Udacity Machine Learning Engineer

Nanodegree Project Report

Pyae Phyo Kyaw

October 28, 2020

Dog Breed Classification

Project Overview

Dog Breed Classification project is one of the interesting and famous ML project. We use supervised machine learning to solve this problem. We input one human image and our program test it and give back the resulted output it is human and similarly, when we put the dog image as an input and the problem is identify the breed of dog. This is pretty big project and we need to use a good method like transfer learning and use some interesting algorithm.

Problem Statement

The project will detect dog and human and predict dog's breed. The steps are:

1. Import dog dataset and human dataset and preprocess the image and use augmentation techniques to prevent overfitting.
2. Detect human
3. Detect Dogs
4. Create a CNN model to Classify Dog Breeds (from scratch)
5. Create CNN model to Classify Dog Breeds (using Transfer Learning)
6. Write your Algorithm
7. Test Your Algorithm

The output will be the resulted image and print out the input image is dog or human and dog's breed, even if the detect image is human,

Metric

The data is trained into train validate and test dataset liked Machine Learning approach. The dataset I got was already split up and so I don't need to do it manually. The model is trained using the train dataset. We use the testing data to predict the performance of the model on the unseen data. The evaluation metric is the accuracy. To get the best trained model during training, I compared the best loss with the validation loss and think as best trained model so far if validation loss is less than best loss.

Best loss = min (Best loss, Validation loss)

Accuracy = corrected prediction / size of dataset

Analysis

Data Exploration

Human Datasets

Human pictures are sorted by name of each human. We have 13233 images of 5750 people with 250 * 250 pixels in all images. Some people have only one image but some are not. We want to detect human and going to use OpenCV library, I won't split it into train, valid and test.

Dog Breed Datasets

There are 8531 dog images of 133 breeds. Dog images have different images, different background, some dogs are in full sizes and some just ahead. Lighting is not the same. I prefer using 6680 images in train, 836 images in valid and 835 images in test. The major goal of this project is to classify dog breed, I'll create CNN both from scratch and transfer learning.

Sample data of human and dog

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 input image, assign importance (learnable weights and biases) to various aspects in the image and be able to differentiate one from other. Firstly, we used OpenCV's implementation of Haar feature-based cascade classifiers to detect human in an image. OpenCV provide many haarcascades. I use this method and detect imaged with multiscale method. Then, I detect dog with using pertained VGG16 architecture. 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 Preprocessing

In implementation of CNN, the first step in preprocessing is resizing to (224 * 224) because I want to follow the VGG16 architecture so I need to gradually reduce the image size into half by half and finally reached to (7 * 7) in the fully connected layer. Also, I need (224 * 224) in transfer learning as resnet101 that I used is trained with this image size. Moreover, I add other augmentation techniques such as RandomHorizontalFlip and RandomRotation not to be overfitting. Finally, all images are converted to tensor before passing into the model.

Implementation

I built a CNN architecture for multiclass classification; this architecture is based on VGG16 architecture. I decreased the image by half in every layer to fully connected layer. I used 64, 128, 256, and 512 filters respectively in layers. I set kernel size 3 and stride 1. The first conv layer takes the 224 * 224 input image and the final conv layer 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. I also added dropout with p = 0.5 to avoid overfitting. I trained the model with 40 epochs and the validation loss is not good enough. I wanted to train with more epochs but it would take too long. So I got that I need to refine with transfer learning.

Refinement

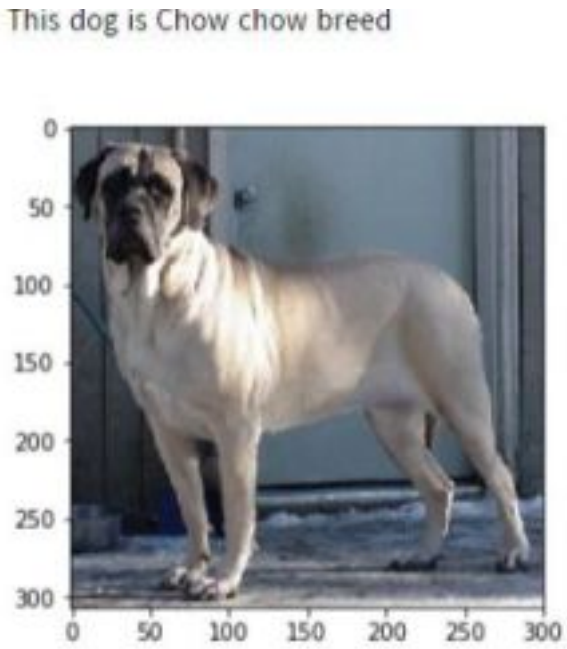
At first I chose VGG16 for transfer learning but after a lot of trails, I found out Restnet 101 can give the best result for me and I could even get a better validation loss. I choose this and I got good accuracy that is much better than threshold. The Resnet101 architecture which is pre-trained on ImageNet dataset, the architecture is 101 layers deep. The last convolutional output of Restnet101 is fed as input to our model. We only need to add a fully connected layer to produce 133-dimensional output. I just modify a bit for training the pretrained model for my dataset. I set parameter.require_grad = False and modified the output channel of fully connected layer to 133 that is the number of class in

my dataset.

Result

Sample Results

For final , I'll show the input image to the user and say "Hello, You looks like a …." If the model detects human image.



Sample out predicted using the model

If it doesn’t detect a dog or human, it’ll say 'Sorry! We cannot detect a human or a dog in this image'.

Model Evaluation and Validation

Human Face Detector: The human face detector function was created using OpenCV’s implementation of Haar feature based cascade classifier. I got 20% of fall short in dog image wrongly detected as human. But I can say that this is good enough for detecting human.

Dog Detector: From directly prediction from pretrained VGG16 model, I got 100 percent dog detection and only one percentage of human detected as dog.

CNN-Scratch: In my own CNN model, performance is not pretty well. Although it decreased training loss in mostly epoch, the validation loss is between 3.8 and 4.2 until

the training finished. I got 13 percent accuracy.

CNN-Transfer Learning: As the pre-trained model that I chose is trained on ImageNet. I can be better than mine. The training loss and validation loss are not that much different and I could see validation loss decreasing in every epoch. The final model produced an accuracy of 88% on the test data.

Test Accuracy: 88% (740/836)

Justification

I think model performance is better than expected. I realized the power of transfer learning. It can save my time a lot and model from transfer learning can beat easily to my scratch-model.

Improvements

The next improvement may be customized CNN architecture instead of using transfer learning only. I'll add more layer to my own model and integrate with transfer learning. Moreover, I'll add more data to the dataset to get a better accuracy.

References

1. Learn CNN from Deep Learning Specialization course by Andrew Ng

2. Restnet101
   https://www.kaggle.com/pytorch/resnet101
3. OpenCV Course
   https://www.alexsnowschool.org/courses/take/opencv-with-python
4. Artificial Neural Network by Zaw Min Khaing (book)