Machine Learning Nanodegree Capstone

Report: Dog Breed Classifier

Mohaned Mohamed El-Hadidi

1. Definition

1.1 Project Overview

Images are now a crucial part of everyday life for most people. Every day, individuals take millions of images with their cellphones and digital cameras. The meaning of a picture, its content, and the information it conveys is simple for humans to decipher. Images, on the other hand, are nothing more than large matrices of pixels to a computer, which has no concept what they mean. It is critical for computers to be able to extract information from images to create artificially intelligent systems.

Dog owners' number in the millions all over the world. A dog-breed classifier can assist dog owners in identifying their dog's breed if they are unsure. This would be extremely beneficial to first-time dog owners.

We explain the use of Convolutional Neural Networks in this project by demonstrating an example picture classification problem with one of the ImageNet competition-winning deep learning models. We employ pre-learned weights from a ResNet-152 model that was trained on a massive dataset to tackle the problem of "classifying dog breed." Udacity has provided us with the data we'll need to train our model.

1.2 Problem Statement

The project's goal is to create a pipeline that can process real-world, user-supplied photos. Given a photograph, the algorithm will determine an estimate of the dog's breed. When the image is of a person, the algorithm will select a dog breed that closely matches the person. If neither a dog nor a human can be found, an error message is displayed. As a result, the models in place should be capable of recognizing a dog or human in an image, classifying the dog by breed, and classifying a dog breed that resembles the human.

1.3 Metrics

"Accuracy" is the metric we use to measure how effectively our algorithm performs in classifying dog breeds in photos. Our model's accuracy is the proportion of total items properly categorized. The accuracy would be a good statistic to use for judging our model because it gives us a good idea of how well it works and gives us a value between 0 and 100. The greater the accuracy, the better would be our model in classifying dog images.

The accuracy is defined as:

$$Accuracy\ \% = \frac{total\ number\ of\ correct\ predictions}{total\ number\ of\ images}$$

Also, accuracy won't perform well if the dataset classes are imbalanced.

But as of improving the accuracy can also be thought of as minimizing our loss metric, which is log loss. The log loss error considers how different the predicted probability which is 0 or 1. This is directly usable in our problem.

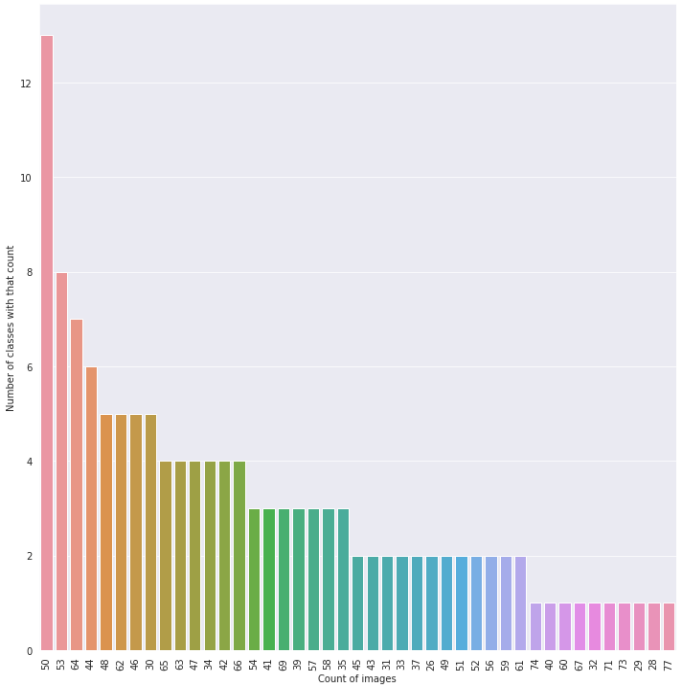
2. Analysis

2.1 Data Exploration

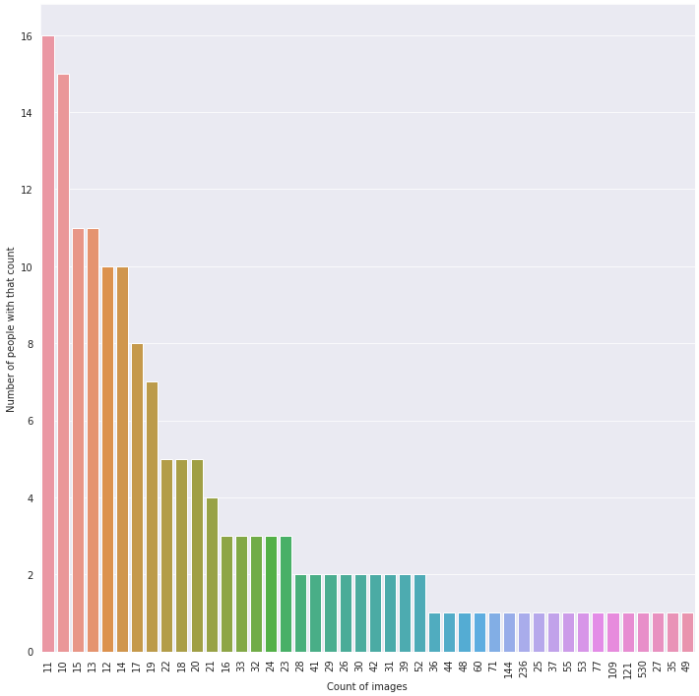
Udacity provided all the datasets used to train, test, and validate the CNN model. The dataset has pictures of dogs and humans.

Dog images dataset: The dog images dataset contains a total of 8351 images of dogs. These images have been split into training, testing, and validation images with a split of 80% training data, 10% testing data, and 10% validation data. Thus, there are 6680 training images, 836 testing images, and 835 validation images. Each of this directory (train, test, valid) have 133 folders corresponding to dog breeds. The images are of different sizes and different backgrounds, some images are not full-sized. Also, to be noted that after analysis of the data, it was found that the number of images in each folder is different, thus making the training set highly imbalanced.

Human images dataset: The human dataset contains 13233 total human images which are sorted by names of human (5750 folders). All images are of size 250x250. Images have different background and different angles. When looking at the amount of images in each folder, it becomes clear that the dataset is highly unbalanced, and the quantity of images varies greatly from person to person. This is highlt noticed in the below figure.



histogram of the counts of images in different classes in dog dataset



histogram of counts of images for different humans in the human dataset

2.2 Exploratory Visualization

The job of identifying a dog's breed from an image is described as "very difficult." Consider how difficult it would be for a human to tell the difference between a Brittany and a Welsh Springer Spaniel.

It is not difficult to find other dog breed pairs with minimal inter-class variation (for instance, Curly-Coated Retrievers and American Water Spaniels).

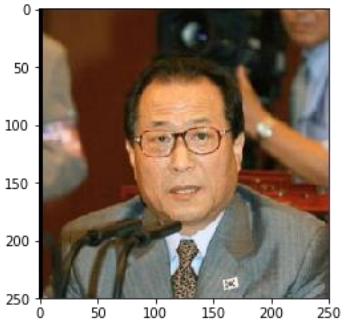
Likewise, recall that Labradors come in yellow, chocolate, and black. Your vision-based algorithm will have to conquer this high intra-class variation to determine how to classify all these different shades as the same breed.

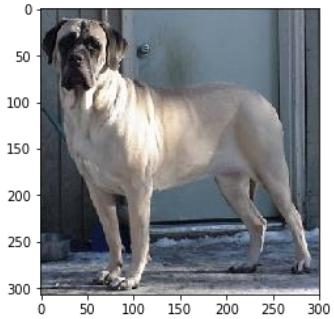
Yellow Labrador

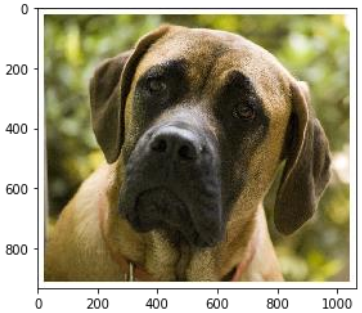
Chocolate Labrador

Black Labrador

Some sample images from the dataset:







2.3 Algorithms and Techniques

This section discusses the algorithms and techniques used to achieve the goals of our project.

- The solution of this project will include a trained detector using CNN to detect dog breeds based on the supplied picture. CNN stands for convolutional neural network, a deep learning framework which is widely used in image classifications. Convolutional layers are one of the basic components of a CNN model. CNN uses multiple layers of convolutional filters to extract features out of an image such as vertical/horizontal edges, groups, certain shapes, and colors, etc. A complete CNN model also features hyperparameters such as number of epochs, batch size, pooling, number of layers, weight, and dropouts. CNN works extremely well with image classification applications because it follows a hierarchical model which resembles human brain. This makes CNN super effective in image classification applications that is why it was chosen.
- We use OpenCV's implementation of Haar feature-based cascade classifiers to detect humans. Many pre-trained face detectors are available from OpenCV as XML files on GitHub. The XML file containing the pre-trained face detector is the Haar classifier's only input parameter.

- We use the VGG 16 model, which was trained on the ImageNet dataset, a very big, highly popular dataset used for picture classification and other vision tasks, to detect dogs. We feed our input image into the VGG-16 model, which produces numeric output that matches to the expected output's class. The input to our VGG-16 model is a 224×224 normalized image tensor. The VGG16 model won the 2014 Imagenet competition and achieves 92.7% top-5 test accuracy in the ImageNet dataset, which is a dataset of over 14 million images belonging to 1000 classes, making it a very good choice.
- In this section, we'll use the transfer learning technique. In transfer learning, a pretrained model is used which is then only modified according to the intended use. In my case we will use the pretrained ResNet-152. A pretrained model is utilized in transfer learning, and it is merely updated according to the intended purpose. The model we use here is the ResNet-152 model, because of its high accuracy. We make use of the pre-trained weights from the convolutional layer of the model and retrain our own fully connected layer added to it

2.4 Benchmark

The accuracy of the Convolutional Neural Network that we build from the ground up should be at least 10%. This was chosen in consideration of the difficulty of distinguishing distinct dog breeds even for humans, and so getting high accuracy using bare-CNN is not a simple task. To be effective in a dog breed classifier app, the pre-trained model must have an accuracy of at least 60%

3. Methodology

3.1 Data Preprocessing

Not all photographs have the same resolution, as we've already seen. In addition, the distribution of the photos varies between dog breeds. There is a little data imbalance in this case.

For our human face detector, we use a pre-trained classifier. As a result, no human dataset photos are required for model training. The photos, on the other hand, are used to evaluate the performance of our human face detector. Converting the image to grayscale is the only preparatory step necessary to achieve the same result.

The photos are used to train the dataset for the dog detector and the dog breed classifier. We do some pre-processing on the photos before feeding them to the model, the images are resized to 224×224, the images are cropped to the center, the images are also randomly rotated, and the images are normalized.

These pre-processing stages give us a wider range of data while also preventing overfitting in our model.

3.2 Implementation

For the implementation of algorithms and techniques:

- **The Human Face Detector**: we use OpenCV's implementation of Haar feature-based cascade classifiers to detect human faces in images. The image was converted into grayscale as mentioned above. Also to find faces number in the image this method detectMultiScale() was used. This returns the number of human faces that the classifier has detected.

- **The Dog Detector**: a VGG16 pre-trained model is used to create the dog detector. The VGG 16 model was obtained and loaded with pre-trained weights before being transferred to the GPU. The transformations mentioned in the predict function are used in the predict function. The image is loaded into the model after the "Data Preprocessing" stage is completed. We assert that the image contains a dog if the output falls between 151 and 268. And as for the dog Breed Classifier from Scratch PyTorch was used to design, train, and test a model from scratch. So, the model was created as follows:

```
def __init__(self):
    super(Net, self).__init__()
    ## Define layers of a CNN

    self.conv1 = nn.Conv2d(3, 32, 3, stride=2, padding=1)
    self.conv2 = nn.Conv2d(32, 64, 3, stride=2, padding=1)
    self.conv3 = nn.Conv2d(64, 128, 3, padding=1)

    self.fc1 = nn.Linear(128 * 7 * 7, 500)
    self.fc2 = nn.Linear(500, 133)

    self.dropout = nn.Dropout(p = 0.25)

    self.pool = nn.MaxPool2d(2, 2)
```

```
def forward(self, x):
    ## Define forward behavior

    x = self.pool(F.relu(self.conv1(x)))
    x = self.pool(F.relu(self.conv2(x)))
    x = self.pool(F.relu(self.conv3(x)))
    x = x.view(-1, 128 * 7 * 7)
    x = self.dropout(x)
    x = F.relu(self.fc1(x))
    x = self.dropout(x)
    x = self.fc2(x)
    return x
```

- **The Dog Breed Classifier:** in this case we applied the technique transfer learning, and the dog classifier makes use of a ResNet-152 model. The pre-trained weights from the convolutional layer of the ResNet-152 model were taken and used for the dog breed classifier model. We train our own fully connected layer with 2048 input features, 512 hidden layer nodes, and 133 output classes in the final layer. The model was trained for 20 epochs, and the log loss error was used to evaluate its performance.

- **Dog Breed Classifier app:** all the preceding methods are combined in the final dog breed classification app to create a single sample app. Using our detectors, we determine

whether the image is of a human or a dog. We utilize our dog breed classifier software, which takes the image path, loads it, and uses the predict_breed_transfer() function to determine the projected breed. The predict_breed_transfer() function resizes and normalizes the image to the size predicted by the model, which is 224224. Then, using our dog breed classifier model, it generates a forecast for the image. The anticipated dog breed or the closest match to the human in the image is then displayed to the user.

3.3 Refinement

The CNN created from scratch have accuracy of 11%, Though it meets the benchmarking, the model can be significantly improved by using transfer learning. To create CNN with transfer learning, we have selected the Resnet152 architecture which is pre-trained on ImageNet dataset. The initial model had a learning rate of 0.001 was used for up to 20 epochs, resulting in an accuracy of 80%. We were able to boost it to 85% by employing a learning rate of 0.05, which fulfils our benchmark expectations.

4. Results

4.1 Model Evaluation and Validation

OpenCV's implementation of Haar feature-based cascade classifiers was used to develop the human face detector function. In the first 100 photographs of the human face dataset, 98 percent of human faces were detected, while 17 percent of human faces were discovered in the first 100 images of the dog dataset.

The pre-trained VGG16 model was used to construct the dog detecting function. In the first 100 photographs of the dog dataset, 100% of dog faces were spotted, but just 3% of dog faces were discovered in the first 100 images of the human dataset.

During training, the model was assessed on a validation set, and subsequently on the test set. From all the combinations, the best performing hyperparameters were chosen.

The final model employs Stochastic Gradient Descent with a learning rate of 0.05 and achieves an 85 percent test accuracy, correctly predicting 714 of the 836 photos tested. This satisfies our requirements and may be utilized in our final project.

The model was trained with 834 photos representing all 133 dog breeds to ensure its robustness.

4.2 Justification

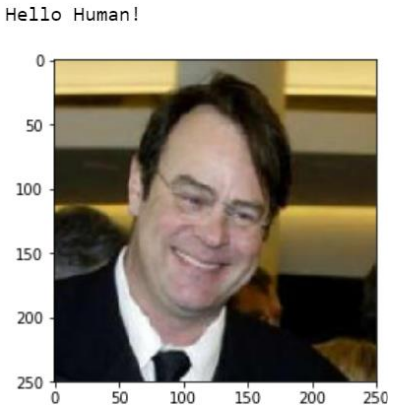
On the test dataset, our final model has an accuracy of 85 percent, which is higher than our target of 60 percent. Given the challenge of classifying dog photographs discussed above, this is a reasonable accuracy that we can use in our dog breed classifier software.

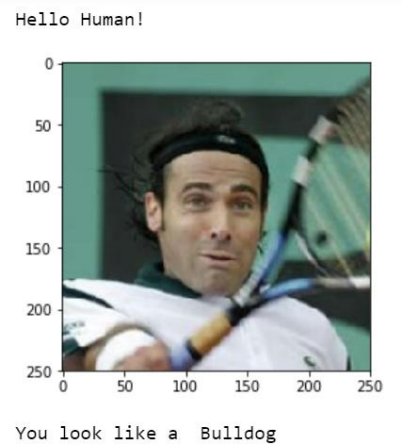
The model performed better than expected compared to the CNN model created from scratch which had only 13% accuracy.

5. Conclusion

5.1 Free-Form Visualization

Some screenshots of outputs are provided here:

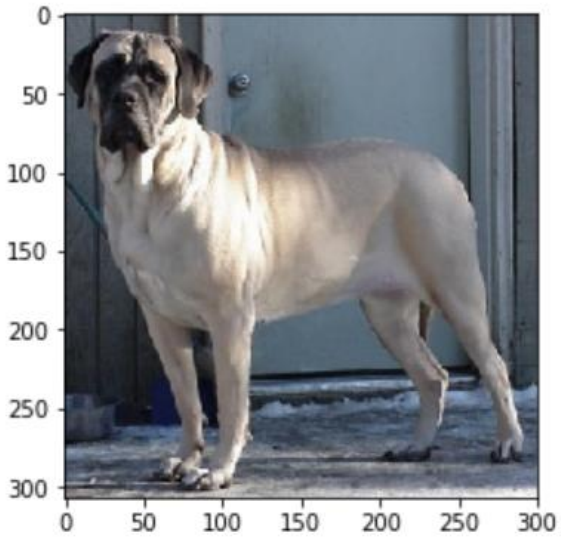




```
Hello Dog!
```


```
You're a  Mastiff
```

```
Hello Dog!
```




```
You're a  Mastiff
```

Here you can see both dogs are being recognized as dogs and their breed is shown successfully.

Also for the two humans they are recognized and their corresponding breed is submitted successfully.

5.2 Reflection

The steps used to complete this project can be described as follows:

- The data was gathered and analyzed.
- The data was pre-processed.
- A reliable method for recognizing humans in images has been implemented.
- A dog detector was created using VGG 16 model.
- For our models, we chose a benchmark.
- A dog breed classifier was built from scratch using CNN.
- Transfer learning was used to create a final dog breed classifier for usage in our final app.
- The project's various components were all linked together.

The most difficult part regarding the project was to tune the god breed classifier.

5.3 Improvement

Regarding improvement there are somethings that can provide better results:

- Improving the accuracy of the dog breed classifier using another model.
- Improving the accuracy of the human detector using another algorithm than haar
- More manipulation on the training images to make input dataset more robust.
- choose another hyperparameters.

6. References

1. Original repo for Project - GitHub: https://github.com/udacity/deep-learning-v2-pytorch/tree/master/project-dog-classification
2. Udacity: The dog dataset. url: https://s3-us-west-1.amazonaws.com/ udacity-aind/dog-project/dogImages.zip.
3. Udacity: The human dataset. url: https://s3-us-west-1.amazonaws.com/ udacity-aind/dog-project/lfw.zip.
4. OpenCV. Face Detection using Haar Cascades. https://docs.opencv.org/trunk/db/d28/tutorial_cascade_classifier.html.
5. Ayanzadeh, Aydin and Sahand Vahidnia. Modified Deep Neural Networks for Dog Breeds Identification. May 2018. url: https://www.researchgate.net/profile/Aydin_Ayanzadeh2/publication/325384896_Modified_ Deep_Neural_Networks_for_Dog_Breeds_Identification/links/5cd0345ea6fdccc9dd90690c/Modified-Deep-Neural-Networks-forDog-Breeds-Identification.pdf
6. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).