Dog Breed Classifier App Project

Project Definition:

Oveview

For the Capstone Project in the Data Scientist Nanodegree program, we were given a choice of different projects including the writing an algorithm to classify dog breeds. I had worked on this project in the Advanced Machine Learning Nanodegree program in a python notebook, and thought it would be great to take that code further and develop an app using the learnings from Data Engineering / Software Engineering lessons in the Data Scientist Nanodegree program.

The given dataset consists of 8,351 dog images for 133 dog breeds, and 13,233 human which is pretty limited.

Problem Statement

Using this data, I am required to build a pipeline that is able to process a user supplied image, and return the predicted breed of the dog if the image is that of a dog, or return the predicted breed of the dog that the human resembles if the image is that of a human, or return a message to user if the image is neither of a dog nor human.

This is an image classification problem, and with 133 different breeds, this is a multi-class classification problem.

Image is a type of unstructured data. The user can upload a black and white or color image, or can upload a small or large image, high-quality or low-quality image. Color images have 3 channels, so the model would be more efficient if it can directly work on a matrix or multi-dimensional data rather than a vector. Given these challenges, a Convolutional Neural Network would be ideal to use in this case.

Choice of Metrics

I could use precision, recall and accuracy as metrics to gauge the model performance on a binary classification problem. For a binary classification problem:

- Precision is a measure of proportion of predicted positives being truly positive
- Recall is a measure of proportion of actual positives that are correctly classified
- Accuracy is a measure of proportion of total observations (positive and negative) that were correctly classified

However, given that the problem at hand is a multi-class classification, precision and recall might be too complex to compute and assess because these metrics will have to be computed for each of the 133 dog breeds in our case. However, I can still use accuracy to evaluate model performance.

Analysis:

Data Exploration

As stated earlier, there are a total of 8,351 dog images which have been split into 6,680 images in our training set, 835 in the validation set and 836 in the testing set.

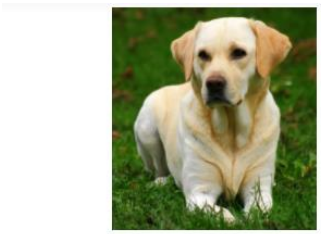
The training set contains 133 dog breeds but it is unbalanced. If each category had the same number of images or observations, one would expect approx.. 50 images for each of the 133 dog breeds in the training set. However, there are some dog breeds that have over 70 images while some have less

than 30. This could impact model performance on some of these dog breeds with lower samples to train on.

Data Visualization

Another point to keep in mind is that this is a more granular classification problem. We are no longer dealing with having to detect differences between species but go deeper and find differences within species. In other words, finding differences between dogs is a harder problem than finding differences between a dog and a cat. Some dog species can be very similar which makes it harder for the model to differentiate between them. For example, 'Brittany' and 'Welsh Springer Spaniel' can look very similar and hence would be difficult to tell apart.

Some dog breeds can come in different colors, for example Yellow, Chocolate and Black Labradors

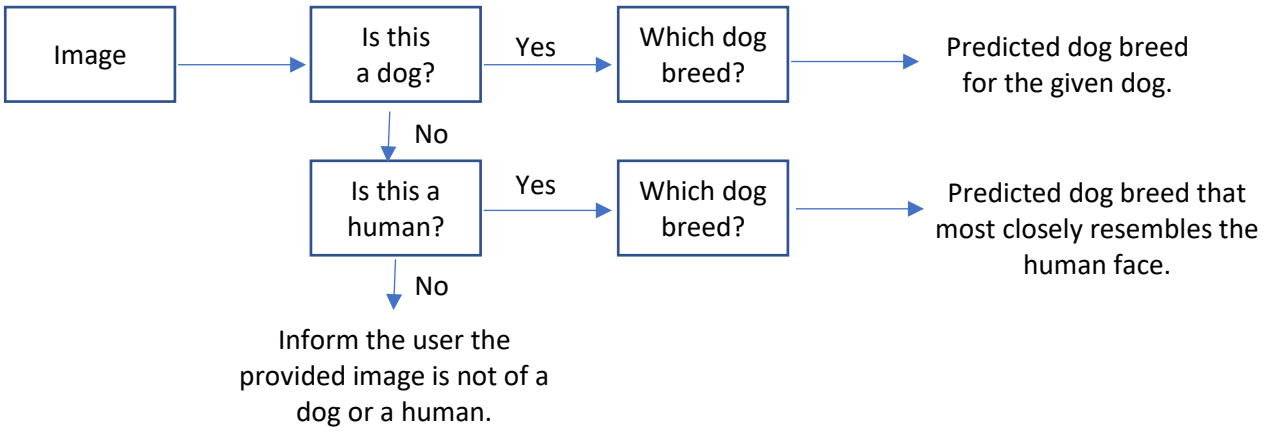


These accentuate the problem given that these is a limited data to work with.

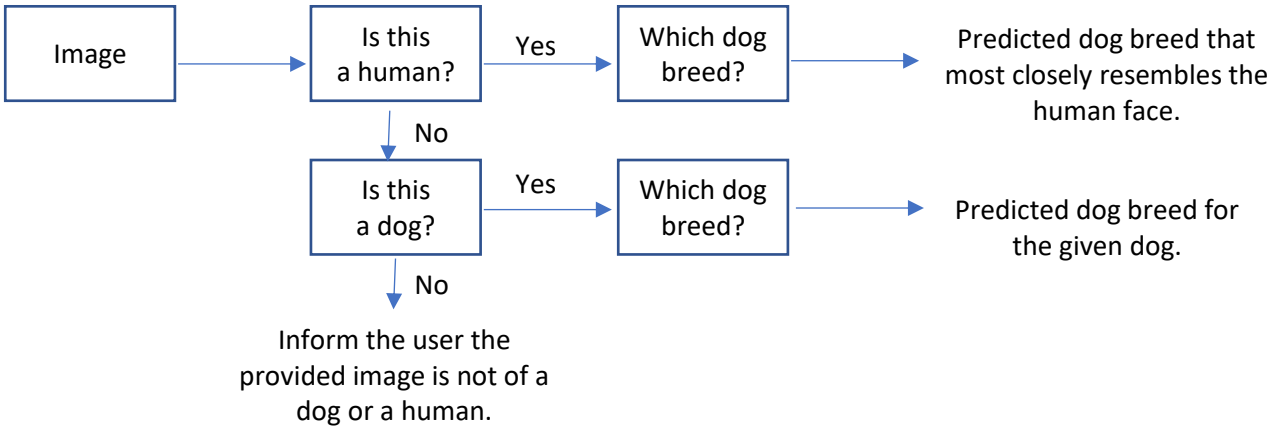
Methodology

We can think of designing the app in one of the two ways:

1)



2)



Whichever option we choose, we need 3 image classifiers or at the bare minimum 2 image classifiers. The first two or first image classifier will help determine whether the image is that of a dog or a human. The last classifier will only be invoked if the image is that of a dog or a human and will determine the appropriate dog breed.

I am choosing to go with 3 image classifiers – 1) detects whether the image has a human face (human face detector), 2) detects whether the image is that of a dog (dog detector) and 3) predict the dog breed (dog breed predictor). And as described below, because the ResNet50 dog detector performs better than the OpenCV face detector, I chose to use the 1st option above (check for a dog image before checking for a human face).

	Human face detector	**Dog detector**	**Dog breed predictor**
Implementation	OpenCV's implementation of Haar feature-based cascade classifier	ResNet50 model pretrained on ImageNet database. Categories 151 to 268 in the ImageNet database are for dogs.	Transfer learning using InceptionV3 model pretrained on ImageNet database with last layer retrained on classifying dog breeds

Data Pre-processing	Images are converted to grayscale using the cvtColor method of OpenCV	The ResNet50 model from Keras using a Tensorflow backend requires a 4D array (number of sample, rows, columns, channels). I wrote a custom function that takes an image, resizes it to 224 x 224 pixels. This image is converted into 4D tensor of shape (1, 224, 224, 3). Color images have 3 channels. The 'preprocess_input' function from keras is used to reorder the channels (RGB to BGR) and normalize the tensor by subtracting mean pixel (calculated from all pixels in all images in ImageNet) from every pixel in each image.	I reused the custom function that was built for ResNet50 model that resizes the image to 224 x 224 pixels and converts it into 4D tensor shape. I further used the 'preprocess_input' function from keras to reorder the channels (RGB to BGR) and normalize the tensor by subtracting mean pixel (calculated from all pixels in all images in ImageNet) from every pixel in each image.
Model Evaluation and Validation	Tested on 100 human and 100 dog images Accuracy = 94.5% (All 100 human images were correctly classified as human images, but 11 out of 100 dog images were incorrectly classified as human images)	Tested on 100 human and 100 dog images Accuracy = 100% (All 100 dog images were correctly classified as dog images, while all 100 human images were correctly detected as non-dog images.)	Tested on 836 images on test set, we get an accuracy of 78% in the InceptionV3 model. The target accuracy for the project was 60%.
Refinement	The accuracy is pretty high for the purpose of this app implementation. No further refinement was done.	The accuracy is pretty high for the purpose of this app implementation. No further	First approach was to train a CNN from scratch, but with the limited resources (time, computing power and training

		refinement was done.	set) I could only achieve an accuracy of 6%. The model architecture and details are given below the table. The next model I tried was transfer learning with VGG16. With this model I was able to get an accuracy of 48%. The model architecture is given below.

Dog Breed Classifier Models:

First – CNN from scratch

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 224, 224, 16)      208
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 112, 112, 16)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 112, 112, 32)      2080
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 56, 56, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 56, 56, 64)        8256
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 28, 28, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 28, 28, 64)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 50176)             0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                2508850
_________________________________________________________________
dropout_2 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 133)               6783
=================================================================
Total params: 2,526,177
Trainable params: 2,526,177
Non-trainable params: 0
_________________________________________________________________
```

This was a very simple model with 3 (convolutional + maxpooling) layers followed by two dense (fully connected) layers. We were training 2,526,177 parameters. Though I used dropouts, this model architecture was too simple for task at hand, and we only had 6,680 training images to train the model on.

Model settings:

- optimizer – 'rmsprop'
- loss – 'categorical_crossentropy' (for multiclass classification)
- metrics – 'accuracy'
- epochs – 7 (this was purposely kept low because we were training 2.5 Mn parameters and each iteration was taking a long time to run)
- batch_size - 20

Second – Transfer Learning with VGG16

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
global_average_pooling2d_1 ( (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 133)               68229
=================================================================
Total params: 68,229
Trainable params: 68,229
Non-trainable params: 0
_________________________________________________________________
```

The pre-trained VGG16 model was frozen, and only the parameters for newly added global average pooling layer and a dense layer were trained (68,229 parameters) on the 6,680 training images. The convolutional layers in VGG16 models were able to do a better job of extracting features from the images, and the newly added layer were able to take these and predict on the 133 dog breeds helping us pull the accuracy up to 48%.

Model settings:

- optimizer – 'rmsprop'
- loss – 'categorical_crossentropy' (for multiclass classification)
- metrics – 'accuracy'
- epochs – 20 (we could run more iterations because with fewer parameters to train, each iteration was running much faster)
- batch_size - 20

Third and Final – Inception V3

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
global_average_pooling2d_2 ( (None, 2048)              0
_________________________________________________________________
dense_4 (Dense)              (None, 133)               272517
=================================================================
Total params: 272,517
Trainable params: 272,517
Non-trainable params: 0
_________________________________________________________________
```

Similar to VGG16 model, the pre-trained InceptionV3 model was frozen, and only the parameters for newly added global average pooling layer and a dense layer were trained (272,517 parameters) on the 6,680 training images. Just by comparing the number of parameters (272,517 vs 68,229) I could tell that the InceptionV3 was a much complex model, and the model architecture was able to extract more complex features compared to VGG16.

Model settings:

- optimizer – 'rmsprop'
- loss – 'categorical_crossentropy' (for multiclass classification)
- metrics – 'accuracy'
- epochs – 20 (we could run more iterations because with fewer parameters to train, each iteration was running much faster)
- batch_size – 20

Conclusion

The final model with InceptionV3 had an accuracy of 78% vs the goal of 60% so I didn't try to refine it further. However, given time and resources, I would have tried the following to improve the model performance:

- Increase the training set
- Augment the data
- Oversample on dog breeds with less representation (have fewer examples in the training set)

This was an interesting project to me as I was able to bring learnings from Advanced Machine Learning (Convolutional Neural Networks) and Data Scientist Nanodegree (Software/Data Engineering) programs together. By reusing a pre-trained image classification model, I was amazed to have built a model with only 6,680 images that performed with approx.. 80% accuracy.