MACHINE LEARNING ND

DOG BREED CLASSIFIER BY USING CNNs

REPORT

Yasir Almutairi
30 December 2020

A. Definition

1. Project Overview

Computer vision is the field to make computers recognize and identify objects, such as people, dogs, cars. It might be easy for you or me to identify for a computer it is not a trivial task. There have been many breakthrough in this area by leveraging deep learning in order to classify objects correctly by the use of concepts such as Convolutional neural networks (CNN) which have pushed the capabilities of computers forward and it has many application in self-driving cars, And object classification, And object classification is the domain of the problem I'm trying to solve falls into. In which I will use CNNs in order t0 identify not only a dog but its breed from 133 breeds.

2. Problem Statement

The objective is to build an algorithm that can accept an image and transforms it to an appropriate format. And classify per conditions given:

- If the image contains a dog then the algorithm have to be able identify the dog, then be able to identify its breed.
- If the image contains a human then the algorithm would return a resembling dog breed.

3. Metrics

The metric in which will determine the effectiveness of our prediction is accuracy, which would yield the the percentage of correctly classified breeds across all predictions.

$$\text{Accuracy} = \frac{\text{True Positives + True Negatives}}{\text{All Samples}}$$

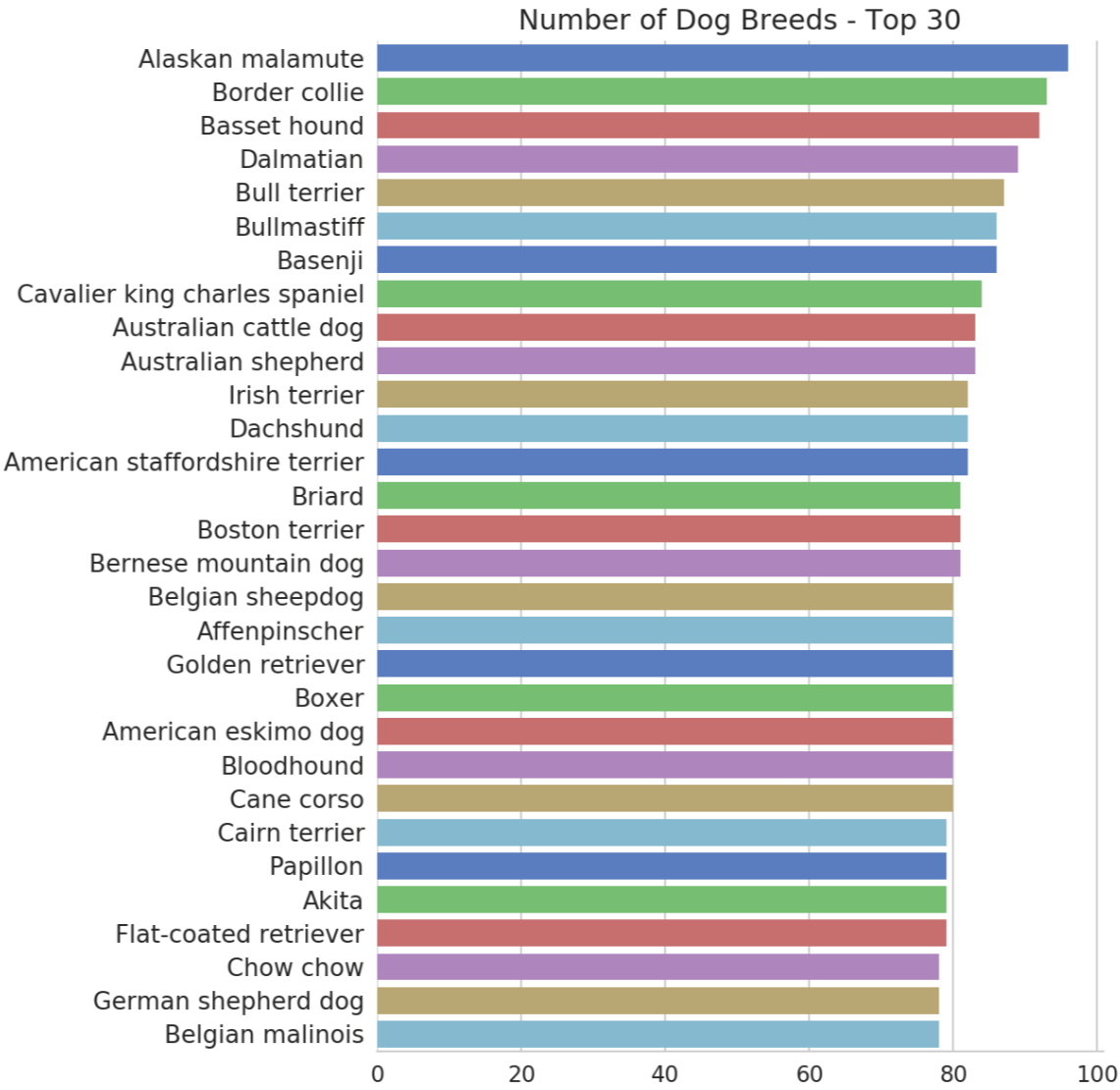
B. Analysis

1. Data Exploration

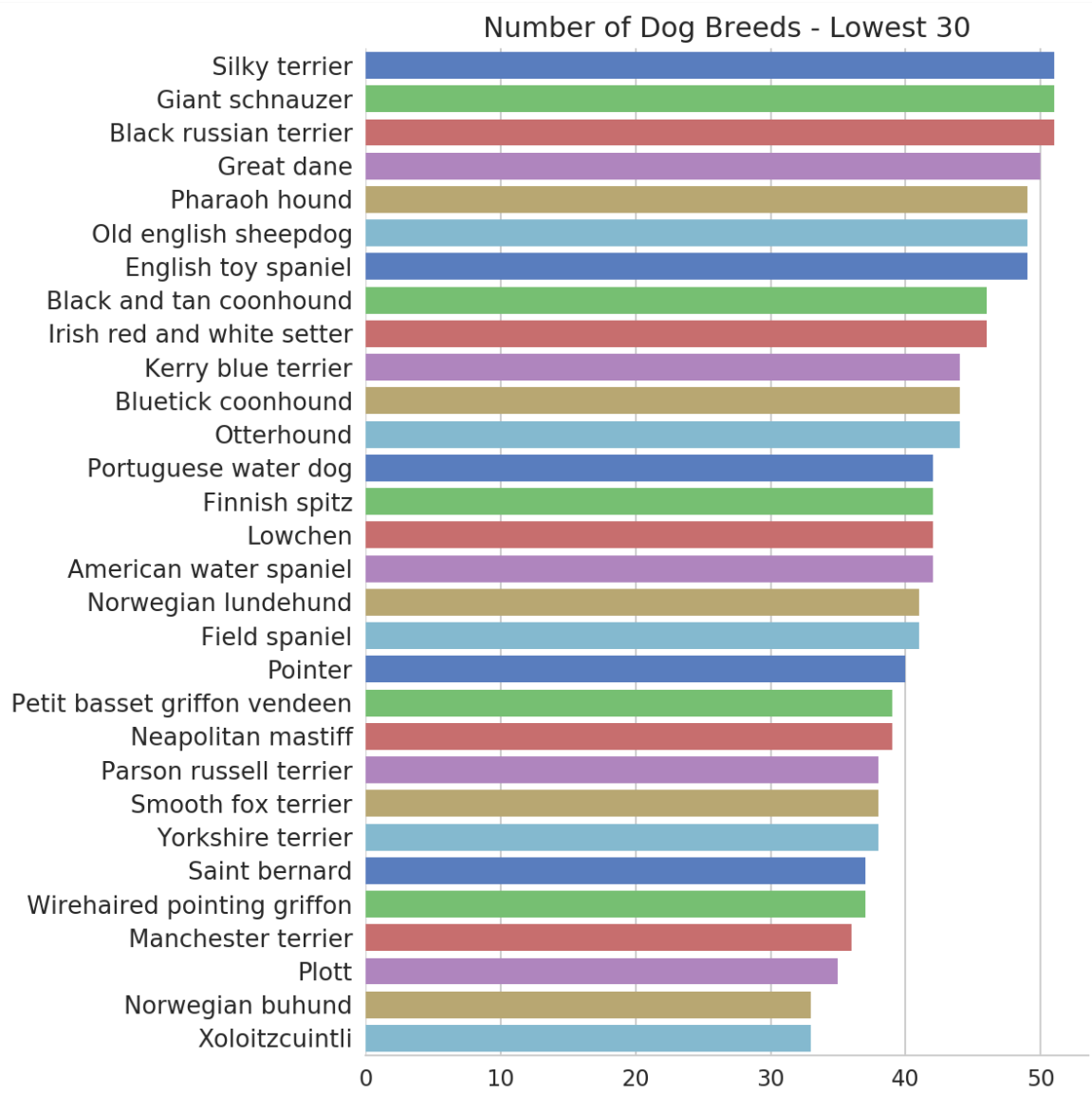
There are two datasets provided one contains the datasets consist of human image and dog images, both provided by Udacity. Human datasets contains 13,233 images of humans. While dog dataset contains 8,351 images of dogs. To better view the types of breed provided by dog datasets.

2. Exploratory Visualization

To better view the distribution of 8,351 dogs over 133 breeds, View the below figures.



his figure shows the number of top 30 dog breed available in the datasets. To have a better a view of the data the next figure shows the lowest 30 breeds in our data.



what these two figures tells us, Is we have fewer examples for Plott breed than for instance Basenji which might effect the performance of the model in predicting these dog breed that we few examples of them.

3. Algorithms

Here I will outline the algorithms and techniques I used:

1. **Humans Face Detector**:

Using OpenCV's pre-trained Haar Cascade classifier to detect the face of human in given image.

2. **Dog Detector:**

By using a PyTorch pre-trained model VGG-16. It has been trained on a large dataset. Its classification categories is of 1,000 if we look at the sequence from 151, to 268 we will notice they are all dogs. So we can check if the image contains a dog if the category falls between these two categories.

3. **Dog Breed Classifier by using CNN from Scratch:**

By using PyTorch to build a CNN from scratch the minimum accepted test score is 10%, which the model scored 21% test accuracy.

4. **Dog Breed Classifier by using Transfer learning:**

I used transfer learning in order to transfer the pre-trained weights of ConvNets of RestNet model to classify dogs breed.

C. Methodology

1. **Data Preprocessing:**

In order to use the images for training or prediction I preprocessed the images by using PyTorch transforms.

```
transforms.RandomResizedCrop(224),
transforms.RandomHorizontalFlip(),
transforms.ToTensor(),
transforms.Normalize(mean=[0.5, 0.5, 0.5],
```

Preprocessing Images

to add randomness to the data and avoid overfitting. which i believe it will help in generality. it also worth to mention i have used a variety of image augmentation order to have better affect in training the model.

2. Implementation:

- **Dog Breed Classifier from scratch:**

The below figure shows the architecture of the CNN model. Which outline 3 Convolutional Networks and 2 fully Connected Linear Layer, and a dropout of probability 30% to avoid overfitting.

```
Net(
  (conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1
))
  (conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1,
1))
  (conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1,
1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_
mode=False)
  (fc1): Linear(in_features=6272, out_features=500, bias=True)
  (fc2): Linear(in_features=500, out_features=133, bias=True)
  (dropout): Dropout(p=0.3)
)
```

D. Results

1. Model Evaluation

During the training the model were tested on a validation dataset after each epoch.

The model from scratch had a test score of 21% after 100 epochs and learning rate of 0.05 and as an optimizer I used stochastic gradient descent (SGD) and as a loss function Cross Entropy was used.

The model from transfer learning using ResNet architecture had a test score of 75% on the test data set. after 15 epochs which I stopped the training after 9 epochs.

2. Justification

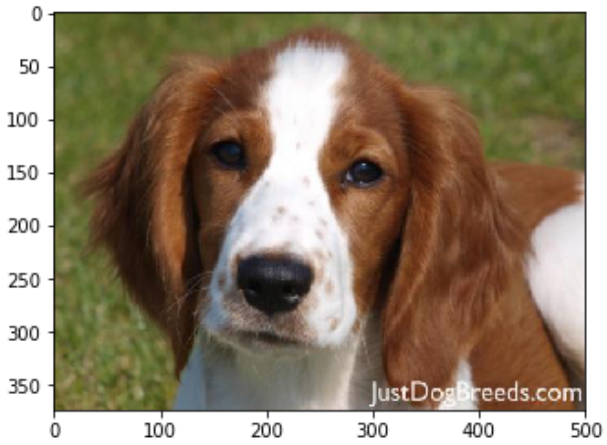
Due to the accuracy results of ResNet transferred model on unseen data which achieved an accuracy of 75% I have used it in the final application algorithm.

E. Conclusion

- **Free-Form Visualization**

Here I will show the final outputs of the developed application, Which takes an image as input and produce a predicted breed, if instead the image contains a human then produced the similar dog breed.

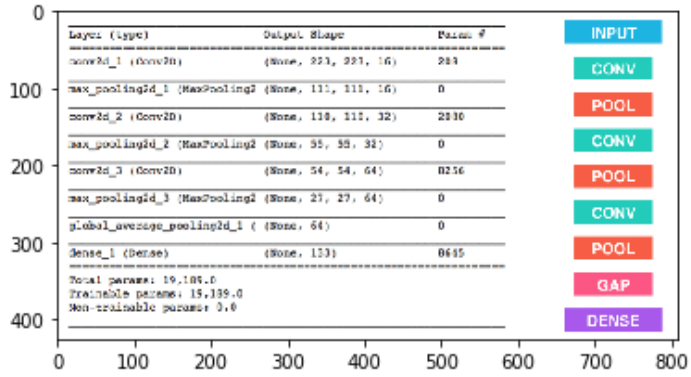
Below is an example for a dog.




```
Dog is detected!          It is of Australian shepherd breed
./images/Welsh_springer_spaniel_08203.jpg
```

And this is what happens if the images provided contains neither an face of dog or human

.




```
Error... Neither where human or dogs were detected.
./images/sample_cnn.png
```

- **Reflection**

To summarize the workflow I have done in this project:

1. Understanding the problem.
2. Importing required data.
3. Developing human face detector.
4. Developing dog detector.
5. Augmenting the image.
6. Define a CNN model and trained it.
7. Define a Transfer learning model and trained it.
8. Design the behaviour of the application.
    1. Takes input as image
    2. Detected human face
    3. Detect dog
    4. Predict dog breed
    5. Return prediction

For me the most challenging part for me was working with images and understanding different types of augmentation and how it affect the model and its performance. My first choice of image augmentation steps resulted in the model training never went lower than 8.9. and test accuracy never managed to score a result even close to 10% however after iterations I managed to fix my errors. It was great learning journey I spent a lot of time reading through amazing papers of the pre-trained models.

- **Improvement**
  - Build an application to inference the dog breed in real-time
  - Get a data set contents a lot more types of dog breeds.
  - And probably use ensemble of models in order to get a more a accurate results.