Capstone Project

Machine Learning Engineer Nanodegree

Yisakor Weldegebriel
October 3rd, 2020

Definition

Overview

Extracting meaning from images and classifying them has been an area of interest for decades. The first major breakthrough came about with the use of Convolutional Neural Networks (CNNs) in the early 1990s[1].
A famous benchmark is the MNIST dataset used to classify handwritten digits.

In this project I worked on the classification of dog breeds using a CNN.

There are two main classification models in this project:

- The first one is a CNN architecture I built from scratch
- The second is built off of the VGG model using transfer learning

The program will take an image or an array of images as input and display:

- The breed in the case of a dog input
- The closest dog breed in the case of a human input
- An "I don't recognize this object." message if the input is neither

[1] Most of the information on the history of CNNs were taken from the following website: https://hackernoon.com/a-brief-history-of-computer-vision-and-convolutional-neural-networks-8fe8aacc79f3

Problem Statement

The objective is to create a program that predicts dog breeds. The steps taken to do so were:

1. Download and preprocess data
2. Build human face and dog detectors based on pretrained models
3. Build a dog breed classifying model from scratch
4. Build a dog breed classifying model with Transfer Learning
5. Code a final function that outputs predictions

Metrics

Accuracy is the metric used for this project and it's calculated as follows:

$$accuracy = \frac{true\ predictions}{total\ predictions}$$

Analysis

Data Exploration

The dataset used for this project was provided by Udacity.
It contains training, validation and testing data for 133 dog breeds as well as a human face dataset.

Figure 1- Images from the dataset

Algorithms and Techniques

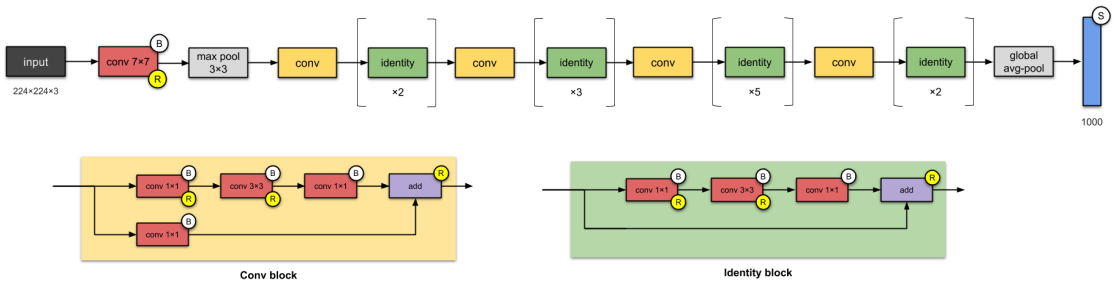
The main component to creating a classification model is a Convolutional Neural Network. We have an ample amount of data to train the CNN on, and its performance as seen in pretrained models is remarkable.
There are 133 possible predictions for each image, so the algorithm assigns a probability for each breed and outputs the breed with the maximum amount.

An accurate CNN model takes a lot of time and parameter tuning so in addition to creating a model from scratch, I used transfer learning to obtain a more accurate model on the back of an existing model.

A CNN will be composed of the following parameters that can be tuned for optimal accuracy:

- Layer types (convolutional, fully connected, maxpool)
- Number of Layers
- Batch size

- Epochs
- Learning rate
- Dropout rate



[2]

Figure 2 - Example of CNN architecture

Benchmark

The benchmark for the model built from scratch is an accuracy rate of 10%. For the Transfer Learning model, the bar is set at 60%.

Methodology

Data Preprocessing

For the human face detection method, the downloaded dataset is loaded into arrays. For the rest of the detection methods data is preprocessed through transforms and separated into training, validation and testing data before being fed to the CNN.

The transform consists of:

1. Resizing images into squares
2. Randomly rotating images 10°
3. Randomly flipping images horizontally
4. Turning them into Tensors and normalizing

[2] The example image of a CNN architecture was taken from: https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

Then the data is shuffled in the loaders.

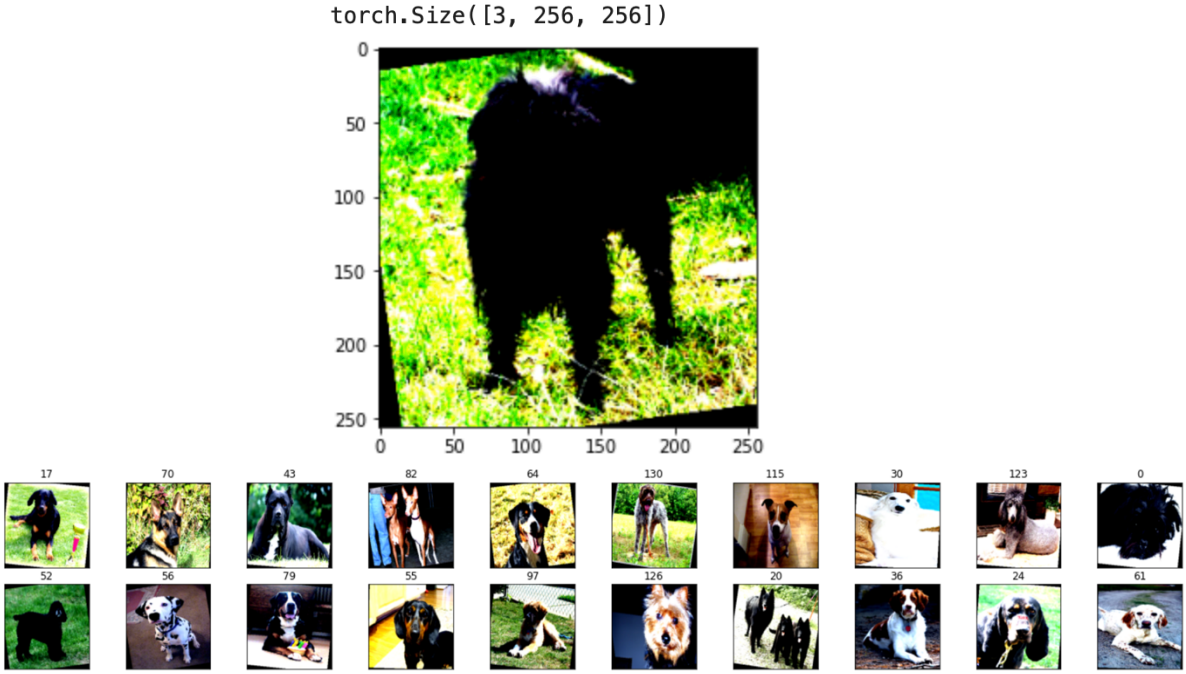


Figure 3 - Data after preprocessing

Implementation

The first model architecture I used contains 6 layers:

- 3 Convolutional layers
- 1 Max Pooling layer
- 2 Fully Connected layers
- Dropout rate of 0.25

```
(classifier): Sequential(
  (0): Linear(in_features=25088, out_features=4096, bias=True)
  (1): ReLU(inplace)
  (2): Dropout(p=0.5)
  (3): Linear(in_features=4096, out_features=4096, bias=True)
  (4): ReLU(inplace)
  (5): Dropout(p=0.5)
  (6): Linear(in_features=4096, out_features=133, bias=True)
)
```

Figure 4 - Recap of CNN architecture

After a bit of research on Pytorch, I opted for a CrossEntropyLoss function because it is best suited for multi-class classification.
I also picked an SGD optimizer with a learning rate of 0.01

For the Transfer Learning portion, the existing model I chose to adapt to my project is VGG16. This model is already optimized to classify 1000 different objects including different types of dogs so repurposing it as a dog breed classifier should work out well.

```
(classifier): Sequential(
  (0): Linear(in_features=25088, out_features=4096, bias=True)
  (1): ReLU(inplace)
  (2): Dropout(p=0.5)
  (3): Linear(in_features=4096, out_features=4096, bias=True)
  (4): ReLU(inplace)
  (5): Dropout(p=0.5)
  (6): Linear(in_features=4096, out_features=1000, bias=True)
)
```

Figure 5 - Recap Transfer Learning CNN

Improvement/Iterations

In order to achieve better results on the model built from scratch, I decided to experiment with the number of convolutional layers. I increased it from 3 to 6 and added one more fully connected layer. But it did not yield better results.

Results

Model Evaluation and validation

The final architecture of the CNN is the best performing out of the iterations I tried.

I defined training and testing methods and I was able to observe that for the model built from scratch, validation loss is at a minimum after about the 10th epoch, after which the model starts over-fitting.

Depending on the training cycle, accuracy varies from 10% to 11% on the test data.

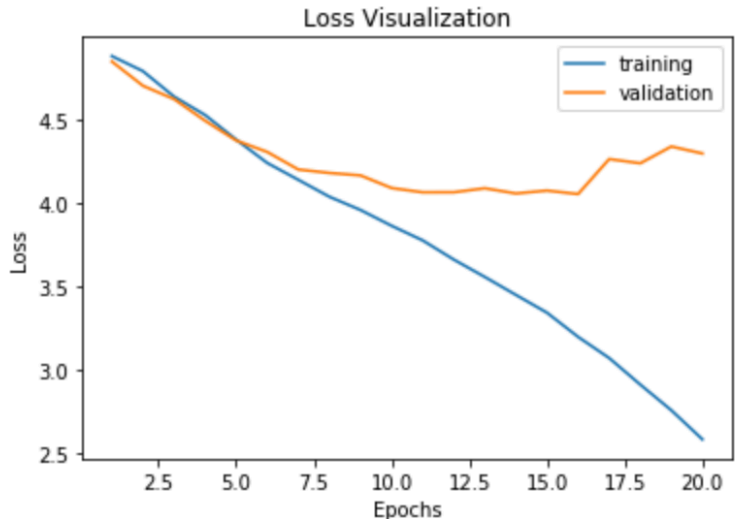


Figure 6 - Model_Scratch

The transfer Learning model outperforms the models built from scratch by a lot so that is what I used for the final prediction methods at the end of the notebook.

I was able to reach an accuracy rate of about 85% using transfer learning after about 15 epochs.

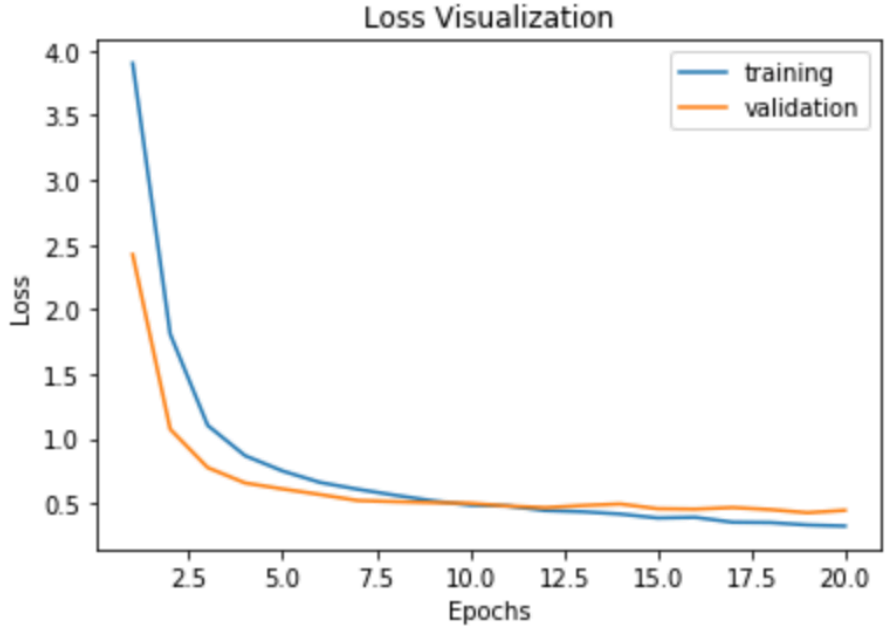


Figure 7- Transfer_Model

Conclusion

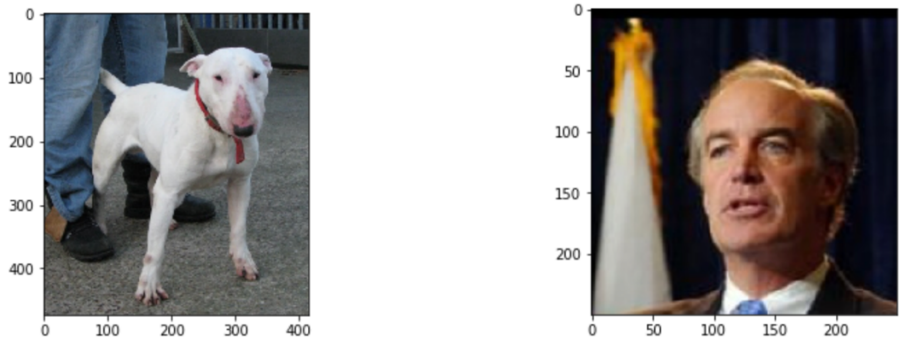


Figure 8 – Sample predictions made from the model

Summary

In summary I was able to implement code that for an input image handled 3 cases:

- If the input is a dog image, it will predict what the dog's breed is.
- If it's a human face, it will predict the dog breed closest to that face.
- If it's neither, it returns an error messaging stating that it does not recognize the object.