Machine Learning Engineer Nanodegree

Capstone Project Report

Dog Breed Classification

Brian Collins

Definition

Project Overview

In this project I will be building an app that is able to process an unseen image of a dog, and predict the breed of the dog from the image. Additionally the app will accept images of people, and pick the breed of dog that the person resembles the most. The image classification will be achieved using a Convolutional Neural Network (CNN). Convolutional Neural Networks are a class of Deep Neural Networks that are frequently used for image processing tasks.
Convolutional Neural Networks were inspired by studies of the brain's visual cortex, in particular the discovery that many neurons in the visual cortex have a small local receptive field, meaning they only react to visual stimuli located in a limited region of the visual field. Also discovered was that some neurons have larger receptive fields, and they react to more complex patterns that are combinations of the lower-level patterns [1].

The dataset that will be used in this project has been provided by Udacity, and it contains approximately 8000 images of dogs from 133 different breeds. The CNN will be trained on these images.

Problem Statement

The core problem that I am trying to solve in this project is to create an app that can correctly classify dog breeds from an unseen/unlabelled image of a dog. This will be achieved using a branch of Machine Learning called Deep Learning, in particular Convolutional Neural Networks which are especially used for image classification tasks.
In order to do this I will use PyTorch - an open-source machine learning library.

A dataset of approximately 8000 labelled images of dogs from various breeds has been provided by Udacity. I will use this dataset to train a CNN. Once the CNN has been trained, I will be able to provide it with new unseen and unlabelled images of dogs, and it will try to predict the breed of dog. A byproduct of this is that I will also be able to provide it with images of humans, and it will predict the breed of dog the human most resembles.

Therefore, the application will serve two functions - one more practical in nature, and the other more humorous:

1) Classification of dog breeds from actual images of dogs
2) Predicting which breed of dog a person most resembles from images of people

Metrics

The Convolutional Neural Network created using transfer learning must have an accuracy of 60%. Since there are 133 classes, randomly guessing would produce an accuracy of 0.75%. The CNN created from scratch must have an accuracy of at least 10%. Since the training data that I am using is unbalanced across the classes, I will use Cross Entropy Loss as an evaluation metric to quantify the performance of both the benchmark model and the solution model. Cross Entropy Loss is useful for classification problems with multiple classes and there is an unbalanced training set. I will also compare the accuracy of the transfer-learning model and the from-scratch model.

Analysis

Data Exploration

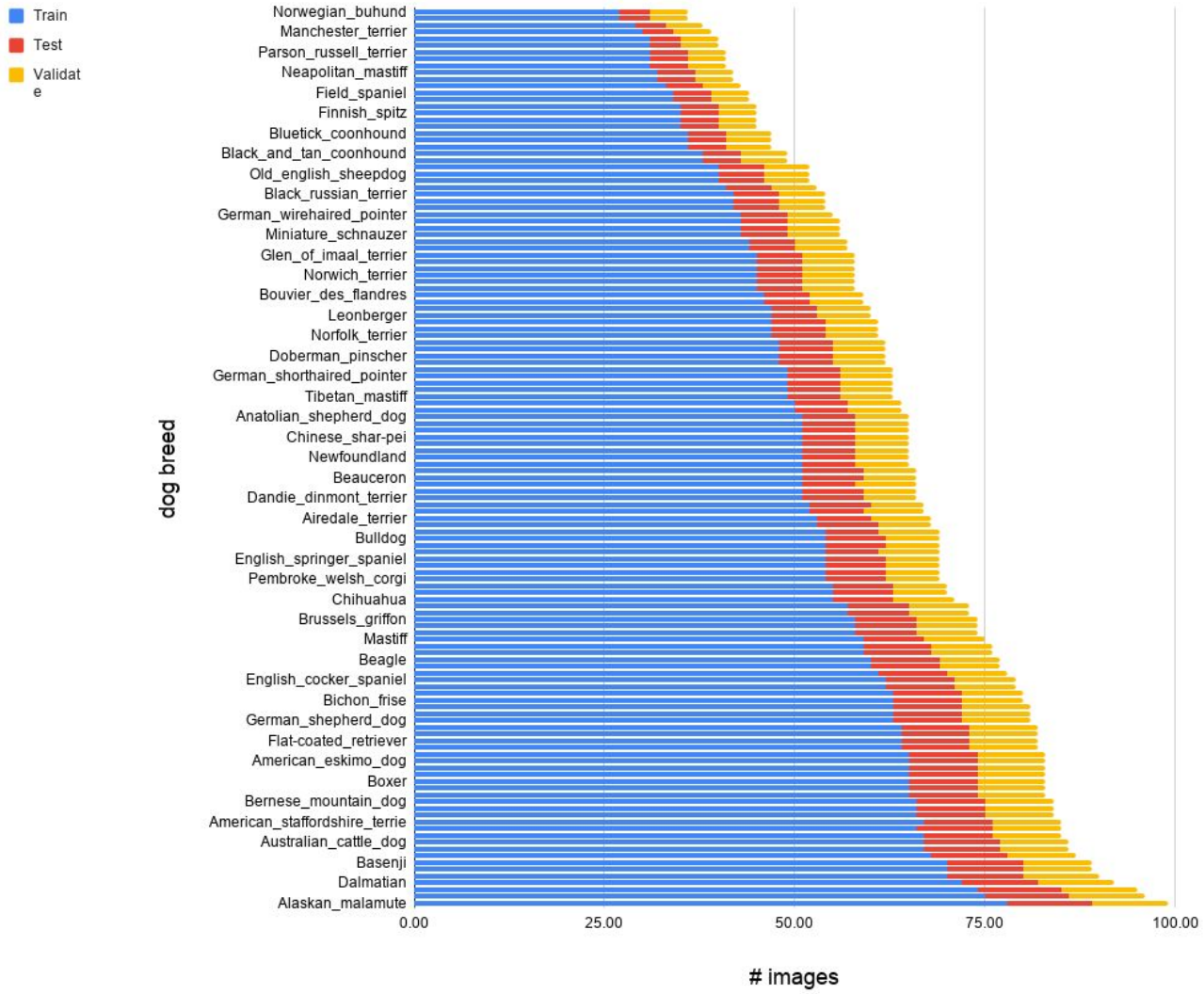
The dataset that I will be using for training consists of 8351 images of dogs, from 133 different breeds (classes). The dataset is unbalanced across the classes, i.e. some breeds have more images than others. The dog images are of various sizes and aspect ratios. During training, the images will be augmented in various ways (such as rotating, horizontal flipping etc) to help the algorithm generalise to the data. To test out the application I will also be using some images of human faces from a database of 13233 images. These datasets have been proved by Udacity.

Also provided by Udacity is a dataset of 5750 images of human faces. This data will not be used to train the models, but rather will be used to try out on the final trained model.

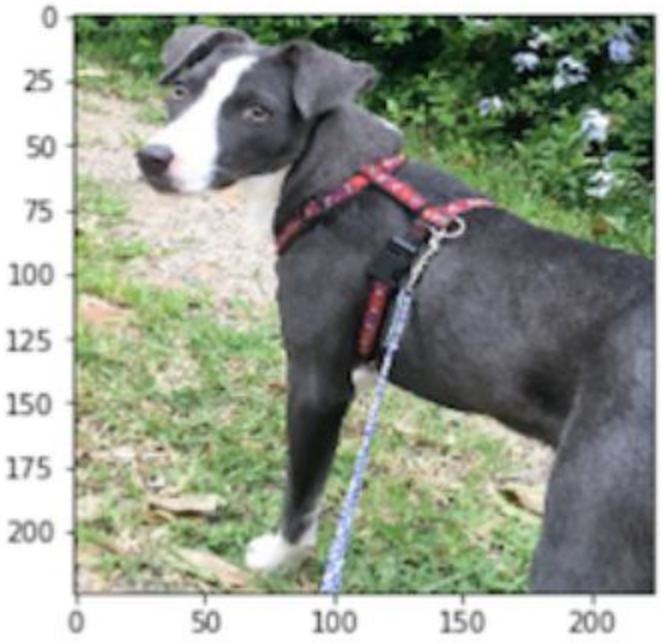
Exploratory Visualization

There are 133 different breeds of dog in this dataset, which represent 133 classes. There are 8351 images in total, and these are split into 3 groups of images, for training, testing, and validation. Not all dog breeds have the same number of images, and thus there is an imbalance in the classes. For example there are 36 images of Norwegian Buhunds, whereas there are 99 images of Alaskan Malamutes. The stacked bar chart below shows the number of images of each breed. This visualisation was chosen because it shows the total number of images in each class, and it also shows the breakdown between training, testing, and validation. Approximately 75% of

the images for each class are dedicated to training, while the other 25% is split between training and validation.



The image below shows what a typical image looks like. The images can range in size and aspect ratio, but various preprocessing steps will be carried out to normalise the images before training occurs.



Algorithms and Techniques

Four different algorithms will be used throughout the course of the project.

- For detecting dogs within an image, I will use a pretrained VGG16 Convolutional Neural Network (CNN). This model has been trained on ImageNet - a very large benchmark dataset used for image classification. ImageNet contains over 10 million images of objects from 1000 categories.

- If a dog is not detected within an image, I will check if there is a human face in the image using the Haar Cascade Classifier. This is a pre-trained face detector provided by OpenCV [4].

- If a dog is detected in the image, then the breed of the dog will be predicted using a ResNet50 CNN. ResNet-50 is a convolutional neural network that is 50 layers deep, and pretrained on more than a million images from the ImageNet database [2]. This pretrained network will be suitable for this problem because it has learned rich feature representations for a wide range of images. Although this model has already been trained for image classification, I will retrain it for this specific dog breed classification task, through a process called transfer-learning. In order to retrain the model, I replace the fully connected portion of the network with one that has 133 outputs for our 133 classes.

- I will also try to build a CNN from scratch where I define the architecture myself. The performance of this 'from scratch' CNN will be compared against the performance of the pretrained ResNet50 CNN. The model has three convolutional layers, with a kernel size of 3, a stride of 2. The first convolutional layer has 3 input channels, and the last convolutional layer has 128 output channels. Between each convolutional layer is a pooling layer which reduces the input by a factor of 2. Finally there are two fully connected layers with an output of 133, which predicts our classes. Dropout is applied to avoid overfitting, with the probability of an element being zeroed set to 0.25.

Benchmark

The Convolutional Neural Network created using transfer learning must have an accuracy of 60%. Since there are 133 classes, randomly guessing would produce an accuracy of 0.75%. The CNN created from scratch must have an accuracy of at least 10%.

I will also use the 'from scratch' CNN as a benchmark by which to compare the performance of the ResNet50 CNN trained with transfer learning. The expectation is that this model will outperform the 'from scratch' model.

Methodology

Data Preprocessing

The images in our dataset vary in size and aspect ratio. I take various steps to preprocess these images in order to put them into a more suitable format for training and prediction.
For the training data I resize each of the images to a size of 256x256 pixels. I then take a random resized crop of 224px. I then take various data augmentation steps to diversify the training data. Data Augmentation is a strategy for increasing the diversity of data available for training, without actually collecting new data. The data augmentation steps that a take are applying a random horizontal flip to the image, and applying a random rotation of 10 degrees. These steps will help diversify the dataset and help the algorithm generalise better during training.
Finally I convert the images to tensors and apply normalization. All of these transformations are applied using PyTorch’s Torchvision.Transforms module.

During testing and validation, there is no need for the data augmentation steps. Therefore I simply resize the images to 256x256px and take a center crop of 224px. I then convert to a tensor, and normalize each image.

Implementation

The main function that the application runs is called run_app. The run_app function has three subroutines (dog_detector, face_detector, and predicted_breed).

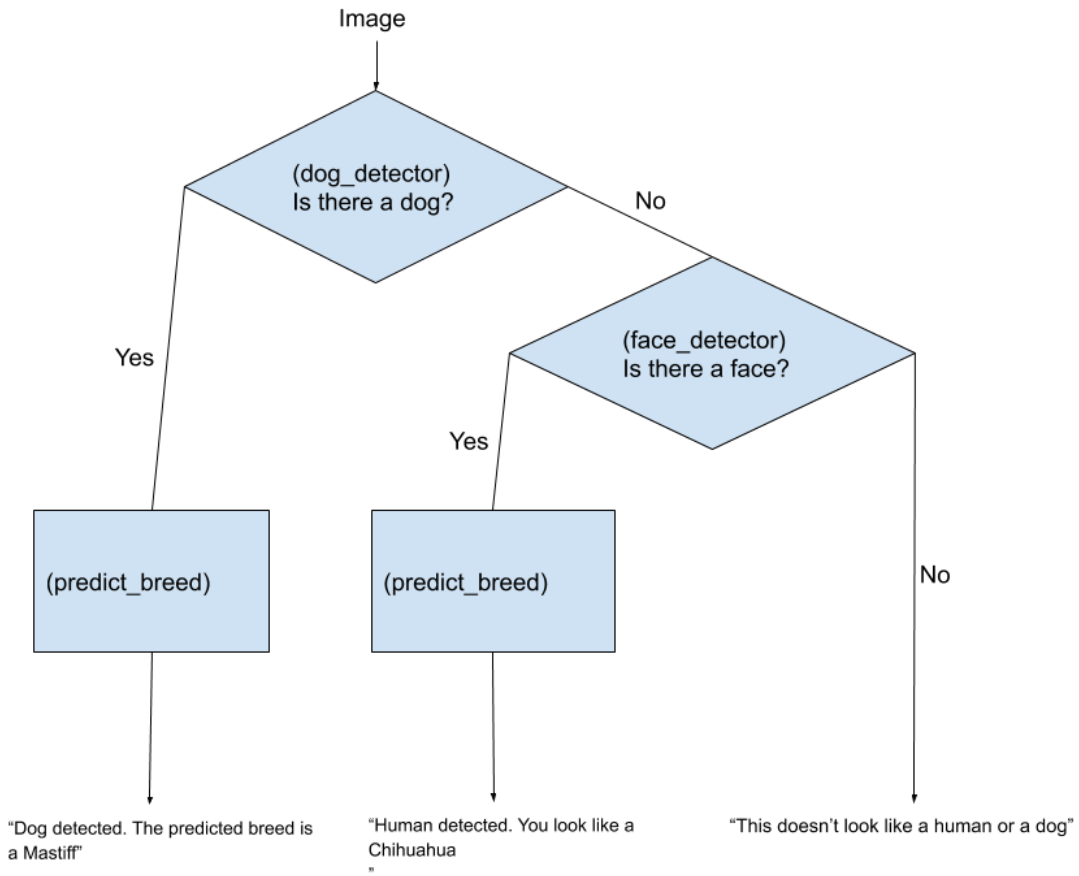
- When the run_app function receives an image, it will first try to detect if there is a dog in the image using the function dog_detector. The dog_detector function uses a pretrained VGG16 CNN to detect whether or not a dog is detected.

- If a dog is detected in the image, the predicted_breed function is used to make a prediction of the breed of the dog. The predicted_breed function uses the ResNet50 CNN that has been retrained for dog breed classification using our dataset.

- If the dog_detector doesn't detect a dog, then the function face_detector is used to detect if there is a human face in the image. The face_detector function uses the Haar Cascade Classifier to detect human faces in the image. If a human face is detected, the predicted_breed function is used to predict a breed of dog that most resembles the face.

- If neither a human or a dog is detected in the image, an appropriate message is returned.

The diagram below illustrates the workflow of the application.



- Refinement

During training of the ResNet50 CNN, I tuned various model parameters in order to improve the performance of the model. I initially trained the model for 5 epochs, with a learning rate of 0.05. I achieved better performance by increasing the number of training epochs to 10, and reducing the learning rate to 0.01.

Results

Model Evaluation and Validation

The ‘model from scratch’ was trained for 10 epochs. It was then tested using the test dataset of dog images, and the test loss and accuracy was measured. The final Test Loss for this model was 3.928150, and the Accuracy was a mere 12% (103/836).
The Transfer Learning model was also trained for 10 epochs with a learning rate of 0.001. After testing this model on the test dataset of dog images, it was found that the Test Loss was 0.527502 and the Test Accuracy was 83% (698/836). This represents a significant improvement over the ‘from scratch’ model.

Justification

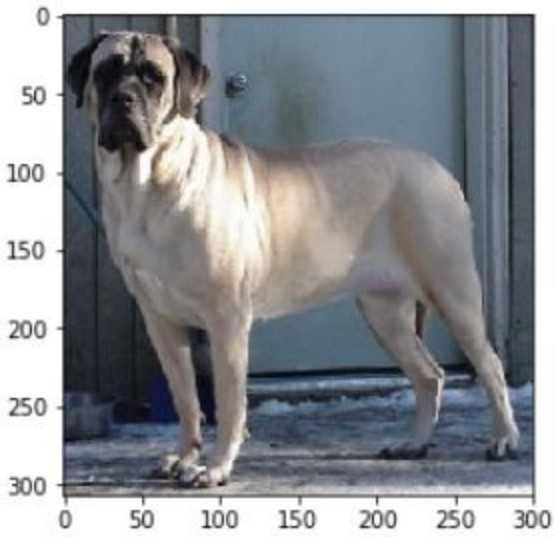
Since there are 133 classes, randomly guessing a dog breed would produce an accuracy of 0.75%. The ‘model from scratch’ performs better than randomly guessing, with an accuracy of 12%, but this performance is not sufficient for the purposes of the application. The model trained using transfer learning had significantly better performance than the benchmark ‘from scratch’ model with an accuracy of 83%. This performance is sufficient for the application.

Conclusion

Free-Form Visualization

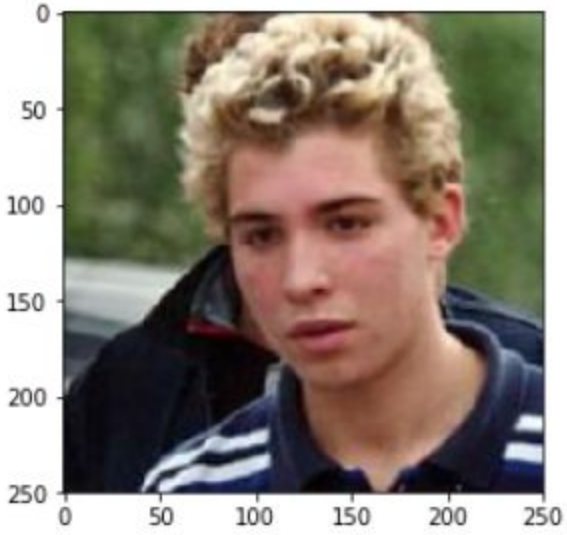
Overall both aspects of the application worked surprisingly well - both the classification of actual dogs, and the matching of dog breed to human subjects.
When provided with an image with a dog, the app would correctly predict the breed most of the time. With the application’s ability to accurately

classify dogs, one could imagine this app being useful for amatuer dog breeders, or for estimating the makeup of a mixed-breed dog etc.




```
Dog detected. The predicted breed is a Mastiff
```

When provided with an image of a human the results were often humorous, with the subject often resembling the predicted breed in surprising ways. One could imagine this being a popular humorous app that people would use for fun.




```
Human detected. You look like a  American water spaniel
```

Reflection

Upon reflection on the project as a whole, my biggest take away was the power of Transfer Learning, and the difficulty of trying to architect a Convolutional Neural Network from scratch. I was disappointed that the CNN I created from scratch only achieved an accuracy of 12%, but I suppose this speaks to the difficulty of image classification.
On the other hand the model that used transfer learning, and was simply retrained for this specific classification task, achieved a very impressive accuracy of over 80%. It is interesting that a model that has been pre-trained on a large benchmark dataset can be re-purposed just by retraining the 'Classifier' section of the CNN on a new dataset.

Improvement

For this application to be useful in practice, it would need to be capable of recognising all breeds of dogs. The model was trained on data for 133 breeds of dogs, whereas Worldwide, the FCI lists 360 officially recognized breeds. This is one area of improvement that I would like to make.
It would be interesting to try other pre-trained models to see if they yield better performance.

References

[1] Wurtz, Robert H. "Recounting the impact of Hubel and Wiesel." The Journal of physiology 587.12 (2009): 2817-2823.
[2] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[3] Alom, Md Zahangir, et al. "The history began from alexnet: A comprehensive survey on deep learning approaches." arXiv preprint arXiv:1803.01164 (2018).

[4] https://docs.opencv.org/trunk/db/d28/tutorial_cascade_classifier.html