Capstone Project
Data Scientist Nanodegree

Mohammed Alnegheimish
June 25, 2020

Dog Breed Classification

Project Overview

With the advancements of self-driving cars, robotics, and other sectors that rely on capturing the essence and contents of images and videos, the need for the improvement of computer vision algorithms surged. Computer vision has evolved in numerous methods in the past decade. Additionally, it allowed machines to accurately assess the contents of images that can be used in innovative applications.

In this project, Convolutional Neural Networks (CNNs) will be used to correctly identify different breeds of dogs in an image. It will involve training a CNN from scratch and using transfer learning to greatly boost the performance of the model.

Project Statement

The goal is to build a model that will predict the breed of the canine, the tasks will involve the following:

1- Load the datasets
2- Use Haar Cascade to detect human faces
3- Import a pretrained Resnet classifier for the purpose of detecting dogs
4- Create a CNN from scratch to classify dog breeds
5- Use transfer learning to classify canine breeds.
6- Build an algorithm that will apply the previous steps
7- Assess our implementation

The built algorithm will check if the dog is in the image, then it will predict its breed using the pre-trained network. If the image is of a human, it will return the resembling dog breed. However, if the image contains none of the two, it will ask the user to provide another input.

Metrics

The metric used in this project is the accuracy, as it is the most intuitive metric to be used to assess the predictions of the model.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

During training the best parameters will be picked when the loss function is minimum. The loss used for this project is categorical cross-entropy, which will heavily penalize misclassifications during training.

Data Exploration

There are two datasets used in this project, the dog images dataset, and the human images dataset.

The dog dataset contains 8351 total images of dogs, that will be split into 6680 for the training, 835 for validation, and 836 for the test subset. The dimensions of the images varies, which will be addressed later. This dataset will be used for both dog detection and breed classification using CNNs.

The human dataset contains 13233 total images. However, this dataset will only be used to refine and choose our human face detector for this project.

Data Visualization

To further understand the of the dog dataset, the distribution of the different breeds in the training subset was visualized.

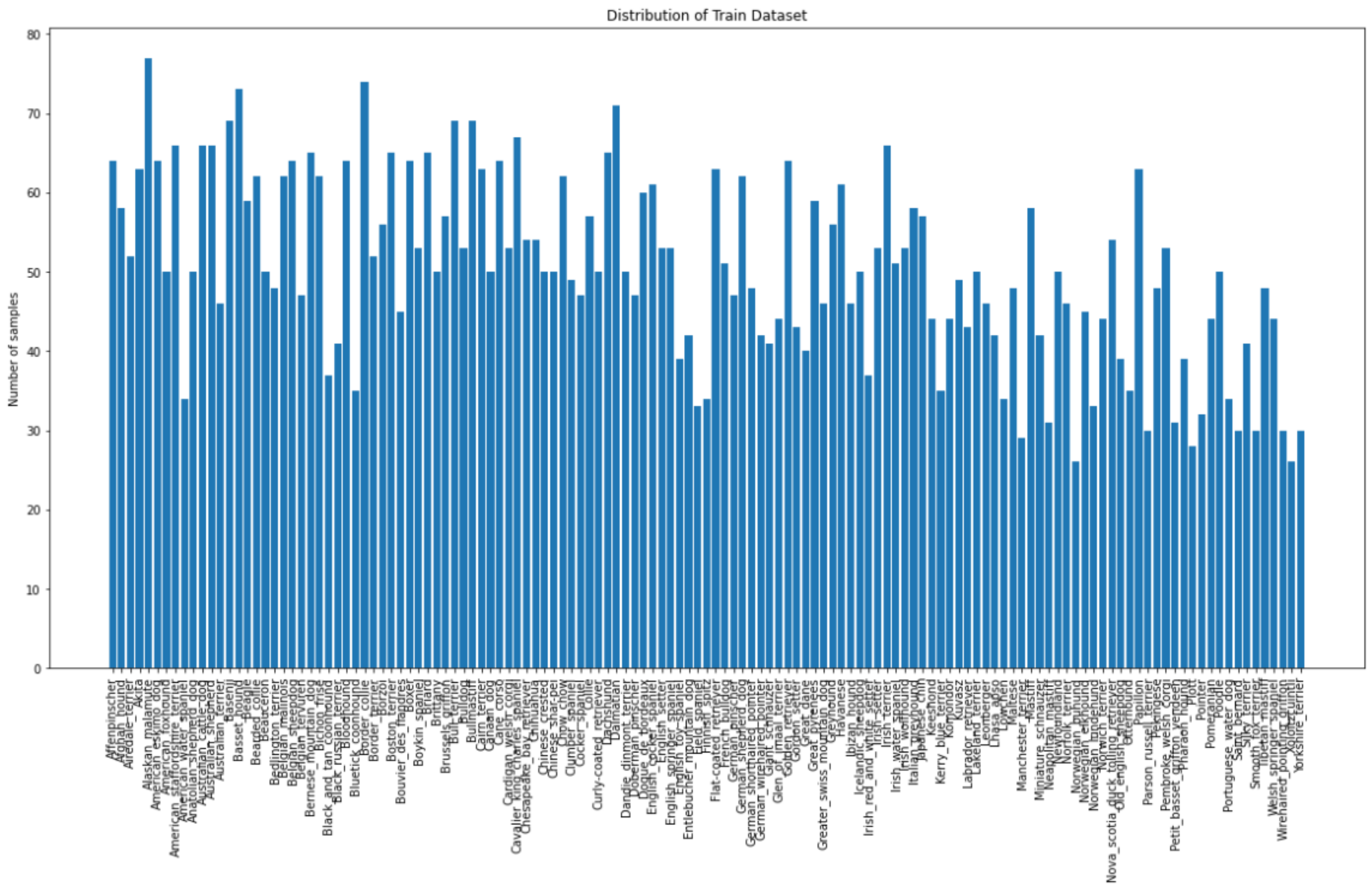


Figure 1 Distribution of Train Dataset

After looking at the plot, we notice that there are breeds that are oversampled and the others are undersampled, with the average number of samples across all breeds is 50.26.

Methodology and Approach

Data Preprocessing

The data was preprocessed using TensorFlow as a backend, where it will be converted to a tensor with the dimensions of 224x224x3, they are the rows, columns, and number of channels, respectively. If a pretrained network was used, the features of the image will undergo a pipeline where it will extract the features before proceeding the FC layers.

Implementation

1. Human Face Detection

In this section, two widely used face detection algorithms will be compared. The first is Haar cascades, where it will collect the Haar features of the image, adds up the pixel intensities in the region, then calculates the difference of the sum [1].

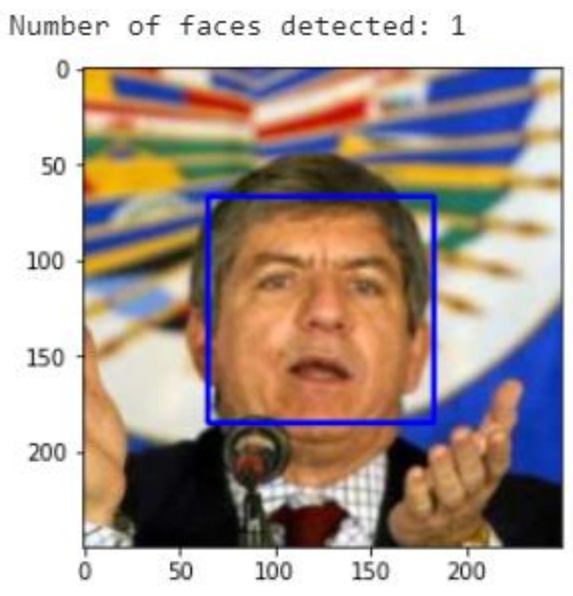


Figure 3 Haar face detection

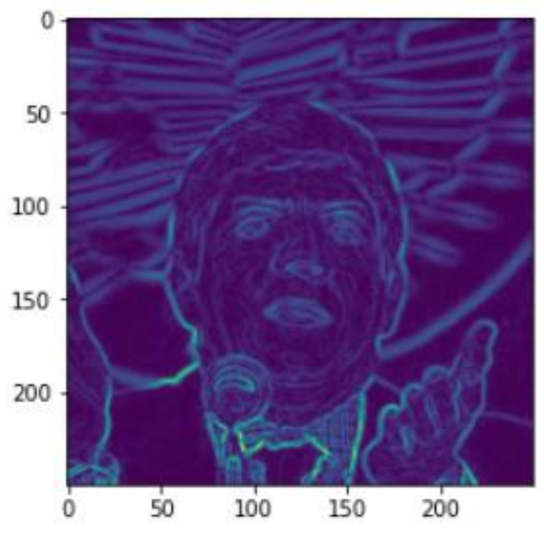


Figure 2 HOGs featuers of an image

The second face detector used is Histogram of Oriented Gradients (HOG). HOG is a feature descriptor that is widely used in computer vision for the purpose of object detection, where it can capture the contours and some texture information of the image by using gradients [2].

2. Dog Detection

Before the classification of the dog breeds, we must first look if the dog exists in the image. The detector used in the project was provided by Udacity. It is a pre-trained ResNet-50 model that accurately detected all the dogs in the first 100 images in the dataset and did not detect any dogs in the first 100 human images. Since it is highly accurate, it will be the first step of the algorithm later on.

3. *Creating CNN from Scratch*

Training a CNN from scratch on a low number of samples is challenging, this is even harder when the model has to predict between 133 different classes. One way to tackle this issue is to generate new augmented data from the original dataset. But even then, the results may vary.

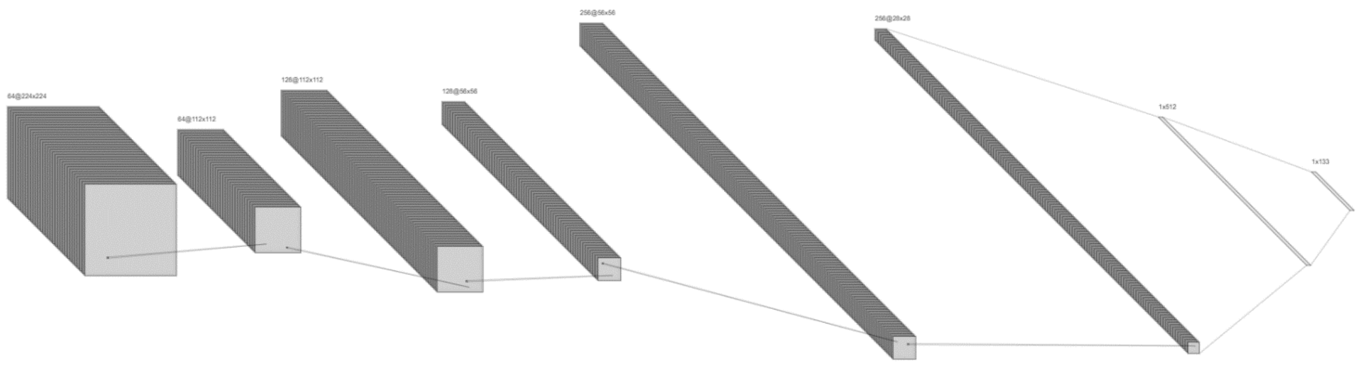


Figure 4 CNN architecture used

CNNs are proven as viable methods to extract features from the input. The input passes through a sequence of layers, alternating between convolutional layers, and max-pooling layers. Convolutional layers are the feature extractors of the model, while the max-pooling layers reduce dimensionality of the image in order to decrease the computational power, while also serving as a noise suppressant. After each layer, it will have a ReLU activation. The last layer will have a Softmax activation to output the probability of each label. Dropouts were also used as a regularization method to prevent overfitting. The expectations of this model are not high, as it will not attain the best parameters for feature extraction using a low number of samples. The performance of the model will be measured in test accuracy.

4. *Transfer Learning to Classify Dog Breeds*

In this section, transfer learning will be utilized in our classifier. A pre-trained model that was trained on the ImageNet dataset, which had over 1.2 million images with around 1000 separate object category. After comparing VGG19, ResNet50, Xception, and Inception, Xception was chosen due to getting the best top-1 score out of the rest of the models.

Global Average Pooling (GAP) is used to minimize overfitting by reducing the number of parameters in the model, it will also flatten the features dimensions into one. This is followed by the fully connected layers that will classify the dog breed. The model is expected to perform way better than the last method used, as the images will have gone through a pretrained CNNs to extract their features. Test accuracy will be used to measure the performance of the model.

5. *Setting Up the Algorithm*

The goal of the algorithm is to determine the following:

- If the image contains a dog, predict its breed.
- If the image contains a human, output the resembling breed.
- If the image does not contain a human or a dog, output a message indicating an error.

Since we are confidant of the dog detector the most, we will be using it first followed by the human detector. If both detectors returned false values, print a message for the user to try another picture.

Refinement

Before choosing a face detector, both Haar Cascades and HOG will be compared.

Metric / Method	Haar Cascades	HOG
Percentage of human faces detected in human dataset.	99.0%	99.0%
Percentage of human faces detected in dog dataset.	12.0%	9.0%
Time elapsed during detection.	34.33s	104.87s

The tests above were conducted on the first 100 images of each dataset. The performances of both methods were almost identical, with HOG having a slight edge. However, it takes 3 times longer to process the same number of images as Haar. Thus, Haar cascades will be chosen as the face detector for the rest of the project.

Results

Model Evaluation and Validation

1. *CNN from Scratch*

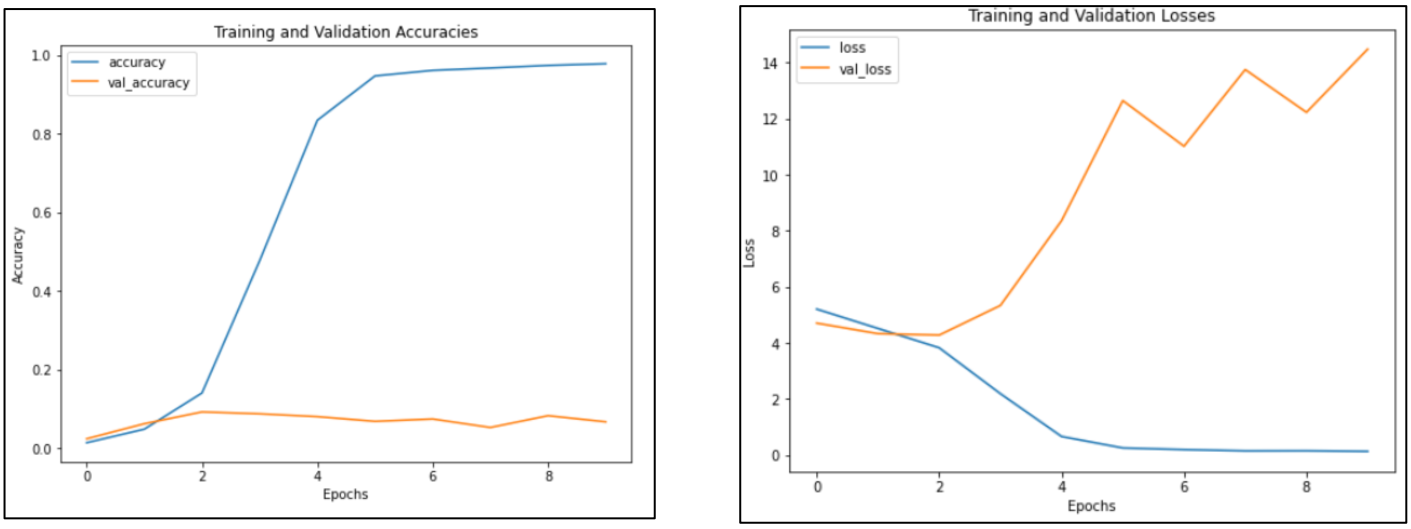


Figure 5 CNN Learning curve

After training the model, the performance was abysmal and it only overfitted to the training data. The achieved accuracy on the test dataset was 8.01%. It is expected from a model that is trained on a small number of samples

2. *Xception Network*

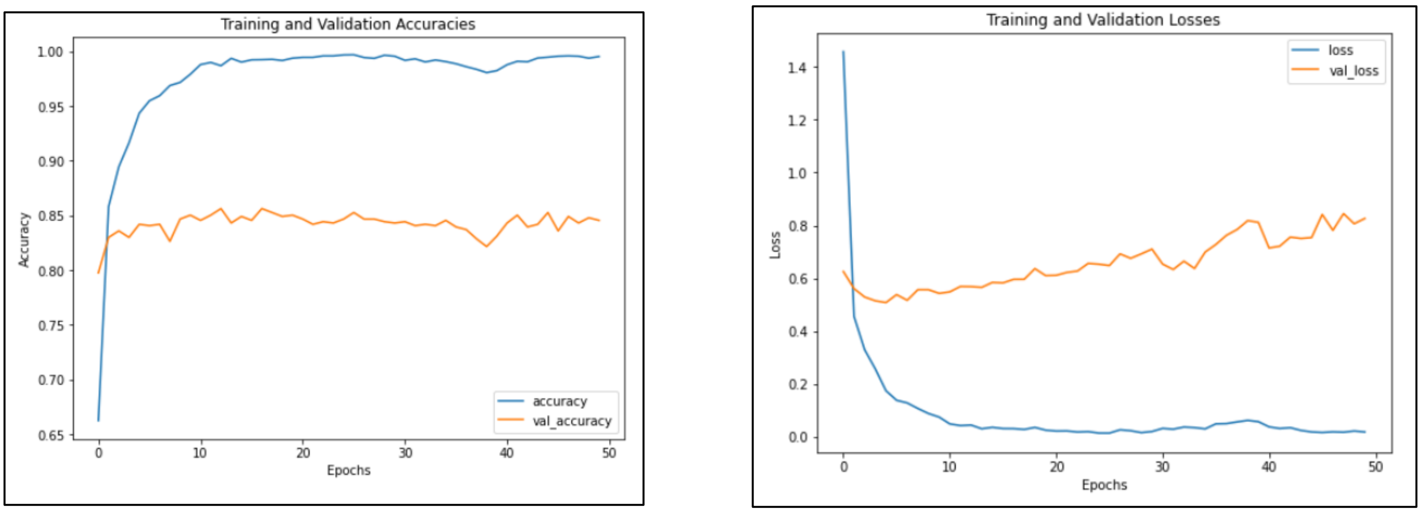


Figure 6 Xception learning curve

After training the Xception model, it has performed exceptionally well. Obtaining a test accuracy of 83.97%, which is a significant improvement than the last model. This is due to using a feature extractor that was trained on a large dataset.

3. *Implementation Assessment*

Before going through with the algorithm, I have written a code that will help understand the predictions of the model better. It will output the image itself, the true label, and the top 5 predictions with their probabilities like in the following figures.

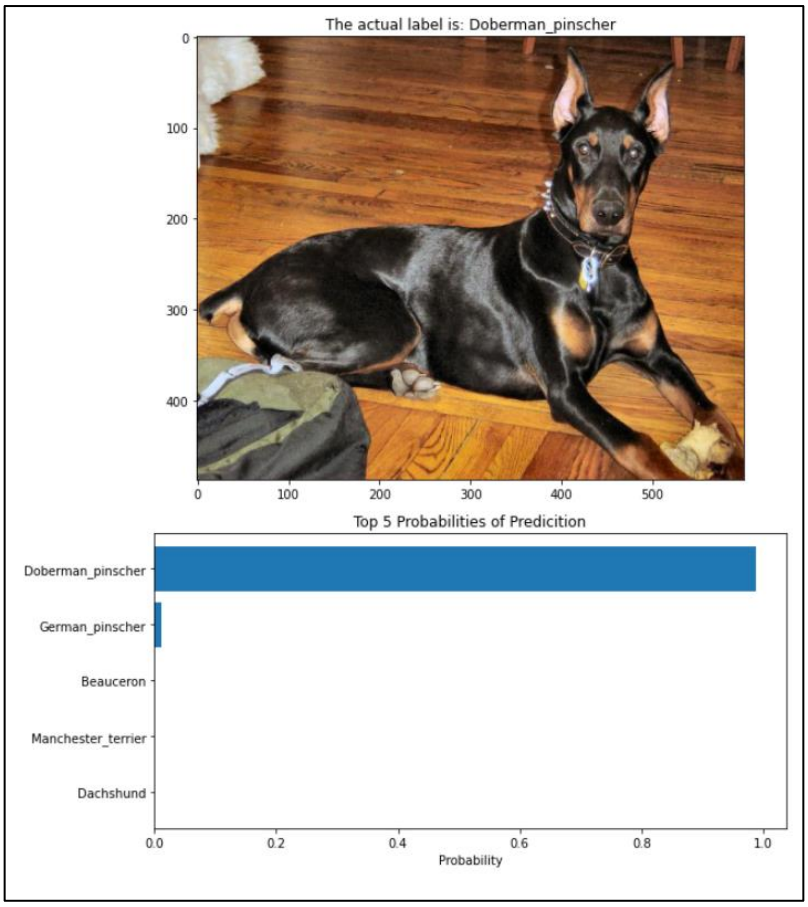
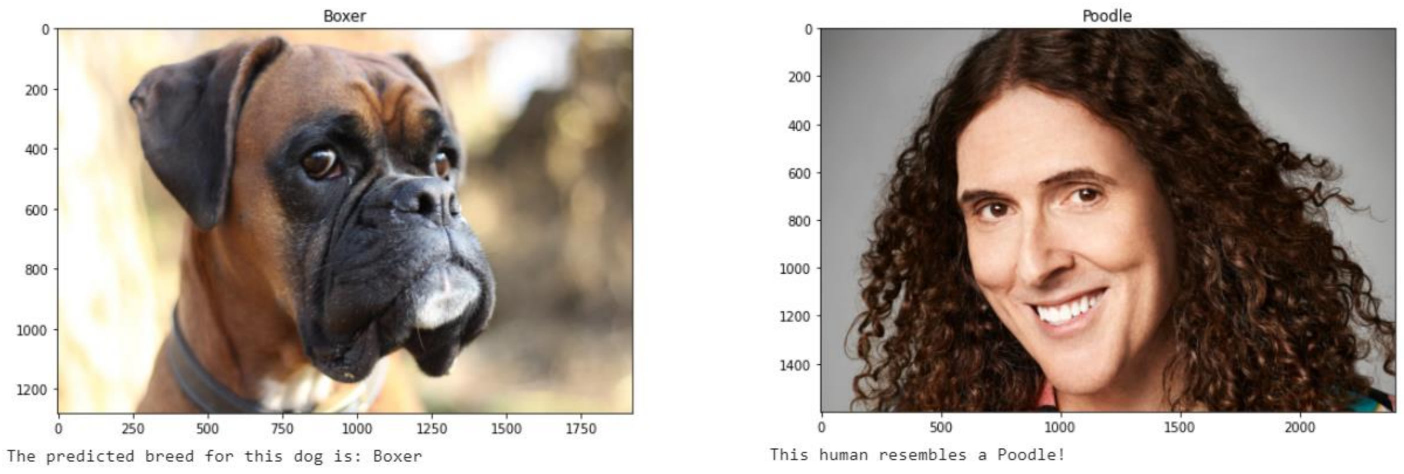


Figure 7 Sanity Check

After viewing some of the results of the model, I have come to understand the model struggles when the dog is not in focus, multiple dogs, or breeds that are almost identical.

Justification

The final model has achieved a test accuracy of 83.97%, when compared to the CNN that was built from scratch that got an accuracy of 8.01%, we observe a significant difference in performance. One of the factors of why the Xception architecture performed better is that it had 36 convolutional layers that was trained in the ImageNet dataset.



```
The predicted breed for this dog is: Boxer
```

```
This human resembles a Poodle!
```

Figure 8 Algorithm output

Finally, after using some images to try out the algorithm, the results were outstanding. The model was able to classify all shades of Labrador retrievers as the same breed. It also detected whether the picture contains a dog, human, or neither.

Conclusion

Reflection

The final model was built using a pre-trained CNN that has achieved a fine accuracy of 83.97%, which might be above human capabilities. The model assesses the image and looks for either a dog or a human, then gives a prediction for either. This model performed significantly better than the model that was built from scratch that got 8.01% accuracy.

Improvement

There are still improvements that may add more performance to the model some of them are:

- Using data augmentation as a regularization method.
- Increasing the number of samples, as we have very little data compared to the number of breeds.
- Experimenting with other optimizers during training like Adam.
- Fine-tuning the parameters of the model.

References

[1] http://www.willberger.org/cascade-haar-explained/

[2] https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html