Abbas Shakiba

Machine Learning Engineer Nanodegree

Dog Breed Classifier

Capstone Project Proposal

Domain Background

The dog breed classifier is one of the classic supervised multi-class classification problems in modern machine learning. The classifier has to identify the breed of the bogs from their pictures. In this project, the Convolutional Neural Network (CNN) algorithm, one of the most advanced machine learning algorithms, is being used.

Problem Statement

The goal of this project is to classify pictures of dogs based on their breeds if the picture is a dog and flag the picture as human if the picture is human. To add more fun to this project, if pictures 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.

Datasets and Inputs

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.

Solution Statement

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.

For the dog breed classification, two models are being used: One model from the scratch inspired by famous CNN architecture. The second model for taking advantage of well-known CNN models, Transfer Learning is applied to transfer weight and filters from the VGG16 model.

For Human Detection Haar Cascades classifier is used, which is one of the common OpenCV algorithms for humans' face detection.

Benchmark Model

The first model has 5 convolutional layers, each doubling the depth of the next layer by adding more filters. Each of the layers has a kernel with the size of (3,3) and having 1 stride and padding. To reduce the width and height of each layer a 2D max-pooling is being applied. Eventually, the network will be flattened and followed by a 4 layer fully connected network with drop-out in between each layer. The last layer of the network has 133 nodes which represents the number of classes.

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

Evaluation Metrics

For the model evaluation Cross-Entropy loss function is being applied which is most common loss function for the multi-class classification.
Eventually, to evaluate the performance of the model on test data set we just care about accuracy of the model for all the classes: the ratio of correctly predicted images over all the images. Since the dataset is not balanced within the classes it may not perform at the same level for all breeds.

Machine Learning Engineer Nanodegree
Capstone Project Proposal, March 2021
Abbas Shakiba

Project Design

Step 1: Import datasets and data exploration
Step 2: Detect humans' faces by Haar Cascade classifier, OpenCV
Step 3: Detect dogs pictures by pre-trained ImageNet VGG16 model
Step 4: Create a CNN from scratch to classify dog breeds. Creating image loaders, Net class, loss function, optimizer, train, and test.
Step 5: Create a CNN using Transfer Learning (Pre-trained VGG16). Creating image loaders, Net class, loss function, optimizer, train, and test.
Step 6: Implement the main function to detect the picture as a human or a dog and return the breed of the dog if the picture is a dog or resembling the dog breed if the picture is human.
Step 7: Test the model and the main with random pictures

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.