Sam Heidler & Kashir Khan
CS 4372 - Data Science
24 October 2024

Model Output

```
Loading and preparing data...
Total images in labels.csv: 10222
Valid images found: 10222
Number of unique breeds: 120
Training samples: 8177
Validation samples: 2045
Found 8177 validated image filenames belonging to 120 classes.
Found 2045 validated image filenames belonging to 120 classes.

Verifying generators:
Train generator classes: 120
Steps per epoch: 256
Validation steps: 64
Creating model...

Model summary:
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_6 (InputLayer)         [(None, 300, 300, 3)]     0

efficientnetb3 (Functional)  (None, 10, 10, 1536)      10783535
)

global_average_pooling2d_3   (None, 1536)              0
(GlobalAveragePooling2D)

batch_normalization_2 (Bat   (None, 1536)              6144
chNormalization)

dropout_4 (Dropout)          (None, 1536)              0
```

dense_6 (Dense) (None, 512) 786944

batch_normalization_3 (Bat (None, 512) 2048
chNormalization)

dropout_5 (Dropout) (None, 512) 0

dense_7 (Dense) (None, 120) 61560

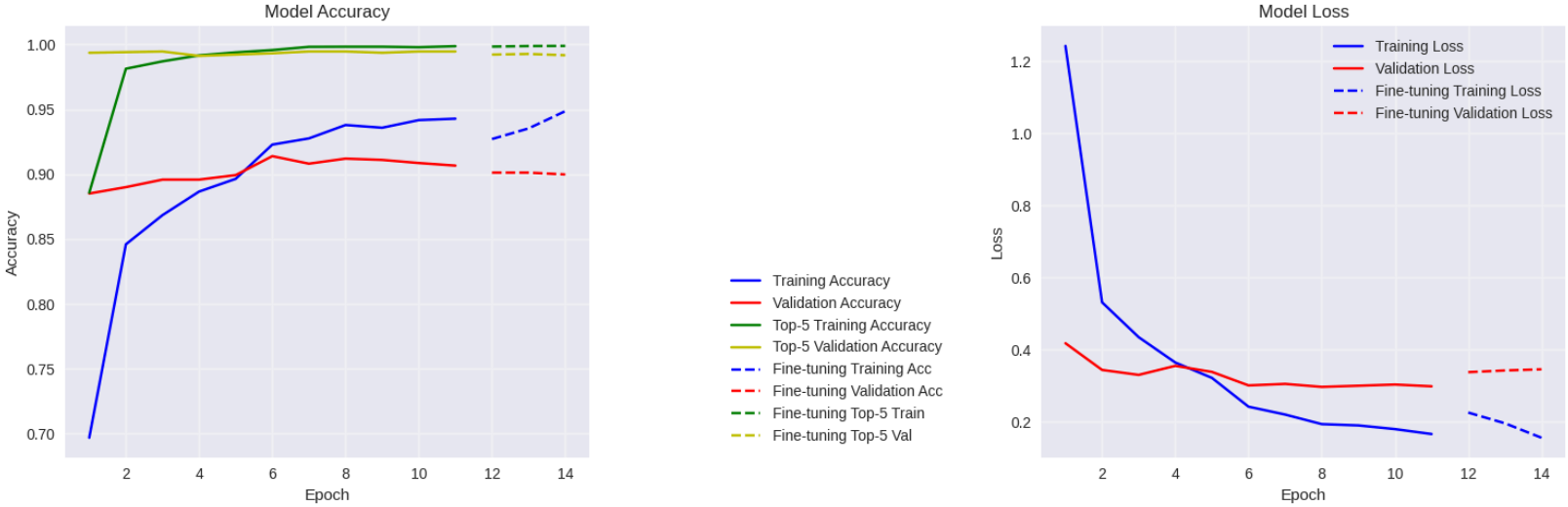
===
Total params: 11640231 (44.40 MB)
Trainable params: 852600 (3.25 MB)
Non-trainable params: 10787631 (41.15 MB)
_____
Starting initial training...
Epoch 1/13
256/256 [==============================] - 415s 2s/step - loss: 1.2423 - accuracy: 0.6968 - top_5_accuracy: 0.8857 - val_loss: 0.4176 - val_accuracy: 0.8851 - val_top_5_accuracy: 0.9936 - lr: 0.0010
Epoch 2/13
256/256 [==============================] - 395s 2s/step - loss: 0.5311 - accuracy: 0.8459 - top_5_accuracy: 0.9813 - val_loss: 0.3436 - val_accuracy: 0.8900 - val_top_5_accuracy: 0.9941 - lr: 0.0010
Epoch 3/13
256/256 [==============================] - 402s 2s/step - loss: 0.4344 - accuracy: 0.8684 - top_5_accuracy: 0.9870 - val_loss: 0.3298 - val_accuracy: 0.8958 - val_top_5_accuracy: 0.9946 - lr: 0.0010
Epoch 4/13
256/256 [==============================] - 399s 2s/step - loss: 0.3641 - accuracy: 0.8866 - top_5_accuracy: 0.9916 - val_loss: 0.3549 - val_accuracy: 0.8958 - val_top_5_accuracy: 0.9912 - lr: 0.0010
Epoch 5/13
256/256 [==============================] - 407s 2s/step - loss: 0.3215 - accuracy: 0.8964 - top_5_accuracy: 0.9939 - val_loss: 0.3381 - val_accuracy: 0.8993 - val_top_5_accuracy: 0.9922 - lr: 0.0010
Epoch 6/13
256/256 [==============================] - 398s 2s/step - loss: 0.2416 - accuracy: 0.9228 - top_5_accuracy: 0.9957 - val_loss: 0.3006 - val_accuracy: 0.9139 - val_top_5_accuracy: 0.9932 - lr: 2.0000e-04
Epoch 7/13
256/256 [==============================] - 393s 2s/step - loss: 0.2198 - accuracy: 0.9276 - top_5_accuracy: 0.9982 - val_loss: 0.3048 - val_accuracy: 0.9081 - val_top_5_accuracy: 0.9946 - lr: 2.0000e-04
Epoch 8/13

256/256 [==============================] - 403s 2s/step - loss: 0.1932 - accuracy: 0.9379 - top_5_accuracy: 0.9983 - val_loss: 0.2965 - val_accuracy: 0.9120 - val_top_5_accuracy: 0.9946 - lr: 2.0000e-04
Epoch 9/13
256/256 [==============================] - 402s 2s/step - loss: 0.1896 - accuracy: 0.9358 - top_5_accuracy: 0.9983 - val_loss: 0.2997 - val_accuracy: 0.9110 - val_top_5_accuracy: 0.9936 - lr: 2.0000e-04
Epoch 10/13
256/256 [==============================] - 396s 2s/step - loss: 0.1794 - accuracy: 0.9417 - top_5_accuracy: 0.9979 - val_loss: 0.3030 - val_accuracy: 0.9086 - val_top_5_accuracy: 0.9946 - lr: 2.0000e-04
Epoch 11/13
256/256 [==============================] - 393s 2s/step - loss: 0.1658 - accuracy: 0.9428 - top_5_accuracy: 0.9987 - val_loss: 0.2981 - val_accuracy: 0.9066 - val_top_5_accuracy: 0.9946 - lr: 4.0000e-05

Starting fine-tuning...
Epoch 12/23
256/256 [==============================] - 444s 2s/step - loss: 0.2253 - accuracy: 0.9271 - top_5_accuracy: 0.9984 - val_loss: 0.3374 - val_accuracy: 0.9012 - val_top_5_accuracy: 0.9922 - lr: 1.0000e-04
Epoch 13/23
256/256 [==============================] - 423s 2s/step - loss: 0.1956 - accuracy: 0.9353 - top_5_accuracy: 0.9988 - val_loss: 0.3421 - val_accuracy: 0.9012 - val_top_5_accuracy: 0.9927 - lr: 1.0000e-04
Epoch 14/23
256/256 [==============================] - 428s 2s/step - loss: 0.1555 - accuracy: 0.9486 - top_5_accuracy: 0.9989 - val_loss: 0.3453 - val_accuracy: 0.8998 - val_top_5_accuracy: 0.9917 - lr: 1.0000e-04
Epoch 15/23
180/256 [====================>.....] - ETA: 1:42 - loss: 0.1164 - accuracy: 0.9622 - top_5_accuracy: 0.9995

The notebooks runtime maxed out here, at 96% accuracy. This was simply due to Google Colabs' limit on free notebooks - it demanded we pay for 'Pro' to continue running models.

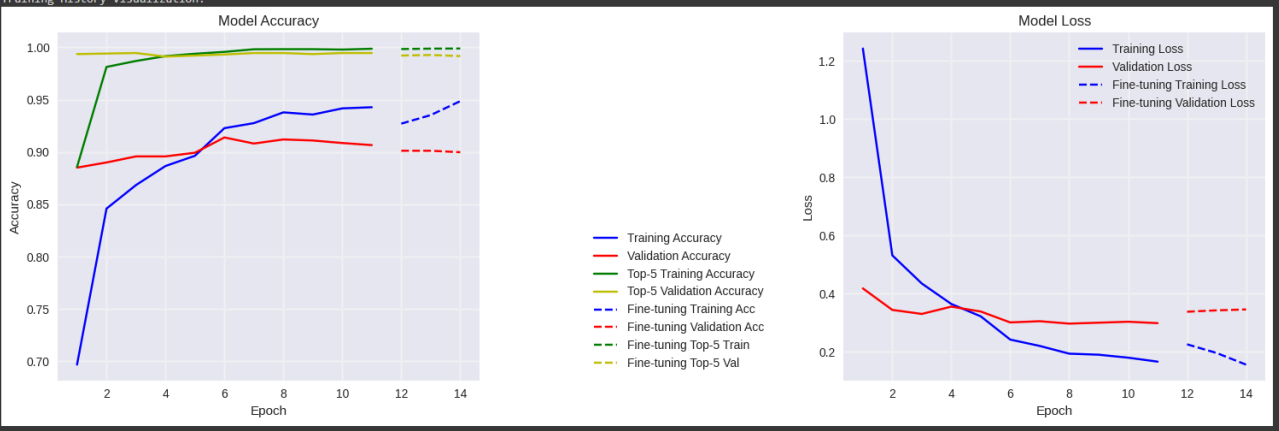
1. History Plots



Model Architecture and Parameters:

	Iteration	Base Model	Input Size	Layers Added	Initial Learning Rate	Batch Size	Total Parameters	Trainable Parameters	Best Training Accuracy	Best Validation Accuracy	Best Top-5 Training	Best Top-5 Validation
0	1	EfficientNetB3	300x300	- GlobalAveragePooling2D - BatchNorm - Dropout(0.3) - Dense(512) - BatchNorm - Dropout(0.4) - Dense(120)	0.001000	32	11.64M	852.6K	94.86%	91.39%	99.89%	99.46%

Training History Visualization:



Parameter Testing and Tuning Summary:

	Parameter Configuration	Epochs	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
0	Initial Training	1-5	89.64%	89.93%	0.3215	0.3381
1	Learning Rate Reduction (0.001 → 0.0002)	6-9	93.58%	91.10%	0.1896	0.2997
2	Further LR Reduction (0.0002 → 0.00004)	10-11	94.28%	90.66%	0.1658	0.2981
3	Fine-tuning (LR: 0.0001)	12-14	94.86%	89.98%	0.1555	0.3453

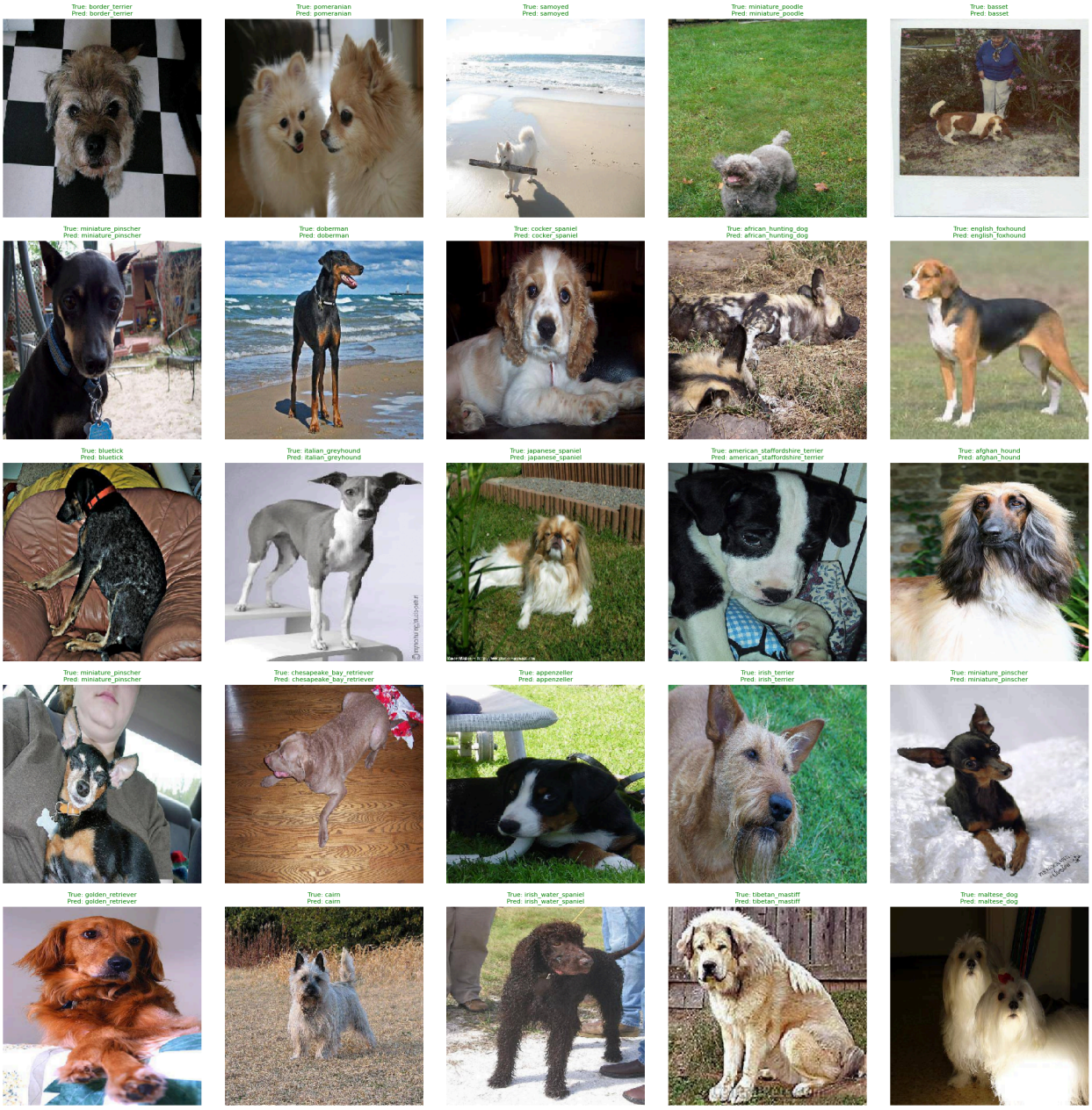
2. Test Data Points (25)

Loading saved model...
Loading and preparing test data...
Total images in labels.csv: 10222
Found 2045 validated image filenames belonging to 120 classes.

Example of 25 test data points from test dataset:
==========
1/1 [==========] - 4s 4s/step

Test Results Table:

Sample #	True Label	Predicted Label	Correct?
1	border_terrier	border_terrier	True
2	pomeranian	pomeranian	True
3	samoyed	samoyed	True
4	miniature_poodle	miniature_poodle	True
5	basset	basset	True
6	miniature_pinscher	miniature_pinscher	True
7	doberman	doberman	True
8	cocker_spaniel	cocker_spaniel	True
9	african_hunting_dog	african_hunting_dog	True
10	english_foxhound	english_foxhound	True
11	bluetick	bluetick	True
12	italian_greyhound	italian_greyhound	True
13	japanese_spaniel	japanese_spaniel	True
14	american_staffordshire_terrier	american_staffordshire_terrier	True
15	afghan_hound	afghan_hound	True
16	miniature_pinscher	miniature_pinscher	True
17	chesapeake_bay_retriever	chesapeake_bay_retriever	True
18	appenzeller	appenzeller	True
19	irish_terrier	irish_terrier	True
20	miniature_pinscher	miniature_pinscher	True
21	golden_retriever	golden_retriever	True
22	cairn	cairn	True
23	irish_water_spaniel	irish_water_spaniel	True
24	tibetan_mastiff	tibetan_mastiff	True
25	maltese_dog	maltese_dog	True


True: border_terrier
Pred: border_terrier
True: pomeranian
Pred: pomeranian
True: samoyed
Pred: samoyed
True: miniature_poodle
Pred: miniature_poodle
True: basset
Pred: basset
True: miniature_pinscher
Pred: miniature_pinscher
True: doberman
Pred: doberman
True: cocker_spaniel
Pred: cocker_spaniel
True: african_hunting_dog
Pred: african_hunting_dog
True: english_foxhound
Pred: english_foxhound
True: bluetick
Pred: bluetick
True: italian_greyhound
Pred: italian_greyhound
True: japanese_spaniel
Pred: japanese_spaniel
True: american_staffordshire_terrier
Pred: american_staffordshire_terrier
True: afghan_hound
Pred: afghan_hound
True: miniature_pinscher
Pred: miniature_pinscher
True: chesapeake_bay_retriever
Pred: chesapeake_bay_retriever
True: appenzeller
Pred: appenzeller
True: irish_terrier
Pred: irish_terrier
True: miniature_pinscher
Pred: miniature_pinscher
True: golden_retriever
Pred: golden_retriever
True: cairn
Pred: cairn
True: irish_water_spaniel
Pred: irish_water_spaniel
True: tibetan_mastiff
Pred: tibetan_mastiff
True: maltese_dog
Pred: maltese_dog

3. Parameter Table

As stated in the Google Colab notebook, our model performed incredibly well, but it maxed out the available runtime that is offered with the free version of Colab (i.e. Google Colab halted the execution and demanded we pay for the 'Pro' version to keep running more iterations). Due to this extenuating circumstance, we were unable to run multiple iterations to get multiple parameters to create a comprehensive table. Fortunately, we were able to run enough of the model to **achieve over 90% accuracy (94% training and 91% validation accuracy)**, so hopefully that is a sufficient outcome for this homework assignment as it demonstrates excellent capability and understanding of the subject material.

Parameter	Details
Architecture Parameters	
Base Model	EfficientNetB3
Input Size	300x300
Added Layer 1	GlobalAveragePooling2D
Added Layer 2	BatchNorm
Added Layer 3	Dropout(0.3)
Added Layer 4	Dense(512)
Added Layer 5	BatchNorm
Added Layer 6	Dropout(0.4)
Added Layer 7	Dense(120)
Training Parameters	
Initial Learning Rate	0.001
Batch Size	32
Model Size	
Total Parameters	11.64M (44.40 MB)
Trainable Parameters	852.6K (3.25 MB)
Non-trainable Parameters	10.79M (41.15 MB)
Best Performance Metrics	
Best Training Accuracy	94.86% (Epoch 14)
Best Validation Accuracy	91.39% (Epoch 6)
Best Top-5 Training Accuracy	99.89% (Epoch 14)
Best Top-5 Validation Accuracy	99.46% (Multiple epochs)
Final Training Loss	0.1555 (Epoch 14)
Final Validation Loss	0.3453 (Epoch 14)