MACHINE LEARNING ENGINEER NANODEGREE

Capstone Report

Dog Breed Classifier with CNNs

Navjot Singh Bajaj

(8th August 2020)

1. 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.

2. Problem Statement

The thought is that we snap a photo with a telephone and the application mentions to us what canine variety is on the image. Also, we need that application to advise us to what exactly canine variety a human is most clone. We need to have a response to a couple of inquiries here. The principle question is What canine variety is on the image? The subsequent inquiry is: How might you look on the off chance that you were a canine? To respond to these two inquiries we initially need to address the inquiry: Is on the image human or a canine? This is a managed learning issue and on the grounds that we have our canine pictures isolated into breed classes we will utilize grouping prescient displaying all the more decisively multi-class prescient model.

3. Matrics

The provided data is split into three datasets which are train, test and valid dataset. The model is trained using the train dataset. We use the testing data to predict the performance of the model on unseen or new data. We will use accuracy as a metric to evaluate our model on test data. The formula is

Accuracy=Number of items correctly classified/ All classified items

Also, during model training, we compare the test data prediction with validation dataset and calculate Multi class log loss to find the best performing model. Log loss takes into the account of uncertainty of prediction based on how much it varies from actual label and this will help in evaluating the model.

4. Datasets and Inputs

To take care of our concern our info information must be pictures since we need to client takes a picture of a canine (or human) with his telephone, sent it to our worker and we would return what canine variety is in all probability in an image (or to which canine variety is human most resemble). All information for this task is given by Udacity. We have pictures of canines and pictures of people. All canine pictures are arranged in train(6,680 Images),

test(836 Images) and valid(835 Images) catalog, and all the pictures in these indexes are arranged in breed registries. We have 133 organizers (canine varieties) in each train, test and substantial index. Human pictures are arranged by name of every human. We have 13,234 Files (Images), 5,749 Folders(Humans)

Our information isn't adjusted in light of the fact that we have one picture of certain individuals and a few for other people. The equivalent is for canine pictures. (the thing that matters is from 1 to 9 pictures by and large). Canine pictures have diverse picture sizes, various foundations, a few canines are in full sizes and some simply ahead. Lightning isn't the equivalent. That is quite on the grounds that we don't have a clue how clients' pictures will be, and we need that our model chips away at various kinds of pictures. Human pictures are the entirety of a similar size 250×250.

Sample Images from the dataset

5. Algorithms and techniques

We will utilize Convolutional Neural Networks (CNN) to make a model. For performing this multiclass classification, we can use Convolutional Neural Network to solve the problem. A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The solution involves three steps. First, to detect human images, we can use existing algorithm like OpenCV's implementation of feature based cascade classifiers. Second, to detect dog-images we will use a pretrained VGG16 model. Finally, after the image is identified as dog/human, we can pass this image to an CNN model which will process the image and predict the breed that matches the best out of 133 breeds.

6. 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.

7. Data Preprocessing

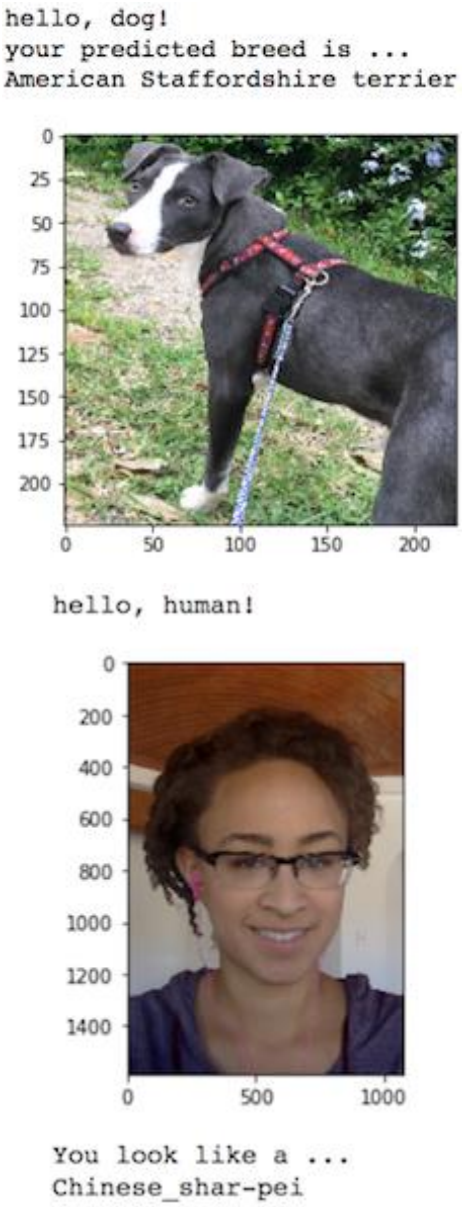
All the pictures are resized to 224*224, at that point standardization is applied to all pictures (train, substantial and test datasets). For the preparation information, Image growth is finished to lessen overfitting. The train information pictures are arbitrarily pivoted and irregular even flip is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

8. Implementation

I have made a CNN model without any planning to deal with the issue. The model has 3 convolutional layers. All convolutional layers have divide size of 3 and stage 1. The first conv layer (conv1) takes the 224*224 data picture and the last conv layer (conv3) produces a yield size of 128. ReLU inception work is used here. The pooling layer of (2,2) is used which will diminish the data size by 2. We have two totally related layers that finally conveys 133-dimensional yield. A dropout of 0.25 is added to keep up a key good ways from over overfitting.

9. Refinement

With 10 epochs, The CNN made without any planning have accuracy of 21% (183/836),Though it meets the benchmarking, the model can be inside and out improved by using move learning. To make CNN with move learning, I have picked the Resnet101 building which is pre-arranged on ImageNet dataset, the designing is 101 layers significant. The last convolutional yield of Resnet101 is dealt with as commitment to our model. We simply need to add a totally related layer to convey 133-dimensional yield (one for every canine arrangement). The model performed very well when diverged from CNN without any preparation. With just 10 epochs, the model got 84% accuracy.



Sample Predictions of this Model

10. 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 10 epochs, and the final model produced an accuracy of 84% on test data. The model correctly predicted breeds for 709 images out of 836 total images.

Accuracy on test data: 84% (709/836)

11. Justification

I think the model execution is better than foreseen. The model made using move learning have an accuracy of 84% diverged from the CNN model made without any planning which had quite recently 21% accuracy.

12. Improvement

The model can be improved by including all the all the more planning and test data, at the present time the model is made using only 133 sorts of canine. Also, by performing more picture extension, we can swear off overfitting and improve the accuracy. I have endeavored in a manner of speaking with ResNet 101 plan for feature extraction, May be the model can be improved using assorted building.

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