Tuesday, May 19, 2020

Elias Junior Ghantous
Udacity Machine Learning Engineer Nanodegree
Capstone Project

Dog Breed Classifier Using Convolutional Neural Networks

I. Definition

Project Overview

Classifying dog breeds has long been a popular problem in the machine learning industry. In fact, "man's best friend" has countless Kaggle datasets and submissions dedicated to them pertaining to a multitude of questions one might ask about dogs. Ultimately, the question of "what breed is that dog?" is one I chose to answer, by suggestion of the Udacity Machine Learning Nanodegree course. My goal is to develop a machine learning model to answer the question, taking the guesswork out of the situations people face when they fail to recognize the breed of dogs passing them by. This is the Final Report analyzing the results and detailing the process of developing said model.

Problem Statement

The goal is to build a machine learning model which could be deployed via a web app, helping users predict and classify dog breeds. This is a multiclass classification problem where supervised learning can be used. The final algorithm must perform two tasks:

- ***Detection of human faces:*** If supplied an image of a human, the model will identify the resembling dog breed, similar to that of popular Buzzfeed questionnaires e.g. "Which type of garlic bread are you?"
- ***Detection of dog faces:*** If supplied an image of a dog, the model will identify the dog breed.

These are the potential solutions to the problem which the model would solve.

Evaluation Metrics

I am using multiclass log loss as the evaluation metric. Due to the varying number of pictures for each breed in the dataset, accuracy would not be suitable. The metric log loss is such that it takes into account the uncertainty of the prediction made relative to the actual label and so the model would be evaluated fairly. The metric is fairly simple being that it is

$$Accuracy = \frac{Number\ of\ items\ classified\ correctly}{Total\ number\ of\ classified\ items}$$

In order to find the optimally performing model, multi class log loss is used to compare results of both validation set and test set.

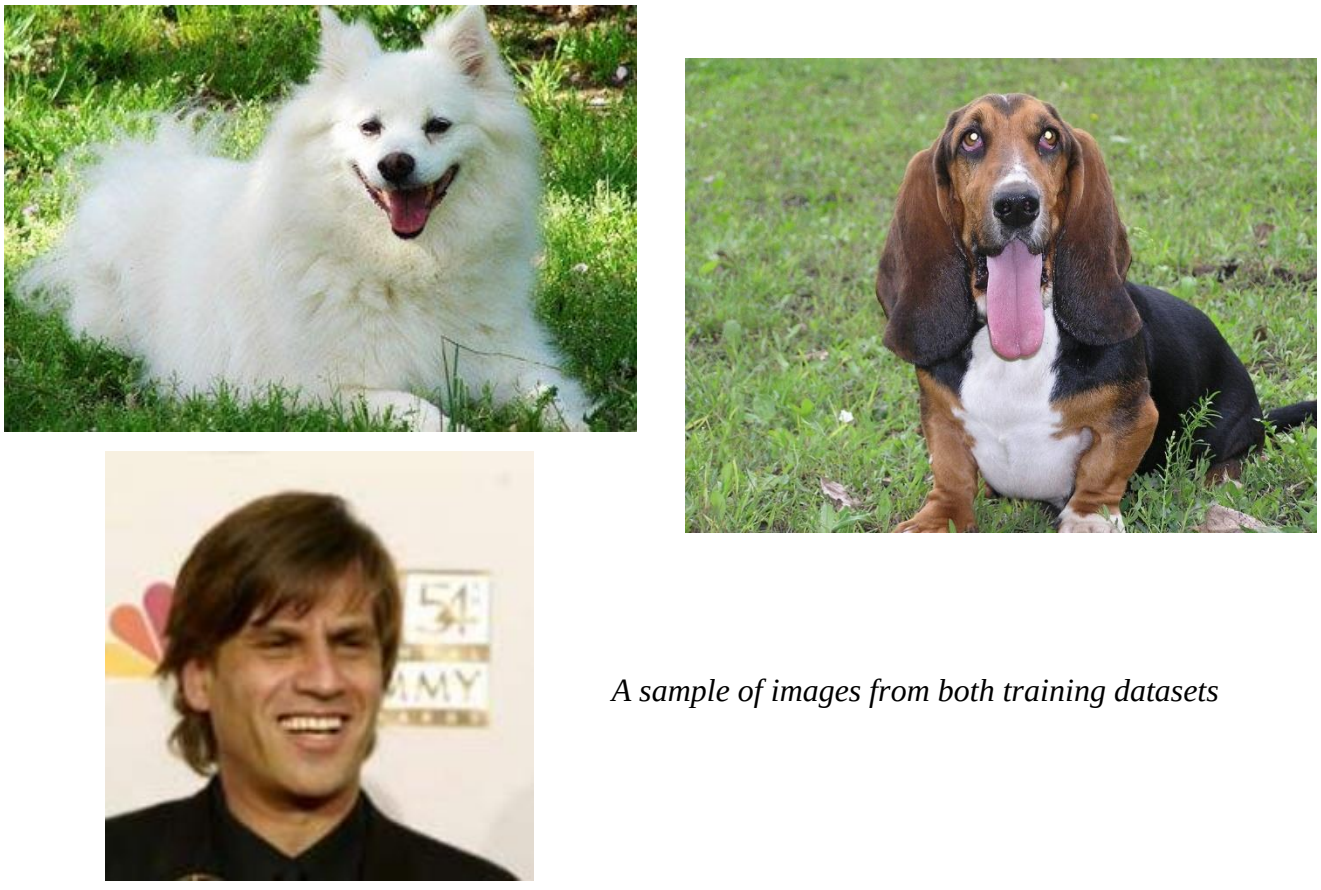
II. Analysis

Data Exploration and Visualization

The dataset for this project which includes images of both dogs and humans is provided by Udacity.

'dogImages' dataset: The dog image dataset has 8351 total images which are sorted into train (6,680 Images), test (836 Images) and valid (835 Images) directories. Each directory has within it 133 folders corresponding to dog breeds. The images are of different sizes and different backgrounds with some of the images being of smaller sizes than the others. The number of images per breed varies as well, an issue which could possibly affect the accuracy of the model produced due to unbalanced samples.

'lfw' dataset: The human dataset contains 13233 total human images which are sorted in alphabetical order of the names of the humans (5749 folders). All images are of size 250x250. Images have different background and different angles. The number of images per person varies, making the dataset unbalanced and potentially affecting the model accuracy.



A sample of images from both training datasets

Algorithms and Techniques

In order to perform the multiclass classification, I am going to use a convolutional neural network. In deep learning, a **convolutional neural network** (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. The network applies weights to specific objects in each image in order to 'see' the image differently to others.
First, we must be able to detect human images. To do this, I will use an existing algorithm, OpenCV's implementation of Haar feature based cascade classifiers. Second, in order to detect dog images, I will use a pretrained VGG16 model.
Finally, after identifying the image as either dog or human, we pass this image to a CNN which will process the image and predict the breed of dog out of all possible 133 breeds.

Benchmark

The CNN model created from scratch must have accuracy of at least 10%. This would confirm that the model is working, ruling out the chance of it guessing randomly which would return a correct answer in roughly 1/133 times (<1% accuracy).
The CNN model created using transfer learning must have accuracy of 60% or higher.

III. Methodology

Data Preprocessing

I normalized all the datasets. Then, I implemented RandomRotation and RandomHorizontalFlip to training data so as to avoid overfitting. After that, the next step was to resize and then CenterCrop was used to make images 224x224 as that is the optimal size for VGG16 models.
I didn't apply any image augmentation apart from CenterCrop to the validation and testing datasets.
All of this was done alongside the creation of three separate data loaders that incorporated the use of tensor so that the images could be used by the model. This meant that all of the images were valid and any of the abnormalities would not affect the model significantly.

Implementation

Step 1: Import the necessary dataset and libraries. Pre-process the data and create the train, test and validation datasets. Perform image augmentation on training data.

Step 2: Detect human faces using OpenCV's implementation of Haar feature based cascade classifiers.

Step 3: Create a dog detector using the pretrained VGG16 model.

Step 4: Create a CNN to classify dog breeds from scratch. Train, validate and test the model.

Step 5: Create a CNN to classify dog breeds using Transfer Learning with resnet101 architecture. Train and test the model.

Step 6: Write an algorithm to combine the dog detector and human detector.

a) If a dog is detected in the image, return the predicted breed.
b) If a human is detected in the image, return the resembling dog breed.
c) If neither is detected, provide output that indicates the error.

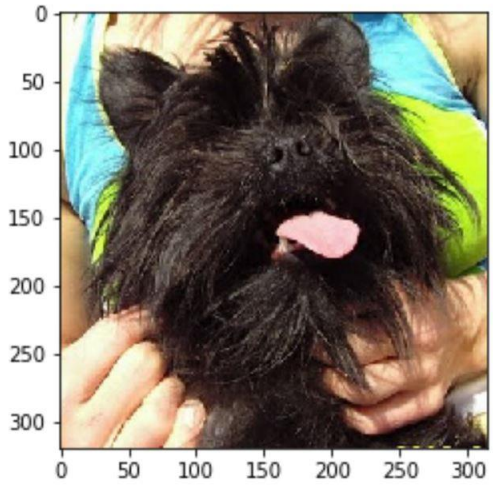
Within each of these steps, there were smaller tasks to do, one of which was to create a CNN from scratch. This is documented in the notebook. The CNN I created is a 3-layer network. All 3 layers have the same kernel size of 3, the first and second layers having strides of 2 while the final 3rd layer has a stride of 1. A pooling layer of kernel size 2 and stride 2 is used which will reduce the input size by 2. We have two fully connected layers that produce a 133-dimensional output. Dropout of 0.25 is used to avoid overfitting. A Relu activation function is used for the layers, similar to that used later in the pretrained resnet101 model.

Refinement

For my scratch CNN, it managed to exceed the requirements of the benchmark model with its accuracy of 11% (95/836). However, this is not a viable solution to the problem due to the fact that having an accuracy of 11% is extremely low. To improve this, I used a resnet101 pretrained model and used transfer learning to optimize model performance. The resnet101 architecture is the most suitable for this classification problem seeing that it has been pre-trained on the ImageNet dataset. It has 101 connected layers and can be adjusted to suit this specific problem. After importing the model, I changed the output features of the final layer to 133 to match that of this classification problem so now the output is 133 dimensions similarly to the above output with the scratch CNN. This resnet101 based model achieved an accuracy of 86% (726/836) in 5 epochs, an astonishingly better result than the scratch CNN.

Below are two examples of outputs the model predicted correctly.

```
Hey there Doggy!
Your predicted breed is:  Affenpinscher
```




```
Hey there Doggy!
Your predicted breed is:  Affenpinscher
```

IV. Results

Model Evaluation and Validation

Firstly, the human face detector was made. This function used OpenCV's implementation of Haar feature based cascade classifiers. Of the first 100 human images in the 'lfw' dataset, 96% of them were detected correctly as humans. Of the first 100 dog images in the 'dogImages' dataset, 17% of them were detected as having human faces.

Second, I moved on to making the dog face detection function. This was created using the pretrained VGG16 model mentioned earlier. Of the first 100 human images, none of them were detected as dogs, a 100% success rate. Of the first 100 dog images however, 93% were detected as dogs.

Lastly, the final model used for prediction is the resnet101 CNN model built using transfer learning. Although it was only trained for 5 epochs, this model classified 86% of the dogs correctly when applied to the test data. Of the 836 images, it classified 726 correctly. This accuracy rate is significant enough for the model to be confidently used because this is more of an application with no consequences of guessing wrong. However, if this accuracy were to be such in the medical field, as an example, the model would be far from finished due to the fact that those applications would be more significant and consequential if the model were to predict incorrectly.

Justification

Overall, for a model that predicts dog breeds, it performed very well. An 86% accuracy is impressive, especially since it jumped from 11% to 86% after using resnet101 and transfer learning. I would be confident using this model to predict breeds of dogs I couldn't immediately discern. In order to potentially improve the model, some additions and changes can be made such as tuning hyperparameters, increasing the size of the training data, and possibly applying some more augmentation to the images. Furthermore, I only used the resnet101 architecture. Perhaps other pretrained models could perform better on these images.