Abbas Shakiba

Machine Learning Engineer Nanodegree

Dog Breed Classifier

Capstone Project Report

1. Overview

Image classification is one of the most critical research areas in machine learning, and it has numerous applications in different fields. E.g., Face recognition, Health care, weather, education, industry, etc.

In this project, I will build a dog breed classifier to determine a dog's breed from an image. It is a challenging problem since there are hundreds of different breeds in different sizes, colors. Additionally, the pictures' size, location of the dog in the photos, and the dog's angle add more complexity. Thus, I will use a deep neural network model, known as Convolutional Neural Network, or CNN.

This project aims to classify pictures of dogs based on their breeds if the picture is a dog and flag the picture as human if the image is human. To add more fun to this project, if images of humans have been detected, the algorithm assigns a breed to that picture. Thus, the two main tasks are the human face detector and the dog breed detector.

2. Analysis

The input for this problem is color pictures. There are two datasets of humans and dogs. The human dataset has 13233 pictures, and the dog image dataset has 8351 pictures. There are 133 different classes of breeds among dogs pictures. The dog data is split into three folders: Train (6680 pictures), Vali (835 pictures), and test (836) pictures. I resized and normalized all the images before using them in the model. Additionally, to reduce the chance of overfitting and more information to the training dataset, I applied random horizontal flip and random rotation.

3. Methodology

Multi-class image classification is one of the most common problems in computer vision and has several applications. As mentioned previously, CNN is being used in this project. CNN used convolutional layers to create filters and, by adding depths to the

input, extracts additional information, which eventually makes the model predict the class accurately.

The first model has five convolutional layers, each doubling the following layer's depth by adding more filters. Each of the layers has a kernel with the size of (3,3) and having one stride and padding. A 2D max-pooling is applied to reduce the size of each convolutional layer. Eventually, the network will be flattened and followed by a four-layer fully connected network with a drop-out in between each layer. The last layer of the network has 133 nodes which represents the number of classes.

The second model for taking advantage of well-known CNN models, Transfer Learning, is applied to transfer weight and filters from the VGG16 model. Additionally, for Human Detection Haar Cascades classifier is used, one of the standard OpenCV algorithms for humans' face detection.

For the model evaluation, the Cross-Entropy loss function is applied, which is the most common loss function for the multi-class classification. On average, there are 50 images for each breed, yet, some breeds have more pictures, and the data is only slightly imbalanced. Thus, to evaluate the model's performance on the test data set, I just cared about the model's accuracy for all the classes: the ratio of correctly predicted images over all the images.

4. Results

The benchmark model trained on 100 epochs optimized by SGD (alpha=0.05) and Cross-Entropy as the loss function is helpful for multi-class classification problems. The test accuracy after 100 epochs was 31% (267/836), and the loss value was 2.739. Based on each epoch's loss values during the training part, it seems the model could perform better with the higher number of epochs.

The second model, the one with Transfer Learning, has ten epochs only, optimized by SGD (alpha=0.01), and uses the Cross-Entropy loss function, which has an accuracy of 81% (685/836).

Both models work well enough -- nor really!. Yet, there is always room for improvement:

- Using a more sophisticated pre-trained algorithm, preferably on dog images. e.g., ResNet50
- Hypertuning; find the values for the parameters to get the highest accuracy
- Tuning the number of epochs, adding more data augmentation, etc.

5. Reference

https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html
https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d
https://stackoverflow.com/questions/64629702/pytorch-transform-totensor-changes-image
https://pytorch.org/vision/stable/transforms.html
https://pytorch.org/vision/stable/models.html
https://arxiv.org/abs/1409.1556
https://deepai.org/machine-learning-glossary-and-terms/accuracy-error-rate#:~:text=Accuracy%20in%20Machine%20Learning,of%20all%20the%20data%20points.