

UNIVERSITY OF BUEA

P.O Box 63,
Buea, South West Region



REPUBLIC OF CAMEROON

PEACE-WORK-FATHERLAND

FACULTY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER ENGINEERING

COURSE CODE: CEF 444

COURSE TITLE: AI AND MACHINE LEARNING

TRAINING AN ANIMAL CLASSIFIER MODEL USING KERAS

GROUP MEMBERS

NDZO DANIEL UGHE (FE18A045)

OMBUCHUM MICAH ANJI (FE18A059)

CHI CLAUDETTE MAH (FE18A018)

BANTI JEANIFAR NAHBILA (FE18A015)

Course Instructor: **Dr SOP DEFFO LIONEL**

What is Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. It is user friendly, easily extensible, and works with python.

Overview of the project

This animal classifier is designed to classifier 5 different animals, that is; tiger, camel, sheep, pig and horse. The five different types of animals were downloaded in bulk separately. The images where then converted into a format that was suitable for the model. After the conversion, the images were divided into test and training dataset (90% was used for training and 10% used for testing). After some analysis, the model was compiled, trained, tested, and did predictions on some images.

Dataset

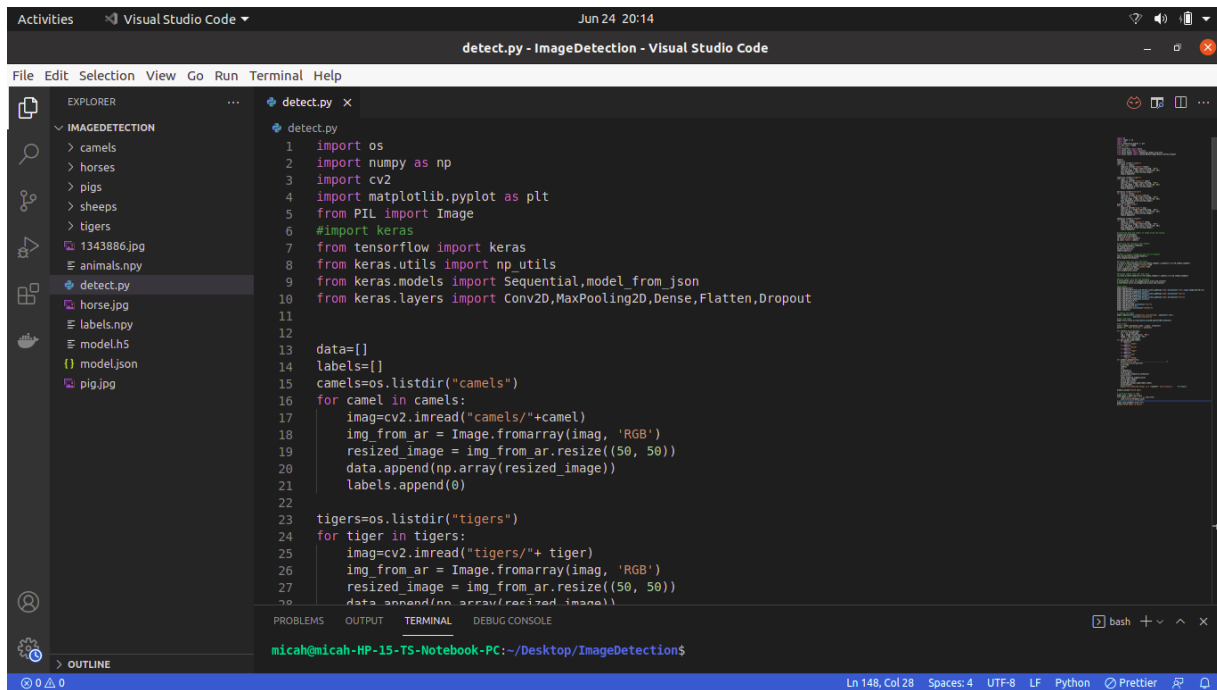
- 100 photos of tiger
- 100 photos of pigs
- 100 photos of sheep
- 90 photos of camel
- 90 photos of horse
- 90% of each animal was used for training and 10% for testing.

Libraries used

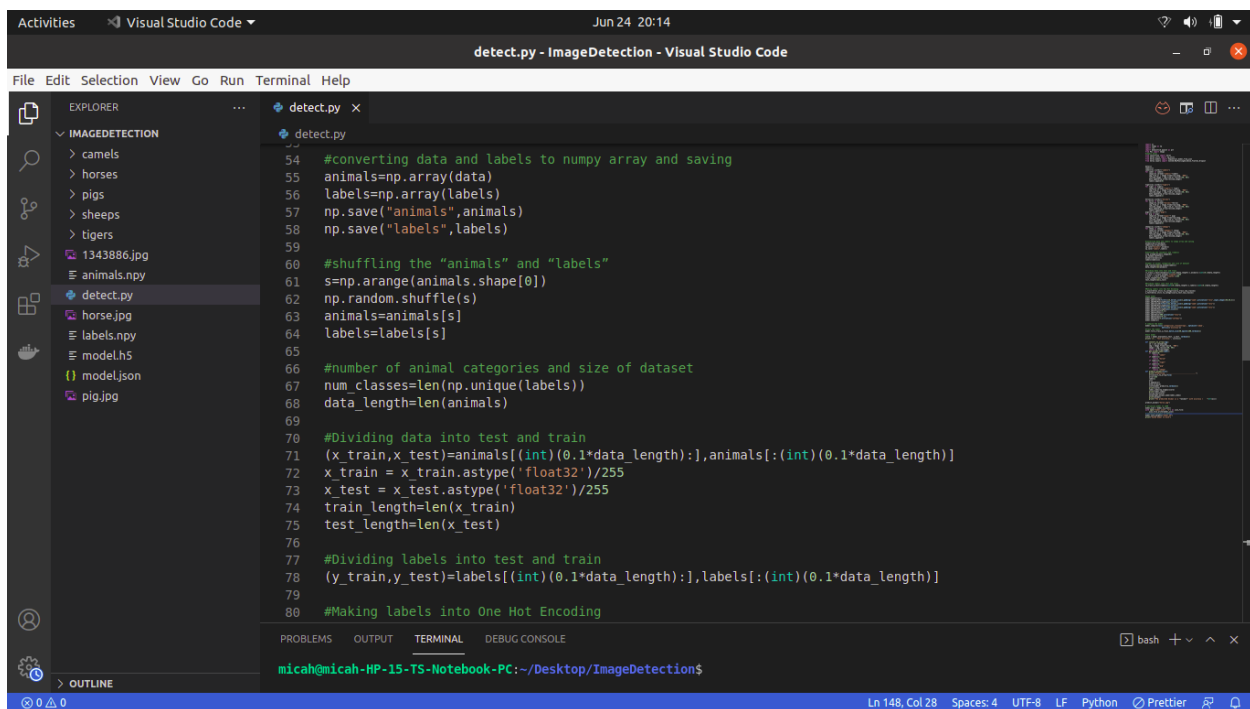
- Tensorflow (for building models. Keras runs on top of it)
- Keras(Developing and evaluating deep learning models)
- OpenCv (For computer vision)
- Numpy(For working with arrays)
- Pillow(For image manipulation)
- Matplotlib(Plotting library for pyhton)

Screenshots of the working classifier

Screenshots of the source code



```
1 import os
2 import numpy as np
3 import cv2
4 import matplotlib.pyplot as plt
5 from PIL import Image
6 #import keras
7 from tensorflow import keras
8 from keras.utils import np_utils
9 from keras.models import Sequential, Model from json
10 from keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout
11
12
13 data=[]
14 labels=[]
15 camels=os.listdir("camels")
16 for camel in camels:
17     img=cv2.imread("camels/"+camel)
18     img_from_ar = Image.fromarray(img, 'RGB')
19     resized_image = img_from_ar.resize((50, 50))
20     data.append(np.array(resized_image))
21     labels.append(0)
22
23 tigers=os.listdir("tigers")
24 for tiger in tigers:
25     img=cv2.imread("tigers/"+tiger)
26     img_from_ar = Image.fromarray(img, 'RGB')
27     resized_image = img_from_ar.resize((50, 50))
28     data.append(np.array(resized_image))
29     labels.append(1)
```



```
54 #converting data and labels to numpy array and saving
55 animals=np.array(data)
56 labels=np.array(labels)
57 np.save("animals",animals)
58 np.save("labels",labels)
59
60 #shuffling the "animals" and "labels"
61 s=np.arange(animals.shape[0])
62 np.random.shuffle(s)
63 animals=animals[s]
64 labels=labels[s]
65
66 #number of animal categories and size of dataset
67 num_classes=len(np.unique(labels))
68 data_length=len(animals)
69
70 #Dividing data into test and train
71 (x_train,x_test)=animals[(int)(0.1*data_length):],animals[::(int)(0.1*data_length)]
72 x_train = x_train.astype('float32')/255
73 x_test = x_test.astype('float32')/255
74 train_length=len(x_train)
75 test_length=len(x_test)
76
77 #Dividing labels into test and train
78 (y_train,y_test)=labels[(int)(0.1*data_length):],labels[::(int)(0.1*data_length)]
79
80 #Making labels into One Hot Encoding
```

```
83
84 #make model
85 model=Sequential()
86 model.add(Conv2D(filters=16,kernel_size=2,padding="same",activation="relu",input_shape=(50,50,3)))
87 model.add(MaxPooling2D(pool_size=2))
88 model.add(Conv2D(filters=32,kernel_size=2,padding="same",activation="relu"))
89 model.add(MaxPooling2D(pool_size=2))
90 model.add(Conv2D(filters=64,kernel_size=2,padding="same",activation="relu"))
91 model.add(MaxPooling2D(pool_size=2))
92 model.add(Dropout(0.2))
93 model.add(Flatten())
94 model.add(Dense(500,activation="relu"))
95 model.add(Dropout(0.2))
96 model.add(Dense(5,activation="softmax"))
97 model.summary()
98
99 # compile the model
100 model.compile(loss='categorical_crossentropy', optimizer='adam',
101               metrics=['accuracy'])
102 #train the model
103 model.fit(x_train,y_train,batch_size=50,epochs=100,verbose=1)
104
105 #test model
106 score = model.evaluate(x_test, y_test, verbose=1)
107 print('\n', 'Test accuracy:', score[1])
108
109 def convert_to_array(img):
110     im = cv2.imread(img)
111     img = Image.fromarray(im, 'RGB')
112     image = img.resize((50, 50))
113     return np.array(image)
114
115 def get_animal_name(label):
116     if label==0:
117         return "camel"
118     if label==1:
119         return "horse"
120     if label==2:
121         return "tiger"
122     if label==3:
123         return "pig"
124     if label==4:
125         return "sheep"
126
127 def predict_animal(file):
128     print("Predicting .....")
129     ar=convert_to_array(file)
130     ar=ar/255
131     label=1
132     a=[]
133     a.append(ar)
134     a=np.array(a)
135     score=model.predict(a,verbose=1)
136     print(score)
137     label_index=np.argmax(score)
```

```
108
109 def convert_to_array(img):
110     im = cv2.imread(img)
111     img = Image.fromarray(im, 'RGB')
112     image = img.resize((50, 50))
113     return np.array(image)
114
115 def get_animal_name(label):
116     if label==0:
117         return "camel"
118     if label==1:
119         return "horse"
120     if label==2:
121         return "tiger"
122     if label==3:
123         return "pig"
124     if label==4:
125         return "sheep"
126
127 def predict_animal(file):
128     print("Predicting .....")
129     ar=convert_to_array(file)
130     ar=ar/255
131     label=1
132     a=[]
133     a.append(ar)
134     a=np.array(a)
135     score=model.predict(a,verbose=1)
136     print(score)
137     label_index=np.argmax(score)
```

```
129 label = 1
130 a = []
131 a.append(ar)
132 a = np.array(a)
133 score = model.predict(a, verbose=1)
134 print(score)
135 label_index = np.argmax(score)
136 print(label_index)
137 acc = np.max(score)
138 animal = get_animal_name(label_index)
139 print(animal)
140 print("The predicted Animal is a "+animal+" with accuracy = "+str(acc))
141
142 predict_animal("horse.jpg")
143
144 # serialize model to JSON
145 model_json = model.to_json()
146 with open("model.json", "w") as json_file:
147     json_file.write(model_json)
148 # serialize weights to HDF5
149 model.save_weights("model.h5")
150 print("Saved model to disk")
```

micah@micah-HP-15-TS-Notebook-PC:~/Desktop/ImageDetection\$

Output of the code

```
9/9 [=====] - 1s 75ms/step - loss: 0.0031 - accuracy: 1.0000
Epoch 92/100
9/9 [=====] - 1s 76ms/step - loss: 0.0021 - accuracy: 1.0000
Epoch 93/100
9/9 [=====] - 1s 76ms/step - loss: 0.0117 - accuracy: 0.9935
Epoch 94/100
9/9 [=====] - 1s 77ms/step - loss: 0.0119 - accuracy: 1.0000
Epoch 95/100
9/9 [=====] - 1s 76ms/step - loss: 0.0106 - accuracy: 0.9990
Epoch 96/100
9/9 [=====] - 1s 76ms/step - loss: 0.0046 - accuracy: 0.9990
Epoch 97/100
9/9 [=====] - 1s 75ms/step - loss: 0.0108 - accuracy: 1.0000
Epoch 98/100
9/9 [=====] - 1s 78ms/step - loss: 0.0098 - accuracy: 0.9993
Epoch 99/100
9/9 [=====] - 1s 85ms/step - loss: 0.0101 - accuracy: 0.9971
Epoch 100/100
9/9 [=====] - 1s 77ms/step - loss: 0.0045 - accuracy: 1.0000
2/2 [=====] - 1s 9ms/step - loss: 1.5905 - accuracy: 0.6522

Test accuracy: 0.6521739363670349
Predicting .....
1/1 [=====] - 1s 586ms/step
[[4.2311193e-08 9.9996960e-01 3.1875363e-06 2.7126971e-05 2.8319913e-10]]
1
horse
The predicted Animal is a horse with accuracy = 0.9999696
Saved model to disk
micah@micah-HP-15-TS-Notebook-PC:~/Desktop/ImageDetection$
```

Conclusion

The model created is around 80% accurate because it was not fed with sufficient data. But it is still able to classify the different types of animals it was trained with.