Machine Learning Engineer Nanodegree Udacity

Capstone Project

Vishal Sharma 21st, 2021

Dog Breed Classifier

I. Definition

Project Overview

The requirement of this project is to classification of the different dog breeds. In our world, from east to west and north to south exists a lot of breeds of dogs that differs one to another by a lot of parameters such as size , height, weight ,type of employment, type of intelligence etc.

For our project, the classification is absolutely based on images of the dogs and the goal defined for this is to classify 133 dog breeds using state of art deep learning algorithms to teach and train the computer how to give an estimation of a particular dog breed from an dog image provided.

Image classification is one of the most vibrant and dynamic field in Machine Learning, computer vision and one other very important challenge is the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) where each year review of the new classification algorithms[1].

Problem Statement

The ultimate purpose of this project is to write an algorithm that accepts a file path to an image and first determines and detects whether the image contains a human, dog, or neither.

After a detection process, we could have following:

1. Dog detection in the provided image, return the predicted breed .
2. Human detection in the provided image, return the resembling dog breed.
3. Neither detection in the provided image, provide output that indicates an error.

Metrics

In order to work and deal with a multi-class classification problem, the negative log-likelihood loss function will be used as the evaluation metrics.

Using the negative log-likelihood loss function as evaluation metrics, the algorithm will calculate each iteration and the distance of a predicted output to the corresponding label.

In this manner, the algorithm will learn from it and it will adjust the predictions in order to minimize this distance (that is loss).

II. Analysis

Data Exploration

The datasets required for project are two: dogs dataset and human dataset.

The dogs dataset is composed by following :

- Training: 6680 images
- Validation: 835
- Test: 836
- Total images: 8351
- Classes: 133

The human dataset is composed by 13233 human images. First of all images are resized to 244x244 and then normalized before being used by the model.

Exploratory Visualization

We mention that the task of assigning breed to dogs from images is considered exceptionally challenging. To see why, consider that even a human would have trouble distinguishing between a Brittany and a Welsh Springer Spaniel.

It is not difficult to find other dog breed pairs with minimal inter-class variation (for instance, Curly-Coated Retrievers and American Water Spaniels).

Likewise, recall that Labradors come in yellow, chocolate, and black. The vision-based algorithm will have to conquer this high intra-class variation to determine how to classify all of these different shades as the same breed.

Yellow Labrador	Chocolate Labrador	Black Labrador
		

The random chance presents an exceptionally low bar: setting aside the fact that the classes are slightly not in balance, a random guess will provide a correct answer roughly 1 in 133 times, which corresponds to an accuracy of less than 1%.

Algorithms and Techniques

There are 7 main steps for the solution of the project as follows.

Step 1 : We need to explore and check the datasets in order to understand how to use them and choose the proper algorithms for this.

Step 2 : Then Implementing a Haar feature-based cascade classifier by using OpenCV in order to detect faces in the human dataset provided.

Step 3 : Then I will use a pre-trained VGG16 model so that it can detect dogs in the dogs dataset provided.

Step 4 : Then I will try to create a LeNet[2] like architecture that uses CNN in order to classify the 133 dogs breeds and have an accuracy more than 10% as required.

Step 5 : I will use the transfer learning technique in order to get a pre-trained ResNet50 architecture and then continue the training with the dogs dataset provided. The minimum required accuracy is 60% in the test set given.

Step 6 : I will write a custom algorithm that accepts a file path for the image and first of all it will detect whether the image provided contains a human, dog, or neither.

Step 7 : And also I will Test the Algorithm with some random images found from online.

Benchmark

This model will be compared with various benchmark models in a Kaggle competition (https://www.kaggle.com/c/dog-breed-identification/discussion)

III. Methodology

Data Preprocessing

For the Haar feature-based cascade classifier no data preprocessing is necessary.

For the pre-trained VGG16 model, is used only an image resize transformation to 224 x 224 pixels because is the input size of the network.

In order to increase and uplift the performance of the custom CNN model and the ResNet50 model, I have introduced an ETL pipeline to the dog's dataset where:

1. For all the three datasets i.e. train, validation, test.I put the data a mean to 0 and a standard deviation to1 (i.e. Z-score normalization)

2. For the training dataset is also used the following transformation:

    (i). A Random rotation of a maximum of 30 degrees is done

    (ii). A Random resize crop of 224 x 224pixels is done

    (iii). A Random horizontal flip is done

3. following transformation for the validation dataset is also used:

    (i). Image resize to 256 x 256 pixels

    (ii). Centre crop of 224 x 224 pixels

4. For the test dataset is also used the following transformation:

    (i). Image resize to 224 x 224

After all of this transformation, the data is put into three different data loaders and a batch size of 64 is used.

Implementation

The Haar feature-based cascade classifier is used a computer vision pre-trained face detector.

For the VGG16 pre-trained model is used PyTorch in order to make the predictions.

For the custom CNN model, I consider and followed the LeNet architecture and I split the task into two main parts:

For classification, some fully connected layers are used.

For the feature, extraction is used some convolution layers.

I also implemented maxpooling in order to reduce the dimension of the layers and the dropout in order to prevent overfitting.

The loss function used for Cross Entropy.

The optimizer used for: Stochastic gradient descent (SGD).

The learning rate is set to 0.05.

Here the net summary:

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))

For the last model (ResNet50), I have changed the last layer of the architecture to:

Linear(in_features=2048, out_features=133, bias=True)

This modification is made because the output nodes of ResNet50 is 2048 and whereas we have 133 dog breeds.

The loss function used is: Cross Entropy.

The optimizer used is: Stochastic gradient descent (SGD).

The learning rate is set to 0.001.

Refinement

Especially for the custom CNN and ResNet50 models, some hyperparameters tuning is made during all the training and design processes.

These are some parameters tuned during the process: Training epochs, dropout value, learning rate, etc.

IV. Results

Model Evaluation and Validation

All the models trained meet the expectation end the results are:

1. For the pre-trained VGG16 the dogs detected in dogs images is the 98.0%.
2. For the custom CNN model, the accuracy is: 17% (143/836)
3. For the ResNet50 model, the accuracy is: 75% (635/836)

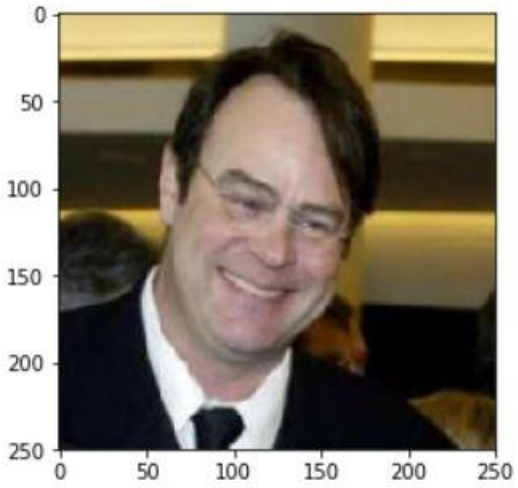
Justification

All the model trained and meet the expectation of the project requirement but there is a room for improvement in respect to the benchmark.

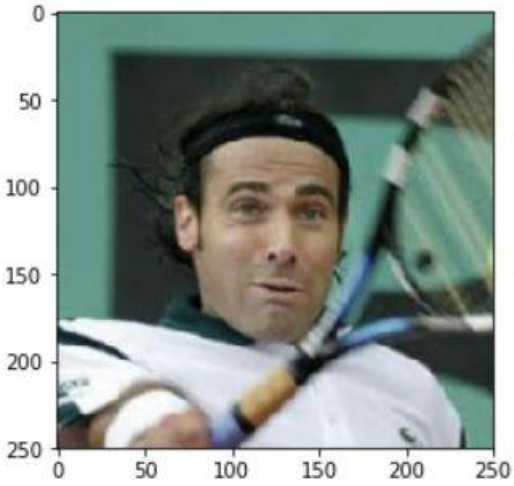
V. Conclusion

Free-Form Visualization

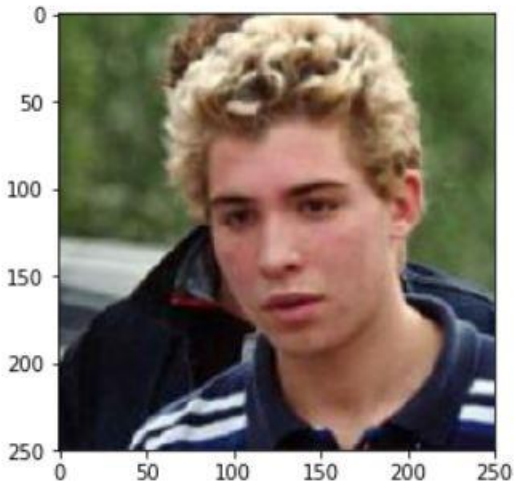
Following are some results from the final algorithm that combine the face detector with the ResNet50 dog breeds model:



Face detected! Looks like a Dachshund

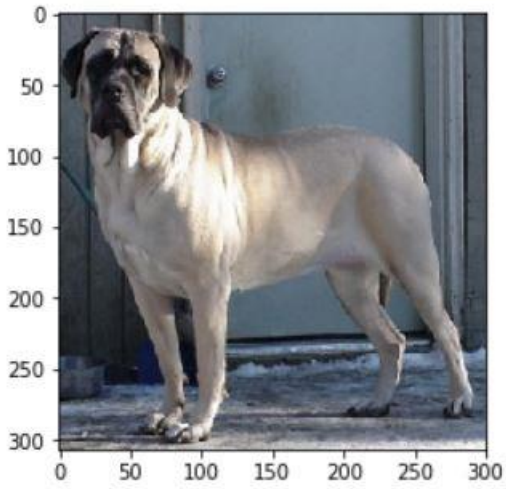


Face detected! Looks like a Dachshund

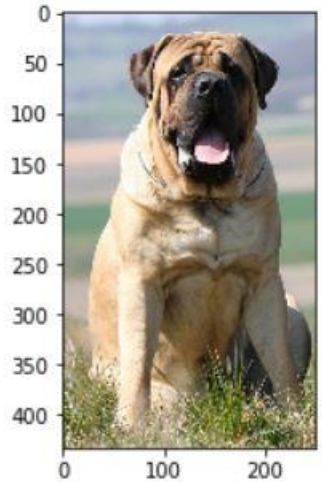


Face detected! Looks like a American water spaniel

Dogs Detected: Bullmastiff



Dogs Detected: Bullmastiff



Dogs Detected: Bullmastiff

Reflection

So finally ,The solution designed for this project has followed these steps:

Step 1: Datasets exploration and check.

Step 2: Determines and detect Humans using a Haar feature-based cascade classifiers

Step 3: Determines and detect Dogs using a pre-trained network

Step 4: Create a CNN for Classification of Dog Breeds (from Scratch) using a LeNet like architecture.

Step 5: Create a CNN for Classification of Dog Breeds using Transfer Learning and a using a ResNet50 architecture.

Step 6: Write a custom Algorithm that accepts a file path to an image and first of all it will determines whether the image contains a human, dog, or neither.

Step 7: Test the Algorithm with some random sample images found online.

In this project, I have used a lot of different approaches for managing the dog breeds classification task and this final solution, with a little bit more fine tuning, could be used in a general setting to solve these kind of tasks.

Improvement

For this the dog breeds classification task ,possible scenario to improve the model in my opinion are as follows:

1. We have to try some different model architectures like resnext and other like this.

2. We have to try hyper-parameters fine tuning (by changing the optimizer, batch size, learning rate, etc.)

3. By improving the dataset with more images and more data augmentation training.

Reference

1. https://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html

2. Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015

3. https://en.wikipedia.org/wiki/LeNet

4. https://hackernoon.com/a-brief-history-of-computer-vision-and-convolutional-neural-networks-8fe8aacc79f3

5. https://medium.com/nanonets/how-to-easily-build-a-dog-breed-image-classification-model-2fd214419cde