DOG BREED CLASSIFIER (CNN)

Ninad Deshpande

February 6, 2021

I. DEFINITION

Project Overview

Image classification has always been one of the hot topics in the machine learning field. It wasn't until recently that computers were able to perform seemingly trivial tasks such as detecting an object in a picture or recognizing spoken words. But why are these tasks so effortless to us humans? It comes down to the fact that perception largely takes place outside the realm of our consciousness, within specialized visual, auditory, and other sensory modules in our brains.

Convolutional neural networks (CNNs) emerged from the study of the brain's visual cortex, and they have been used in image recognition and classification since the 1980's. A paper on CNNs to classify dog breeds published by David Hsu made me realize that this classification is in fact challenging especially due to the minimal inter-class variations and existence of a huge variety of species.

In this project, I have utilized CNNs in order to identify the canine's breed for a given dog image. Moreover, if supplied an image of a human, the code identifies the resembling dog breed. Datasets used in this project Is provided by Udacity.

Problem Statement

The aim of the project was to build a machine learning model which can be used to process the user-supplied images and perform the following two tasks:

- **Dog face detection:** Algorithm identifies an estimate of the canine's breed, if an image of a dog is supplied.
- **Human face detection:** Algorithm identifies the resembling dog breed, if an image of a human is supplied.

The steps involved in this project are as follows,

1. Import dataset.
2. Create train, test and validation dataset after preprocessing the data.
3. Detect humans using OpenCV's HAAR feature based cascade classifiers.
4. Detect dogs using a pretrained convolutional network for classification and detection, VGG16 model.
5. Create a CNN to classify dog breeds from scratch.
6. Create a CNN to classify dog breeds using transfer learning followed by training and testing the model.
7. Write an algorithm which returns the predicted breed, if a dog or a human face is detected and an error when neither of them is detected in an image.
8. Test the algorithm.

Metrics

Multi class log loss will be used for the model evaluation. Even though we will use accuracy as a metric for the model evaluation. In this case, accuracy may not be a good indicator to measure the performance because of the data imbalance.

In case of a classification model, we have a multitude of metrics of performance available to optimize our models, quantify their performances, compare them and improve them. The log loss metric considers the probabilities underlying our model, and not only the final output of the classification. Thus, it will consider the uncertainty of prediction based on how much it fluctuates from the actual label. This will help us in the model evaluation.

The required accuracy benchmarks for this project are as follows,

- The CNN model created in this project must have the accuracy of at least 10%. As there are 133 dog breeds, a random guess will give a correct answer once in 133 tries, which is equivalent to the accuracy of less than 0.80%.
- To make a good model, the CNN created with transfer learning must have at least 60% accuracy.

II. ANALYSIS

Data Exploration

The dataset consists of images of dogs and humans and is provided by Udacity.

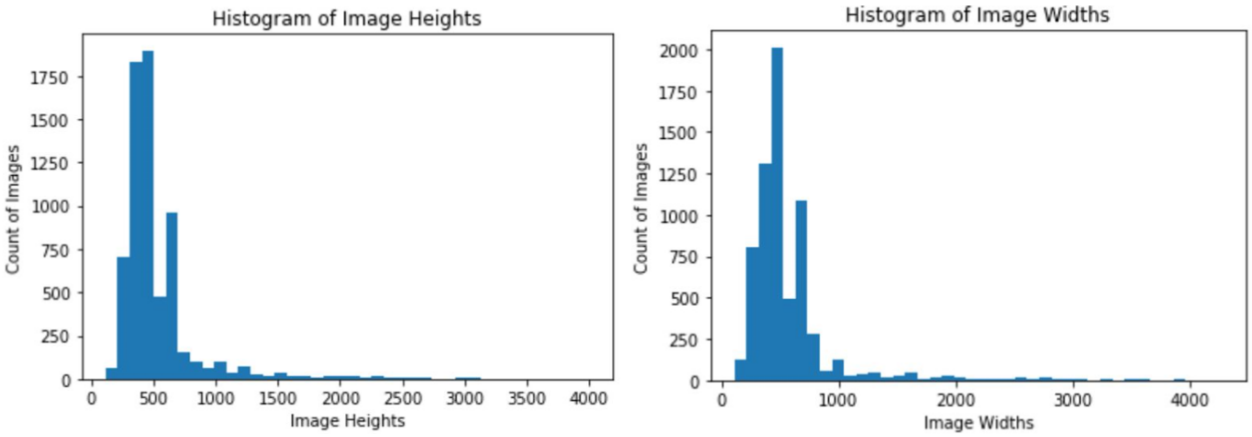
- **Dog images:** The dataset has 8351 images which are sorted into train with 6,680 Images, valid with 835 Images and test with 836 images. Each directory has 133 subdirectories corresponding to each dog breed. These images are of different sizes and backgrounds and as the number of images provided for each breed is not the same, the data is not balanced.
- **Human images:** The dataset contains 13233 human images sorted by names of size 250x250 with different backgrounds and postures. As the number of images provided in each folder is different, the data is not balanced.

Following are some of the images from the dataset,



Exploratory Visualization

After analyzing the dimensions of the images in the given dataset, we can conclude that these images have different heights and widths with most of them less than 1k pixels. There are some outliers as well.



Algorithms and Techniques

Object Detection using OpenCV's Haar feature-based cascade classifiers is an effective object detection method. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images and it is then used to detect objects in other images. I have used it for detecting human faces.

Similarly, for detecting dogs in the images, I have used the VGG16, which is a convolutional neural network model proposed by K. Simonyan and A. Zisserman. This model makes improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. Also, the VGG16 significantly outperforms the previous generation of models making it a better candidate for many deep learning image classification problems.

Furthermore, I have created two CNNs for classifying the dog breeds, one from scratch with three convolutional layers and another using transfer learning to attain greatly improved accuracy followed by training and testing the model.

Benchmark

- The CNN model created in this project was expected to have the accuracy of at least 10%. As there are 133 dog breeds, a random guess would give a correct answer once in 133 tries, which is equivalent to the accuracy of less than 0.80%. The model I created from scratch has 16% accuracy.
- To make a good model, the CNN created with transfer learning was expected to have at least 60% accuracy. The model I created has 78% accuracy.

III. METHODOLOGY

Data Preprocessing

Size of 224,224 is picked for the input tensor as VGG16 takes that size as input. Apart from that I have used RandomResizedCrop() method to crop the given image, RandomHorizontalFlip() method to horizontally flip the given image randomly with the default probability of 0.50 and the RandomRotation() method to rotate the image by 15-degree angle. I have applied these transformations to the training data to augment the dataset and to prevent the model from overfitting. Image resizing has been done for the validation as well as test data.

Implementation

For implementing a CNN for scratch, I have defined three convolutional layers. As the input images will be of size 224x224 and depth 3 (for RGB), the first convolutional layer will produce the depth of 36 (i.e., 36 filtered images), second will produce the depth of 64 and the third layer with the depth of 128. All the layers are using filters of size 3x3 and I have also added 1 pixel of padding on the border to keep the output layer with the x, y size as the input image. I have defined the max pooling layer with a kernel size and stride of 2, which will down sample the input dimensions by a factor of 2. Finally, I have flattened and applied a fully connected layer to produce the desired number of classes (i.e., dog breeds which are 133). The model got 16% accuracy.

Refinement

Transfer learning is used to further improve the model accuracy. I have used a Residual Network (also known as ResNet), even though there are various variants with 34, 50, 101 and 152, I decided to go with the one which is 101 layers deep as it would suffice for the given problem. As increasing network depth does not work simply by stacking layers together and the deep networks are hard to train because of the vanishing gradient problem (as the gradient is backpropagated to earlier layers), repeated multiplication may make the gradient infinitively small. As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly. The model I used got 78% accuracy with only 5 epochs. With more epochs, we can certainty improve this accuracy. Steps include importing the pre-trained ResNet model, excluding subgraphs from gradient computation to increase the efficiency, replacing the last fully connected layer by setting the size of each output sample to 133 and later using CrossEntropyLoss() as it is useful when training a classification problem with multiple classes.

IV. RESULTS

Model Evaluation and Validation

OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces gave 98% accuracy in detecting human faces in the first 100 images of humans. While the pre-trained VGG-16 model achieved 100% accuracy in detecting dog faces in the first 100 images of dogs.

The CNN model created from scratch to classify dog breeds has 16% test accuracy with 3.600283 test loss and the model created with transfer learning indeed archived much better test accuracy of 78% along with only 0.714229 test loss.

Justification

Both CNNs models performed better than expected.

- CNN created from scratch:
    - Benchmark: 10%
    - Achieved accuracy: 16%
- CNN created from transfer learning:
    - Benchmark: 60%

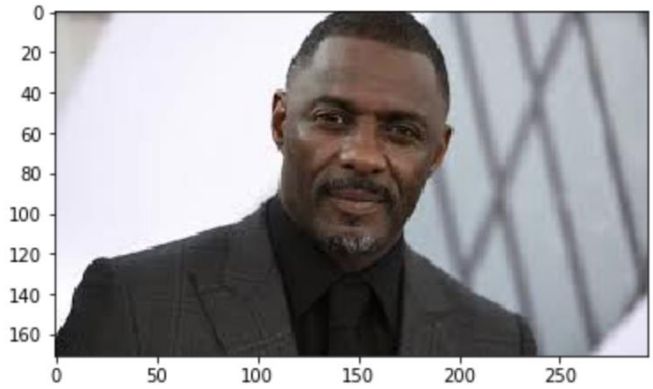
- Achieved accuracy: 78%

V. CONCLUSION

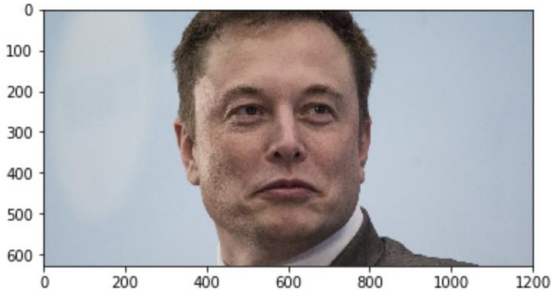
Free-Form Visualization

The predictions by the final model to classify dog breeds shows that it performs much better than expected,

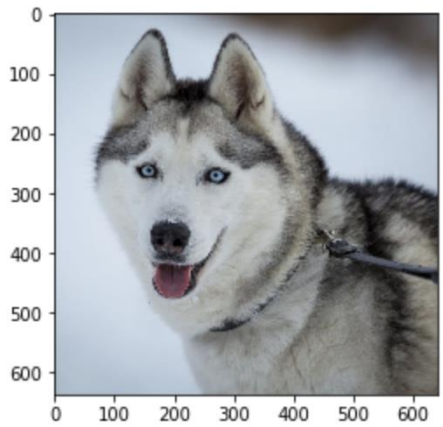
```
I see a Human in this picture!
No offense but I think this human looks like a  Irish wolfhound
```



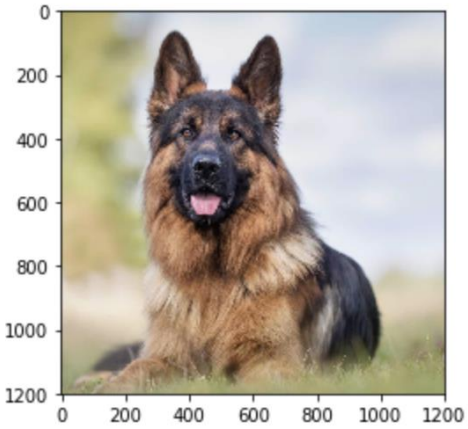

```
I see a Human in this picture!
No offense but I think this human looks like a  Cavalier king charles spaniel
```



```
I see a dog in this picture!
I think it is a  Alaskan malamute
```




```
I see a dog in this picture!
I think it is a  German shepherd dog
```



Reflection

In this project, I have learned how to build a pipeline to process real-world, user-supplied images. Along with exploring CNN models for classification, I understood the challenges involved in piecing together a series of models designed to perform various tasks in a data processing pipeline. Each model has its strengths and weaknesses, and engineering a real-world application often involves solving many problems without a perfect answer.

I started with importing the datasets and preprocessing the data, later I created train, test and validation datasets. I used the popular OpenCV's HAAR feature based cascade classifiers to detect humans followed by using a pretrained convolutional network model called VGG16 to detect dogs in the images. I have created a CNN to classify dog breeds from scratch and improved its performance significantly by creating a CNN using transfer learning followed by training and testing the model. Finally, I designed an algorithm which returns the predicted breed, if a dog or a human face is detected and an error when neither of them is detected in an image.

Improvement

I could have certainly improved the model accuracy by adding more training data, further hyperparameter tuning and more data augmentation.

REFERENCES

- GitHub repository of the original project, https://github.com/udacity/deep-learning-v2-pytorch/blob/master/project-dog-classification/
- Eugenio Culurciello. The History of Neural Networks. dataconomy.com. Apr. 19, 2017, https://dataconomy.com/2017/04/history- neural- networks/
- Hsu, David. “Using Convolutional Neural Networks to Classify Dog Breeds”, Stanford University. http://cs231n.stanford.edu/reports/2015/pdfs/fcdh_FinalReport.pdf
- PyTorch Documentation: https://pytorch.org/docs/master/
- Muhammed Talo. “Convolutional Neural Networks for Multi-class Histopathology Image
- Classification”, Munzur University Turkey. https://arxiv.org/ftp/arxiv/papers/1903/1903.10035.pdf