Capstone Project
Machine Learning Engineer
Nanodegree Program

Anup Joseph Sebastian

22nd March, 2020

Dog Breed Classification using CNNs

Project Definition

Project Overview

Over the past few years, development in Artificial Intelligence and Machine Learning has improved our lives in many different ways. Its impact has been far and wide, in both industry and research, and has allowed us to solve problems that were considered impossible just a few years ago. One of the biggest winners of this new AI revolution has been the field of Computer Vision.

In this project, Convolutional Neural Networks and a Haar Cascades OpenCV detector is used to distinguish between various dog breeds and humans. The application is first successfully able to distinguish between a dog and a human and then classifies detected dog images based on it’s breed, and human images as the dog breed that the human looks most similar to.

I have used two provided datasets from Udacity – a dog image dataset and a human image dataset – as well as a few additional images for testing which I have included along with this project.

Problem Statement

The goal of the project is to build an image classifier, which can perform a series of tasks as follows.

- Given an input image of either a dog or a person, the application should classify whether it is a dog or a human. In other words, it should calculate the probability that it is a dog.

- If the image is of a dog, the application should classify, which dog breed it belongs to from a number of possible classes. This may be extended to include mixed breed dogs. In a mathematical sense, given an image the CNN should output the probability that the image belongs to a particular breed of dog.

- If the input image was of a human, the CNN should output the dog breed that the human most resembles.

Metrics

In order to evaluate the performance of our model, I have used accuracy as a metric to justify acceptable performance of the model.

Accuracy is just a simple measure of how many images the model classifies correctly out of the total number of images.

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

Analysis

Data Exploration

This CNN accepts images as inputs and it will be trained using two datasets, which were provided by Udacity. They datasets used are as follows:

Dog image dataset: This dataset consists of a total of 8351 images, split into three sets – train (6,680 images) , test (836 images) and valid (835 images). Each of these directories are further split into 133 folders each, which contain images of various dog breeds. The dataset can be found here.

Human image dataset: The dataset contains 18982 images of people, with directory names corresponding to the persons name. Most of the directories consist of only a single image, but there are some which have multiple images. The data consists of images of 13233 unique people. The dataset can be found here.

Both these datasets are going to be used to train neural networks for classification.

Data Visualization

The data contains 133 different breeds of dogs, but some breeds have more examples than others. The histogram below shows the number of examples for each breed of dog in the dataset. As you can see from the histogram below there is a disparity between the number of examples available for training, but at the same time it is also clear that there are more than enough examples to train for every dog breed, so it should not be a problem. A sample of the dataset is shown below.

Fig 1: Sample of images from dog dataset

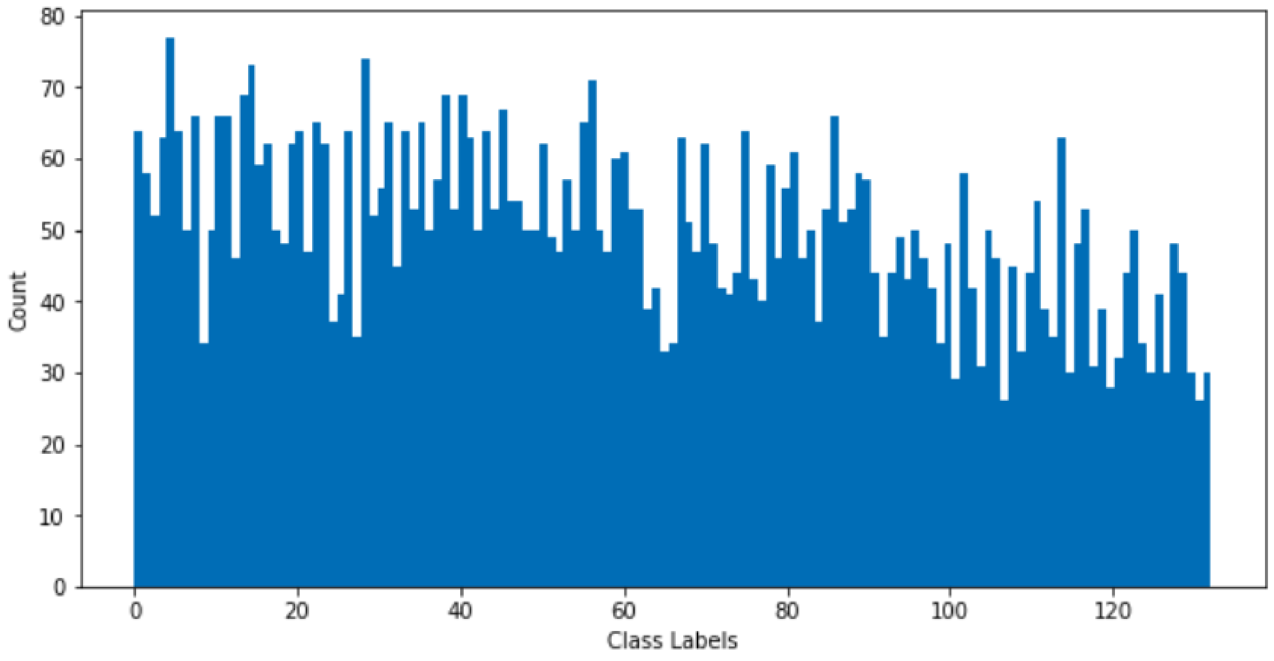
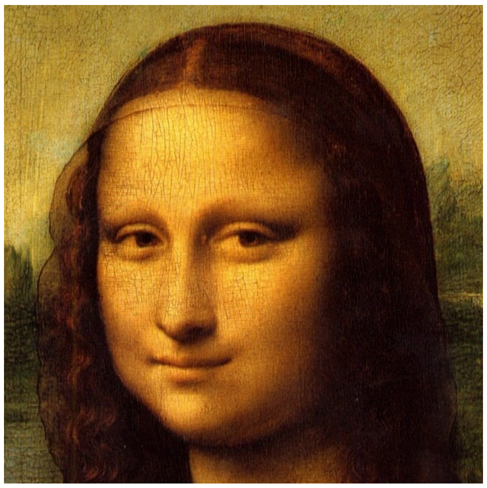
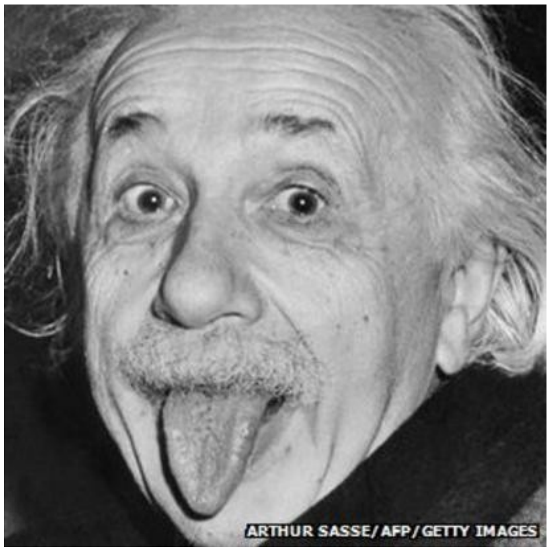


Fig 2: Number of images of dogs by breed

For human images, there is a much greater disparity. As mentioned in the previous section, there are 13233 unique people and a total of 18982 images. However, this should was not a problem as these images were not used to train the neural network but only to identify whether humans were detected using the OpenCV detector or not detected using the VGG-16 pretrained neural network.

In addition to these images, I used some other images to test the application at the end. Some of them are shown below.





Algorithms and Techniques

The solution to this problem can be split into three main parts, and the techniques used to solve each part is different. The algorithms and techniques for each part are detailed below:

1. Distinguishing between humans and dogs.

   There are two possible ways to solve this problem. The first one is to use an OpenCV detector to identify human faces from the input images and the next one was to use a pre-trained CNN classifier to identify dogs in the images. Each one has its advantage. For instance, the OpenCV detector can run faster than the CNN, however, the CNN provided better performance. The CNN was chosen in the end as it was the first step in the problem and the performance of the rest of the model depends on the first step being correct.

2. Classifying dog breeds

   For this task another CNN was used. CNNs are the best solution for this task because of their ability to understand spatial information in images. There was a choice between training a network from scratch or to use transfer learning. Both methods were tested and the transfer learning model provided much better performance compared to the one trained from scratch. This CNN is trained on only dog images, but is fed in both human and dog images for classification in the final application.

3. Writing an Algorithm to produce the correct output

   This is just a simple algorithm that contains a conditional statement that uses the first model to detect if the image is a human or dog and produces the correct output accordingly. It does the task of passing in the input image to the models and also formatting the output to look presentable.

Benchmark

To make sure that the model performs well, there are a number of target benchmarks that the classifier models should achieve.

- It is expected that a CNN trained from scratch, should achieve a minimum accuracy of 10%. Considering that the dataset has 133 classes, randomly guessing should give us an expected accuracy of 0.0075%. Therefore this is considered acceptable for first model.

- However, when leveraging transfer learning, it is expected that the model should obtain an accuracy above 60% for it to be considered performing fairly well.

Methodology

Data Preprocessing

Before feeding images into the CNNs for training, various augmentations are done to it as part of the preprocessing stage. These image augmentations are done to improve the generalization capability of the model. In other words, it helps the model train on a wider variety of images than is present in the training set, which will help it be more resilient to various real world variations.

The augmentations done were:

- A random rotation by 15 degrees

- A random resized crop, which crops (224, 224) images. I chose this size simply because I it allows me to use the same transformations in the second model while doing transfer learning.
- A random horizontal flip, which gives mirror images of the training set.
- Normalization with the same mean and standard deviation that the PyTorch pretrained models require. Normalization is also generally done to allow the model to train faster as smaller values can converge faster.

For the testing and validation sets I decided not do add any augmentations, but only just resize to (224, 224) so that it input tensor is in the correct size. This is because I would want to evaluate performance on the provided dataset accurately, which is representative of a real world test dataset.

Implementation

The main workflow for the design and implementation of this project was as follows:

1. Import and preprocess the data. The steps for preprocessing could include converting to tensors, resizing, augmentation and normalization. The data should also be split into train, validation and test sets.

2. Evaluate the performance of the OpenCV face detector and compare it to a pretrained neural network for performance (accuracy and inference time). Choose the most suitable method.

3. Train a CNN from scratch to classify dog breeds and evaluate its performance.

4. Leverage transfer learning to create a similar dog classifier using one of the suitable options in Pytorch. A balance of accuracy and inference time should be considered, to ensure a good user experience.

5. Write an algorithm to make the logical decisions on what model to invoke the logic detailed in the Problem Statement.

These steps are detailed in order in the provided Jupyter Notebook.

Refinement

The initial CNN model trained from scratch had consistently given accuracy values ranging from 13% to 16% on various runs. This was not acceptable for a final application. For a classification model with 133 classes it is doing much better than random guessing, but it gets far more classes wrong than right.

Therefore, a transfer learning model using the architecture and weights from a ResNet101 classifier, trained on the ImageNet dataset was used. The final fully connected layer was replaced from having 1000 outputs to 133 which is what our problem requires. The weights in the Resnet101 classifier have been trained to be robust to a wide variety of different image classes in the ImageNet dataset, and the initial layers act as an excellent feature detector for our images. Using transfer learning allowed the model to obtain a much higher accuracy of 85%.

I also tried using other ResNet models such as ResNet34, ResNet50 and ResNet152, but ResNet101 struck the right balance between training and inference time, and accuracy.

Results

Model Evaluation and Validation

The metrics obtained by the various stages of this project are as follows:

- Dog/Human Detector:
  I tried multiple solutions for the first stage to classify an input image as a dog or a human. The table below shows the results of the three OpenCV methods. It shows the percentage of humans detected in two datsets.

HaarCascades Detector	Dog Image Results (%)	Human Image Results (%)
Original (alt)	14	99
Detector_A (default)	53	98
Detector_B (alt2)	26	99

These detectors performed well with human images but also detected a lot of dogs as humans, so a pretrained VGG-16 model was tried next. The VGG-16 model did much better. It was able to detect dogs in 98% of the dog images and 0% of the human images. This score may vary with a different set of images, but not drastically enough to justify using OpenCV based methods. So this model was chosen as the preferred method to distinguish between dogs and humans.

- Dog Breed Classifier:
  Two methods were used here. A CNN trained from scratch and a transfer learning CNN using the ResNet101 architecture. Their performance metrics are shown below:

Model	Accuracy (%)
CNN from scratch	16
ResnNet101 Transfer Learning	85

The transfer learning model performed much better and was the obvious choice to implement in the application.

Justification

The final models performance exceeded well beyond the benchmark expectations set set for it. It was able to obtain a final accuracy of 85% which is much higher than the expected accuracy of 60%. The model was also tested using a few additional images, which it performed well on to. These images were a good mix between typical images and atypical images. For example, the model was correctly able to identify a picture of the Mona Lisa as a person.

On the other hand, the detector also performed as expected in images with both dogs and humans. The algorithm for the model was written in such a way that it gives priority to dogs in the images, as it is primarily a dog breed classifier. In images with dogs and humans, the model generally ignores humans and classifies dogs. It only classifies the similar dog breed when there are no dogs detected in images.

The level of accuracy (85%) reached by the model is a justifiable accuracy to implement in a final application. It gives acceptable results most of the time similar to most real world apps. The application is not exactly something critical, but something to just have some fun with images. A higher accuracy (above 90%) would be expected if it were to be used for a different application such as dog breed identifier at a pet shelter.

Conclusion

Reflection

I found this project to be challenging and interesting. It allowed me to explore ideas on my own and experiment and play with different CNN architectures. I would have expected a network trained from scratch to have performed better, by intution, but the final accuracy of around 15% gave me a realization of the difficulty of the problem and also how effective transfer learning can be in solving problems. I hope to make use of transfer learning solutions more in my own future projects.

The hardest part of the project was having to wait for models to run, while trying out different architectures . Since there is a lot of trial and error involved, it can be hard to predict what sort of performance you can expect from a particular architecture without having run it first. Atleast, without significantly more experience with CNNs.

This project also introduced me to the OpenCV library and some of it’s capabilities as a different approach to solve image based problems. This made me spend some time looking at the OpenCV library and its uses. I hope to spend some time to learn more about it to gain a deeper understanding of Computer Vision on a more fundamental level.

Improvement

The model performed fairly well, but after finishing it I noticed that there was some room for improvement. Some of the ways the model could be improved are as follows:

- Train for a few more epochs. The validation loss was still consistently decreasing after 15 epochs of training the transfer learning model, so another 5 or 10 epochs might have allowed the model to converge fully.

- Some of the last few layers of the transfer learning model can be un-frozen and allowed to also train to make the model adapt better to the problem. This may however, be at the cost of training time.

- Additional image augmentation could be done such as stretching and scaling. This will help the model generalize better to images that get 'compressed' or 'stretched' during the resizing process in the test and validation sets.

References

1. Udacity Project Repository: https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification

2. Pytorch Documentation: https://pytorch.org/docs/stable/index.html

3. Torchvision Library: https://pytorch.org/docs/stable/torchvision/models.html

Images From:

1. Mona Lisa: http://lookatherbeautifulface.blogspot.com/2010/08/one-of-most-famous-female-faces-in.html

2. Albert Einstein: https://www.bbc.com/news/health-23665502

3. Pug: https://www.smithsonianmag.com/smart-news/excessive-vitamin-d-pet-food-may-be-making-dogs-sick-180970963

4. Golder Retreiver: https://santansun.com/2018/11/06/valley-fever-storm-brewing-for-chandler-dog-owners/