 (SeparableConv  (None, 14, 14, 728)  536536     ['block9_sepconv3_act[0][0]']
2D)

block9_sepconv3_bn (BatchNorma  (None, 14, 14, 728)  2912       ['block9_sepconv3[0][0]']
lization)

add_7 (Add)                     (None, 14, 14, 728)  0          ['block9_sepconv3_bn[0][0]',
                                                                 'add_6[0][0]']
```

```
block10_sepconv1_act (Activati  (None, 14, 14, 728)  0          ['add_7[0][0]']
on)

block10_sepconv1 (SeparableCon  (None, 14, 14, 728)  536536     ['block10_sepconv1_act[0][0]']
v2D)

block10_sepconv1_bn (BatchNorm  (None, 14, 14, 728)  2912       ['block10_sepconv1[0][0]']
alization)

block10_sepconv2_act (Activati  (None, 14, 14, 728)  0          ['block10_sepconv1_bn[0][0]']
on)

block10_sepconv2 (SeparableCon  (None, 14, 14, 728)  536536     ['block10_sepconv2_act[0][0]']
v2D)

block10_sepconv2_bn (BatchNorm  (None, 14, 14, 728)  2912       ['block10_sepconv2[0][0]']
alization)

block10_sepconv3_act (Activati  (None, 14, 14, 728)  0          ['block10_sepconv2_bn[0][0]']
on)

block10_sepconv3 (SeparableCon  (None, 14, 14, 728)  536536     ['block10_sepconv3_act[0][0]']
v2D)

block10_sepconv3_bn (BatchNorm  (None, 14, 14, 728)  2912       ['block10_sepconv3[0][0]']
alization)

add_8 (Add)                    (None, 14, 14, 728)  0          ['block10_sepconv3_bn[0][0]',
                                                                'add_7[0][0]']

block11_sepconv1_act (Activati  (None, 14, 14, 728)  0          ['add_8[0][0]']
on)

block11_sepconv1 (SeparableCon  (None, 14, 14, 728)  536536     ['block11_sepconv1_act[0][0]']
v2D)

block11_sepconv1_bn (BatchNorm  (None, 14, 14, 728)  2912       ['block11_sepconv1[0][0]']
alization)

block11_sepconv2_act (Activati  (None, 14, 14, 728)  0          ['block11_sepconv1_bn[0][0]']
on)

block11_sepconv2 (SeparableCon  (None, 14, 14, 728)  536536     ['block11_sepconv2_act[0][0]']
v2D)
```

```
block11_sepconv2_bn (BatchNorm  (None, 14, 14, 728)  2912       ['block11_sepconv2[0][0]']
alization)

block11_sepconv3_act (Activati  (None, 14, 14, 728)  0          ['block11_sepconv2_bn[0][0]']
on)

block11_sepconv3 (SeparableCon  (None, 14, 14, 728)  536536     ['block11_sepconv3_act[0][0]']
v2D)

block11_sepconv3_bn (BatchNorm  (None, 14, 14, 728)  2912       ['block11_sepconv3[0][0]']
alization)

add_9 (Add)                    (None, 14, 14, 728)  0          ['block11_sepconv3_bn[0][0]',
                                                                'add_8[0][0]']

block12_sepconv1_act (Activati  (None, 14, 14, 728)  0          ['add_9[0][0]']
on)

block12_sepconv1 (SeparableCon  (None, 14, 14, 728)  536536     ['block12_sepconv1_act[0][0]']
v2D)

block12_sepconv1_bn (BatchNorm  (None, 14, 14, 728)  2912       ['block12_sepconv1[0][0]']
alization)

block12_sepconv2_act (Activati  (None, 14, 14, 728)  0          ['block12_sepconv1_bn[0][0]']
on)

block12_sepconv2 (SeparableCon  (None, 14, 14, 728)  536536     ['block12_sepconv2_act[0][0]']
v2D)

block12_sepconv2_bn (BatchNorm  (None, 14, 14, 728)  2912       ['block12_sepconv2[0][0]']
alization)

block12_sepconv3_act (Activati  (None, 14, 14, 728)  0          ['block12_sepconv2_bn[0][0]']
on)

block12_sepconv3 (SeparableCon  (None, 14, 14, 728)  536536     ['block12_sepconv3_act[0][0]']
v2D)

block12_sepconv3_bn (BatchNorm  (None, 14, 14, 728)  2912       ['block12_sepconv3[0][0]']
alization)

add_10 (Add)                   (None, 14, 14, 728)  0          ['block12_sepconv3_bn[0][0]',
```

```
                                                                 'add_9[0][0]']

block13_sepconv1_act (Activati  (None, 14, 14, 728)  0          ['add_10[0][0]']
on)

block13_sepconv1 (SeparableCon  (None, 14, 14, 728)  536536     ['block13_sepconv1_act[0][0]']
v2D)

block13_sepconv1_bn (BatchNorm  (None, 14, 14, 728)  2912       ['block13_sepconv1[0][0]']
alization)

block13_sepconv2_act (Activati  (None, 14, 14, 728)  0          ['block13_sepconv1_bn[0][0]']
on)

block13_sepconv2 (SeparableCon  (None, 14, 14, 1024  752024     ['block13_sepconv2_act[0][0]']
v2D)                           )

block13_sepconv2_bn (BatchNorm  (None, 14, 14, 1024  4096       ['block13_sepconv2[0][0]']
alization)                     )

conv2d_3 (Conv2D)               (None, 7, 7, 1024)   745472      ['add_10[0][0]']

block13_pool (MaxPooling2D)     (None, 7, 7, 1024)   0           ['block13_sepconv2_bn[0][0]']

batch_normalization_3 (BatchNo  (None, 7, 7, 1024)  4096        ['conv2d_3[0][0]']
rmalization)

add_11 (Add)                    (None, 7, 7, 1024)   0           ['block13_pool[0][0]',
                                                                 'batch_normalization_3[0][0]']

block14_sepconv1 (SeparableCon  (None, 7, 7, 1536)  1582080     ['add_11[0][0]']
v2D)

block14_sepconv1_bn (BatchNorm  (None, 7, 7, 1536)  6144        ['block14_sepconv1[0][0]']
alization)

block14_sepconv1_act (Activati  (None, 7, 7, 1536)  0           ['block14_sepconv1_bn[0][0]']
on)

block14_sepconv2 (SeparableCon  (None, 7, 7, 2048)  3159552     ['block14_sepconv1_act[0][0]']
v2D)

block14_sepconv2_bn (BatchNorm  (None, 7, 7, 2048)  8192        ['block14_sepconv2[0][0]']
alization)
```

```
block14_sepconv2_act (Activati  (None, 7, 7, 2048)  0           ['block14_sepconv2_bn[0][0]']
on)

global_average_pooling2d (Glob  (None, 2048)        0           ['block14_sepconv2_act[0][0]']
alAveragePooling2D)

flatten (Flatten)              (None, 100352)       0           ['block14_sepconv2_act[0][0]']

dense (Dense)                  (None, 1)            2049        ['global_average_pooling2d[0][0]'
                                                                ]

dense_1 (Dense)                (None, 4)            401412      ['flatten[0][0]']

==================================================================================================
Total params: 21,264,941
Trainable params: 5,152,261
Non-trainable params: 16,112,680
```

Fitting the Model

In [44]:

```
model.compile(optimizer='adam', metrics=['accuracy'],loss=['binary_crossentropy', 'mse'] )
history=model.fit(x_train, [y_class_train, y_box_train], validation_data=(x_validation, [y_class_validation, y_box_validation]),ep
history
```

```
Epoch 1/20
92/92 [==============================] - 433s 5s/step - loss: 991.3856 - dense_loss: 0.2650 - dense_1_loss: 991.1207 - dense_accur
acy: 0.9231 - dense_1_accuracy: 0.8032 - val_loss: 534.2087 - val_dense_loss: 0.1049 - val_dense_1_loss: 534.1038 - val_dense_accu
racy: 0.9837 - val_dense_1_accuracy: 0.8760
Epoch 2/20
92/92 [==============================] - 413s 4s/step - loss: 154.1748 - dense_loss: 0.0995 - dense_1_loss: 154.0753 - dense_accur
acy: 0.9813 - dense_1_accuracy: 0.9105 - val_loss: 266.8718 - val_dense_loss: 0.0687 - val_dense_1_loss: 266.8030 - val_dense_accu
racy: 0.9932 - val_dense_1_accuracy: 0.8951
Epoch 3/20
92/92 [==============================] - 402s 4s/step - loss: 94.8177 - dense_loss: 0.0827 - dense_1_loss: 94.7350 - dense_accurac
y: 0.9823 - dense_1_accuracy: 0.9213 - val_loss: 143.6665 - val_dense_loss: 0.0593 - val_dense_1_loss: 143.6072 - val_dense_accura
cy: 0.9905 - val_dense_1_accuracy: 0.9210
Epoch 4/20
92/92 [==============================] - 408s 4s/step - loss: 71.0263 - dense_loss: 0.0681 - dense_1_loss: 70.9582 - dense_accurac
y: 0.9857 - dense_1_accuracy: 0.9282 - val_loss: 123.4953 - val_dense_loss: 0.0523 - val_dense_1_loss: 123.4429 - val_dense_accura
cy: 0.9905 - val_dense_1_accuracy: 0.9196
```

```
Epoch 5/20
92/92 [==============================] - 410s 4s/step - loss: 70.5779 - dense_loss: 0.0622 - dense_1_loss: 70.5157 - dense_accurac
y: 0.9823 - dense_1_accuracy: 0.9367 - val_loss: 174.5141 - val_dense_loss: 0.0489 - val_dense_1_loss: 174.4653 - val_dense_accura
cy: 0.9891 - val_dense_1_accuracy: 0.9251
Epoch 6/20
92/92 [==============================] - 383s 4s/step - loss: 48.7992 - dense_loss: 0.0612 - dense_1_loss: 48.7380 - dense_accurac
y: 0.9850 - dense_1_accuracy: 0.9404 - val_loss: 109.3161 - val_dense_loss: 0.0458 - val_dense_1_loss: 109.2703 - val_dense_accura
cy: 0.9918 - val_dense_1_accuracy: 0.9251
Epoch 7/20
92/92 [==============================] - 412s 4s/step - loss: 47.0993 - dense_loss: 0.0508 - dense_1_loss: 47.0485 - dense_accurac
y: 0.9901 - dense_1_accuracy: 0.9435 - val_loss: 167.4933 - val_dense_loss: 0.0374 - val_dense_1_loss: 167.4559 - val_dense_accura
cy: 0.9918 - val_dense_1_accuracy: 0.9210
Epoch 8/20
92/92 [==============================] - 393s 4s/step - loss: 36.3960 - dense_loss: 0.0527 - dense_1_loss: 36.3432 - dense_accurac
y: 0.9840 - dense_1_accuracy: 0.9544 - val_loss: 130.7314 - val_dense_loss: 0.0401 - val_dense_1_loss: 130.6913 - val_dense_accura
cy: 0.9905 - val_dense_1_accuracy: 0.9087
Epoch 9/20
92/92 [==============================] - 402s 4s/step - loss: 42.6278 - dense_loss: 0.0477 - dense_1_loss: 42.5801 - dense_accurac
y: 0.9898 - dense_1_accuracy: 0.9537 - val_loss: 120.6181 - val_dense_loss: 0.0356 - val_dense_1_loss: 120.5824 - val_dense_accura
cy: 0.9905 - val_dense_1_accuracy: 0.9210
Epoch 10/20
92/92 [==============================] - 435s 5s/step - loss: 32.9885 - dense_loss: 0.0445 - dense_1_loss: 32.9440 - dense_accurac
y: 0.9888 - dense_1_accuracy: 0.9687 - val_loss: 130.0070 - val_dense_loss: 0.0424 - val_dense_1_loss: 129.9647 - val_dense_accura
cy: 0.9918 - val_dense_1_accuracy: 0.9169
Epoch 11/20
92/92 [==============================] - 385s 4s/step - loss: 27.2425 - dense_loss: 0.0402 - dense_1_loss: 27.2022 - dense_accurac
y: 0.9915 - dense_1_accuracy: 0.9663 - val_loss: 133.1241 - val_dense_loss: 0.0371 - val_dense_1_loss: 133.0870 - val_dense_accura
cy: 0.9877 - val_dense_1_accuracy: 0.9128
Epoch 12/20
92/92 [==============================] - 403s 4s/step - loss: 27.1508 - dense_loss: 0.0426 - dense_1_loss: 27.1082 - dense_accurac
y: 0.9908 - dense_1_accuracy: 0.9714 - val_loss: 116.0304 - val_dense_loss: 0.0352 - val_dense_1_loss: 115.9952 - val_dense_accura
cy: 0.9918 - val_dense_1_accuracy: 0.9155
Epoch 13/20
92/92 [==============================] - 399s 4s/step - loss: 25.4699 - dense_loss: 0.0387 - dense_1_loss: 25.4312 - dense_accurac
y: 0.9905 - dense_1_accuracy: 0.9694 - val_loss: 94.7238 - val_dense_loss: 0.0365 - val_dense_1_loss: 94.6873 - val_dense_accurac
y: 0.9905 - val_dense_1_accuracy: 0.9251
Epoch 14/20
92/92 [==============================] - 364s 4s/step - loss: 24.7988 - dense_loss: 0.0358 - dense_1_loss: 24.7630 - dense_accurac
y: 0.9911 - dense_1_accuracy: 0.9741 - val_loss: 90.3766 - val_dense_loss: 0.0353 - val_dense_1_loss: 90.3413 - val_dense_accurac
y: 0.9891 - val_dense_1_accuracy: 0.9223
Epoch 15/20
92/92 [==============================] - 420s 5s/step - loss: 22.4888 - dense_loss: 0.0375 - dense_1_loss: 22.4513 - dense_accurac
y: 0.9901 - dense_1_accuracy: 0.9741 - val_loss: 101.4216 - val_dense_loss: 0.0359 - val_dense_1_loss: 101.3857 - val_dense_accura
cy: 0.9891 - val_dense_1_accuracy: 0.9210
```

```
Epoch 16/20
92/92 [==============================] - 370s 4s/step - loss: 22.3997 - dense_loss: 0.0386 - dense_1_loss: 22.3611 - dense_accurac
y: 0.9901 - dense_1_accuracy: 0.9704 - val_loss: 105.7388 - val_dense_loss: 0.0334 - val_dense_1_loss: 105.7055 - val_dense_accura
cy: 0.9918 - val_dense_1_accuracy: 0.9237
Epoch 17/20
92/92 [==============================] - 408s 4s/step - loss: 24.1302 - dense_loss: 0.0343 - dense_1_loss: 24.0959 - dense_accurac
y: 0.9901 - dense_1_accuracy: 0.9741 - val_loss: 98.4875 - val_dense_loss: 0.0312 - val_dense_1_loss: 98.4564 - val_dense_accurac
y: 0.9918 - val_dense_1_accuracy: 0.9251
Epoch 18/20
92/92 [==============================] - 385s 4s/step - loss: 17.9329 - dense_loss: 0.0367 - dense_1_loss: 17.8962 - dense_accurac
y: 0.9891 - dense_1_accuracy: 0.9762 - val_loss: 85.9742 - val_dense_loss: 0.0379 - val_dense_1_loss: 85.9363 - val_dense_accurac
y: 0.9905 - val_dense_1_accuracy: 0.9210
Epoch 19/20
92/92 [==============================] - 479s 5s/step - loss: 17.0046 - dense_loss: 0.0340 - dense_1_loss: 16.9706 - dense_accurac
y: 0.9905 - dense_1_accuracy: 0.9765 - val_loss: 105.6739 - val_dense_loss: 0.0368 - val_dense_1_loss: 105.6371 - val_dense_accura
cy: 0.9891 - val_dense_1_accuracy: 0.9169
Epoch 20/20
92/92 [==============================] - 488s 5s/step - loss: 17.0071 - dense_loss: 0.0308 - dense_1_loss: 16.9762 - dense_accurac
y: 0.9922 - dense_1_accuracy: 0.9700 - val_loss: 111.0330 - val_dense_loss: 0.0313 - val_dense_1_loss: 111.0016 - val_dense_accura
cy: 0.9905 - val_dense_1_accuracy: 0.9169
```

Out[44]: <keras.callbacks.History at 0x293db9b5700>

In [45]:
```
history.history.keys()
```

Out[45]:
```
dict_keys(['loss', 'dense_loss', 'dense_1_loss', 'dense_accuracy', 'dense_1_accuracy', 'val_loss', 'val_dense_loss', 'val_dense_1_
loss', 'val_dense_accuracy', 'val_dense_1_accuracy'])
```

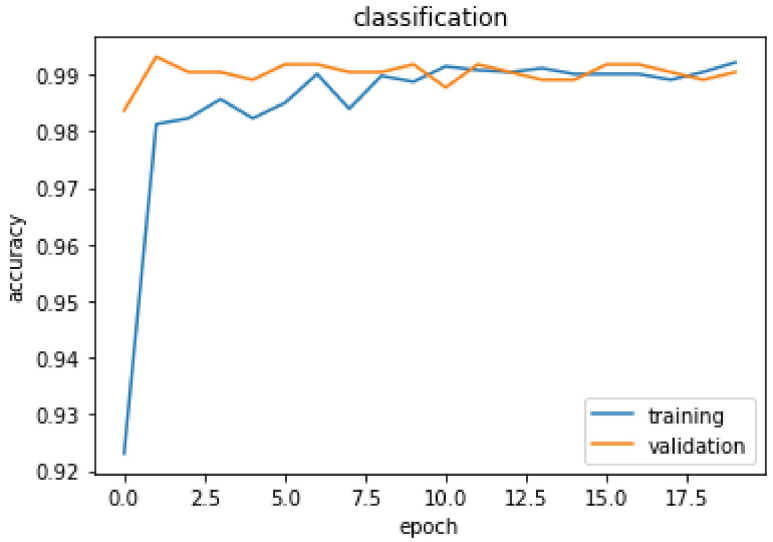
Training History Plot for Classification

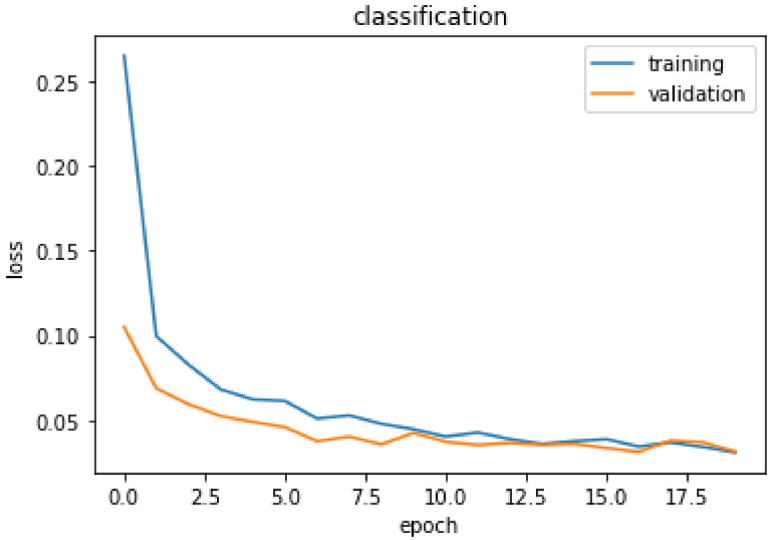
In [46]:
```
def plot_training_history(history, model):
    plt.plot(history.history['dense_accuracy'])
    plt.plot(history.history['val_dense_accuracy'])
    plt.title('classification')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['training', 'validation'], loc='best')
    plt.show()

    plt.plot(history.history['dense_loss'])
    plt.plot(history.history['val_dense_loss'])
    plt.title('classification')
```

```
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['training', 'validation'], loc='best')
    plt.show()

plot_training_history(history, model)
```





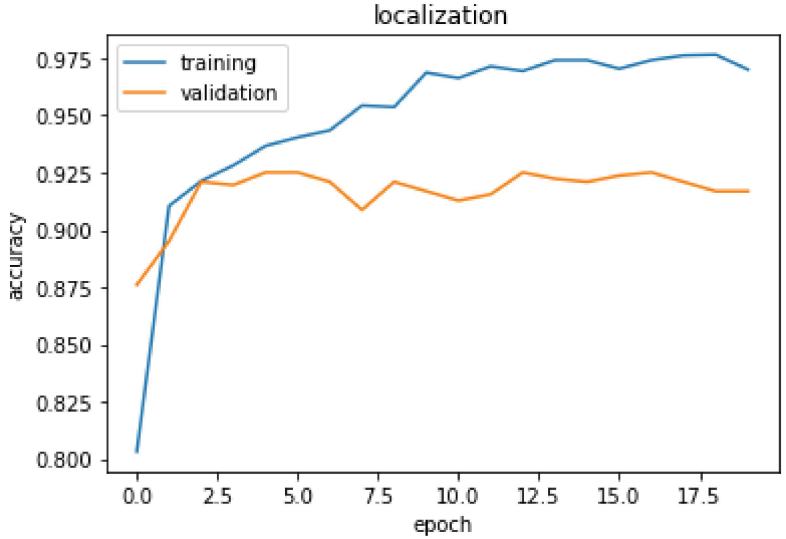
Training and Validation Accuracy are very close to each other for classification task, indicating ideal convergence

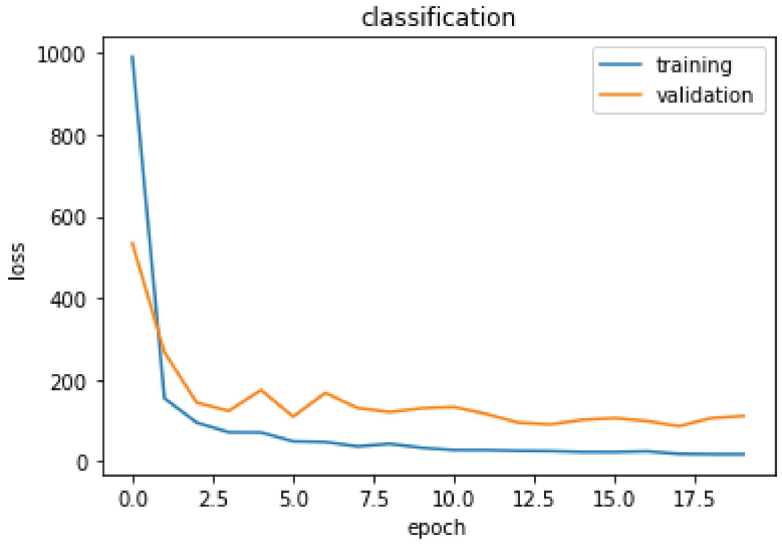
Training History Plot for Localization

In [47]:
```
def plot_training_history(history, model):
    plt.plot(history.history['dense_1_accuracy'])
    plt.plot(history.history['val_dense_1_accuracy'])
    plt.title('localization')
    plt.ylabel('accuracy')
    plt.xlabel('epoch')
    plt.legend(['training', 'validation'], loc='best')
    plt.show()

    plt.plot(history.history['dense_1_loss'])
    plt.plot(history.history['val_dense_1_loss'])
    plt.title('classification')
    plt.ylabel('loss')
    plt.xlabel('epoch')
    plt.legend(['training', 'validation'], loc='best')
    plt.show()

plot_training_history(history, model)
```



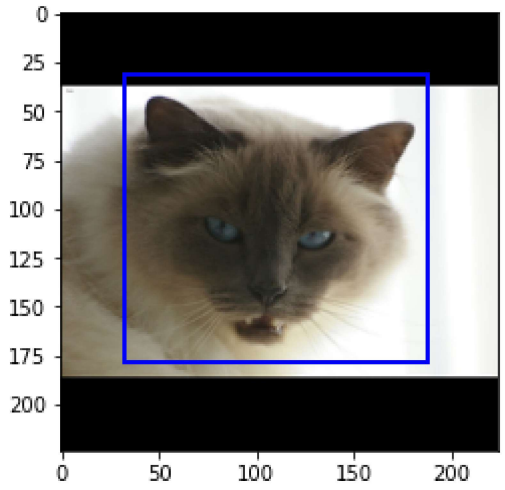


Training and Validation Accuracy are less close to each other for localization task, indicating some degree of overfitting

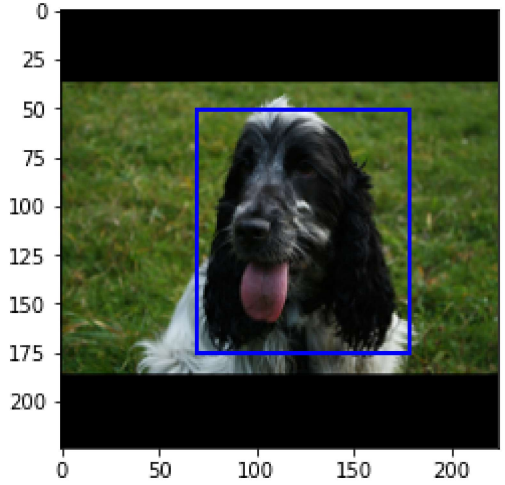
Testing Model on Unseen Data

In [56]:
```
for i in range(6):
    n=np.random.randint(len(x_test))
    pred = model.predict(np.array([preprocess_input(x_test[n])]))
    if pred[0][0]>0.5:
        print('it is a dog')
    else:
        print('it is a cat')
    image_with_bndbox(x_test[n],pred_bndbox=Bounding_Box(*pred[1][0]))
```

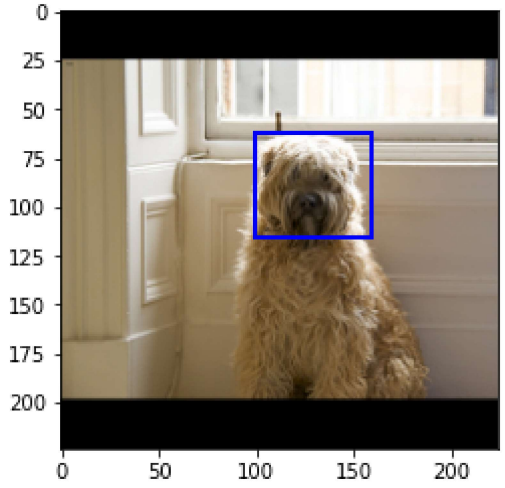
```
it is a cat
```



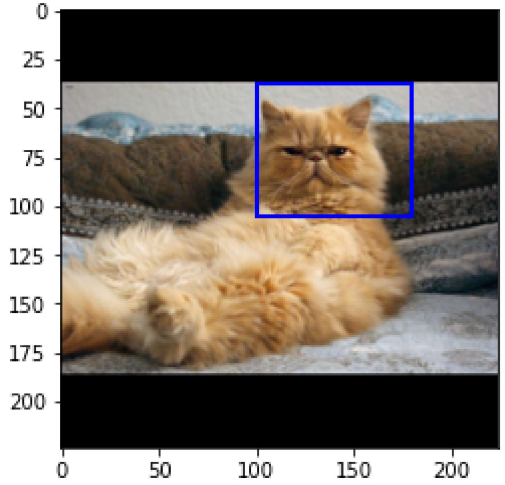

```
it is a dog
```



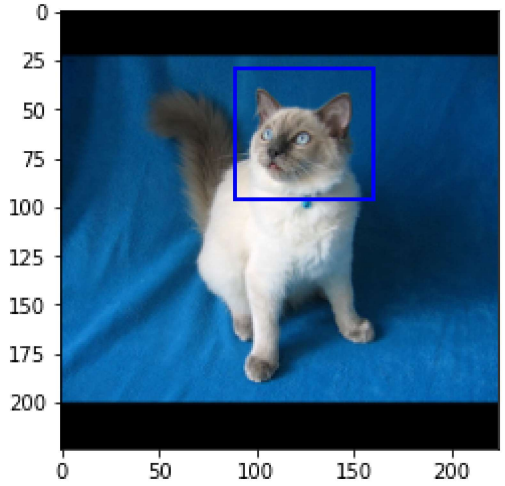

```
it is a dog
```



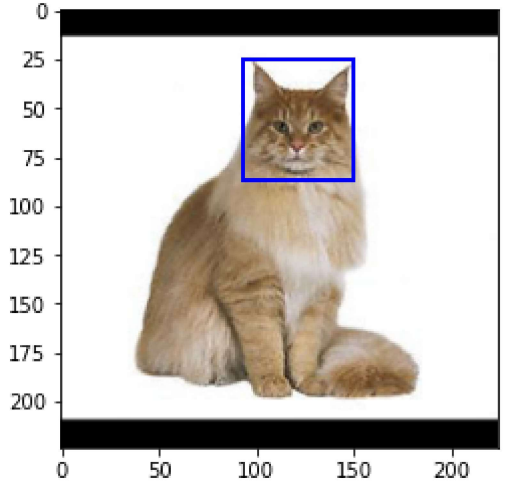

```
it is a cat
```




```
it is a cat
```




```
it is a cat
```



It can be clearly observed that the model shows promising results even on unseen data

Further Suggestions to improve model performance:

More varied data can be added to tackle the problem of overfitting

Data Augmentation can be performed

Freezing of different no. of layers can be tried for fine tuning

Different coordinate setup can be tried for localization problem

In [ ]: