Machine Learning Nanodegree Capstone Report:

Dog Breed Classifier using CNN

Anurag Gupta

02-July-2021

Project Overview:

The Dog breed classifier is a well-known problem in ML. The problem is to identify the 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.

Problem Statement:

The goal of the project is to build a machine learning model that can be used within web app to process real-world, user-supplied images. The algorithm has to perform two tasks:
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 data is split into 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 data. We will use accuracy as a metric to evaluate our model on test data.

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.

Data Exploration:

For this project, the input format must be of image type, because we want to input an image and identify the breed of the dog. The dataset has pictures of dogs and humans.

Dog images dataset: The dog image dataset has 8351 total images which are sorted into train (6,680 Images), test (836 Images) and valid (835 Images) directories. Each of this directory (train, test, valid) have 133 folders corresponding to dog breeds. The images are of different

sizes and different backgrounds, some images are not full-sized. The data is not balanced because the number of images provided for each breed varies. Few have 4 images while some have 8 images.

Human images dataset: The human dataset contains 13233 total human images which are sorted by names of human (5750 folders). All images are of size 250x250. Images have different background and different angles. The data is not balanced because we have 1 image for some people and many images for some.

Sample images from the dataset

Dog images have different image sizes, different backgrounds, some dogs are in full sizes and some just ahead. Lightning is not the same. That is actually ok because we don't know how users' images will be, and we want that our model works on different types of images. Human images are all of the same size 250×250. Images are with different backgrounds, light, from different angles, sometimes with few faces on the image. Here are a few samples of our dog and human images:

Algorithms and techniques:

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 Haar 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.

Benchmark:

For our benchmark model, we will use the Convolutional Neural Networks (CNN) model created from scratch with an accuracy of more than 10%. This should be enough to confirm that our model is working because random guess would be 1 in 133 breeds which are less than 1% if we don't consider unbalanced data for our dog images.

Data Pre-processing:

All the images are resized to 224*224, then normalization is applied to all images (train, valid and test datasets). For the training data, Image augmentation is done to reduce overfitting. The train data images are randomly rotated and random horizontal flip is applied. Finally, all the images are converted into tensor before passing into the model.

Implementation:

I have built a CNN model from scratch to solve the problem. I have two approaches, in the second one where I got accuracy of 13% contains 5 convolutional layers. All convolutional layers have kernel size of 3 and stride 1. The first conv layer (conv1) takes the 224*224 input image and the final conv layer (conv5) produces an output size of 128. ReLU activation function is used here. The pooling layer of (2,2) is used which will reduce the input size by 2. We have two fully connected layers that finally produces 133-dimensional output. A dropout of 0.25 is added to avoid over overfitting.

Refinement:

The CNN created from scratch have accuracy of 13%, Though it meets the benchmarking, the model can be significantly improved by using transfer learning. To create CNN with transfer learning, I have selected the Resnet50 architecture which is pre-trained on ImageNet dataset, the architecture 2048 features. The last convolutional output of Resnet50 is fed as input to our model. We only need to add a fully connected layer to produce 133-dimensional output (one for each dog category). The model performed extremely well when compared to CNN from scratch. With 15 epochs, the model got 86% accuracy.

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 ResNet50 architecture was trained for 15 epochs, and the final model produced an

accuracy of 86% on test data. The model correctly predicted breeds for 720 images out of 836 total images.
Accuracy on test data: 86% (720/836)

Justification:

I think the model performance is better than expected. The model created using transfer learning have an accuracy of 86% compared to the CNN model created from scratch which had only 13% accuracy.

Improvement:

The model can be improved by adding more training and test data, currently the model is created using only 133 breeds of dog. Also, by performing more image augmentation, we can avoid overfitting and improve the accuracy. I have tried only with ResNet 50 architecture for feature extraction, May be the model can be improved using different architecture.

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-neural-networks-the-eli5-way-3bd2b1164a53

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