Dog breed classification using convolutional neural networks

Deep Learning HNDDS 21.1F

Senuli Ratnayake (001)

Sitara Jayasundera (023)

Contents

Abstract

Convolutional Neural Networks(CNN) have been used for dog breed classification and detection. This model supports the classification of 10 dog breeds with a total of 50 images. Consider a dog image is chosen the model can classify and establish an analysis of the dog breed and finally return the estimated dog breed.

Objectives

This report aims to test the capabilities of pre-trained convolutional neural network models to detect and classify as well as to develop a new Convolutional Neural Networking (CNN) model to maximize the accuracy of the proposed CNN. First, we will take a look at what Convolutional Neural Networking is. One of the most popular deep neural networks is Convolutional Neural Networks(CNN/ConvNet), most commonly applied to analyze visual imagery. Convolutional Neural Networks are a special type of neural network that roughly imitates human vision where this was first developed and used around the 1980s. CNN consists of multiple layers of artificial neurons. An artificial neuron is a rough imitation of its biological counterparts which are mathematical functions that calculate the weighted sum of multiple inputs and outputs an activation value. When you input an image in a ConvNet, each layer generates several activation functions that are passed on to the next layer. The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges. Based on the activation map of the final convolution layer, the classification layer outputs a set of confidence scores (between zero and one) that specify how likely the image is to belong to a “class”. The downside of

convolutional neural networks is that sometimes datasets fail to detect objects when they see them under different lighting conditions and from new angles but there is no denying that CNN has caused a revolution in artificial intelligence.

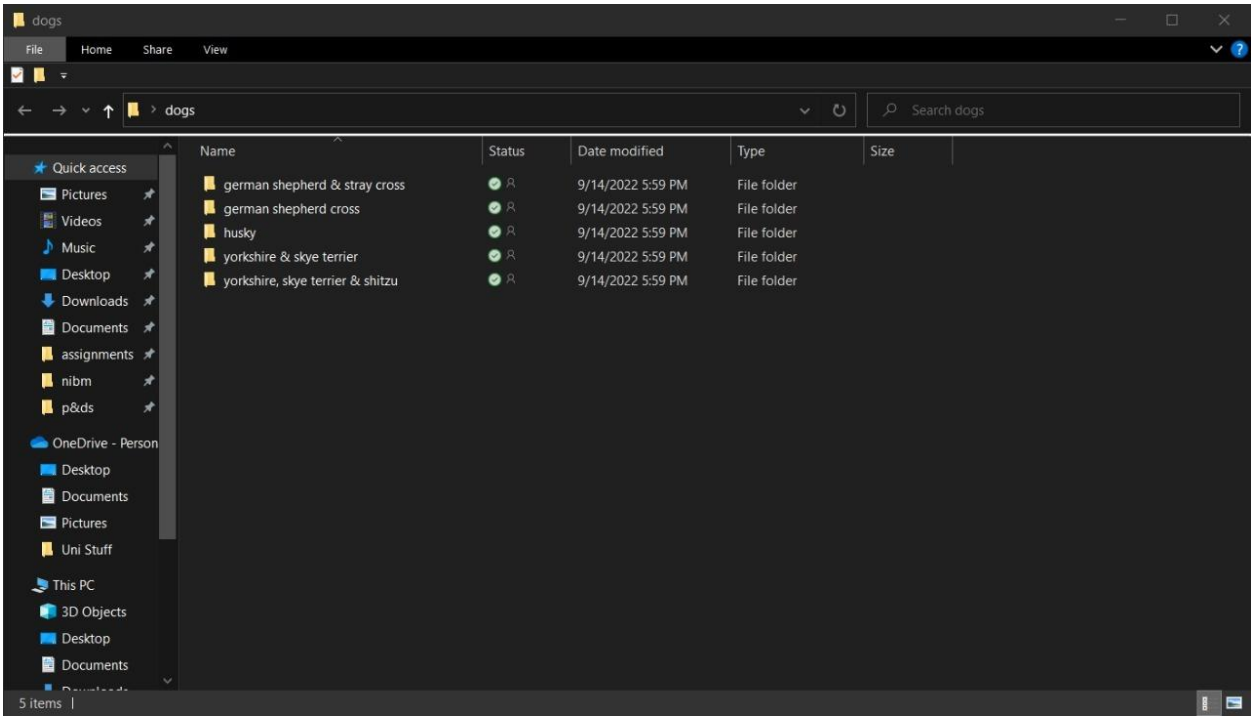
Methodology

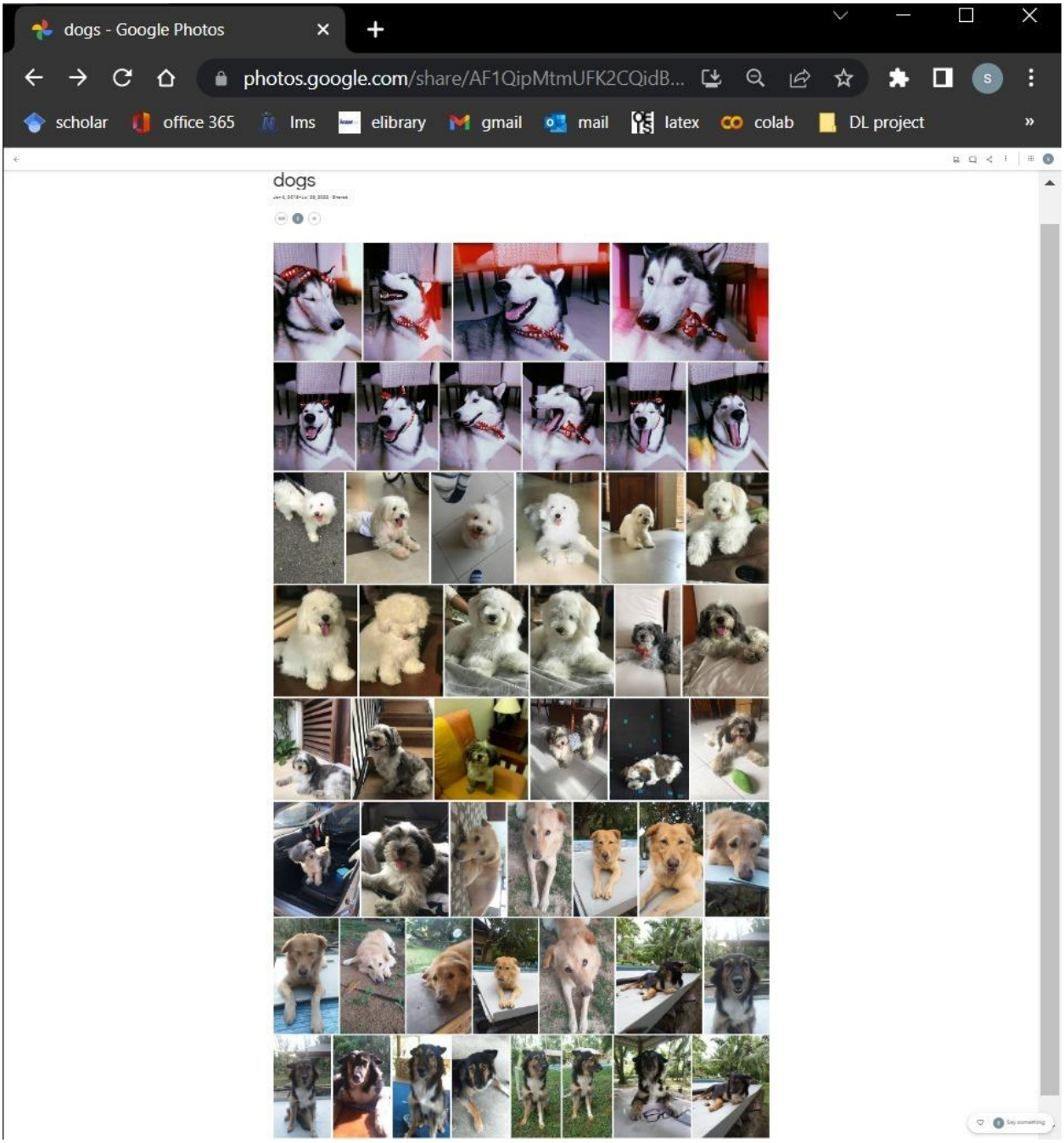
This report will mainly use python and some of its basic python implementations for developing a new Convolutional Neural Networking (CNN). One of which is Keras is a deep learning API (Application Programming Interface) and it will be used to manipulate and analyze the model and will also be used for the visualization framework process.

Algorithm

Step1: Create the dataset

The dataset that has been used in this classification model has been created from scratch with the help of 5 breeds(german shepherd & stray cross, german shepherd cross, husky, Yorkshire & Skye terrier and lastly Yorkshire, Skye terrier & shitzu) totalling up to 50 images. All images were collected using primary sources and compiled into a custom dataset.




dogs - Google Photos
photos.google.com/share/AF1QipMtmUFK2CQidB...
scholar
office 365
lms
elibrary
gmail
mail
latex
colab
DL project
dogs

Step2: importing the relevant libraries

The following libraries were been imported when creating this classification model

In [1]:
```
import tensorflow as tf
from tensorflow.keras.layers import Conv2D,Flatten,Dense,MaxPool2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
import PIL
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import cv2
```

Step3: defining the image properties

the dataset that has been created(i.e dogs) is been uploaded and to get a more refined output the images are then been reshaped and resized.

In [2]:
```
base_dir = r"C:\Users\se\OneDrive\Desktop\dogs"

img_height,img_width=(200,200)
batch_size = 3

train = ImageDataGenerator(rescale = 1/255)
validation = ImageDataGenerator(rescale = 1/255)
```

Step4: training and validation generator

In this step, the dataset is been trained and validated to execute the relevant dog breed. There were initially 50 dog images belonging to five classes in this dataset and with the help of the train and validation 40 images are been trained while the other 10 is been used for validation.

In [3]:
```
train = tf.keras.preprocessing.image_dataset_from_directory(base_dir,
                                                            subset="training",
                                                            seed=123,
                                                            validation_split=0.2,
                                                            image_size=(img_height,img_width),
                                                            batch_size=batch_size)

validation = tf.keras.preprocessing.image_dataset_from_directory(base_dir,
                                                                 subset="validation",
                                                                 seed=123,
                                                                 validation_split=0.2,
                                                                 image_size=(img_height,img_width),
                                                                 batch_size=batch_size)
```

```
Found 50 files belonging to 5 classes.
Using 40 files for training.
Found 50 files belonging to 5 classes.
Using 10 files for validation.
```

In [4]:
```
class_names = train.class_names
print(class_names)
```

```
['german shepherd & stray cross', 'german shepherd cross', 'husky', 'yorkshire & skye terrier', 'yorkshire, skye terrier & shitzu']
```

Step5: Creating the neural net model

Here a CNN architecture has been created from scratch. The idea was simple, stacking up two convolutional layers after the input layer and adding a max pooling layer in between. The Keras sequential module has been used to build the neural network module.

In [5]:
```
model = Sequential()

model.add(tf.keras.layers.Conv2D(16,(3,3),activation='relu',input_shape=(200,200,3)))
model.add(tf.keras.layers.MaxPool2D(pool_size=(2,2)))

model.add(tf.keras.layers.Conv2D(32,(3,3),activation='relu'))
model.add(tf.keras.layers.MaxPool2D(pool_size=(2,2)))

model.add(tf.keras.layers.Conv2D(64,(3,3),activation='relu'))
model.add(tf.keras.layers.MaxPool2D(pool_size=(2,2)))

model.add(tf.keras.layers.Flatten())

model.add(tf.keras.layers.Dense(512,activation='relu'))

model.add(tf.keras.layers.Dense(1,activation='softmax'))
```

In [6]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 198, 198, 16)      448

 max_pooling2d (MaxPooling2D  (None, 99, 99, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 97, 97, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 48, 48, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 46, 46, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 23, 23, 64)       0
 2D)

 flatten (Flatten)           (None, 33856)             0

 dense (Dense)               (None, 512)               17334784

 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 17,358,881
Trainable params: 17,358,881
Non-trainable params: 0
_________________________________________________________________
```

Step6: compile the model and fit the model

compile the model

```
In [7]: model.compile(optimizer=tf.keras.optimizers.Adam(),
                      loss='categorical_crossentropy',metrics=['accuracy'])
```

fit the model

```
In [8]: model.fit(train,epochs=10,validation_data=validation)

Epoch 1/10
14/14 [==============================] - 2s 137ms/step - loss: 0.0000e+00 - accuracy: 0.2000 - val_loss: 0.0000e+00 - val_accuracy: 0.2000
Epoch 2/10
14/14 [==============================] - 2s 130ms/step - loss: 0.0000e+00 - accuracy: 0.2000 - val_loss: 0.0000e+00 - val_accuracy: 0.2000
Epoch 3/10
14/14 [==============================] - 2s 132ms/step - loss: 0.0000e+00 - accuracy: 0.2000 - val_loss: 0.0000e+00 - val_accuracy: 0.2000
Epoch 4/10
14/14 [==============================] - 2s 131ms/step - loss: 0.0000e+00 - accuracy: 0.2000 - val_loss: 0.0000e+00 - val_accuracy: 0.2000
Epoch 5/10
14/14 [==============================] - 2s 129ms/step - loss: 0.0000e+00 - accuracy: 0.2000 - val_loss: 0.0000e+00 - val_accuracy: 0.2000
Epoch 6/10
14/14 [==============================] - 2s 131ms/step - loss: 0.0000e+00 - accuracy: 0.2000 - val_loss: 0.0000e+00 - val_accuracy: 0.2000
Epoch 7/10
14/14 [==============================] - 2s 128ms/step - loss: 0.0000e+00 - accuracy: 0.2000 - val_loss: 0.0000e+00 - val_accuracy: 0.2000
Epoch 8/10
14/14 [==============================] - 2s 131ms/step - loss: 0.0000e+00 - accuracy: 0.2000 - val_loss: 0.0000e+00 - val_accuracy: 0.2000
Epoch 9/10
14/14 [==============================] - 2s 130ms/step - loss: 0.0000e+00 - accuracy: 0.2000 - val_loss: 0.0000e+00 - val_accuracy: 0.2000
Epoch 10/10
14/14 [==============================] - 2s 129ms/step - loss: 0.0000e+00 - accuracy: 0.2000 - val_loss: 0.0000e+00 - val_accuracy: 0.2000
Out[8]: <keras.callbacks.History at 0x22e2a646c40>
```

```
In [23]: score = model.evaluate(train)
         print('Train accuracy:', score[1])

         14/14 [==============================] - 0s 16ms/step - loss: 0.0000e+00 - accu
         racy: 0.2000
         Train accuracy: 0.20000000298023224
```

Step7: visualizing the final results

In this last and final step, the images are been executed and returned with their respective dog breed.

```
In [10]: for images, class_name in train.take(1):
             print(images.shape)
             print(class_name.numpy())

(3, 200, 200, 3)
[2 2 3]
```

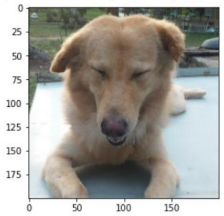
```
In [22]: for images, class_name in train.take(1):

             image = images[0].numpy().astype('uint8')
             label = class_name[0].numpy()

             plt.imshow(image)
             print("actual label:",class_names[label])

             model_prediction = model.predict(images)
             print("predicted label:",class_names[np.argmax(model_prediction[0])])

actual label: german shepherd & stray cross
1/1 [==============================] - 0s 27ms/step
predicted label: german shepherd & stray cross
```



References

Borwarnginn, P., Thongkanchorn, K., Kanchanapreechakorn, S. and Kusakunniran, W., 2019, October. Breakthrough conventional based approach for dog breed classification using CNN with transfer learning. In *2019 11th International Conference on Information Technology and Electrical Engineering (ICITEE)* (pp. 1-5). IEEE.

Mulligan, K. and Rivas, P., 2019, July. Dog breed identification with a neural network over learned representations from the xception cnn architecture. In *21st International conference on artificial intelligence (ICAI 2019).*

Varghese, S. and Remya, S., 2021. Dog Breed Classification Using CNN. *Security Issues and Privacy Concerns in Industry 4.0 Applications*, pp.195-205.

Kumar, R., Sharma, M., Dhawale, K. and Singal, G., 2019, December. Identification of dog breeds using deep learning. In *2019 IEEE 9th International Conference on Advanced Computing (IACC)* (pp. 193-198). IEEE.

Meena, S.D. and Agilandeeswari, L., 2019. An efficient framework for animal breeds classification using semi-supervised learning and multi-part convolutional neural network (MP-CNN). *IEEE Access*, *7*, pp.151783-151802.

Kumar, R., Sharma, M., Dhawale, K. and Singal, G., 2019, December. Identification of dog breeds using deep learning. In *2019 IEEE 9th International Conference on Advanced Computing (IACC)* (pp. 193-198). IEEE.

Albawi, S., Mohammed, T.A. and Al-Zawi, S., 2017, August. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)* (pp. 1-6). Ieee.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J. and Chen, T., 2018. Recent advances in convolutional neural networks. *Pattern recognition*, *77*, pp.354-377.

O'Shea, K. and Nash, R., 2015. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.