Machine Learning Engineer Nanodegree Capstone Project

Dog Breed Classifier

Si Yan

May 21, 2021

Definition

Project Overview

The main challenge of this project is to build an image classifier with sufficient accuracy. Dog breed classification is challenging for a few reasons:

1. Large number of classes – more than 300 breeds have been recognized so far according to the Fédération Cynologique Internationale (FCI).[1] Large size of training data is needed to cover all classes.
2. Low inter-class variation – dogs in different breeds may be very similar in appearance.
3. High intra-class variation – one dog breed may have different colors and sizes.

For such a challenging image classification problem, Convolutional Neural Networks (CNN) is the best choice. CNN has been widely applied in facial recognition, text classification, self-driving cars, medical image processing, and more.[2] Unlike traditional multilayer perceptron (MLP) network that is a fully connected network, CNN has multiple convolution and pooling layers before the last fully connected layer. This architecture takes advantage of the hierarchical pattern in data and assemble patterns using smaller and simpler patterns. Therefore, CNN has high prediction accuracy and computational efficiency on classification problems.

In this project, I will use CNN to build the dog breed classifier for its high accuracy.

Problem Statement

The goal of this project is to build an algorithm that accepts a user-supplied image path as input and makes classification on the image. The algorithm first determines the input image contains a dog, human, or neither. Then, it output predictions based on the detected type:

- If a dog is detected, identify an estimate of the canine's breed.
- If a human is detected, identify the resembling dog breed.
- If neither dog nor human is detected in the image, provide an output that indicates an error.

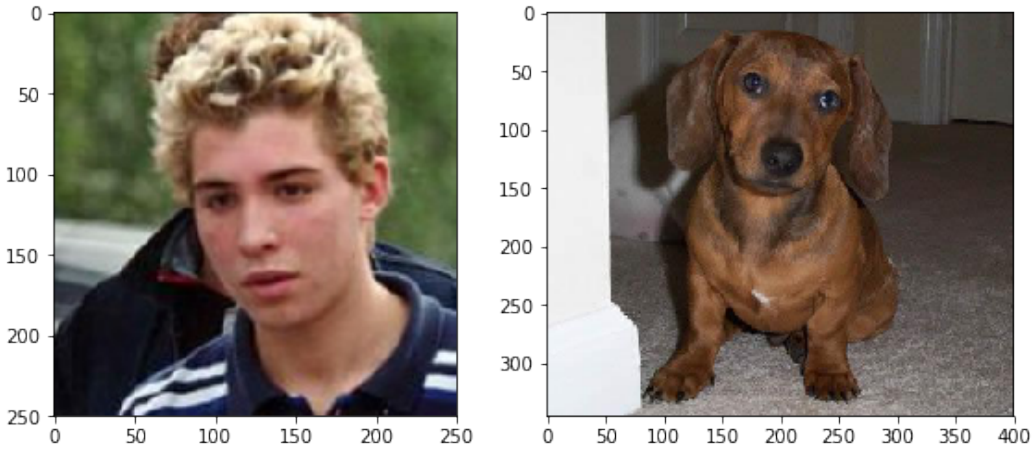
Metrics

The percentage accuracy is used to evaluate the model performance. Accuracy is one of the most widely used metrics for classifier evaluations. This metric is calculated using the number of correct predictions divided by total number of predictions.

Analysis

Data Exploration

All of the human and dog images used in this project are provided by Udacity. The dataset has 13233 human images and 8351 dog images. Some example images of human and dogs are shown in Fig 1. The provided dog image dataset has pre-defined training, validation and test sets of images. Out of the total 8351 dog images, 80% (6680 images) are in the training set, 10% (835 images) are in the validation set, and 10% (836 images) are in the test set. There are 133 classes of dog breed in the dog dataset.



Daniele_Bergamin_0001.jpg Dachshund_03960.jpg

Fig 1. Sample images of human and dog.

Exploratory Visualization

The counts of each dog breed have been plotted in ascending order, as shown in Fig 2. The breed has least number of images is Norwegian Buhund, which has 33 images. The Alaskan Malamute

has the most number of images, with 96 images. The 25 percentile has 53 images and the 75 percentile has 76 images.

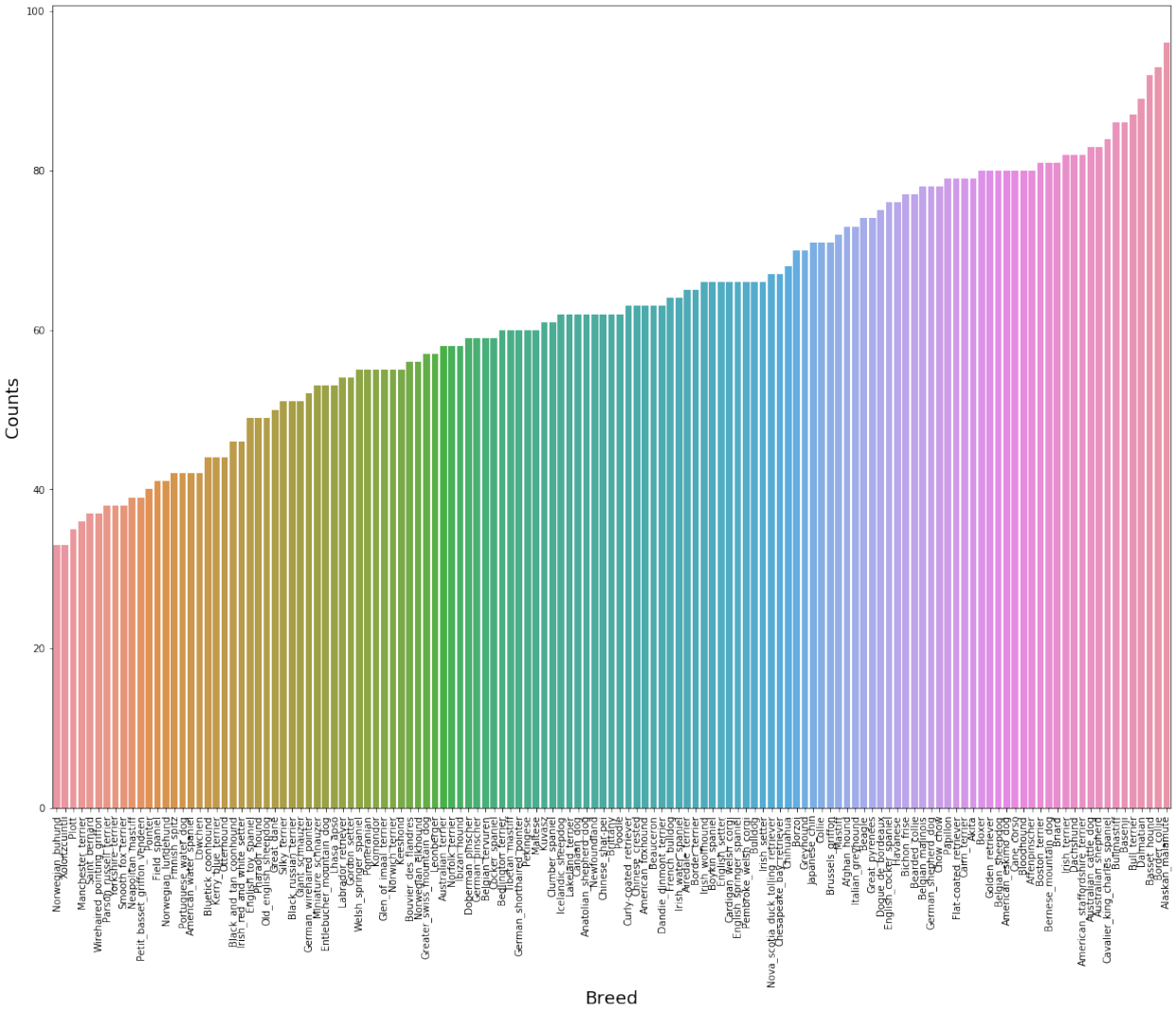


Fig 2. Number of images in each breed class.

One type of problem I observed during data exploration is some dog images contain human faces, which may mislead the first human / dog detection step.

Fig 3. Dog image with both dogs and human

Algorithms and Techniques

As discussed in the Project Overview, CNN is the most suitable technique for this multiclass classification problem on images.

For human detection, pre-trained Haar feature-based cascade classifiers from OpenCV is used. A pre-trained VGG-16 model is used to detect dog in images. To ensure a high classification accuracy with limited computation resource, transfer learning technique is used to build the classifier to identify dog breed from dog images and resembling dog breed from human images. The base classifier is the pre-trained ResNet-50 network, but the final fully-connected layer is replaced with a linear layer with 133 output classes.

Benchmark

The benchmark model is the CNN that I build from scratch, which has accuracy of 15% with the test images. This model has:

- 3 convolutional layers, each followed by a ReLU and a pooling layer.
- 2 dropout layers, with 25% drop-off rate
- 2 fully-connected layers

The details of the CNN is as below:

Net (

(conv1): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
(conv2): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
(conv3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
(pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
(fc1): Linear(in_features=6272, out_features=512, bias=True)
(fc2): Linear(in_features=512, out_features=133, bias=True)
(dropout): Dropout(p=0.25)
)

Methodology

Data Preprocessing

To prepare data for dog detector and breed classification, an image will be resized to 256 x 256 pixels, then crop to the 224 x 224 pixels in the center. After being transformed to Pytorch tensor, a normalization with mean=[0.485, 0.456, 0.406] and std=[0.229, 0.224, 0.225] is performed on the image.

Before training the benchmark model and the real classification model in the final product, augmentation was applied to the training data, including random horizontal flipping and random rotation at 10 degrees.

Implementation

The dog breed classification app was implemented in seven steps.

Step 1: Implement the human face detector. This detector takes an image path as input, and returns true if human face is detected in an image. A pre-trained Haar feature-based cascade classifiers from OpenCV is used to detect the human faces.

Step 2: Implement the dog detector. Similar as the human detector, this dog detector also takes an image path as input, and returns true if dog is detected in an image. The dog detector uses a pre-trained VGG-16 model. If the VGG-16 predictor output an index in the range [151, 268], the dog detector returns true.

Step 3: Create data loaders for the benchmark model training and the final model training.

Step 4: Create the benchmark model to classify dog breeds from scratch. See the details of this CNN in Benchmark Model section. To train the benchmark model, Stochastic Gradient Descent

(SGD) was used with learning rate at 0.05, and CrossEntropyLoss was used as the loss function. 20 epochs were used in training, the validation loss stopped decreasing after 12 epochs.

Step 5: Create a CNN to classify dog breeds using Transfer Learning.

The base classifier is the pre-trained ResNet-50 network[3]. The whole architecture of ResNet-50 were kept, but the final fully-connected layer is replaced with a linear layer with 133 output classes. In model training, Adam optimizer was used with learning rate at 0.001, and CrossEntropyLoss was used as the loss function. 15 epochs were used in training. The validation loss stopped decreasing after 6 epochs and this model was saved for testing. The prediction accuracy of this model on test set is 80%.

Step 6: Implement the algorithm that accepts a file path to an image and first determines whether the image contains a human, dog, or neither. Then,

if a dog is detected in the image by the dog detector, return the breed predicted by the refined model from Step 5.

if a human is detected in the image by the human detector, return the resembling dog breed predicted by the refined model from Step 5.

if neither is detected in the image, provide output that indicates an error.

Step 7: Test the algorithm implemented in Step 6 with some images.

Refinement

For the Benchmark model, I optimized the kernel size and number of output channels of Convolutional layers. For model training, the learning rate and number of epochs were optimized to find the best validation loss efficiently.

For the final model (ResNet-50), I optimized the learning rate and number of epochs were optimized to find the best validation loss efficiently.

Results

Model Evaluation and Validation

The human face detector and dog detector was tested by the first 100 provided human images. The human face detector correctly detected 98 % of the first 100 images in human files. 17 % of the first 100 images in dog files were detected with human face. The 17% human faces detected in

dog images are not because of wrong classification of the pre-trained Haar cascade classifiers, but because of some images contain both human and dogs, i.e. the image in Fig 3.
The dog detector output 0 % of the first 100 images in human_files with a detected dog and 100 % of the first 100 images in dog_files have a detected dog.
During model training, the loss value of validation set was used to compare performance and the model parameters from the training epoch with the lowest validation loss are saved.
In benchmark model training, the lowest validation loss (= 3.798190) is reached after 12 epochs (20 epochs in total). The saved benchmark model correctly classified 126 images out of 836 images in test set. The 15% test accuracy meets the project requirement on benchmark model (10% test accuracy). The test loss of this model is 3.789452, which is a little smaller than the validation loss.
The final CNN model used in the dog app was trained for 15 epochs. The lowest validation loss is 0.632364, which was reached after 6 epochs. The model has test accuracy at 80%, which meets the project requirement of 60% test accuracy. The test loss is 0.680174, comparable with the validation loss. The relatively high test accuracy, and comparable validation and test loss, indicates the robustness of this CNN model.

Justification

The final model performance is 80% accuracy on test dataset, which beats the benchmark model and satisfies the project requirement. For the complexity of dog image classification, this model performance is reasonably good and can be used in the dog breed classifier app.

Conclusion and Discussion

Performance of the Dog Identification App

The final dog identification app was tested with 5 human images and 20 dog images. Representative images and predictions are shown in the Fig 4a-f below.

```
Hello, Human!
You look like a ...
Dachshund.
```

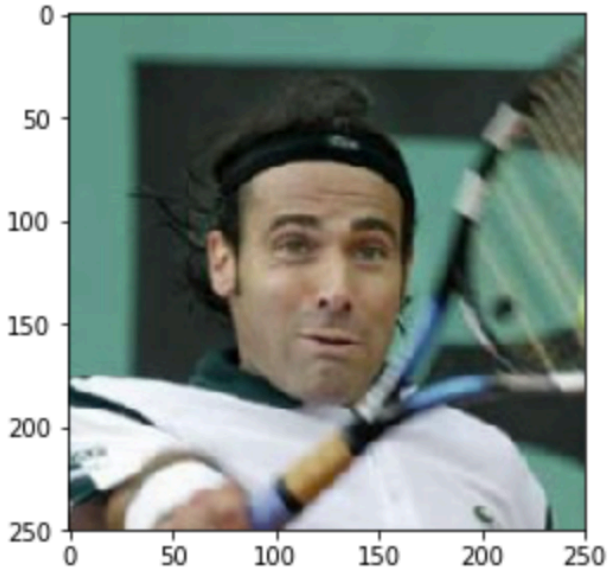


Fig 4a.

```
Hello, Human!
You look like a ...
Norwich terrier.
```

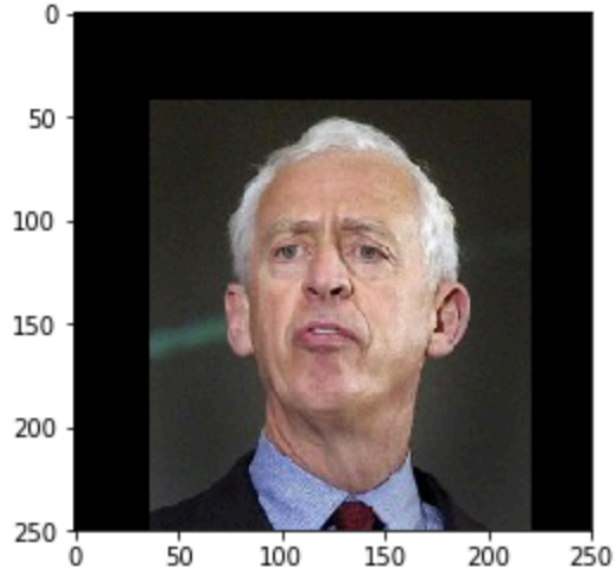


Fig 4b.

```
Hello, Human!
You look like a ...
Wirehaired pointing griffon.
```

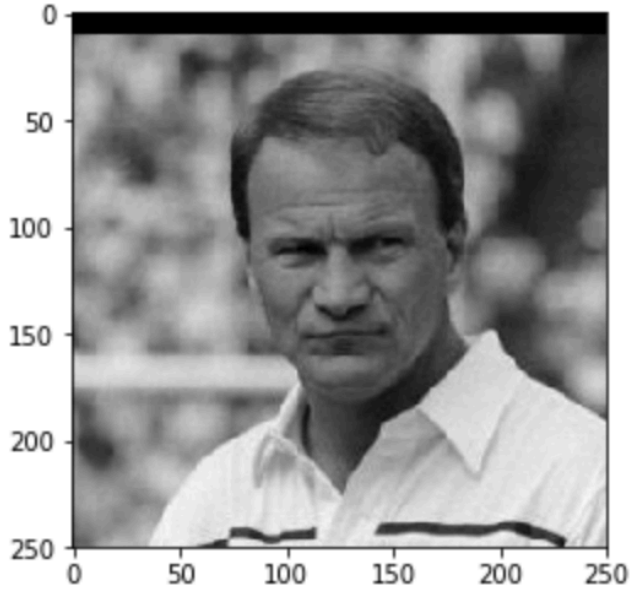


Fig 4c.

```
A Dog!
It looks like a ...
Cane corso.
```

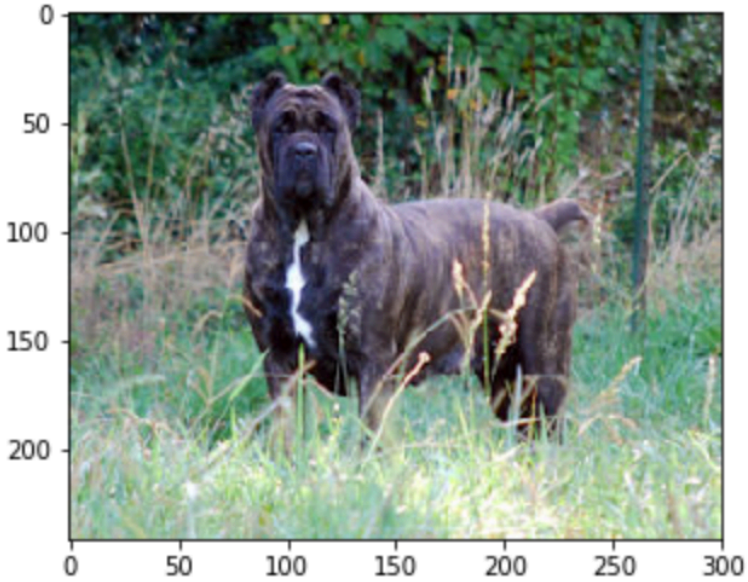


Fig 4d.

```
A Dog!
It looks like a ...
Dachshund.
```

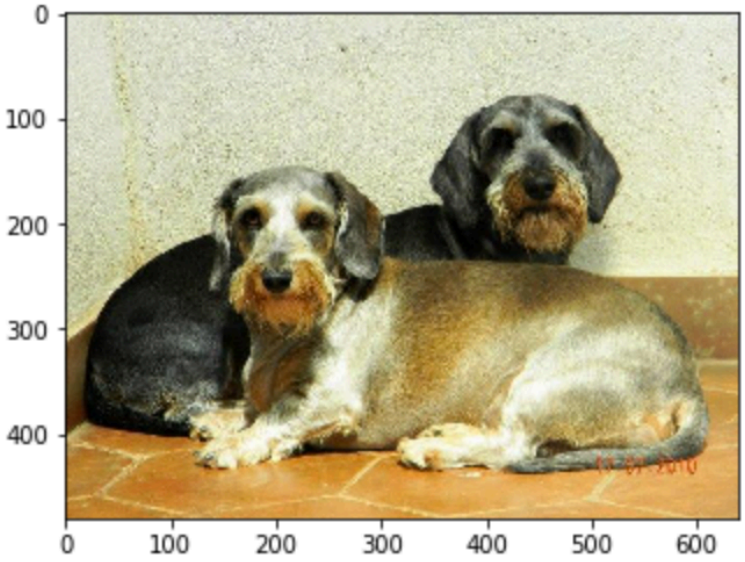


Fig 4e.

```
A Dog!
It looks like a ...
Black russian terrier.
```

Fig 4f.

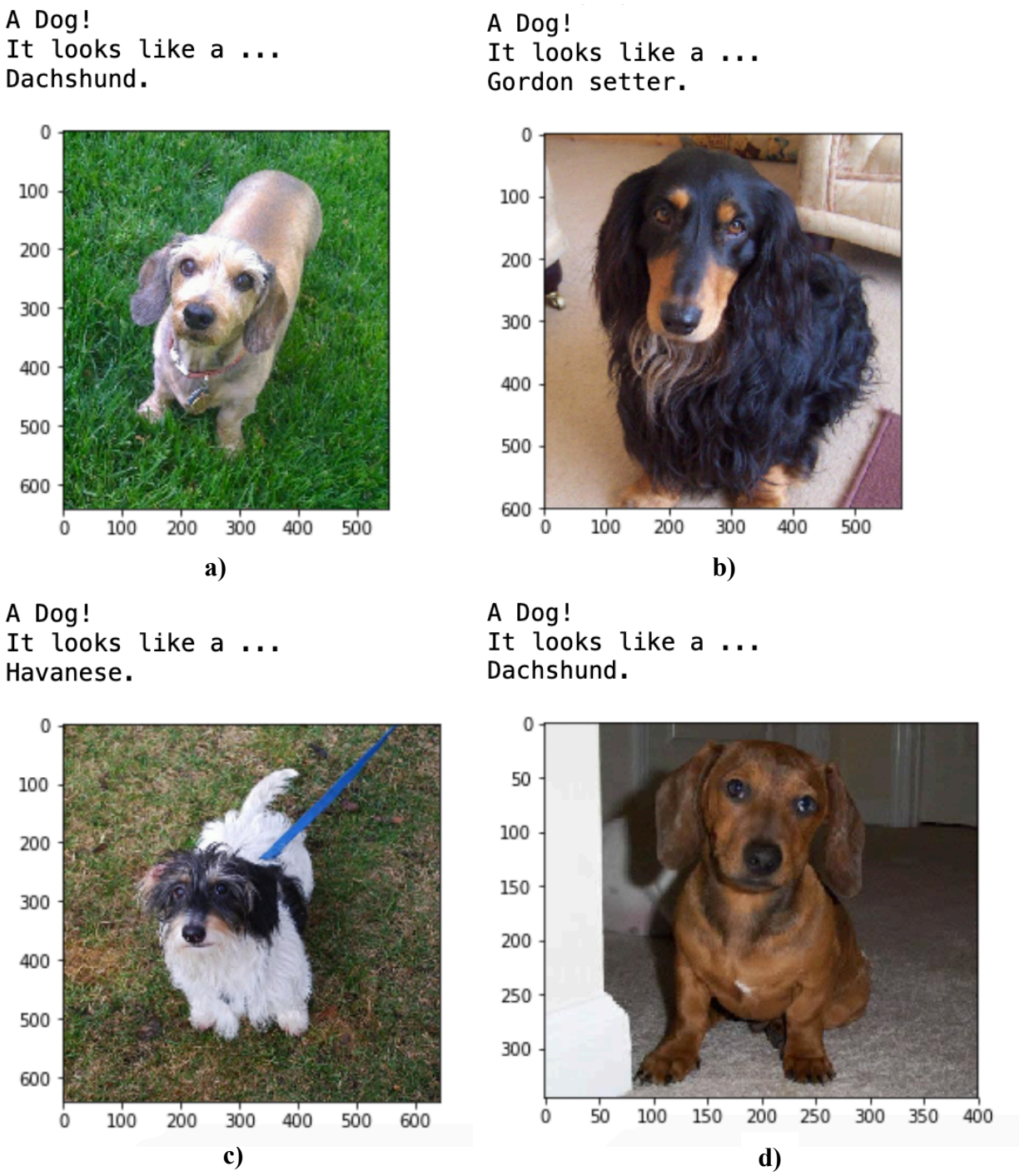


Fig 5. Dachshund in different appearance

The prediction outputs by this app are mostly expected. The app correctly detects all 5 human images and 19 out of 20 dog images. Only one dog image can't be detected by the app. That is an image with low resolution.

The algorithm is able to correctly predict dog class for all Cane Corso pictures and for all Black Russian Terrier pictures. But it makes wrong classifications on two of the Dachshund images. As shown in Fig 5, the four images are all Dachshund, but they all have different looks. b) and c) were classified as different breed. The high intra-class variation of Dachshund causes the miss classification.

With 20 input dog images, 19 of them are detected with dog correctly and 17 out of the 19 images are correctly classified. This result adequately meets the requirement of the project.

Improvements

We can still improve the Dog Classification App to achieve even better classification accuracy from three aspects:

1. Try different pretrained model for dog image detection to minimize the chances that dog can't be detected.
2. Try a CNN model with higher accuracy, and perform hyperparameter fine-tuning.
3. Use more images for training.

Reference

1. http://www.fci.be/en/Nomenclature/

2. Bhandare, A., Bhide, M., Gokhale P., Chandavarkar R., Applications of Convolutional Neural Networks, IJCSIT, Vol. 7 (5) , 2016, 2206-221.

3. https://arxiv.org/abs/1512.03385