Machine Learning Engineer Nanodegree Project Report

DOG BREED CLASSSIFIER

Oluwatola Oluwafemi Tofade

1. Definition

1.1 Project Overview

Dog breed classification is a popular and challenging problem in the Machine Learning Space. Dogs generally look similar but there exist certain features that make it a member of a particular dog breed type. So also, dogs possess varieties of colors, sometimes a particular breed type is identified by its colors. This becomes a complex computer vision task for humans as one would need to remember all the features of a particular breed and also must be able to distinguish these features from other breeds. One solution to this complexity would be to combine various models where each model would be specific such as identifying humans, identifying dog breeds and identifying humans faces that looks like a dog breed. This is a mutli-classification problem where we can use supervised machine learning algorithms.

1.2 Problem Statement

The goal of this project is to build a pipeline to process real-world user-supplied images. The algorithm predicts the breed of a dog given an image and if a human image is passed to it, it predicts the closest matching dog breed. If Neither, it should return that it can't detect if the image is human or a dog. Evaluate a machine learning model to detect dog breeds, humans and subsequently classify dog breeds that resembles humans.

The following are the problems:

- Identification of a dog or human in a provided image
- Identification of closest matching dog breed when given an image of a dog or a human
- Application of transfer learning
- Achieving an accuracy of >= 60% with test data

1.3 Metrics

The data was split into train, test and valid dataset. Model was trained using only the train dataset, while the testing data was used to predict the performance of the model. Accuracy will be used as one of the metrics to evaluate the model.

Accuracy = Number of items correctly classified/All classified items

Also, the Multiclassification Log loss which is calculated by the comparison of the test data prediction with the validation data. The Log loss takes into the account of uncertainty of prediction based on how much it varies from actual label, and this would also help in evaluation of the model.

2. Analysis

2.1 Data Exploration

The input format for this project must be images. The dataset had images of dogs and humans

- Dogs Dataset

  The dog image dataset has 8,351 images which were sorted into training (6,680 Images), testing (836 Images) and validating (835 Images) directories. Each of these directories had 133 folders corresponding to dog breeds. Each image was unique from each other and varied in size from each breed.

Sample Images from the dog dataset

- Humans Faces Dataset

  The humans image dataset has 13,233 images which were stored in 5,750 folders corresponding to people names. The images were all unique from each other.

Sample Images from the human dataset

Data augmentation (such as image flipping) was applied to the testing data so as to have an even number of images during training per dog breed.

2.2 Benchmark

- The CNN model created from scratch must have accuracy of at least 10% on test data, which would be better than a random guess which has an accuracy of 1/133% (less than 1 %). This is the baseline accuracy.
- The CNN model created using transfer learning must have an accuracy of at least 60% on the test data.

3. Methodology

3.1 Data Processing

I implemented torch vision transforms module to resize the images into 256x256 images and then further cropped to 224x224 for the training, validating and testing datasets. I did all these because of the ResNet50 requirement for an input image is 224x224 pixels. These images were then normalized with a standard mean and standard deviation.

3.2 Implementation

Human Detection:

OpenCV implementation of Haar feature-based was used to detect human faces in this project. The 'haarcascade_frontalface_alt.xml" cascade classifier was used and it was able to identify 98% of the human faces from 100 human images and 17% human faces from 100 dog images.

Dogs Detection:

VGG16 model using PyTorch was used to detect dogs in this project. The VGG-16 model has been previously trained on ImageNet, a popular dataset used for computer vision tasks. The images were transformed to

224x224 size, converted to tensors and then normalized using the standard mean and standard deviation before feeding to the VGG model. It then returns an output with index numbers between 151 and 268 (these are the dog categories on ImageNet)

CNN Model:

I created a CNN model using PyTorch. I then trained, validated and tested this model using the respective datasets. The CNN architecture had convolution layers, pooling layers, fully connected layers and drop out layers, all these makes up the standard CNN architecture.

My model had the architecture as shown below:

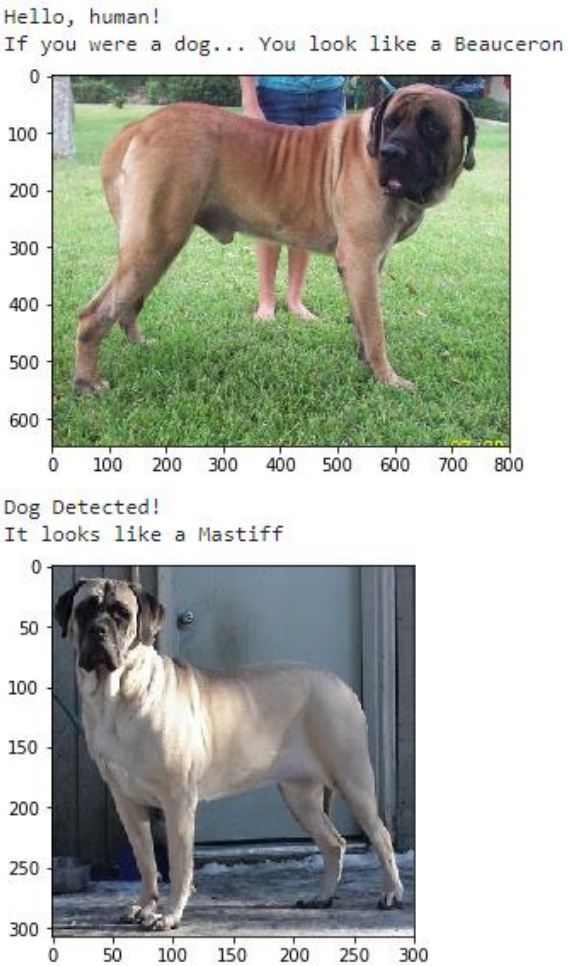
- First Convolutional Layer: 3 inputs, 32 outputs, 3x3 kernel
    - 224 x 224 x 3: 224x224 for image size, 3 for RGB
- Relu Activation function: for non-linearity and better performance over tanh or sigmoid
- Pooling layer: 2x2 kernel: reduces the dimensionality of each map and retaining important information
- Second Convolutional Layer: 32 inputs, 64 outputs,3x3 kernel
- Relu Activation function
- Pooling layer: 2x2 kernel
- Third Convolutional Layer: 64 inputs, 128 outputs, 3x3 kernel
- Relu Activation function
- Pooling layer: 2x2 kernel
- Flatten layer: 6272 length single vector
    - fully connected layer expects vector inputs
- Dropout layer: 30% probability: prevent overfitting
- First Fully connected Linear Layer: 6272 inputs, 500 outputs
- Relu Activation function
- Dropout layer: 30% probability
- Second Fully connected Linear Layer: 500 inputs and 133 outputs (number of dog breeds)

CNN Model using Transfer Learning:

Transfer learning (TL) is a research problem in machine learning (ML) that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. ResNet50 was chosen to be used here cause of it's low Top1 ImageNet error rate. I froze all the parameters from the pre-trained model and then added a fully connected layer which gives an output of 133 classes to correctly classify the images into one of the 133 classes.

3.3 Refinement

The CNN model that was created from scratch met the expected accuracy of >=10% , I had an accuracy of 12% with 30 epochs. I selected ResNet50 for CNN model using transfer learning and with 10 epochs, I was able to achieve 75% accuracy which has met the >=60% accuracy expectation on test dataset.



Sample Results

4. Results

This section describes the evaluation and validation results for the models trained in this project:

4.1 Model Evaluation and Validation

CNN Model:

I trained the CNN model I created for 30 epochs had an accuracy of 12% on the test dataset with a loss of 3.770694. The benchmark here was 10%, it is safe to say that my model passed

CNN Model using transfer learning:

ResNet50 CNN model was trained using transfer learning for 10 epochs, it had an accuracy of 75% with a loss of 1.137982. The benchmark here was 60%, it is safe to say the model passed the benchmark.

4.2 Justification

Both models passed the benchmark set for this project.

4.3 Improvement:

The model can be improved by the inclusion of more training and testing data. Also, more image augmentation can be done here to prevent overfitting and improve the accuracy

5. References

- CNN Model: https://en.wikipedia.org/wiki/Convolutional_neural_network
- Udacity Dogs Data set: Dogs Dataset
- Udacity Human Data set: Humans Dataset
- PyTorch Models - https://pytorch.org/docs/stable/torchvision/models.html
- https://github.com/KwokHing/udacity-project-dog-classification
- https://github.com/Jayshree-S/dog-breed-classifier-CNN