Dog Breed Classification

Capstone Project

Marcelo Hirota

4th April 2020

1. Definition

- Project Overview

Machine Learning (or ML for short) has become one of the most popular areas in computer science because with nowadays powerful machines is possible to process loads of data and predict with high certainty what is going to be the outcome of certain decisions.

Dog Breed Classification is a well-known problem in the ML Community, having even a Kaggle competition about it (Kaggle link to the competition)

The main idea behind this project is to create a model that people are going to be able to use in any website or app they want to create. And also it is going to be a good challenge for my recently acquired knowledge.

- Problem Statement

The idea is to pass a photo through the recognition model and it would tell us what dog breed is on the photo. For every dog breed, there are various parameters along with image and textual data that will become the training parameters. One of the first questions the model has to answer is: is there a dog in the picture? Follow by the question: if there is a dog, what kind of breed it is?

This is a supervised learning problem because I have our dog images divided into breed classes I will use classification predictive modelling more precisely multi-class predictive model.

- Metrics

All datasets were provided by Udacity. There a repository full of pictures of humans and dogs. All dogs pictures are sorted in train, test and valid directories, and all image are sorted in breed directories as well. Humans are sorted by name of each person.
The model is trained using the training dataset. The test dataset is used to predict the performance of the model on unseen data. It was used accuracy as a metric to evaluate the model on test data:

Accuracy = Number of items correctly classified / All classified items

Also, during the training phase, it was compared to the test data prediction with validation dataset and calculate the multi-class log loss to find the best performance model. Log loss takes into the account of the uncertainty of prediction on how much it varies from the actual label and this will help in evaluating the model.

2. Analysis

- Data Exploration

For this project, the input format must be of image type, because the model needs to receive an image in order to identify the breed of the dog. The dataset provided has pictures of dogs and humans.

Dog dataset: The dog dataset is composed of 6,680 train images, 836 testing images and 835 validation images, giving a total of 8,351 images. The images are all sorted by their respective directories and each directory has 133 folders corresponding to the dog breeds. The images have different sizes, different background and lightning. The data is not balanced because few breeds have more images than other breeds. Few have four images while some have 8 images.

Human dataset: this dataset contains 13,233 images in total, which are sorted in folders by name (5750 folders). All images have the same size, 250x250, but they have different backgrounds, lighting and angles. This data is also unbalanced, there are people with only one image and other with more images.

Photos from the dataset

- Algorithms and techniques

Convolutional Neural Network (CNN) has the best performance for multiclass classification and also is excellent with images. CNN is a part of deep neural networks and is perfect for analyzing images and their characteristics. This solution has three steps:

- For humans, I will be using OpenCV model, implementation of Haar feature-based cascade classifiers;
- For dogs, it will be a pretrained VGG16 model,
- The CNN model using transfer learning, to process the image and predict the correct breed.

- Benchmark model

For the benchmark model, it will be used the CNN model with an accuracy of more than 10%.

3. Methodology

- Data Preprocessing

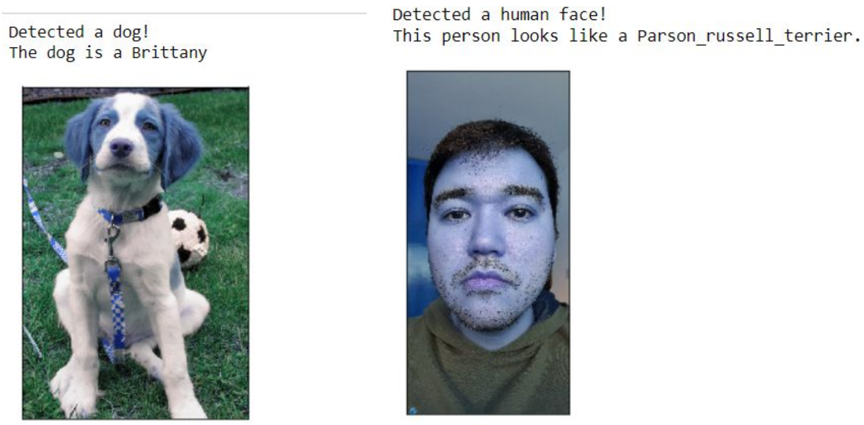
All data are resized to 224x224, then normalization is applied to all images. For training data, image augmentation is done to reduce overfitting, also they are randomly rotated and randomly horizontal flipped. In the end, all images are converted into tensor before passing to the model.

- Implementation

I have built a CNN model from scratch to solve the problem. The model has 3 convolutional layers. All convolutional layers have a kernel size of 3 and stride 1. The first conv layer (conv1) takes the 224*224 input image and the final conv layer (conv3) produces an output size of 128. ReLU activation function is used here. The pooling layer of (2,2) is used which will reduce the input size by 2. We have two fully connected layers that finally produces 133-dimensional output. A dropout of 0.25 is added to avoid over overfitting.

- Refinement

The CNN created from scratch have an accuracy of 1.9%, Though it does not meets the benchmarking, the model can be significantly improved by using transfer learning.
To create CNN with transfer learning, I have selected the Resnet50 architecture which is pre-trained on ImageNet dataset, the architecture is 50 layers deep. The last convolutional output of Resnet50 is fed as input to our model. We only need to add a fully connected layer to produce 133-dimensional output (one for each dog category). The model performed extremely well when compared to CNN from Scratch, resulting in an accuracy of 86%



Outputs predicted by the model

4. Results

- Model Evaluation and Validation

Human detector: with the help of OpenCV, the model was able to detect 97% of humans in the human dataset and 12% of humans detected in the dog dataset (probably humans and dogs together in the photos)

Dog detector: the function was created using the pretrained VGG16 model. In the dog dataset, 100% of dog faces were detected against 1% of dog face found in the human dataset.

CNN: The CNN model was created using transfer learning from ResNet50 architecture and was trained with 5 epochs. The final model produced an accuracy of 86% on the testing data.

- Justification

The model is far from being perfect but performed way better than my own expectations, going from 1.9% accuracy to 86% at the end.

5. Conclusion

- Reflection

In this work, I have found that the two networks used, VGG-16 and ResNet50, are capable of finding patterns that are humanly recognizable when finetuned on the datasets provided. Though over-fitting was present in both networks, we took necessary precautions to prevent and reduce the effects of over-fitting and produced results and analysis to prove that patterns were recognized by both networks, even in the presence of over-fitting. By combining the understanding of both networks and their features, I was able to produce, through training, a model able to recognize different type of dogs and also tell apart dogs from humans.

- Improvement

At the start of the project, my objective was to create a CNN with 90% testing accuracy. Our final model obtained only 86% testing accuracy. The model can be improved by adding more training and test images, also expanding to more than 133 breeds of dog that the initial datasets provided. Also, a Hyperparameter optimization with GridSearch might be able to provide better parameters for higher performance. In this model, I used the ResNet50 architecture for feature extraction, maybe using other architecture will give better results and accuracy.

In addition, I could have looked more deeply into the images themselves that I used for training the dog breeds. I could have looked at a confusion matrix to see which images were giving the biggest errors in the validation data in order to identify possible noise. Maybe some of these images were too blurred and the model was generalizing well but being deceived by unclear images. I could check the training images with random sampling to see the quality of the images and delete images that were badly focused or with more than one breed of dog, reduce noise in the training data. I could check to see if there were sufficient training image of each breed of dog and that the image classes were balanced overall in terms of training numbers. Following the above areas, I am sure I could increase the testing accuracy of the model to above 90%.