Dog Breed Classifier

Files Submitted:

- dog_app.ipynb – Jupyter Notebook file with fully functional code, all code cells executed and displaying output, and all questions answered.
- test_images folder containing all the test images used for measuring the performance of the algorithm
- report.pdf containing the writeup report for the project

Step 1: Detect Humans

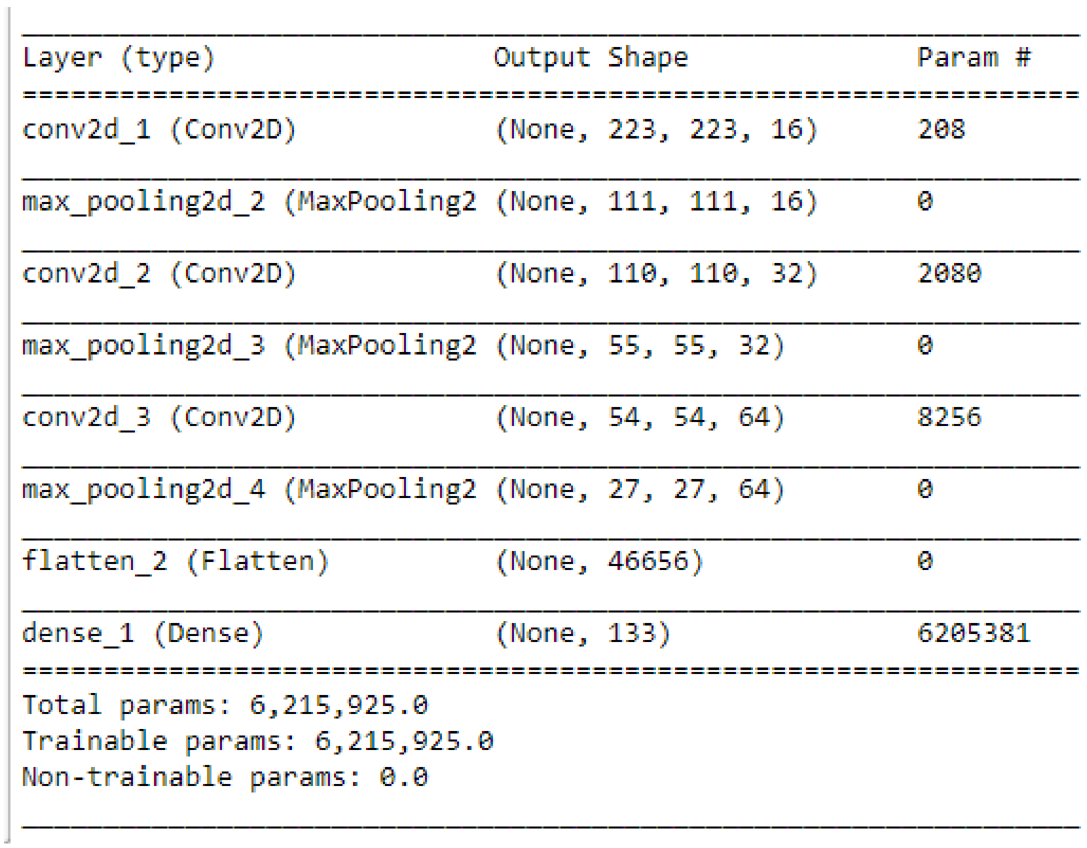
- I have written the code for assessing the given human detector algorithm
  The algorithm successfully detected 99% of the 100 human face images
  The algorithm wrongly predicted 11% of the 100 dog images as human faces

Step 2: Detect Dogs

- I have written the code for assessing the given dog detector algorithm
  The algorithm successfully detected 99% of the 100 dog images
  The algorithm wrongly predicted 11% of the 100 human faces as dogs

Step 3: Create a CNN to Classify Dog Breeds (From Scratch)

- Here is the architecture of the model I have used:

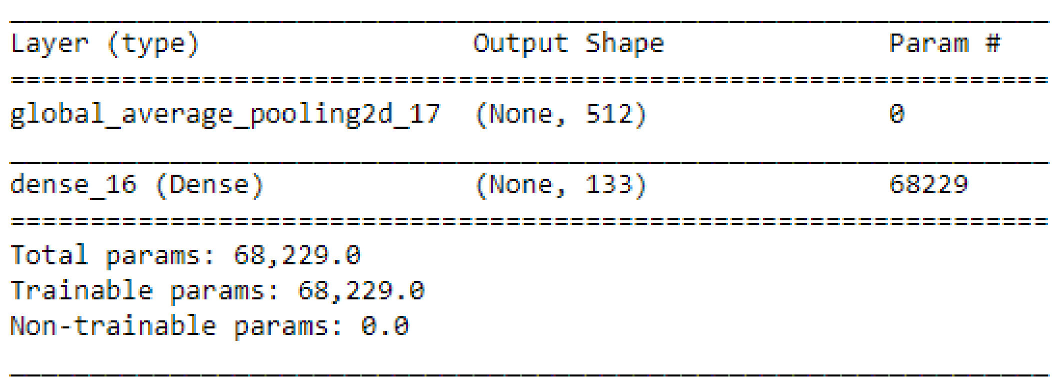


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_1 (Conv2D)            (None, 223, 223, 16)      208       
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 111, 111, 16)      0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 110, 110, 32)      2080      
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 55, 55, 32)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 54, 54, 64)        8256      
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 27, 27, 64)        0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 46656)             0         
_________________________________________________________________
dense_1 (Dense)              (None, 133)               6205381   
=================================================================
Total params: 6,215,925.0
Trainable params: 6,215,925.0
Non-trainable params: 0.0
_________________________________________________________________
```

- The model produced a train accuracy of 1.29 % and a test accuracy of 1.43%
- The current architecture I have chosen is similar to that LeNet model's architecture. It has the minimum required number of layers and parameters to do a good job in the image classification task.

Step 5: Create a CNN to Classify Dog Breeds (Transfer Learning)

- I have used the ResNet50 Model’s and its bottleneck features in this step.
- The architecture for the fine-tuning step for this model is shown below:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
global_average_pooling2d_17  (None, 512)               0
_________________________________________________________________
dense_16 (Dense)             (None, 133)               68229
=================================================================
Total params: 68,229.0
Trainable params: 68,229.0
Non-trainable params: 0.0
_________________________________________________________________
```

- At the end of the ResNet50 model, I added a global average pooling layer as this will greatly reduce the number of parameters and speed up the training time. After the global average pooling layer I added a final output dense layer with 133 output classes. I think the current architecture is suitable for the dog classification problem because the resnet50 model is one of the best image classification models and finetuning it correctly to the given problem would yield high accuracy and using the gap layer speeds up the training process significantly
- The model acquired a Train accuracy of 99% and a test accuracy of 80.86%

Step 6: Write your algorithm

- The algorithm I have written is a simple one. It first check if the given image is a dog image, if it then it prints the breed. If the image is not a dog image then it checks it is a human image, if it is then it prints which breed the human image is similar to. If it fails to detect either a human or a dog image then it prints an error message

Step 7: Test your algorithm

I tested my algorithm on 7 images – 3 humans and 4 dogs.

The algorithms correctly predicted 3 out of 4 dog breeds. It was very close for the image it failed. I presented an image of a Siberaian Husky but it predicted it as an Alaskan Malamute. These two breeds are very similar.

For the human images it predicted three breeds to which those three humans are similar to. It was pretty close too.