Capstone Project

Machine Learning Engineer Nanodegree

Dhanush

May 24th, 2020

Dog Breed Classifier Using CNN

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 apps 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 account the 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) has 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 humans (5750 folders). All images are of size 250x250. Images have different backgrounds and different angles. The data is not balanced because we have 1 image for some people and many images for some.

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 algorithms 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

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%.

Data Preprocessing

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.

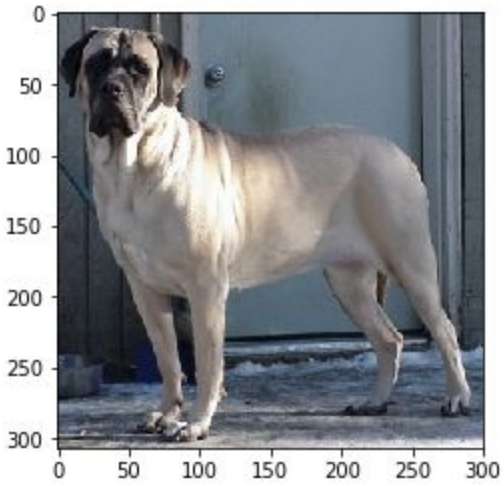
Implementation

I have built a CNN model from scratch to solve the problem. The model has 3 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 (conv3)

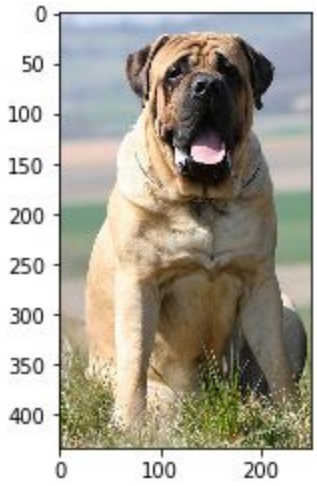
produces an output size of 128. ReLU activation function is used here. The pooling layer of (2,2) is used which will reduce the input size by 2. We have two fully connected layers that finally produce 133-dimensional output. A dropout of 0.3 is added to avoid overfitting.

Refinement

The CNN created from scratch has accuracy of 18%, 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 Resnet101 architecture which is pre-trained on ImageNet dataset, the architecture is 101 layers deep. The last convolutional output of Resnet101 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 just 10 epochs, the model got 80% accuracy.




```
Dogs Detected!
It looks like a Mastiff
```




```
Dogs Detected!
It looks like a Bullmastiff
```

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 the 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 a pre-trained VGG16 model. 100% of dog faces were detected in the first 100 images of dog dataset and 0% 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 for10 epochs, and the final model produced an accuracy of 80% on test data. The model correctly predicted breeds for 676 images out of 836 total images.

Accuracy on test data: 80% (676/836)

Justification

I think the model performance is better than expected. The model created using transfer learning has an accuracy of 80% compared to the CNN model created from scratch which had only 18% 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 101 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. Pytorch Documentation: https://pytorch.org/docs/master/