Machine Learning Engineer Nanodegree

Capstone Project

Dog Breed Classifier using CNN

Parul Agarwal

8th August,2020

Project Overview

The Dog breed classifier is a well-known problem in ML. The problem is to identify a breed of dog if dog image is given as input, if supplied an image of a human, we have to identify the resembling dog breed. The idea is to build a pipeline that can process real world user supplied images and identify an estimate of the canine's breed. This is a multi-class classification problem where we can use supervised machine learning to solve this problem. After completing this model, I am planning to build a web app where user can input an image and obtain prediction from this model. This project gives me an opportunity to build and deploy ML models, so I have chosen this as my capstone project.

Problem Statement

The objective of the venture is to assemble an machine learning model that can be utilized inside web application to process genuine world, client provided pictures. The calculation needs to perform two undertakings:

Dog face detector: Given an image of a dog, the algorithm will identify an estimate of the canine's breed.

Human face detector: If supplied an image of a human, the code will identify the resembling dog breed.

Metrics

The information is part into train, test and legitimate dataset. The model is prepared utilizing the train dataset. We utilize the testing information to anticipate the presentation of the model on concealed information. We will utilize precision as a measurement to assess our model on test information.

Accuracy=Number of items correctly classified/ All classified items

Likewise, during model preparing, we contrast the test information forecast and approval dataset and figure Multi class log misfortune to locate the best performing model. Log misfortune assesses vulnerability of expectation dependent on the amount it fluctuates from real name and this will help in assessing the model.

Datasets and Inputs

For this task, the information group must be of picture type, since we need to enter a picture and distinguish the variety of the canine. The dataset for this task is given by Udacity. The dataset has pictures of canines and people.

Canine images dataset: The canine picture dataset has 8351 all out pictures which are arranged into train (6,680 Images), test (836 Images) and substantial (835 Images) registries. Each of this registry (train, test, substantial) have 133 organizers relating to canine varieties. The pictures are of various sizes and various foundations, a few pictures are most certainly not full-sized. The information isn't adjusted on the grounds that the quantity of pictures accommodated each breed changes. Few have 4 pictures while some have 8 pictures.

Human images dataset: The human dataset contains 13233 complete human pictures which are arranged by names of human (5750 organizers). All pictures are of size 250x250. Pictures have diverse foundation and various edges. The information isn't adjusted since we have 1 picture for certain individuals and numerous pictures for a few.

Sample images in the dataset

Algorithms and Techniques

For playing out this multiclass grouping, we can utilize Convolutional Neural System to take care of the issue. A Convolutional Neural Network (CNN) is a Deep Learning calculation which can take in an info picture, dole out significance (learnable loads and inclinations) to different angles/objects in the picture and have the option to separate one from the other. The arrangement includes three stages. To start with, to distinguish human pictures, we can utilize existing calculation like OpenCV's usage of Haar highlight based course classifiers. Second, to recognize canine pictures we will utilize a pretrained VGG16 model. At last, after the picture is distinguished as canine/human, we can pass this picture to a CNN which will process the picture and anticipate the variety that coordinates the best out of 133 varieties.

Benchmark

The CNN model created from scratch must have accuracy of at least 10%. This can confirm that the model is working because a random guess will provide a correct answer roughly 1 in 133 times, which corresponds to an accuracy of less than 1%. The CNN model created using transfer learning must have accuracy of 60% and above.

Data Preprocessing

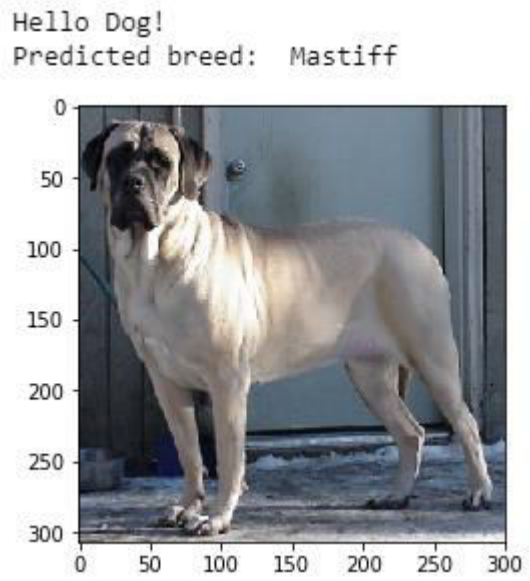
All the pictures are resized to 224*224, at that point standardization is applied to all pictures (train, substantial and test datasets). For the preparation information, Image growth is finished to lessen overfitting. The train information pictures are arbitrarily pivoted and irregular even flip is applied. At long last, all the pictures are changed over into tensor previously going into the model.

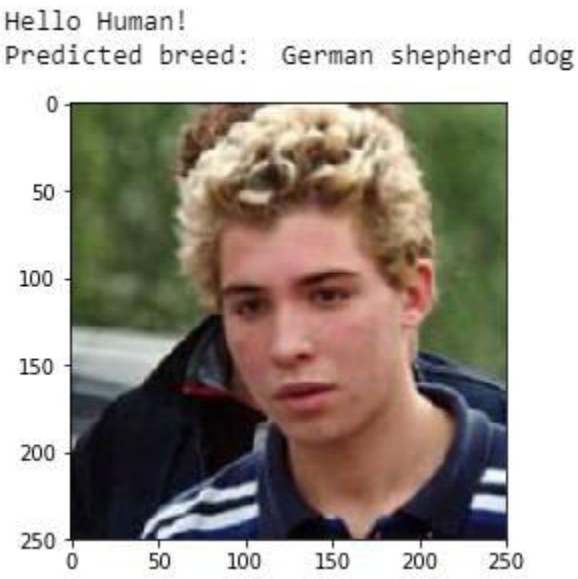
Implementation

I have manufactured a CNN model without any preparation to take care of the issue. The model has 3 convolutional layers. All convolutional layers have portion size of 3 and step 1. The first conv layer (conv1) takes the 224*224 information picture and the last conv layer (conv3) produces a yield size of 128. ReLU initiation work is utilized here. The pooling layer of (2,2) is utilized which will decrease the info size by 2. We have two completely associated layers that at long last delivers 133-dimensional yield. A dropout of 0.25 is added to maintain a strategic distance from over overfitting.

Refinement

The CNN made without any preparation have precision of 13%, Though it meets the benchmarking, the model can be altogether improved by utilizing move learning. To make CNN with move learning, I have chosen the Resnet101 engineering which is pre-prepared on ImageNet dataset, the engineering is 101 layers profound. The last convolutional yield of Resnet101 is taken care of as contribution to our model. We just need to add a completely associated layer to deliver 133-dimensional yield (one for each canine classification). The model performed very well when contrasted with CNN from scratch. With only 5 ages, the model got 81% exactness.





Sample predictions of this model

Model Evaluation and Validation

Human Face detector: The human face detector function was created using OpenCV's implementation of Haar feature based cascade classifiers. 98% of human faces were detected in first 100 images of human face dataset and 17% of human faces detected in first 100 images of dog dataset.

Dog Face detector: The dog detector function was created using pre-trained VGG16 model. 100% of dog faces were detected in first 100 images of dog dataset and 1% of dog faces detected in first 100 images of human dataset.

CNN using transfer learning: The CNN model created using transfer learning with ResNet101 architecture was trained for 5 epochs, and the final model produced an accuracy of 81% on test data. The model correctly predicted breeds for 680 images out of 836 total images.

Accuracy on test data: 81% (680/836)

Justification

I think the model execution is superior to anticipated. The model made utilizing move learning have a precision of 81% contrasted with the CNN model made without any preparation which had just 13% exactness.

Improvement

The model can be improved by including all the more preparing and test information, right now the model is made utilizing just 133 types of canine. Additionally, by performing more picture expansion, we

can abstain from overfitting and improve the exactness. I have attempted as it were with ResNet 101 design for highlight extraction, May be the model can be improved utilizing diverse engineering.

References

1. Original repo for Project - GitHub: https://github.com/udacity/deep-learning-v2-pytorch/blob/master/project-dog-classification/

2. Resnet101: https://pytorch.org/docs/stable/_modules/torchvision/models/resnet.html#resnet101

3. Imagenet training in Pytorch: https://github.com/pytorch/examples/blob/97304e232807082c2e7b54c597615dc0ad8f6173/imagenet/main.py#L197-L198

4. Pytorch Documentation: https://pytorch.org/docs/master/

5. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neuralnetworks-the-eli5-way-3bd2b1164a53

6. http://wiki.fast.ai/index.php/Log_Loss