Capstone Project
Muhammad Faraz Malik

Machine Learning Engineer Nanodegree
April 16th, 2020

Dog Breed Classifier using CNN

Project Overview

The Dog breed classifier is a notable issue in ML. The issue is to distinguish the type of canine, if a dog picture is given as information, whenever provided a picture of a human, we need to recognize the looking like canine breed. The thought is to manufacture a pipeline that can procedure real world user provided pictures and distinguish a gauge of the canine's breed. This is a multi-class classification problem where we can use supervised machine learning to take care of this issue.

The steps that were followed to work through the project were the following:

Step 0: Import Datasets

Step 1: Detect Humans

Step 2: Detect Dogs

Step 3: Create a CNN to classify Dog Breeds (from scratch)

Step 4: Use a CNN to classify Dog Breeds (using Transfer Learning)

Step 5: Create a CNN to classify Dog Breeds (using Transfer Learning)

Step 6: Write an algorithm

Step 7: Test algorithm

Problem Statement

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

Dog face detator: Given a picture of a dog, the calculation will distinguish a gauge of the canine's breed.

Human face detactor: Whenever provided a picture of a human, the code will distinguish the looking like the canine breed.

Metrics

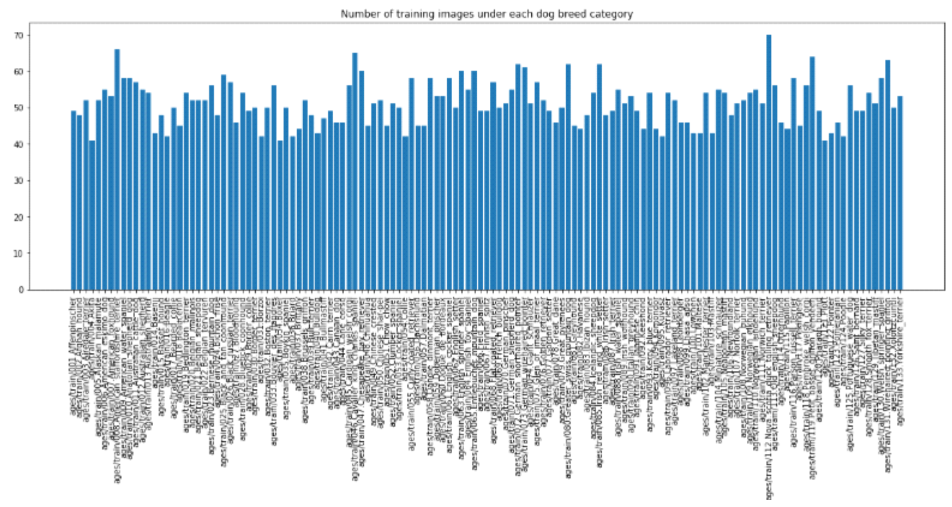
The information is part of the train, test and valid dataset. The model is trained using train dataset. We utilize the testing data to predict the performance of the model on unseen data. We will use accuracy as a measurement to assess our model on test data. Accuracy=Number of things effectively characterized/Every single ordered thing. However, it works well only if there are equal number of samples belonging to each class. In this project, there are total 133 dog breed as class labels. Based on the distribution of training/validation/testing selected, the classes were approximately evenly distributed, except a couple of classes. The number of images under each category seem to be balanced, so we can use accuaracy as our evaluation metric. Likewise, during model training, we compare the test data prediction with the validation dataset and calculate Multi-class log loss to calculate the best performing model. Log loss considers the uncertainity of prediction dependent on the amount it fluctuates from the actual label and this will help in evaluating the model.

Data Exploration

For this task, the input format must be of picture type, since we need to enter a picture and distinguish the type of the dog. The dataset has pictures of dogs and humans:

Dog pictures dataset: The canine picture dataset has 8351 complete pictures that are arranged into a train (6,680 Pictures), test (836 Pictures) and valid (835 Pictures) directories. Every one of these indexes (train, test, valid) has 133 folders corresponding to dog breeds. The pictures are of various sizes and various backgrounds, a few pictures are not full-sized.

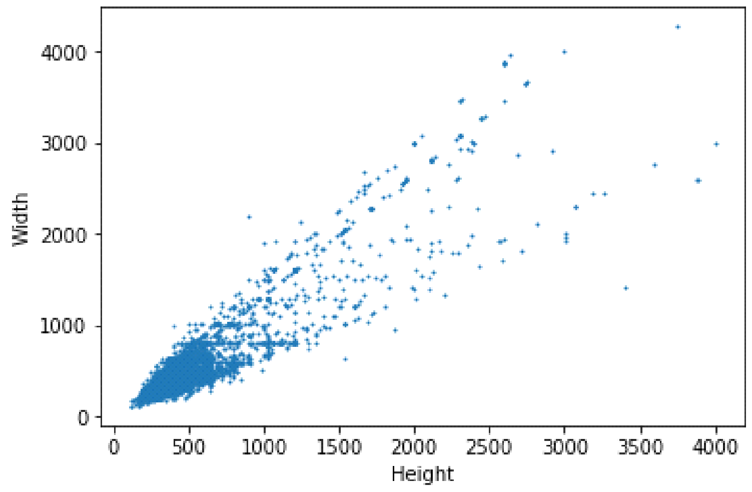
Human pictures dataset: The human dataset contains 13233 total human pictures which are arranged by names of human (5750 folders). All pictures are of size 250x250. Pictures have various backgrounds and various angles. The Humans dataset, additionally given by Udacity, contained 13233 pictures. A bar chart was drawn to figure out the balance of the classess.



As you can see above, the dataset is balanced with an average of 50 images per dog breed class. Since this is a classification problem, I chose to use accuracy as an evaluation metric. If the dataset was overly imbalanced, then precision would have been an appropriate choice.

Now, have a look at some of the dog images:

The individual images were of different sizes and orientations, but most of them were in the range of 300-500 pixels in height and width. As a part of pre-processing step these images were resized to 224x224 pixels to fit the network architecture. The distribution of height and width is shown below using scatter plot.



Algorithms and techniques

For playing out this multiclass order, we can utilize the Convolutional Neural System to tackle the issue. A Convolutional Neural Network (CNN) is a Deep Learning algorithm that takes in a dog image as input image, allot significance (learnable weights and biases) to different aspects/objects in the image and be able to separate one from the other and outputs the breed of the dog. If a human image is given, the algorithm returns the dog name that most resembles to that human. CNNs represent a huge breakthrough in image recognition. They're most commonly used to analyze visual imagery and are frequently working behind the scenes in image classification. The solution involves three stages. To begin with, to identify human images, we can utilize existing calculations like OpenCV's usage of Haar feature based cascade classifiers. Second, to detect dog images we will utilize a pre-trained VGG16 model. At last, after the picture is distinguished as dog/human, we can pass this picture to a CNN model which will process the image and anticipate the breed that coordinates the best out of 133 breeds.

First a CNN was created from scratch that involved some steps that were taken to get to the final CNN architecture are as follows: The model has 3 convolutional layers. All convolutional layers have a kernel size of 3 and stride 1. The principal spread layer (conv1) has in_channels =3 and the last Conv layer (conv3) produces an output size of 128. The ReLU actuation work is utilized here. The pooling layer of (2,2) is utilized which will lessen the information size by 2. We have two completely associated layers that at long last produce 133-dimensional yield. A dropout of 0.25 is added to abstain from overfitting.

Then again a CNN was created, but this time using transfer learning to classify the dog breed. I have chosen to utilize the resnet101 engineering which is pre-prepared on the Imagenet dataset, The design is 101 layers deep, inside only 5 epochs, the model got 74% accuracy. In the event that we train for more epochs, the accuracy can be altogether improved.

Steps:

1) Import pre-trained resnet101 model

2) Change the out_features of a completely associated layer to 133 to solve the classification problem

3) CrossEntropy loss function is picked as the loss function.

Prepared for 5 epochs and got 74% .

Benchmark

To detect dogs in images the VGG-16 model, along with weights that have been trained on ImageNet, a very large, very popular dataset used for image classification and other vision tasks. ImageNet contains over 10 million URLs, each linking to an image containing an object from one of 1000 categories. Given an image, this pre-trained VGG-16 model returns a prediction (derived from the 1000 possible categories in ImageNet) for the object that is contained in the image. The CNN model made from scratch must have an accuracy of at least 10%. This can confirm that the model is working because a random guess will provide an answer about 1 in 133 time, which corresponds to an accuracy of under 1%. For CNN using transfer learning, I have chosen to utilize the resnet101 model which is pre-trained on the Imagenet dataset.

Data Preprocessing

All the images are resized to 224*224, then normalization is applied to all pictures (train, legitimate and test datasets). For the training data, image augmentation is done to diminish overfitting. The train data images are arbitrarily rotated and random horizontal flip is applied. At last, all the images are converted into tensor before passing into the model. A large portion of the Preprocessing models like VGG16 takes the size (224,224) as information, so I have utilized this size.

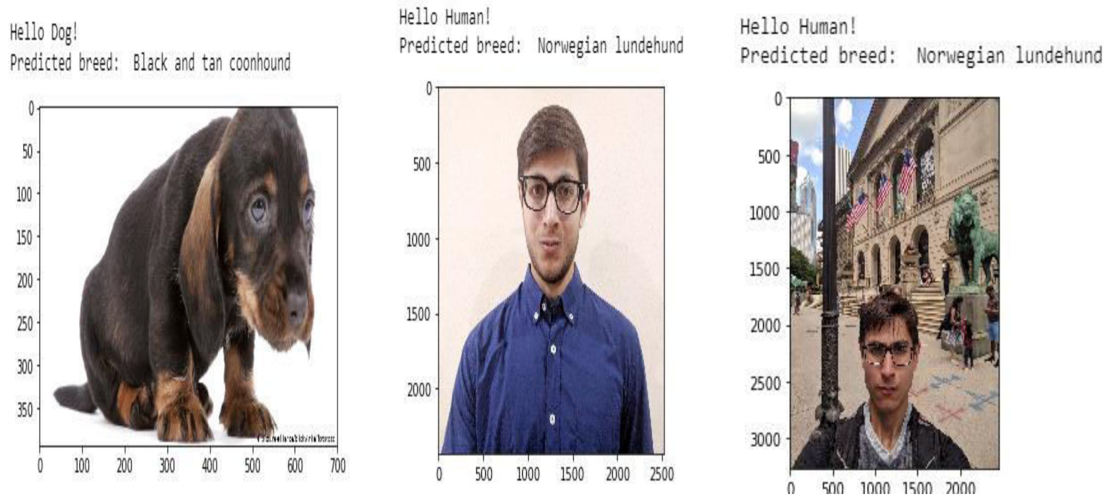
For train information, I have done picture enlargement to abstain from overfitting the model. Changes utilized: Arbitrary resize yield to 224, irregular flipping and arbitrary revolution.For approval and test information, I have done just picture resizing. I have applied standardization to each of the three datasets.

Implementation

I have assembled a CNN model from scatch to take care of the issue. The model has 3 convolutional layers. All convolutional layers have a kernel size of 3 and stride 1. The first conv layer (conv1) takes the 224*224 input image and the last conv layer (conv3) produces an output size of 128. The ReLU activation function is utilized here. The pooling layer of (2,2) is utilized which will diminish the input size by 2. We have two completely connected layers that at long last produce 133-dimensional output. A dropout of 0.25 is added to abstain from overfitting.

Refinement

The CNN made from scratch has an accuracy of 11%, though, it meets the benchmarking, the model can be essentially improved by using transfer learning. To create CNN with transfer learning, I have chosen the Resnet101 architecture which is pre-trained on the ImageNet dataset, the architecture is 101 layers profound. The last convolutional model of Resnet101 is fed as input to our model. We just need to add a completely connected layer to deliver a 133-dimensional output (one for each dog class). The model performed incredibly well when compared with CNN from scatch. With only 5 epochs, the model got 74% precision.


Hello Dog!
Predicted breed: Black and tan coonhound
Hello Human!
Predicted breed: Norwegian lundehund
Hello Human!
Predicted breed: Norwegian lundehund

Sample outputs predicted using the model

Model Evaluation and Validation

Human Face detector: The human face detector function was made utilizing OpenCV's usage of Haar feature based cascade classifiers. 98% of human faces were detected in the initial 100 pictures of the human face dataset and 17% of human appearances recognized in the initial 100 pictures of the dog dataset.

Dog Face detactor: The dog detector function was made utilizing a pre-trained VGG16 model along with weights that have been trained on ImageNet dataset. 100% of dog were detected in the initial 100 pictures of the dog dataset and 1% of dog faces detected in the initial 100 pictures of the human dataset.

CNN using transfer learning: I have chosen to utilize the resnet101 model which is pre-trained on the

Imagenet dataset. The CNN model made using transfer learning with ResNet101 architecture was trained for 5 epochs, and the final model produced an accuracy of 74% on test data. The model correctly predicted breeds for 621 pictures out of 836 complete pictures.

Accuracy on test data: 74% (621/836)

Justification

I think the model performance is superior to anticipated. The model made utilizing transfer learning has an accuracy of 74% compared with the CNN model made from scratch which had just 11% accuracy.

Improvement

The model can be improved by adding more training and test data, presently, the model is made created using just 133 breeds of the dog. Additionally, by performing more picture expansion, we can abstain from overfitting and improve accuracy. I have attempted uniquely with ResNet 101 architecture for feature extraction, Possibly the model can be improved utilizing the distinctive architcture.

References

1. Original repo for Project - GitHub:
https://github.com/udacity/deep-learning-v2pytorch/blob/master/project-dog-classification/

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

3.Imagenet trainingin 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/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a

6. Dog dataset: https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/dogImages.zip

7. Human dataset: https://s3-us-west-1.amazonaws.com/udacity-aind/dog-project/lfw.zip

8. Sample testing images:
https://github.com/farazmalik180/Final-Capstone-Udacity-NanoDegree/tree/master/My_images