DOG CAT CLASSIFICATION MODEL

REPORT

1

INTRODUCTION

This project implement dog cat classification using TensorFlow. TensorFlow is an end-to-end open source platform for machine learning. TensorFlow is a rich system for managing all aspects of a machine learning system; however, this class focuses on using a particular TensorFlow API to develop and train machine learning models. TensorFlow is an open-source machine learning framework, and Python is a popular computer programming language. Keras is another important concept used in this project.  Keras is a high-level neural network library that runs on top of TensorFlow. Both TensorFlow and Keras provide high-level APIs used for easily building and training models, but Keras is more user-friendly because it's built-in Python.

2

2

METHODOLOGY

In this project we are using TensorFlow to classify dogs and cats.

Before we can create and train a model, we must prepare the training data. To enter the image data into the model during training, we first have to load an image from disk and transform it into an array of bytes.

Here we are using a code that imports necessary libraries, defines the image directory path, retrieves image names from the directory, extracts labels from the names, encodes labels, imports additional libraries, initializes feature and label lists, loops through the image names to load, resize, and convert images to arrays, converts lists to NumPy arrays, one-hot encodes labels, imports train-test split functionality, splits the dataset into training and testing sets, further splits the training set into training and validation sets, defines a data augmentation pipeline, defines prediction layers using a dense layer with sotimax activation, loads a pre-trained ResNet50 model and sets it to non-trainable, defines input tensor, applies data augmentation and preprocesses input, passes input through the ResNet50 model, adds a dropout layer, connects prediction layers, creates the final model, compiles the model, trains it using training and validation data, extracts training and validation metrics, plots the training and validation curves, evaluates the model on the test set, predicts labels for the test set, and stores them in y\_pred variable

3

CODE

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import matplotlib.image as image

import glob

import os

images\_fp='./images'

image\_names = [os.path.basename(file) for file in glob.glob(os.path.join(images\_fp,'\*.jpg'))]

image\_names

labels = [' '.join(name.split(\_)[:-1:])for name in image\_names]

labels

def label\_encode(label):

if label == 'Abyssinian' : return 0

elif label == 'Bengal' : return 1

elif label == 'Birman' : return 2

elif label == 'Bombay' : return 3

elif label == 'British Shorthair' : return 4

elif label == 'Egyptian Mau' : return 5

elif label == 'american bulldog' : return 6

elif label == 'american pit bull terrier' : return 7

elif label == 'basset hound' : return 8

elif label == 'beagle' : return 9

elif label == 'boxer' : return 10

elif label == 'chihuahua' : return 11

elif label == 'english cocker spaniel' : return 12

elif label == 'english setter' : return 13

elif label == 'german shorthaired' : return 14

import tensorflow as tf

from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array

4

features = []

labels = []

IMAGE\_SIZE =(224,224)

for name in image\_names :

label = ' ',join(name.split('\_')[:-1:])

label\_encoded = label\_encode(label)

if label\_encoded != None:

img = load\_img(os.path.join(images\_fp, name))

img = tf.img.resize\_with\_pad(img\_to\_array(img,dtype='uint8'),\*IMAGE\_SIZE).numpy().astype('uint8')

image = np.array(img)

features.append(image)

labels.append(label\_encoded)

features\_array = np.array(features)

labels\_array = np.array(labels)

labels\_one\_hot = pd.get\_dummies(labels\_array)

4

from sklearn.model\_selection import train\_test\_split

#Train = 60%,vol = 20% and Test: 20%

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features\_array, labels\_one\_hot, test\_size = 0.2, random\_state =42)

#80%

X\_train, X-val< y\_train, y\_val = train\_test\_split(x\_train,y\_train<test\_size=0.25, random\_state = 1)

from tensorflow.keras import layers, Input, Model

from tensorflow.keras.models import Sequential

from tensorflow.keras.applications import ResNet50

from tensorflow.keras.applications.resnet50 import preprocess\_input as pp\_i

from tensorflow.keras.layers import RandomFlip,RandomRotation, Dense, Dropout

from tensorflow.keras.losses import CategoricalCrossentropy

from tensorflow.keras.optimizers import Adam

5

data\_augmentation = Sequential([RandomFlip("horizontal\_and\_vertical"),RandomRotation(0.2)])

prediction\_layers = Dense(16,activation = 'softmax')

resnet\_model = ResNet50(include\_top = False, pooling = 'avg', weights = 'imagenet')

resnet\_model.trainable = False

preprocess\_input = pp\_i

#Build model

inputs = Input(shape=(224,224,3))

x = data\_augmentation(inputs)

x = preprocess\_input(x)

x = resnet\_model(x,training=False)

x = Dropout(0.2)(x)

outputs = prediction\_layers(x)

model = Model(inputs,outputs)

model.compile(optimizer=Adam(), loss=CategoricalCrossentropy(), metrics = ['accuracy'])

model\_history = model.fit(x=X\_train, y=y\_train, validation\_data=(X\_val, y\_val),epochs=10)

acc = model\_history.history['accuracy']

val\_acc = model\_history.history['val\_accuracy']

loss = model\_history.history['loss']

val\_loss = model\_history.history['val\_loss']

epochs\_range = range(10)

plt.figure(figsize=(15,8))

plt.subplot(1,2,1)

plt.plot(epochs\_range,acc, label='Training accuracy')

plt.plot(epocs\_range,val\_acc,label='Validation accuracy')

plt.legend(loc='lower right')

plt.title('Training and Validation Accuracy')

plt.subplot(1,2,2)

plt.plot(epochs\_range,acc, label='Training loss')

6

plt.plot(epocs\_range,val\_acc,label='Validation loss')

plt.legend(loc='upper right')

plt.title('Training and Validation loss')

model.evaluate(X\_test, y\_test)

y\_pred = model.predict(X\_test)

y\_pred

7

CONCLUSION

The given code effectively classifies a number of dog and cat faces. This is very useful in large pet shops and pet adoption centres.

BIBLIOGRAPHY

* <https://www.1stop.ai/>
* <https://chat.openai.com/>
* <https://www.kaggle.com/competitions/dog-breed-identification/data>

7 7

8