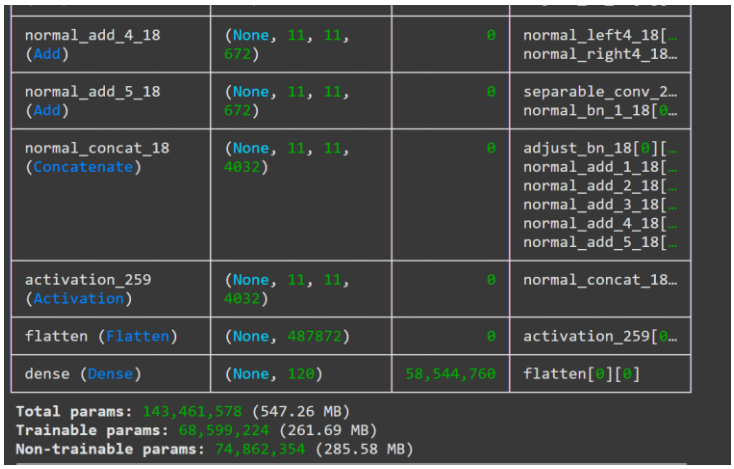
Model Optimization and Tuning Phase

Date	17th June 2025
Team ID	SWTID1749820017
Project Name	Dog Breed Identification using Transfer Learning
Maximum Marks	10 Marks

Motivation for optimization

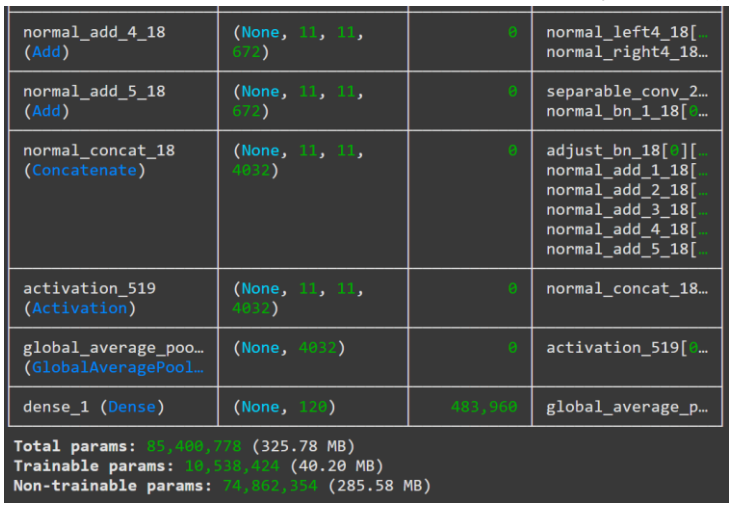
When we first loaded the NASNetLarge model with the flatten() layer we realized that the model was extremely huge even before training had been done(547.26MB). Shown below:



normal_add_4_18 (Add)	(None, 11, 11, 672)	0	normal_left4_18[… normal_right4_18…
normal_add_5_18 (Add)	(None, 11, 11, 672)	0	separable_conv_2… normal_bn_1_18[0…
normal_concat_18 (Concatenate)	(None, 11, 11, 4032)	0	adjust_bn_18[0][… normal_add_1_18[… normal_add_2_18[… normal_add_3_18[… normal_add_4_18[… normal_add_5_18[…
activation_259 (Activation)	(None, 11, 11, 4032)	0	normal_concat_18…
flatten (Flatten)	(None, 487872)	0	activation_259[0…
dense (Dense)	(None, 120)	58,544,760	flatten[0][0]

Total params: 143,461,578 (547.26 MB)
Trainable params: 68,599,224 (261.69 MB)
Non-trainable params: 74,862,354 (285.58 MB)

So we decided to replace the flatten() layer with a GlobalAveragePooling2D() layer which decreased the size of the model by 40% (325.78MB). Shown below

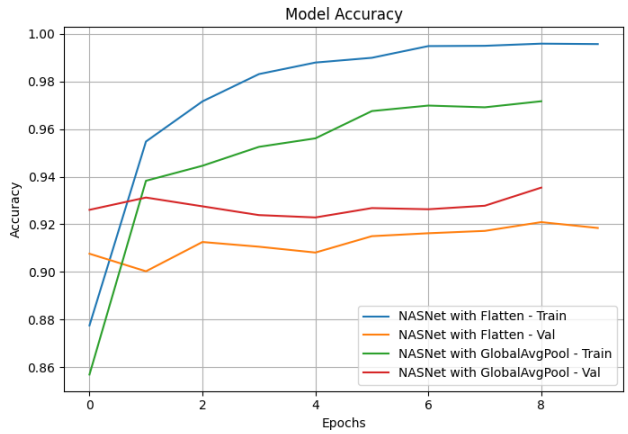


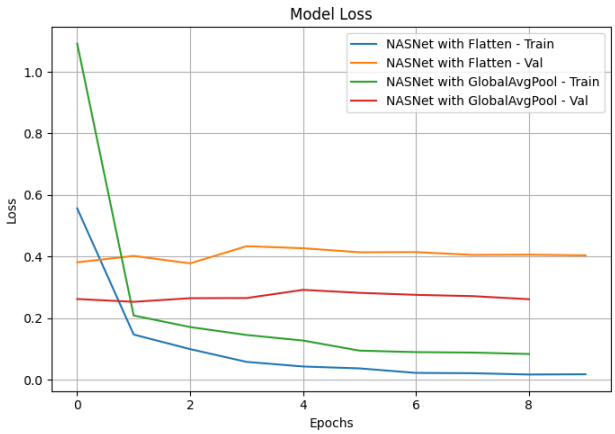
normal_add_4_18 (Add)	(None, 11, 11, 672)	0	normal_left4_18[… normal_right4_18…
normal_add_5_18 (Add)	(None, 11, 11, 672)	0	separable_conv_2… normal_bn_1_18[0…
normal_concat_18 (Concatenate)	(None, 11, 11, 4032)	0	adjust_bn_18[0][… normal_add_1_18[… normal_add_2_18[… normal_add_3_18[… normal_add_4_18[… normal_add_5_18[…
activation_519 (Activation)	(None, 11, 11, 4032)	0	normal_concat_18…
global_average_poo… (GlobalAveragePool…	(None, 4032)	0	activation_519[0…
dense_1 (Dense)	(None, 120)	483,960	global_average_p…

Total params: 85,400,778 (325.78 MB)
Trainable params: 10,538,424 (40.20 MB)
Non-trainable params: 74,862,354 (285.58 MB)

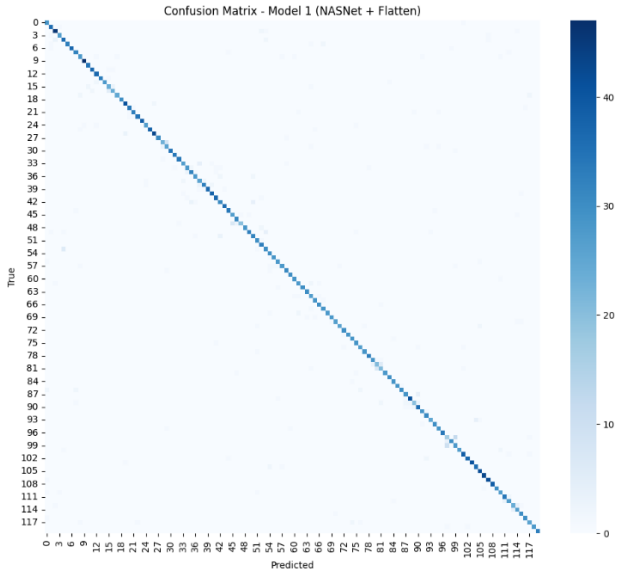
Experimental Comparisons:

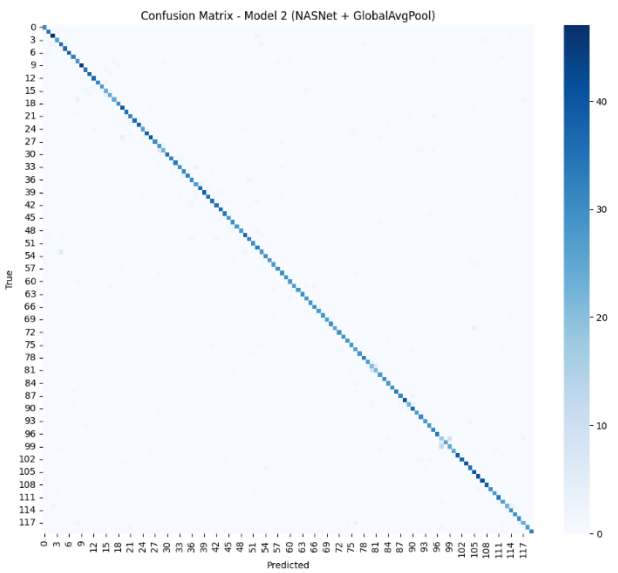
a) Accuracy and Loss





b) Confusion Matrix





c) Validation accuracy

```
val_loss_1, val_acc_1 = model1.evaluate(val_generator)
print(f"Model 1 (NASNet + Flatten) - Val Loss: {val_loss_1:.4f}, Val Accuracy: {val_acc_1:.4f}")

val_loss_2, val_acc_2 = model2.evaluate(val_generator)
print(f"Model 2 (NASNet + GlobalAvgPool) - Val Loss: {val_loss_2:.4f}, Val Accuracy: {val_acc_2:.4f}")
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: User
  self._warn_if_super_not_called()
32/32 ━━━━━━━━━━━━━━━━━━━━ 1891s 59s/step - accuracy: 0.9188 - loss: 0.4051
Model 1 (NASNet + Flatten) - Val Loss: 0.4173, Val Accuracy: 0.9158
32/32 ━━━━━━━━━━━━━━━━━━━━ 130s 3s/step - accuracy: 0.9358 - loss: 0.2428
Model 2 (NASNet + GlobalAvgPool) - Val Loss: 0.2630, Val Accuracy: 0.9278
```

d) Precision, Recall and F1-Score

- Model 1: NASNetLarge with Flatten() layer

```
Weighted F1 Score: 0.9182
Weighted Precision: 0.9206
Weighted Recall: 0.9182
```

- Model 2: NASNetLarge with GlobalAveragePooling2D() layer

```
Weighted F1 Score: 0.9297
Weighted Precision: 0.9324
Weighted Recall: 0.9295
```

e) Model size:

```
from pathlib import Path
path1 = Path('/content/drive/MyDrive/Dog_Classifier/Models/nasnet_flatten_best_model.h5')
path2 = Path('/content/drive/MyDrive/Dog_Classifier/Models/nasnet_globalavgpool_best_model.h5')
size1 = path1.stat().st_size / (1024 * 1024)
size2 = path2.stat().st_size / (1024 * 1024)
print(f"NASNetLarge with Flatten() layer model size: {size1:.2f} MB")
print(f"NASNetLarge with GlobalAveragePooling2D() layer model size: {size2:.2f} MB")
```

```
NASNetLarge with Flatten() layer model size: 1075.35 MB
NASNetLarge with GlobalAveragePooling2D() layer model size: 410.90 MB
```

Conclusion:

From the above experimental results we can conclude that the NASNetLarge with GlobalAveragePooling2D() layer model is not only smaller but also provided high validation accuracy, recall score, precision score and F1-socre than the NASNetLarge with Flatten() layer model.