Udacity Machine Learning Engineer Nanodegree

Capstone Project Report

Dog Breed Classification with Convolutional Neural Networks

Stanislav Medvedev, March 5, 2021

Definition

Project overview

Computer vision is an active research area with applications in medicine, production quality control, and autonomous navigation. It enables automatic extraction and analysis of information from images that can be used to make a medical diagnosis, estimate the quality of a product, or direct a moving vehicle. Image classification through computer vision helps automate a process that can be costly, time-consuming, and could require specific domain knowledge by training a machine learning model, usually a convolutional neural network (CNN), that can label images with high accuracy. For this project, a CNN was trained to differentiate between dog breeds.

Problem Statement

According to a Reader's Digest article, there are at least 195 dog breeds (Gould, 2020). That can make labeling a dog with the correct breed a challenging task, which makes it a perfect candidate for a computer vision problem. The goal of this project was to develop an app that takes an image, and if a dog is present in it, a trained CNN model predicts the dog's breed. The app is also able to detect if an image of a human face was passed. In that case, it returns the dog breed with the most resemblance. If neither a human face nor a dog is detected in the image, the app displays an appropriate notification.

Metrics

The performances of both detectors and the breed classifier were measured using accuracy, which is the ratio of correct predictions over total predictions. More specifically, the dog detector needed to have high precision and high recall, as it was the first to scan the image. A high score in both of those metrics would ameliorate a higher false-positive rate in face detections that can occur with Haar feature-based cascade classifiers used in the following step. Precision and recall are given by TP/(TP+FP) and TP/(TP+FN), where TP = true positives, FP = false positives, and FN = false negatives.

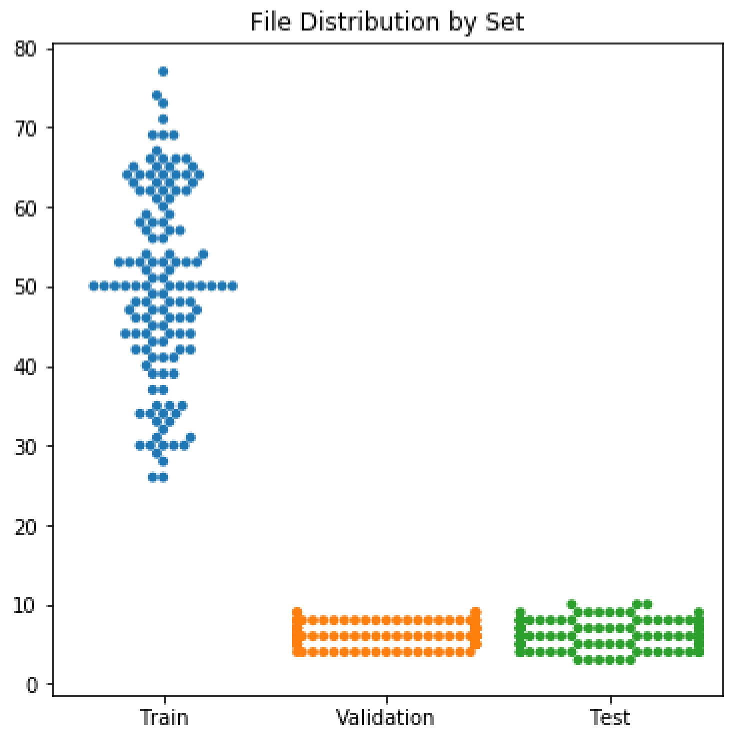
Analysis

Data exploration

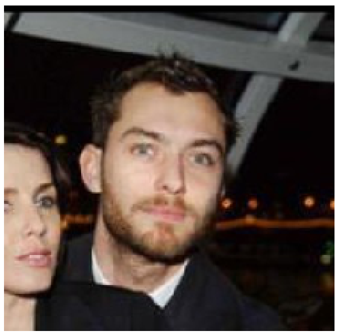
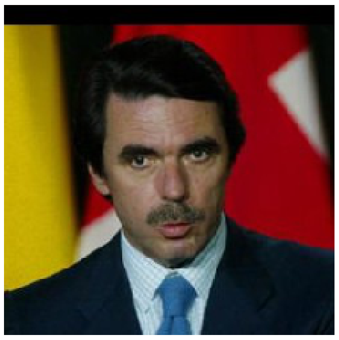
The dataset used in the project is provided by Udacity and consists of 13,233 human and 8,351 dog images. The dog images are split into 6,680 training, 835 validation, and 836 test files with a relatively balanced split among the 133 represented breeds. All images are in color, so they all have 3 channels in depth. The images are of different sizes, which needed to be taken into account while performing image transformations. The dog images were used for training, validating, and testing the CNNs used for classification. The human images were used to test the model's ability to detect a face and return the breed that it resembles the most.

Exploratory visualization

Here is a swarm plot of the file distribution by breed among sets. Each dot is a dog breed.



The following are some examples of images used in the project.

Algorithms and techniques

The app runs through the following steps. The image is loaded and checked for the presence of dogs by the dog detector. If a dog is detected, the image is loaded into the breed classification CNN which outputs the most likely breed. If there is no dog detected, the face detector checks for the presence of a human face, and if it finds one, the image is loaded into the CNN which outputs the breed with the most resemblance. If neither a dog nor a human are detected, the app displays a notification stating that. Dog and face detections are done using the VGG-16 model and `OpenCV`'s Haar feature-based cascade classifiers, respectively. Dog breed classification is done using a CNN built from scratch and with VGG-19 and ResNet-50 models fine-tuned for the task at hand.

Benchmarks

There were two approaches to constructing the CNN. The first approach was to build one from scratch, and the benchmark for it was at least 10% accuracy. Since there are 133 breeds present in the training and testing sets, that benchmark is much higher than 0.75% chance of getting the correct answer by a random guess. The second approach was to use transfer learning by modifying an exciting CNN to perform the task at hand. The aim for that approach was to achieve over 60% accuracy. Both benchmarks were set by Udacity.

Methodology

Data preprocessing

The data preprocessing step consisted of a series of transformations. Those included resizing and then cropping the image down to 224x224 pixel size and normalization for all three datasets. The training set images were also randomly flipped horizontally and rotated five degrees. The VGG-16 model used in dog detection and VGG-19 and ResNet-50 models used for transfer learning expect images 224x224 in size.

Implementation

The app detects dogs by predicting the contents of the image using the pre-trained VGG-16 model. If the predicted class is between 151 and 268, which are various dog breeds that VGG-16 can identify, the detector marks the image as containing a dog.

If no dogs are detected, the app checks the image for human faces using Haar feature-based cascade classifiers through `OpenCV`. If the classifier picks up at least one, the detector marks the image as containing a human face.

If either of those functions returns a positive result, the image is sent to the breed classification function which uses a CNN to analyze it and returns the breed with the most resemblance.

The first CNN that was tested was built from scratch with the following architecture and was trained for 50 epochs:

```
Input Tensor (224, 224, 3)
Conv2d(3, 16, 3, padding = 1)   - ReLU - MaxPool2d(2,2)
Conv2d(16, 32, 3, padding = 1)  - ReLU - MaxPool2d(2,2)
Conv2d(32, 64, 3, padding = 1)  - ReLU - MaxPool2d(2,2)
Conv2d(64, 128, 3, padding = 1) - ReLU - MaxPool2d(2,2)
Linear(14*14*128, 1024)
DropOut(0.3)
Linear(1024, 133)
```

I also tested fine-tuned versions of VGG-19 and ResNet-50 networks, both of which had the outputs of their last fully connected layer changed to 133 and all but that layer frozen before training.

Refinement

I tested several values for the depths of convolutional layers in the scratch CNN before settling on the final architecture. Once they were within a specific range, there were no significant differences in performance between them after ten epochs of training. I also compared Adam and SGD optimizers with different learning rates and chose SGD because it consistently showed slightly better results in all three CNNs. There is room for improvement in the scratch CNN, and choosing a different learning rate for the optimizer should produce better results.

The fine-tuned models were both first trained for 20 epochs. They were later reinitialized with the best-performing weights from the first training session and trained for another 20 epochs. Interestingly, while the validation loss for both of them decreased after the second round of training, their prediction accuracy did not improve.

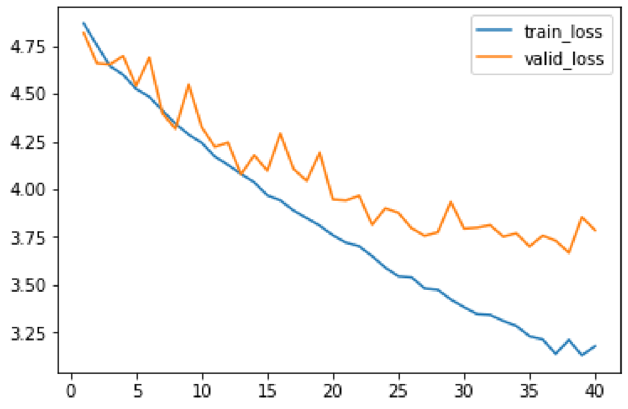
Results

Detectors

Two hundred images split evenly among dogs and humans were used to test the dog and face detectors. They both produced 100 out of 100 possible true positives. The dog detector also returned 0 false positives, while the face detector returned 21. Since the face detector checks the image after the dog detector, the high false-positive rate does not hamper dog breed classification.

Scratch CNN

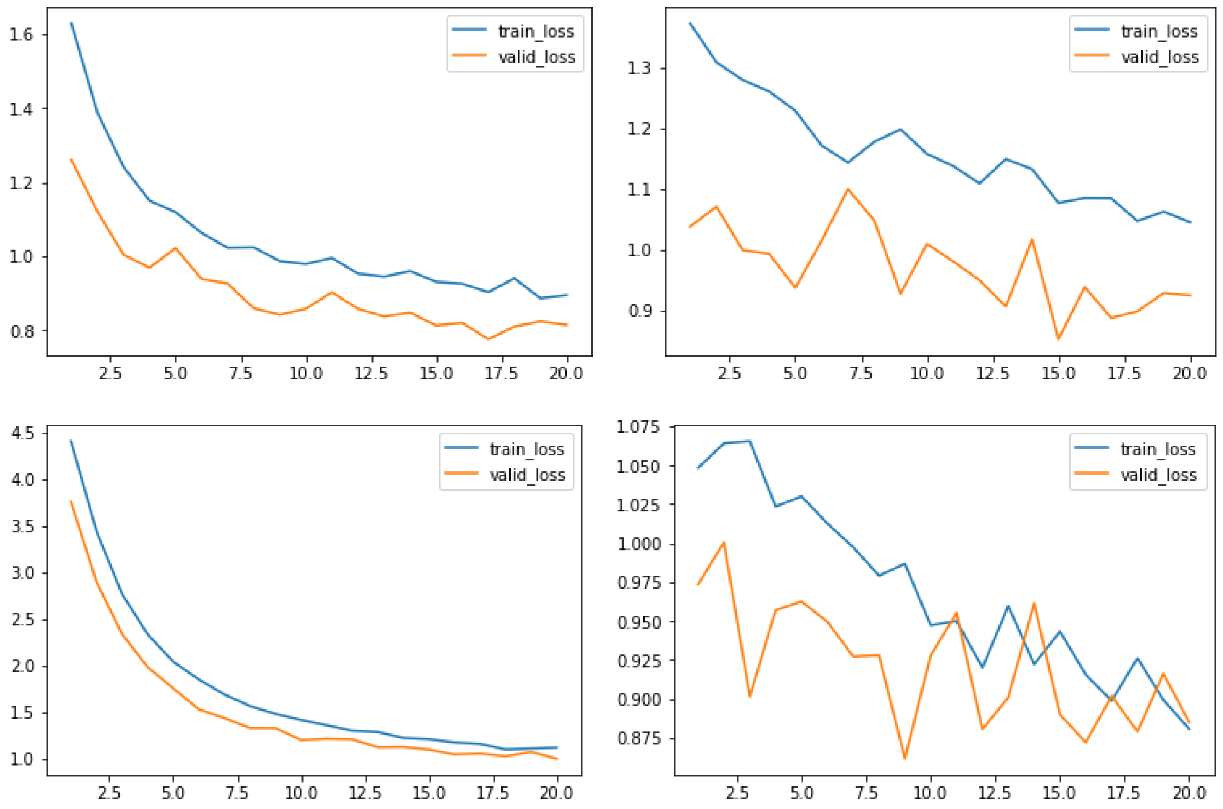
I trained the scratch CNN for 40 epochs. The training and validation losses started diverging around epoch 26, as can be seen on the graph. The model produced 13% accuracy, and picking a lower learning rate for the optimizer and then training the model for a longer duration can most likely improve it.



Scratch CNN training and validation loss graph

Transfer learning CNNs

I tried VGG-19 and ResNet-50 models for this approach. I trained both of them for 20 epochs and measured their performance, and they both produced about 76% accuracy. I later reinitialized them with the best weights from the previous training session and trained them for 20 more epochs. As the loss graphs show, the training and validation losses for both of them decreased; the prediction accuracy, however, remained the same, around 76%.



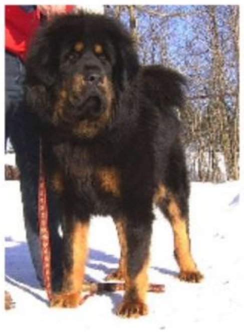
Loss graphs of the models based on VGG-19 (top row) and ResNet-50 (bottom row) architectures

Justification

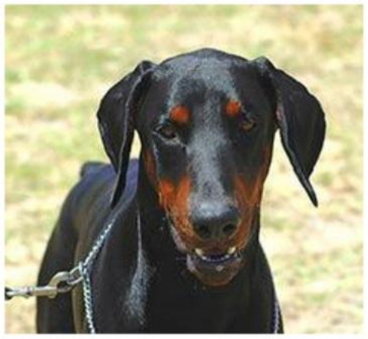
The benchmarks set for the models were 10% accuracy for the scratch CNN and 60% accuracy for the transfer learning approach, and the models surpassed them. The scratch CNN achieved 13%, and the models used in the transfer learning method achieved 76% accuracy, which shows that the chosen approaches worked. Furthermore, there is room for further improvement in all models with hyperparameter tuning and longer training duration. Trying a model with a more complex architecture, like ResNet-101, could also improve results.

Sample app outputs are below.

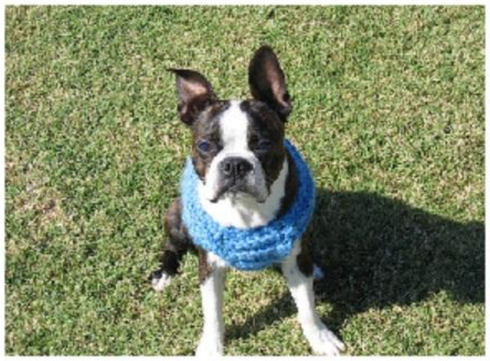
```
It's a dog! Looks like a Tibetan mastiff
```



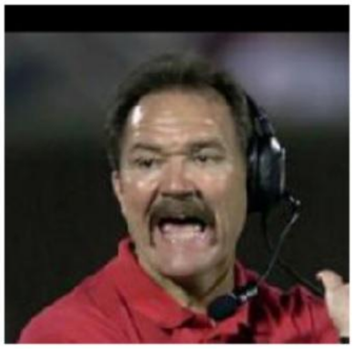
```
It's a dog! Looks like a Doberman pinscher
```



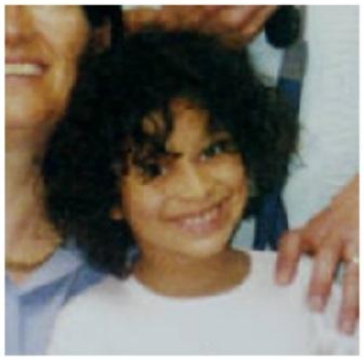
```
It's a dog! Looks like a Boston terrier
```



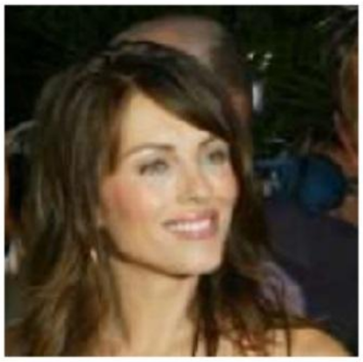
```
Hello, human! You look like a Pharaoh hound
```



```
Hello, human! You look like a Poodle
```

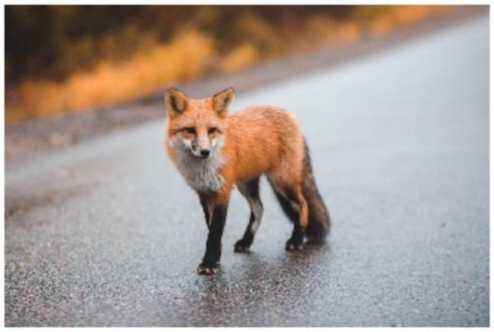


```
Hello, human! You look like a English toy spaniel
```

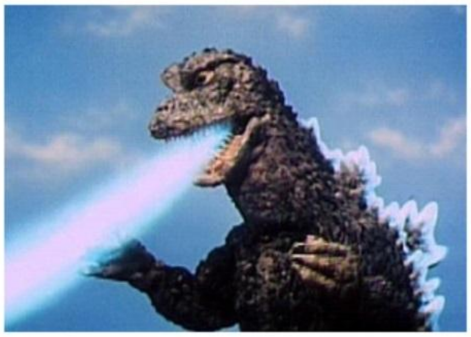


I also tried the app on a few images that didn't have humans or dogs in them with the following results:

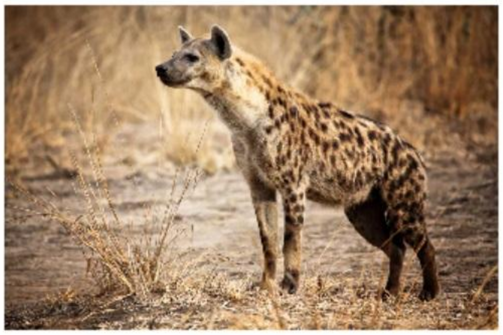
```
Hello, human! You look like a Icelandic sheepdog
```



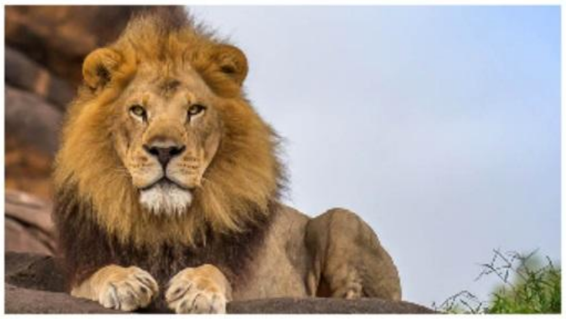
```
Looks like there are no dogs or humans here.
```



```
Looks like there are no dogs or humans here.
```



```
Looks like there are no dogs or humans here.
```



References

Gould, Wendy Rose. “How Many Breeds of Dogs Are There in the World?” *Reader's Digest*, Reader's Digest, 26 Feb. 2020, www.rd.com/article/how-many-dog-breeds-in-the-world/.