PROJECT REPORT

DOG BREED CLASSIFICATION (Image Classification)

Dataset :-

We are using the Stanford Dogs dataset. The Stanford Dogs dataset contains images of 120 breeds of dogs from around the world. This dataset has been built using images and annotations from ImageNet for the task of fine-grained image categorization. Contents of this dataset.

- **Number of categories:** 120
- **Number of images:** 20,580
- **Annotations:** Class labels

Data dictionary:

1) Image of the dog : An RGB image of a dog in various environments and settings.
2) Breed of the dog : The breed label corresponding to the dog in the image.(120 Breeds)

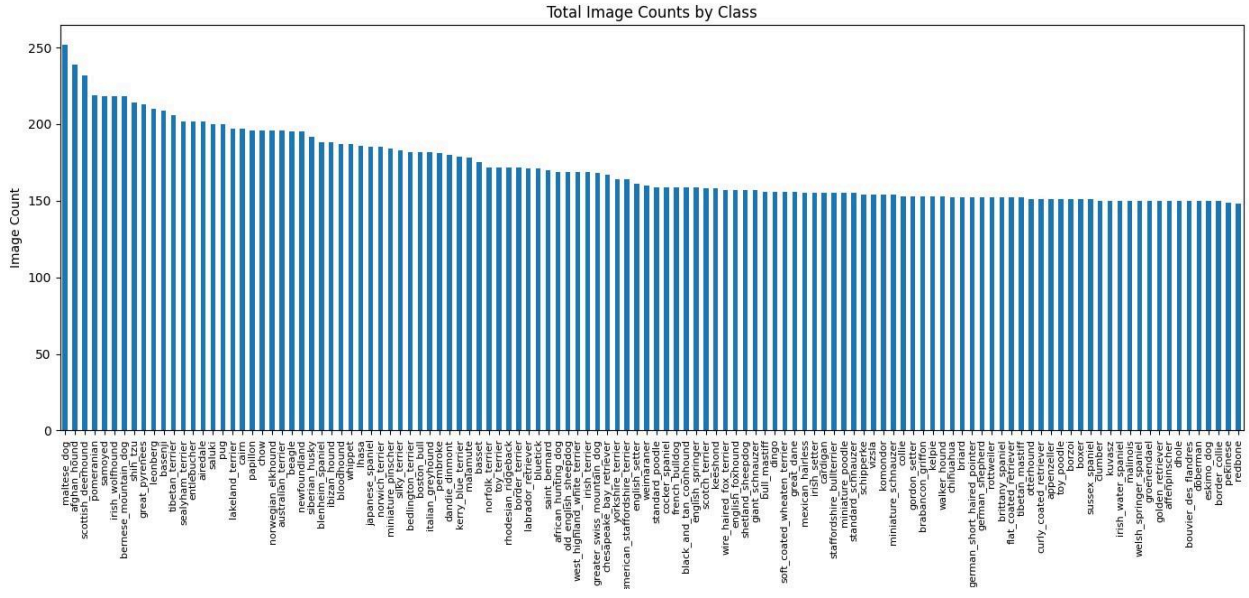
Description of some files inside the zip file :

1) train_list.mat - a list of all the training set images.
2) test_list.mat - a list of all the testing set images.
3) Images/ - a folder containing all of the images of dogs.
4) Annotation/ - a folder containing all of the annotations for each image.
5) file_list.mat - a list of all the files (training and test list combined).

Transfer Learning with EfficientNetV2 for Dog Breed Classification :

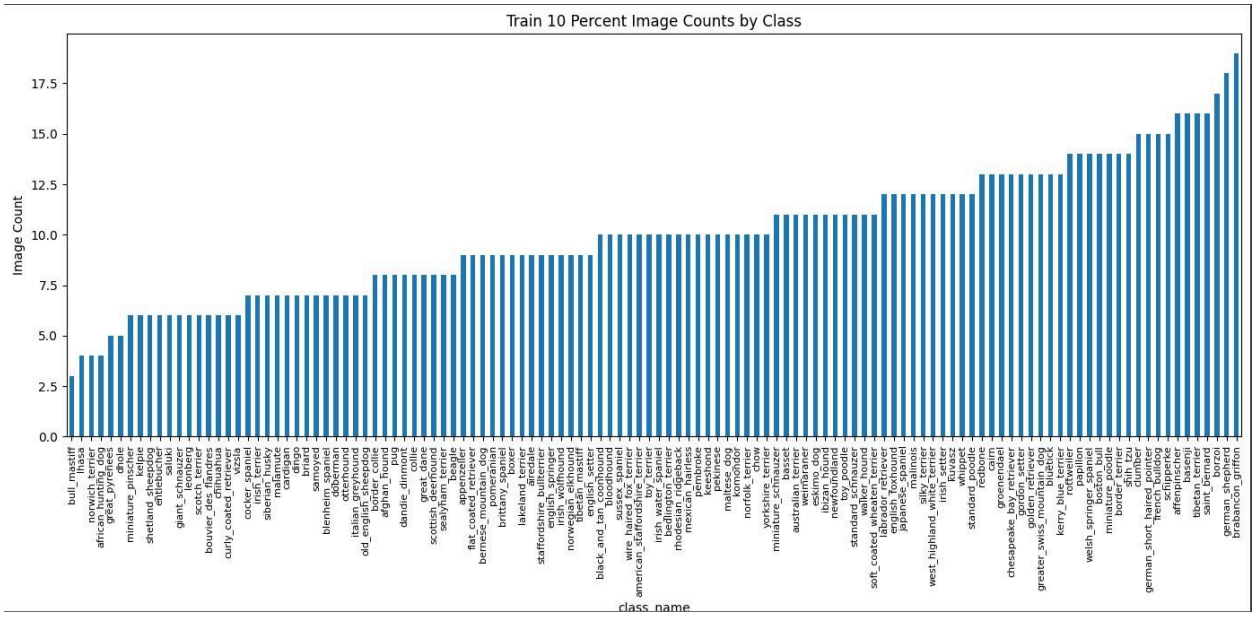
In this project, we leverage **transfer learning** by fine-tuning the pre-trained **EfficientNetV2B0** model from TensorFlow Keras. Trained on **ImageNet1k** (1M+ images, 1000 classes), EfficientNetV2B0 has a strong baseline understanding of visual patterns. We will adapt this model to classify **120 dog breeds**, leveraging its generalized feature extraction to enhance training efficiency and accuracy on our dataset. Fine-tuning involves modifying the model's final layers while keeping the earlier

layers frozen, allowing the model to transfer learned visual patterns effectively to our specific dog image classification task.



We can see that our classes are quite balanced. Each breed of dog has ~150 or more images.

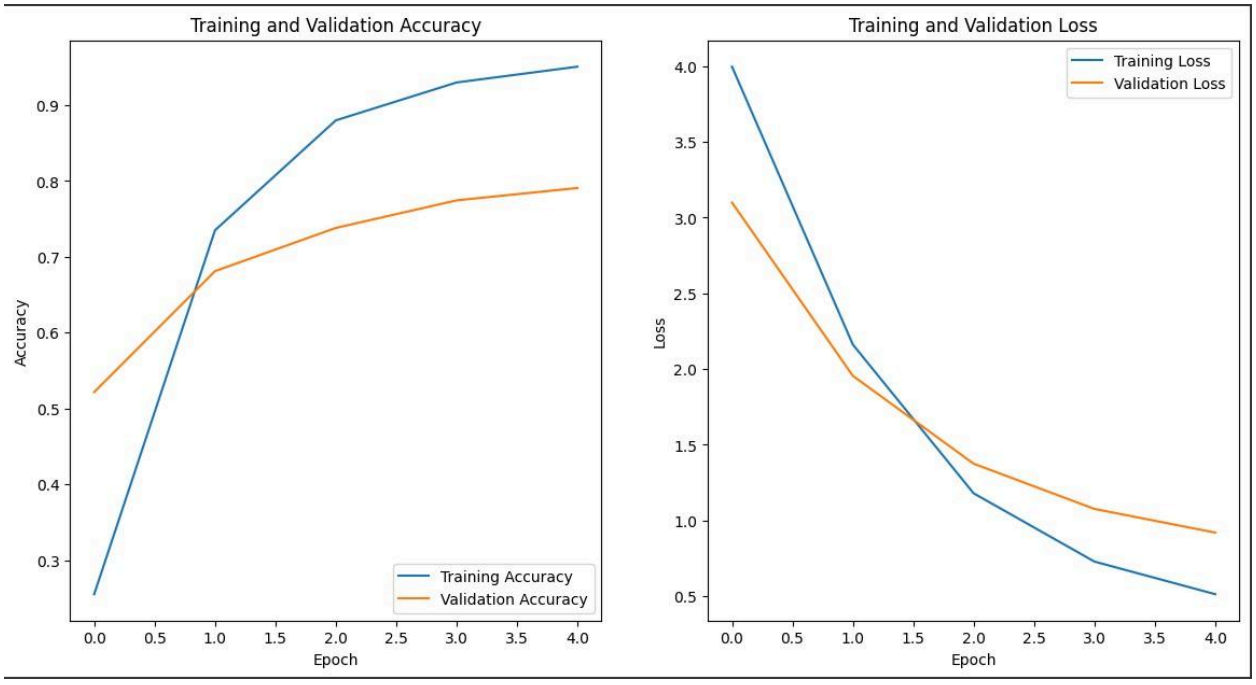
We will first train our model on 10% of our dataset and then will train it on the whole training dataset because more data means longer computation times.



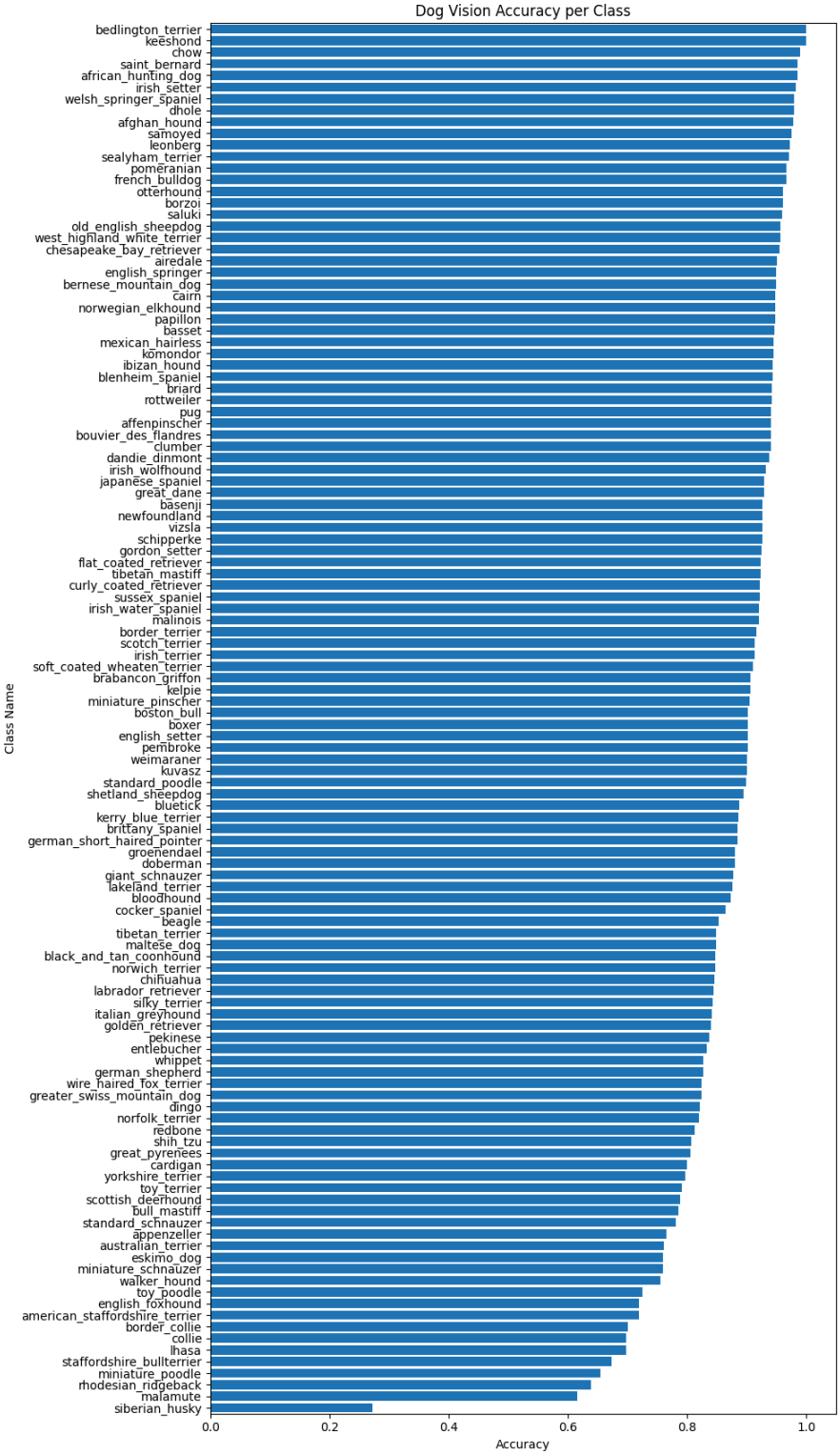
Now since we are using Tensorflow/Keras, we will use a pretrained model from tf.keras.applications.

Our base model has 273 layers.

Plotting Accuracy and Loss :-



Finding Accuracy per class :-



Confusion Matrix : -