Julien Weiss

July 14th, 2020

Machine Learning Engineer Nanodegree

Capstone Proposal – Dog Breed Classifier

Domain Background

Classifying Dog Breeds in pictures is a typical problem that involves computer vision based on Convolutional Neural Networks (CNN). CNNs are a particular class of Deep Neural Networks that have been introduced by Yann LeCun in 1988. They have been increasingly deployed for Computer Vision problems in recent years. They are typically used to classify data that come as 1D arrays (for example, language), 2D arrays (for example pictures) or 3D (for example videos).

There is a parallel between the way CNNs work and the human eye and brain. As explained by Yann Lecun in *Deep Learning Review*, CNNs combine low level features like forms and edges and motifs in order to make up larger features. Ksenia Sorokina in *image-classification-with-convolutional-neural-networks*, explains that CNNs rely on a series of convolutional, non-linear, pooling and fully-connected layers*. ImageNet Classification with Deep Convolutional Neural Networks* details the results made possible by CNNs in image classification task, achieving error rates below 20%. Because computer vision and the implementation and deployment of CNNs are popular in autonomous systems, I find the problem of Dog Breed Classification a very motivating topic for my capstone project.

Problem Statement

Our goal is to build a Dog Breed Classifier that, provided an image, will respectively predict the breed of the dog, if a dog is detected in the image, or the most resembling dog breed if a human face is detected in the image, or output an error message if neither a dog nor a human face are detected in the image.
This multi-class classification problem will involve detecting dogs and human faces in images, training and deploying a CNN model with PyTorch to predict the breed of a dog detected in a picture and building a small app which allows a user to input an image and get a prediction of the dog breed our model thinks it sees. Our problem will be evaluated against its accuracy for an approximate benchmark of our CNN model and then we will use a metric that better suits our dataset and problem.

Datasets and Inputs

As input data for our project we will require pictures, as we aim to classify images. We will rely on the dog pictures and human pictures datasets provided by Udacity: “dog dataset” and “human dataset”.

The dog dataset has a size of 8351 pictures in total which are split into purpose directories for training (80%), validation (10%) and testing (10%).
Within each purpose directory, each dog breed for our multi-class classification problem has its directory of pictures. There are 133 possible dog breeds and thus directories, each containing different number of pictures of different sizes.

The human dataset has a size of 13233 pictures organized in 5750 subdirectories corresponding to a person’s name, containing different number of images. All images are of the same size (250x250). This dataset will only be used to test a pre-trained human face detector.

Solution Statement

Our approach will rely on implementing and using a Convolutional Neural Network (CNN) and will precisely involve the following steps: First, we will detect human faces by using a pre-trained human face detectors. For this, OpenCV provides a Haar feature-based cascade classifier. Second, we will detect dogs on images by using a pre-trained (on ImageNet) VGG-16 model. Third, after having identified if the image represents a dog or a human, we will implement a CNN from scratch using PyTorch and then using transfer learning to classify the dog into one of the possible 133 dog breeds. Last, we will create an app that allows any kind of image to be used as an input and predicts the dog breed if a dog is detected, the most resembling dog breed if a human face is detected and throws an error in any other case.

Benchmark Model

Our created CNN model will be compared against two indicators. The CNN model made from scratch will be compared to a random model: A random guess model would have 1 in 133 chance to predict the correct dog breed (less than 1% accuracy), provided a random dog image. Thus aiming for 10% accuracy for our CNN model from scratch is a good indicator that our model is going in the right direction.
For the CNN model that uses transfer learning, we will aim for at least 60% accuracy.

Evaluation Metrics

We have a classification problem with multiple classes and an unbalanced training set. According to the PyTorch documentation, CrossEntropyLoss is a useful metric for our problem and will be more adapted than accuracy. Thus we are going to train and test our CNN model against CrossEntropyLoss.

Project Design

For our approach, we will follow a 6 steps workflow.

- Step 1: Import datasets.
- Step 2: Detect humans using a pre-trained model.
- Step 3: Detect dogs using a pre-trained VGG-16 model.
- Step 4: Create a CNN from scratch to classify dog breeds.
- Step 5: Create a CNN using Transfer Learning to classify dog breeds.
- Step 6: Write and test an algorithm / app to classify dog images into the detected dog breed or human faces to the most resembling dog breed The app should handle images in which neither a dog nor a human are detected and return an error message to the user.

References

- *Udacity Project GitHub:* *https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification*
- *Image classification with CNNs:* *https://medium.com/@ksusorokina/image-classification-with-convolutional-neural-networks-496815db12a8*
- *ImageNet Classification with CNNs:* *https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf*
- *PyTorch documentation:* *https://pytorch.org/docs/stable/nn.html#loss-functions*
- *Deep Learning Review:* *https://s3.us-east-2.amazonaws.com/hkg-website-assets/static/pages/files/DeepLearning.pdf*