CS 82 Final Project: Dog Breed Detection

Sharjil Khan

1. Introduction & Motivation

Classifying dog breeds correctly allows adoption agencies to look at a dog's likely traits based on their breed. This allows them to explore the dog's differentiating personality as well, so that they can provide a complete picture to possible adopters. Further, adoption agencies often work with volunteers who may not specialize in breed identification and having a program that can evaluate a dog's picture to help determine a breed can allow the volunteers to help in the dog's evaluation.

Detecting the breed of a dog from its visual appearance is not an easy task. The task is made even more difficult by the fact that most dogs are a mixture of different breeds. Having a tool to at least narrow down to a few breeds when trying to determine the breed of a dog would be of great help to many agencies that deal with dogs. This project attempts to solve this problem by identifying dog breeds from their images.

2. Project Outline

Given the complexity of the task at hand, I decided to break it down into some smaller manageable problems so that the issues with the models could be well understood and fixed at easy steps.

I decided to first attempt to identify between only 8 dog breeds using a simple model and then build upwards to more difficult models. My plan was to gradually move towards more difficult problem of detecting breeds between 16 dog breeds and then 32 dog breeds and so on. This strategy allowed me to understand the problems with the model at each step and improve the model performance without having to wait for hours training the model on very large datasets. It also made it easier to see which dogs were being misclassified and how changing the layers of the model where effecting the model performance. Because the problem of detecting dog breeds would be similar for 8 breeds verses many breeds. The convolutional layers would be identifying similar traits, so I believed if a model is fine tuned for a few breeds it could then be scaled easily for the more difficult problems.

The final goal of the project was to detect over a hundred dog breeds with an accuracy level of close to 90%.

3. Data Used

The data used for the project comes from Stanford Dogs Dataset**:** http://vision.stanford.edu/aditya86/ImageNetDogs/main.html.

This dataset contains about 20,580 images of dogs classified into more than 100 breeds. Since the number of dog breeds represented is quite high, the number of images per dog breed in the image set is not that high. It averages to about 170 images per dog breed. The pictures have different pixel counts that make it a challenge to compare them against each other. The dogs we want to detect are also part of a larger environment as seen in the examples in Figure 1. This makes it even harder for the models to classify the dog breeds correctly. I tried to use some libraries in python to try to address some of these issues and focus on the more important parts of the images to improve accuracy scores. These methods are described in detail in the following sections.

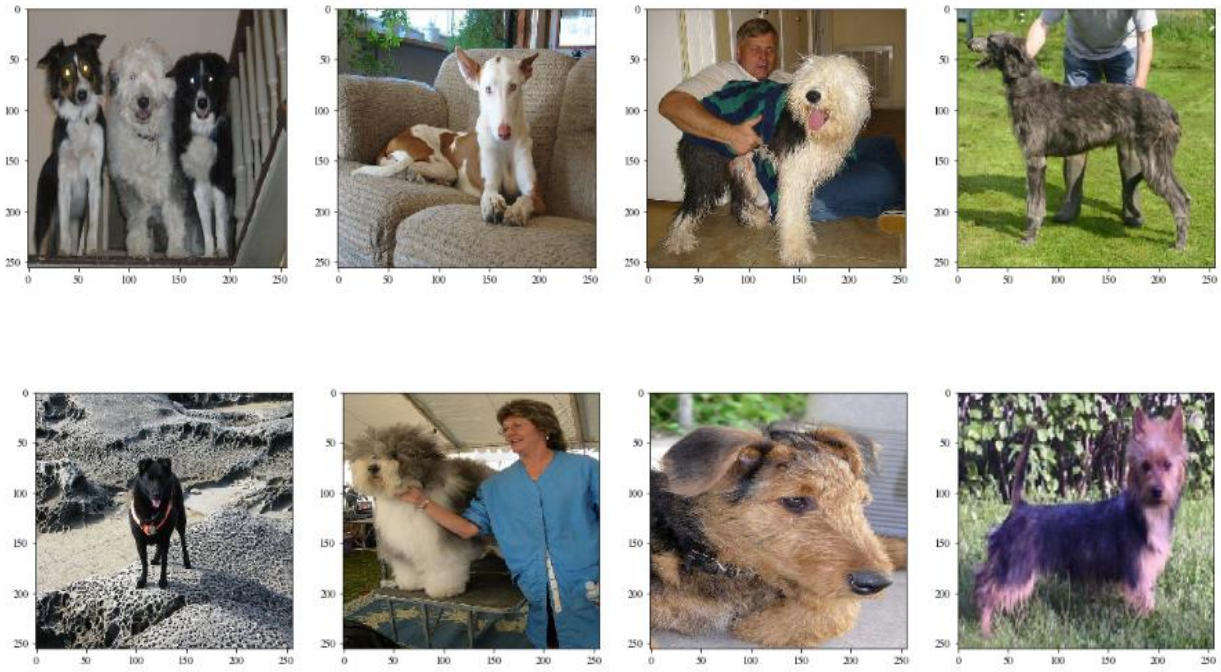

Figure 1 Some examples of the 20,580 dog images in the dataset.

4. Steps taken to Prepare the Data

The data is structured into directories of dog images. The images are in sub folders based on their breeds. As a first step, I wrote a python function to change the directory structure into

training, validation and test and return Scikit-learns data generator modules for each using parameters such as the number of dogs to include and the ratio of the split between training, test and validation. Having a function to quickly produce an image dataset with any number of breeds split into desired proportions for training and validation proved to be very helpful in trying out different combinations.

An initial exploration of the images, made it clear that the data is very noisy. In a lot of the images, the dogs are occupying a small part of the overall image which contains a lot of other objects in the environment. In some of the pictures, there are even objects in front of the dog, partially covering the dogs. For example even in a simple picture like the one shown in Figure 2, more than 50% of the image is occupied by the environment which gives us no information about the dogs breed. I used an open source python module, OpenCV to process the image. The module uses a feature selection function to zoom in on more informative parts of the image. This cuts out the parts that reveal no information about the dog breeds and makes it much easier to learn about the specific dog breed traits with less noise in the training set. I found that applying a simple open source filter like this increased the accuracy results of the models significantly.

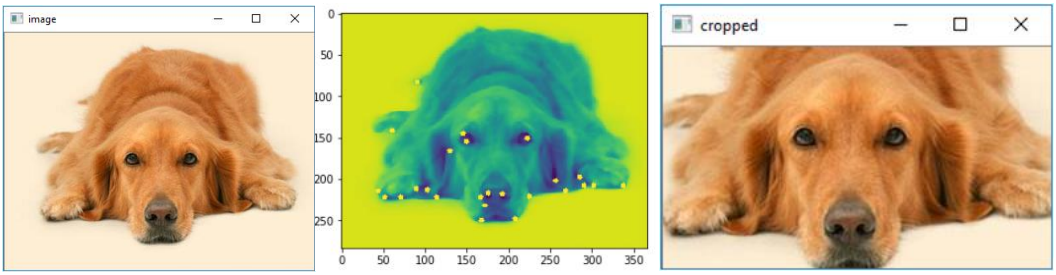

Figure 2 In the left image, the dog is occupying around 50% of image area. The middle image shows how the key features selected by the feature selection function from OpenCV. The image is then cut using the co-ordinates from the feature selection function which results in the dog occupying about 75% of the image on the right.

I also rescaled the images to different sizes for the different models to make the data more manageable. For some of the simple models I rescaled all the images to 64 X 64 and it still performed very well in spite of losing a lot of the details. For the larger more complicated models, I rescaled the images to 256 X 256 or 128 X 128 to achieve better accuracy scores.

5. Models and Approaches

5.1 Convolutional Neural Network:

I used a simple Convolutional Neural Network with 17 hidden layers. It had 6 convolutional layers with 3 X 3 kernels. I found 3 X 3 kernels provided better results than

2 X 2 kernels. Each convolutional layer had varying number of filters ranging from 512 to 64. I added several Dropout layers in between to avoid over fitting. I also added several max pooling layers to narrow down the network as I increased the number of filters for the layers. I used two fully connected layers at the end to output a categorical classification result using SoftMax activation function. I wanted to keep the design simple, so I could easily see the effect of changing a parameter which would then give more insight into how to get the best results out of it. Through trial and error, I tuned other parameters such as dropout rate, number of filters, kernel size and activation functions.

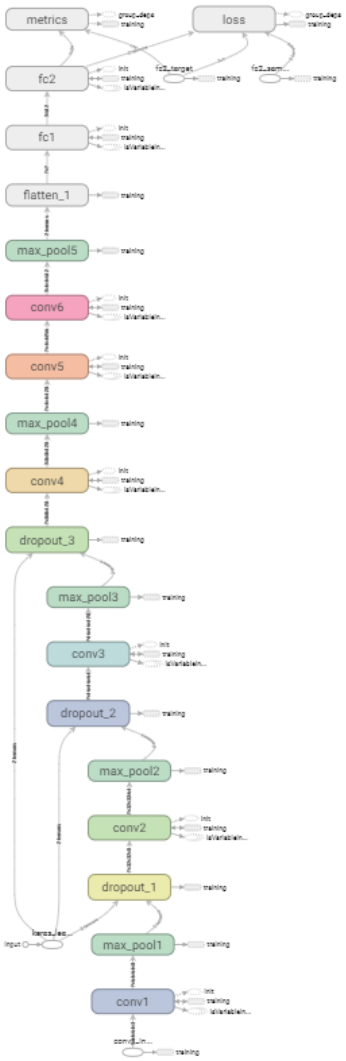

Figure 3 TensorBoard representation of the Convolutional Neural Network used

My eventual goal was to move to a Transfer Learning based model and therefore I wanted to keep the CNN Model as simple as possible. I intended to add components of this model to the end of a pre-trained model provided by Keras library. But this very

simple model performed very well on recognizing dog breeds if the number of classes(breeds) where not too high. The results section describes the accuracy rates achieved in more detail.

5.2 Transfer Learning

To improve scores with a high number of breeds, I tried several pretrained models such as ResNet40, Xception and InceptionResNetV2 provided by Keras. After some initial testing, it was obvious that Xception trained on the imagenet dataset was getting the best test results. So, I focused on the Xception model as my main pre-trained model to build my model.

I removed the top layer for the Xception pre-trained model and added a Convolutional Layer, a max pooling layer and a Batch Normalization layer before outputting the results through two fully connected layer.

The Xception model consists of 154 hidden layers. I also opened the weights for the top 63 layers of the Xception model to be trainable during the training process. Figure 4 shows the architecture of the pre-trained model used based on transfer learning.

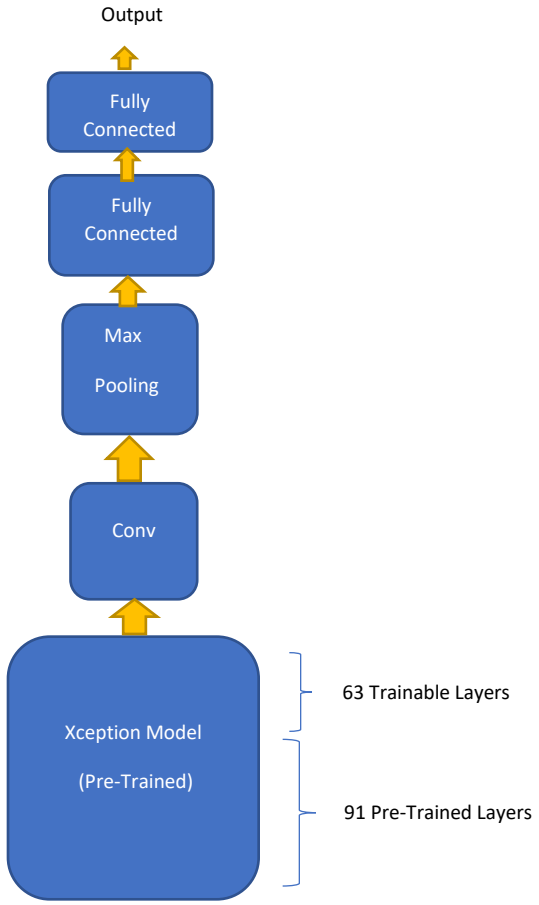

Figure 4 High Level Architecture of the Transfer Learning Model used

6. Results

6.1 Convolutional Neural Network with 8, 16 and 32 dog breeds

Using a Convolutional Neural Network model which was described in Section 5.1, the accuracy score for classifying between 8 dog breeds was almost 100%. The Results below demonstrate that.

Test Accuracy with 8 dog Breeds: 99.6%

Classification Report

	precision	recall	f1-score	support
Airedale	1.00	1.00	1.00	31
Australian_terrier	1.00	1.00	1.00	30
Border_terrier	1.00	1.00	1.00	27
Doberman	1.00	1.00	1.00	23
Ibizan_hound	1.00	0.97	0.98	29
Old_English_sheepdog	1.00	1.00	1.00	26
Scottish_deerhound	0.97	1.00	0.99	36
schipperke	1.00	1.00	1.00	24
avg / total	1.00	1.00	1.00	226

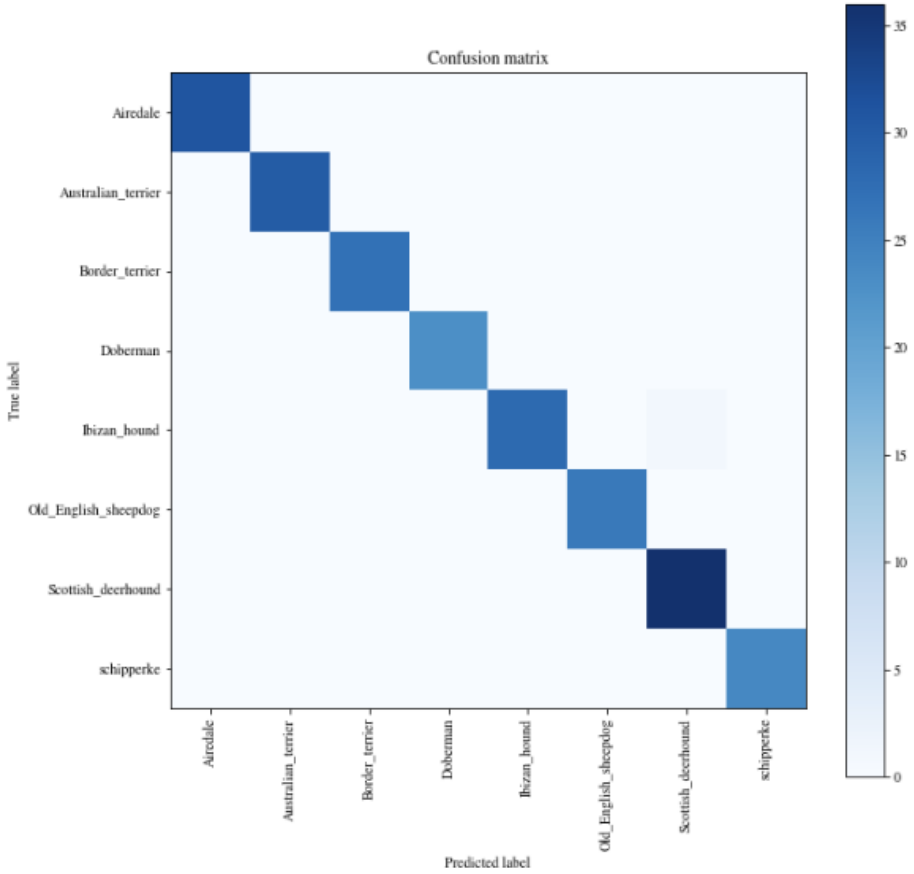

Figure: 5 Results showing near perfect precision, recall and accuracy for 8 dog breeds. The confusion matrix also shows nice deep line along the diagonal.

With such great accuracy scores, I pushed this model to tackle a more difficult task of classifying between 16 dog breeds. Without any change to the model parameters, the accuracy scores on the test set was again surprisingly good with average accuracy levels at over 90%.

Test Accuracy with 16 dog Breeds: 91.17%

Classification Report

	precision	recall	f1-score	support
Airedale	1.00	0.94	0.97	31
Australian_terrier	0.93	0.93	0.93	30
Border_terrier	0.82	1.00	0.90	27
Doberman	0.92	0.96	0.94	23
Ibizan_hound	1.00	0.93	0.96	29
Irish_setter	0.96	0.96	0.96	24
Lhasa	0.92	0.79	0.85	29
Old_English_sheepdog	1.00	0.58	0.73	26
Scottish_deerhound	0.88	0.97	0.92	36
Shih-Tzu	0.78	0.97	0.86	33
beagle	0.96	0.90	0.93	30
borzoi	0.95	0.88	0.91	24
briard	0.96	1.00	0.98	24
miniature_poodle	0.79	0.96	0.87	24
schipperke	1.00	0.92	0.96	24
toy_terrier	0.96	0.96	0.96	27
avg / total	0.93	0.92	0.91	441

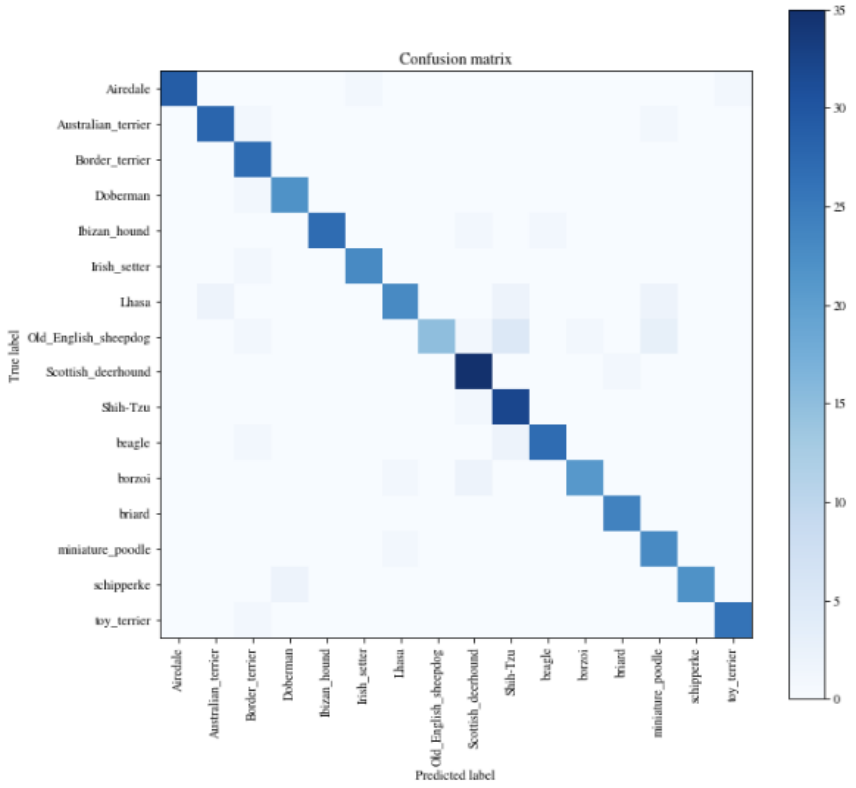

Figure: 6 Classification Report for classifying 16 dog breeds using CNN

Then I doubled the number of dog breeds to 32 making it a much more difficult problem to handle for this relatively simple CNN model. This time the accuracy scores dropped well below 90% and the best accuracy I was able to achieve was about 46.12%. Since the results in Figure 7 fell far below 90% and I have only added 32 dog breeds in the mix, I decided to stop tuning this CNN model anymore and moved to a pre-trained model, that I could tune to my dataset to achieve better scores .The results of the transfer learning approach is described in the next section.

Test Accuracy with 32 dog Breeds: 46.12 %

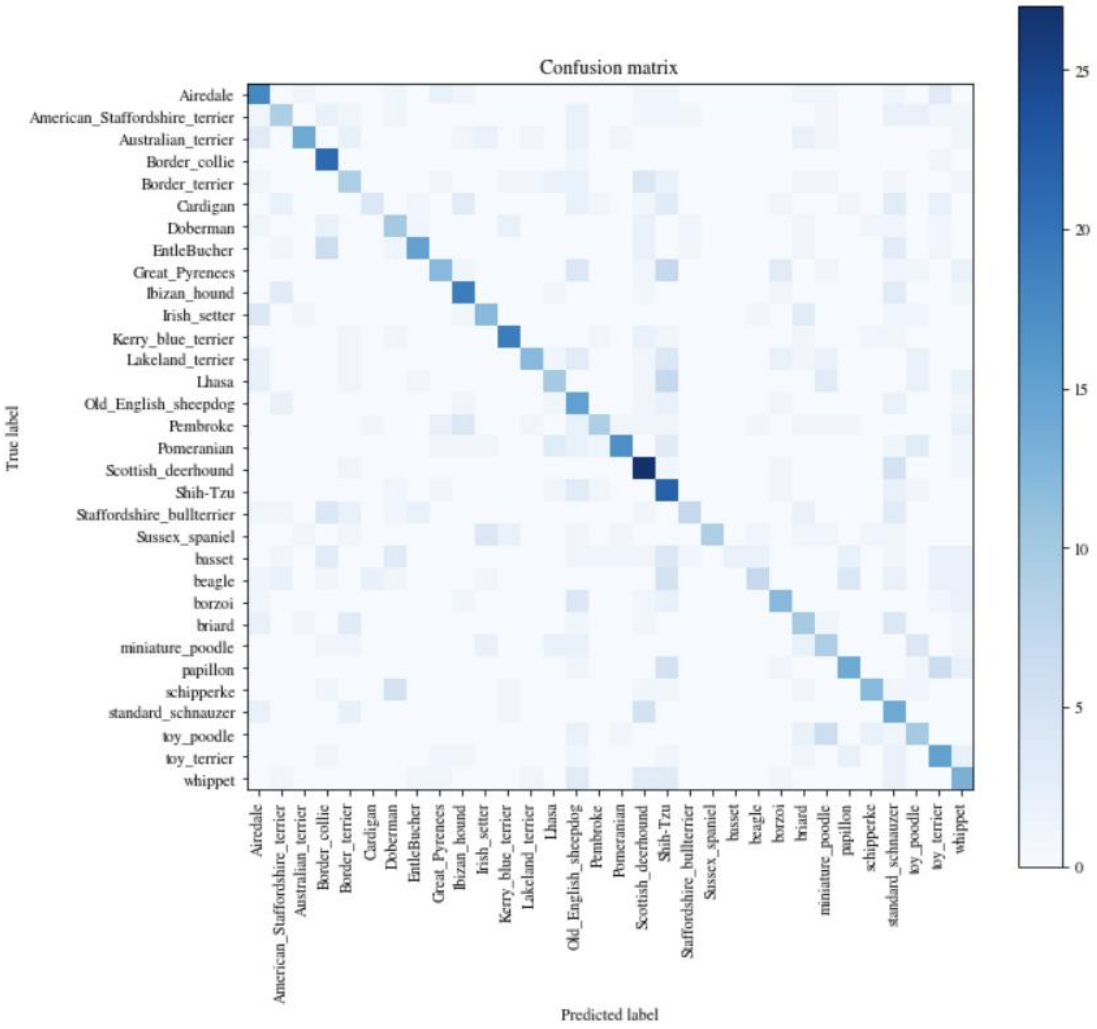

Figure: 7 Confusion Matrix showing results of classification between 32 dog breeds

6.2 Transfer Learning

I used a transfer learning model as described in Section 5.2 and was able to improve the results of the simple CNN model significantly on the problem of detecting between 32 dog breeds.

Test Accuracy with 32 dog Breeds: 97.12 %

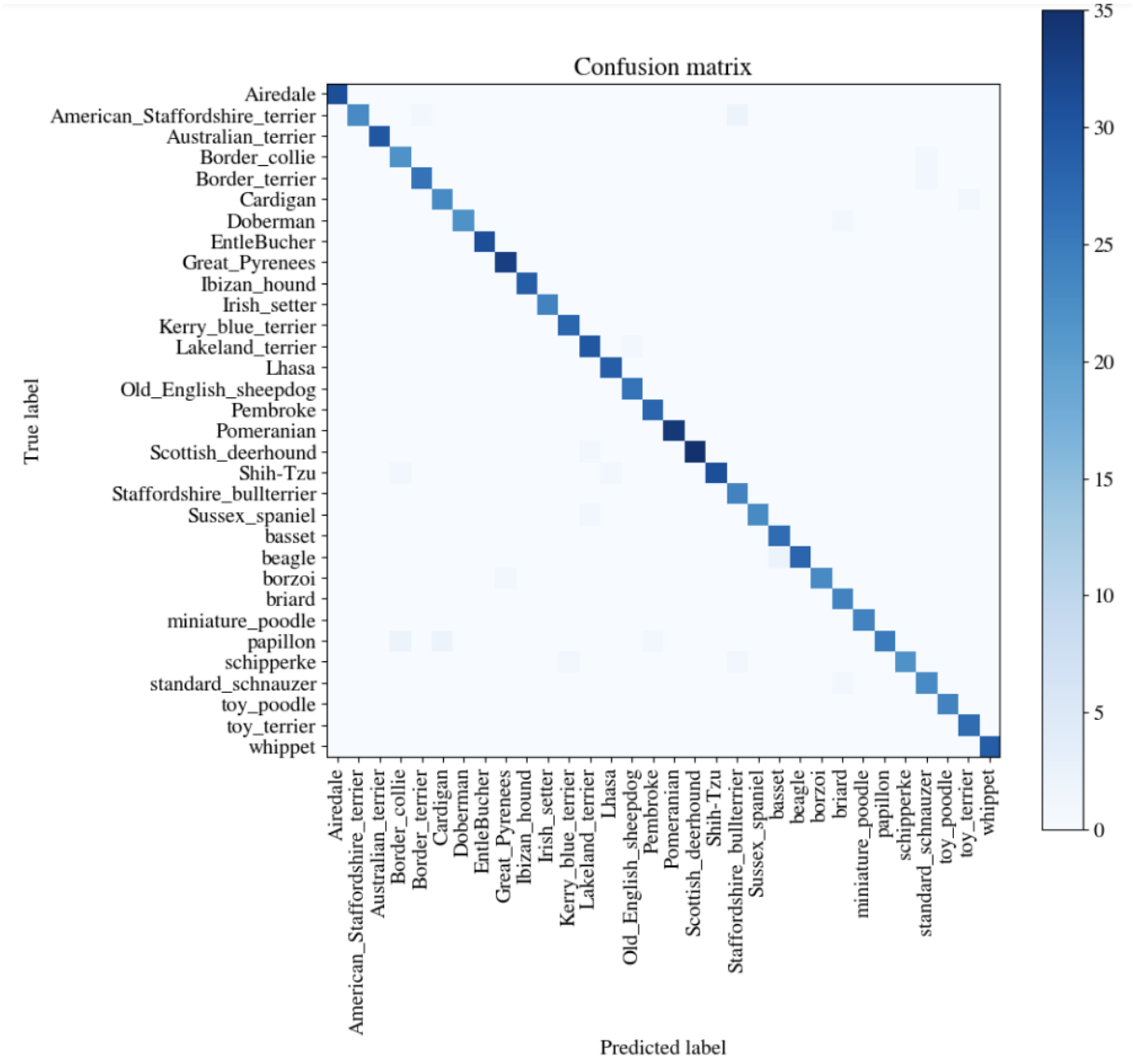

Figure: 8 Confusion Matrix showing results of classification between 32 dog breeds using Xception pre-trained model as a base.

With the partially pre-trained model performing so well on the classifying between 32 breeds, I decided to move forward rapidly and try out a model with 115 dog breeds. I increased the number of epochs to train on and also increased the number

of layers in my convolutional layers to be able to capture more distinct features in order to differentiate between large numbers of dog breeds. It achieved a descent accuracy score of 58.75% with some room for improvement.

Test Accuracy with 115 dog Breeds: 58.75 %

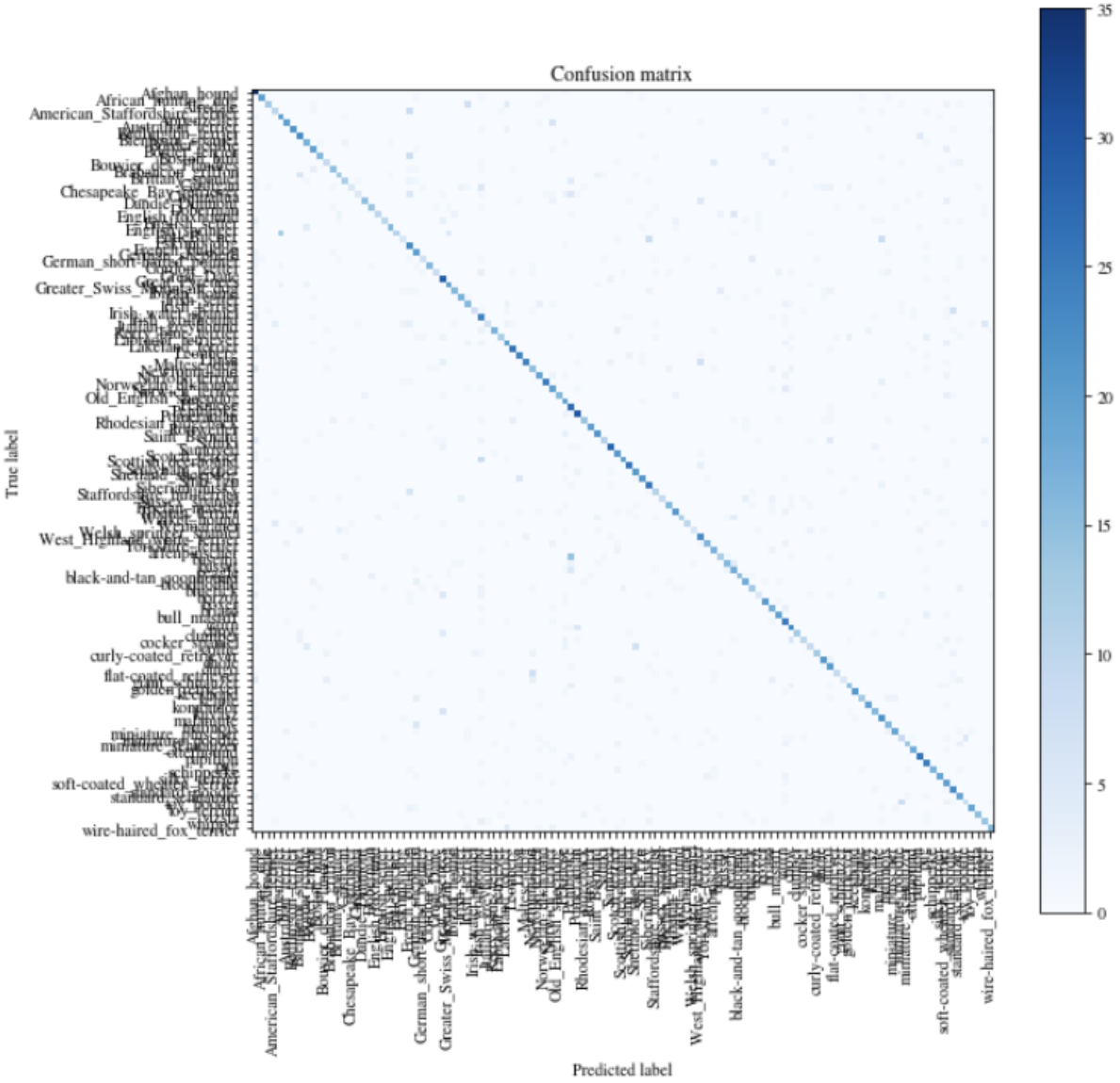

Figure: 9 Confusion Matrix showing results of classification between 115 dog breeds using a pre-trained model as a base.

In the confusion matrix above, we are able to see a nice diagonal deep strip showing a lot of the dogs were being classified correctly in the test dataset. But I believed the

58 % accuracy rate could definitely be improved upon. A look at the training vs validation scores during training showed the model was beginning to overfit as shown in Figure 10 below.

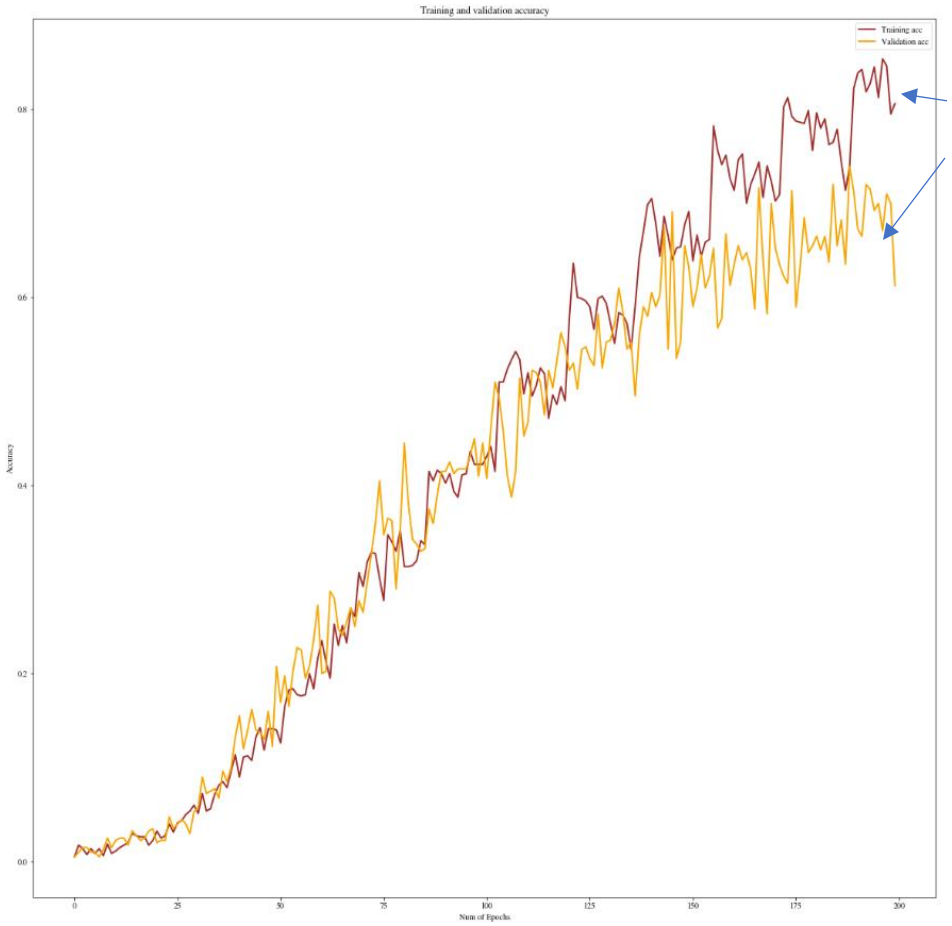

Figure: 10 Training vs Validation scores during training showing signs of overfitting

So I add a batch normalization layer and a drop out layer to try to reduce the overfit and decreased the number of layers in the pre-trained model that I was allowing the new data to train. Having done these I also increased the number of epochs I was training with because I expect the model to start to over fit at a much later time now that I have added the measures to reduce the level of overfitting.

With the new setup I was able to achieve much better results with accuracy rates of around 78% percent. Figure 11 shows the results of running the model on test data for 115 breeds of dogs.

Test Accuracy with 115 dog Breeds: 77.87 %

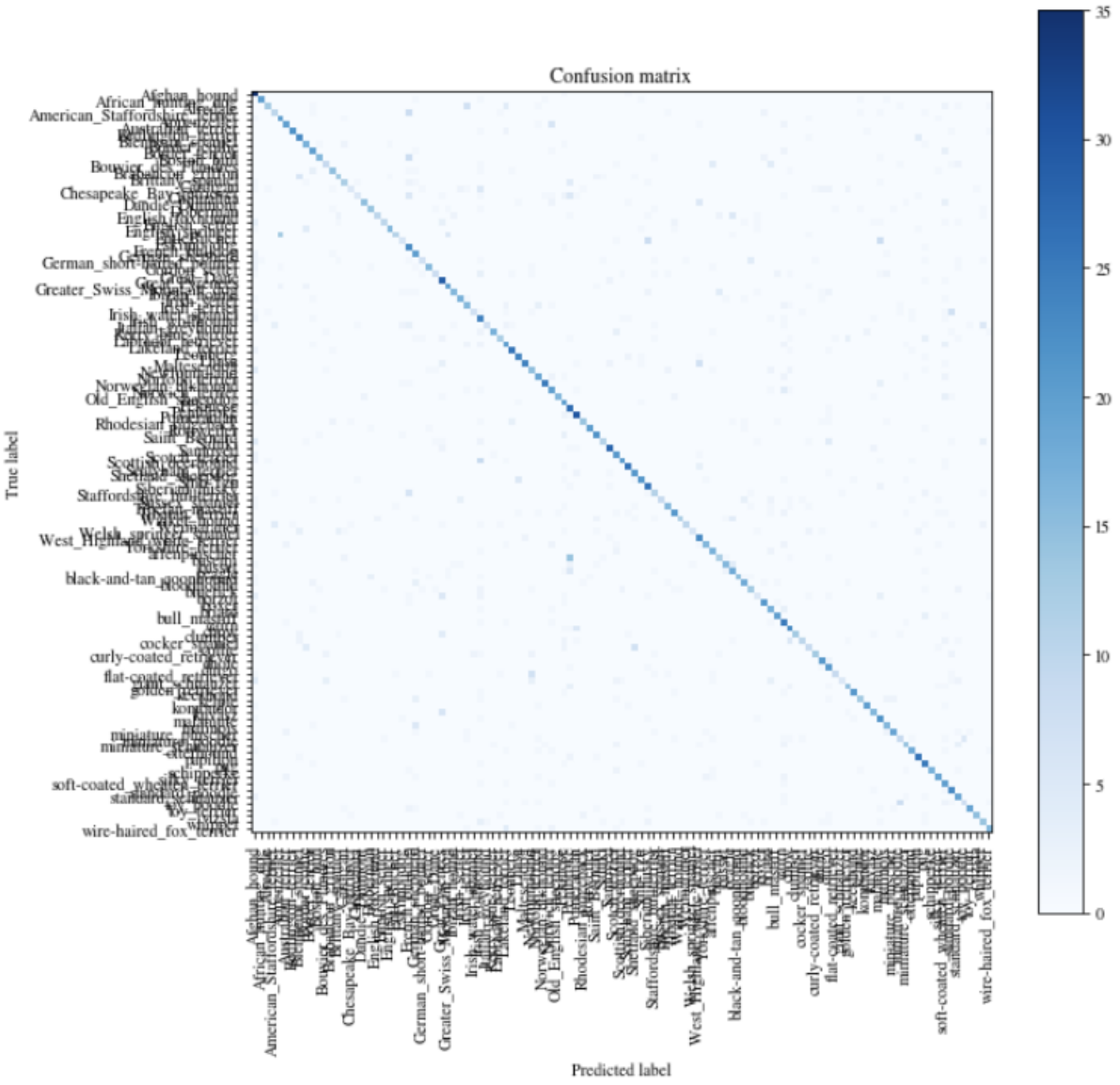

Figure: 11 Confusion Matrix showing results of classification between 115 dog breeds using a pre-trained model as a base and overfit reduced by batch normalization.

Even though the above accuracy rate of 78% did not quite reach the initial goal of 90%, I believe if I had a little more time to tune the parameters, the model would perform at close to 90%. One difficulty with such large networks and huge datasets was that training the model took several hours at a time which made it difficult to iterate through the tuning process in the short time at hand.

7 Conclusion

This problem is particularly difficult because we are trying to differentiate between pictures that are very similar in nature. All the pictures have dogs in them which in general have very similar features with very subtle difference between the breeds. Having said that, I was very surprised by how accurately even a very simple Convolutional Neural Network was able to detect different breeds with very little training. It shows that convolutional Neural Networks are very good at detecting patterns in images.

But for the more difficult problem of detecting all dog breeds which would number in the hundreds we need much larger models with more layers and parameters. We would also need a lot of training data to train such a model. In those cases I was able to demonstrate that transfer learning is probably the best way to solve it where the model is already pre-trained and we don't have to retrain all the parameters ourselves. And the improvement on the results with transfer learning demonstrates that.