Machine Learning Engineer Nanodegree

Capstone Proposal

Yitao Hu
August 6th, 2019

CNN Project: Dog Breed Classifier Proposal

Domain Background

Image recognition is an area of almost unlimited possibilities, whose project usually plays a crucial role in the state-of-art machine learning pipelines. For instance, one key step in self-driving car projects is to make the autonomous car effectively identify various objects including pedestrians, traffic lights, and other vehicles. Another computer vision application, which shares a greater similarity of this project, is human facial recognition because identifying a person requires somehow similar methodologies of detecting dog breed like extracting and comparing the nose, eyes, and ears.

For image recognition, a convolutional neural network or CNN classifier is a common practice, as it utilizes filters to extract high-level features and pooling layers to perform dimension reduction, and thus "(it) lead to savings in memory requirements …(and) even outperform humans in cases such as classifying objects into fine-grained categories such as…breed of dog or species of bird", according to a paper from IP Group, Cadence. In the same essay, the researchers also mentioned several cases, in which a CNN classifier achieved the highest correct detection rates, including a 99.77% in MNIST handwriting recognition, a 97.47% with the NORB dataset. Therefore, although this project aims to merely build a CNN dog breed classifier, the same hypothesized model, learning algorithm, and even learned parameters can be generalized to other broad multi-class fine-grained photo classification projects.

Problem Statement

The problem to be solved in this project is to build an algorithm to be used in a web or smartphone app. This algorithm takes an image as an input and returns an estimated dog breed if the image is detected of a dog, or returns a dog breed the human look like most if a human face is detected. As dogs of various breeds might look similar, and, conversely, dogs from the same breed may look differently, developing a dog breed classifier which performs better than humans will extract great insights for other fine-grained classification projects.

Datasets and Inputs

The project will be developed from the ipython Jupyter Notebook "dog_app", provided by Udacity team, which contains some template codes, guidelines, and suggestions on implementation. Based on that, I will add additional code or markdown cells for other steps such as data exploration and pre-processing. All the image data are provided by Udacity team, including 13,233 human and 8,351 labeled dog images of 133 breeds. Besides these data, I will use 6 pictures of my portraits and took in my life for testing the final algorithm.

The dog images are divided into 6,680 in training set, 835 in the validation set, and 836 in the test set, while the human images are purely used to test the human face detector. As for the formats, the dog images are RBG jpg with size ranging from hundreds by hundreds to thousands by thousands of pixels, whereas human images are uniformly 256 by 256. All the dog images are pre-processed, including resizing and augmentations, before used in training, validation, and testing.

Solution Statement

The solution to the dog classification problem includes two classifiers. The first classifier, which detects the existence of a human face or a dog in the image, will be a pre-trained OpenCV's and Pytorch VGG16 CNN classifier provided by Udacity team for this project. For the second dog breed classifier, following guided steps provided by Udacity team, I will use transfer learning build a classifier transferred from Pytorch pre-trained models. Finally, I will evaluate the performance of models transferred from different nets on test Cross-Entropy

Loss and accuracy, and then decide which one to be used in the final algorithm.

Benchmark Model

The benchmark in this project will be a simple-structure CNN model I will develop from scratch, whose performance will later be compared with that of transfer learned model on test accuracy and Cross-Entropy Loss. Because a plain vanilla CNN provides an unbiased benchmark for further possible improvements, I would argue such a benchmark is appropriate to test how much the model performance can be improved by transfer learning.

Evaluation Metrics

The evaluation metrics will be the test multi-class Cross-Entropy Loss, which is also the optimization object, and test accuracy. The multi-class Cross-Entropy Loss is defined by the formula from Pytorch organization:
$\text{Loss}\,(\text{x}, \text{class}) = -\text{x}[\text{class}] + \log(\text{j}\sum\exp(\text{x}[\text{j}]))$. For the test accuracy, it is the percentage of correctly classified images in the test set.

Project Design

The first step or step 0 of this project is to import dataset and perform some data exploration. Because I plan to work in Udacity workplace, in which the dog and human images are already downloaded in the corresponded directory, I will not need re-download the images. After importing the image directories, I am going to explore the data distribution and image size, so that then decide appropriate pre-processing methods and evaluate my evaluation metrics.

The second step will be to build a human face detector and test its performance. For this detector, I will use the OpenCV's pre-trained human face detector downloaded by Udacity team, and then assemble it into a function, which takes the image directory as input and returns a True or False value, depending on whether or not a human face is detected in the image. Afterward, following the instructions, I will assess the accuracy of this detector function on 100 human face and 100 dog images. If the result is satisfactory, I will use this function as part of the final algorithm.

The third step is to build a dog detector from a Pytorch pre-trained model and test its performance, which is essentially the same as the procedure above.

However, one point worth noticing is that, because PyTorch and CV2 are different ML libraries, the data pre-process procedures will be very different. Here, I plan to resize, convert to numerical values, and normalize the input images, and then use the recommended pre-trained VGG16 model to make an inference. Likewise, I will then assemble the model into a function that returns Boolean values, and evaluate its performance.

The next step is to build a benchmark model or simple CNN model from scratch. First, I will pre-process the image data when building the data loaders. In particular, besides the resize and crop already mentioned before, I am going to augment the images in the training set to reduce the risk of over-fitting. For the net architecture, I plan to start with sample architecture provided by Udacity team, which contains 3 convolution layer and a linear layer, and then modify the architecture, hyper-parameters and/or do regularization, dependent on the model's validation and testing performance. Predictably, its performance will not be very impressive because of the simple architecture, and limited training examples, but I would still argue that it plays a decent role as a plain vanilla benchmark.

Once the benchmark is done, I will start to build the solution model via transfer learning. For the data loaders, I am going to use the same processed data loaders before so that the performances of the benchmark and the solution model are comparable. As for the crucial model architecture, I am going to try and compare several pre-trained PyTorch models by adjusting the output layer size to our 133 classes, and then use the best-performer of the test accuracy. Hopefully, since lots of low-level features are already extracted in a pre-trained model, and so we can use all our training example to fit the parameters in the last layer, the model's test performance will be substantially improved.

The final two steps are to build a dog breed detector from the transferred model obtained above and assemble all the human face detector, dog detector, and dog breed detector into our final algorithm. Afterward, I will test my final algorithm with images provided by Udacity and taken by myself.

References

[1] Hijazi, Kumar, and Rowen. 2015. “Using Convolutional Neural Networks for Image Recognition.” IP Group, Cadence.
https://ip.cadence.com/uploads/901/TIP_WP_cnn_FINAL-pdf

[2] “CrossEntropyLoss”. torch.nn. Docs. Pytorch.
https://pytorch.org/docs/stable/nn.html#crossentropyloss