Abdullah Kamal

abdullah202kamal@gmail.com
t-abdullahkamal@zewailcity.edu.eg
abdullahkamal@aucegypt.edu

Table of Contents

1. Introduction

Deep learning has significantly advanced the field of computer vision, enabling models to perform complex image classification tasks with high accuracy. This report presents an analysis of a notebook that explores the use of **MobileNetV2**, a lightweight yet powerful deep learning model, for image classification. The primary objective of the notebook is to train and evaluate a model capable of accurately classifying images from a given dataset.

The project follows a structured workflow that includes data preparation, model development, training, evaluation, and inference. A key focus of this implementation is leveraging **transfer learning**, where a pre-trained model is fine-tuned to improve classification performance while reducing training time and computational costs.

Dataset Information

The dataset used in this project is the **Oxford-IIIT Pet Dataset**, which consists of 37 different pet breeds, including both cats and dogs. Each class contains approximately **200 images**, with significant variations in **pose, lighting, and background clutter**, making classification a challenging task. The dataset also provides **segmentation masks**, though this notebook primarily focuses on **image classification** rather than segmentation.

Key Dataset Characteristics:

- **Total Classes**: 37 (various breeds of cats and dogs).
- **Total Images**: Around 7,400 labeled images.
- **Labeling**: Each image is assigned a class label corresponding to a pet breed.
- **Challenges**: Class imbalance, variations in background and lighting, and potential mislabeling.

Objectives of the Notebook

The primary goals of the notebook are:

1. **To preprocess and structure the dataset** for efficient training.
2. **To implement MobileNetV2** as a base model for classification.
3. **To train and fine-tune the model** for optimal performance.
4. **To evaluate the model** using various performance metrics.
5. **To analyze the results** and assess potential areas for improvement.

The use of **MobileNetV2** provides a balance between accuracy and efficiency, making it suitable for deployment on edge devices.

2. Environment and Dependencies

2.1 Programming Language & Frameworks

The notebook is developed using **Python**, leveraging **TensorFlow** and **Keras** as the primary deep learning frameworks. These provide an intuitive and scalable interface for building, training, and evaluating deep learning models efficiently.

2.2 Required Libraries

The project depends on several libraries, each serving a specific role in the workflow. The key dependencies are categorized as follows:

Deep Learning Frameworks

- **TensorFlow** – Core framework for deep learning model implementation.
- **Keras** – High-level API for TensorFlow, simplifying model building and training.

Data Handling and Numerical Computations

- **NumPy** – Supports efficient numerical operations and array manipulation.
- **Pandas** – Used for structured data handling, analysis, and processing.

Data Visualization

- **Matplotlib** – Enables visualization of training metrics such as loss and accuracy trends.
- **Seaborn** – Provides enhanced visual representations, including confusion matrices.

Machine Learning Utilities

- **Scikit-learn (sklearn)** – Essential for dataset splitting, evaluation metrics, and preprocessing techniques.

File Management and Dataset Handling

- **OS** – Facilitates interaction with the file system for managing datasets and models.
- **Glob** – Used for retrieving file paths from directories for efficient dataset organization.

To optimize training time, **GPU acceleration** is highly recommended, especially for handling large datasets efficiently.

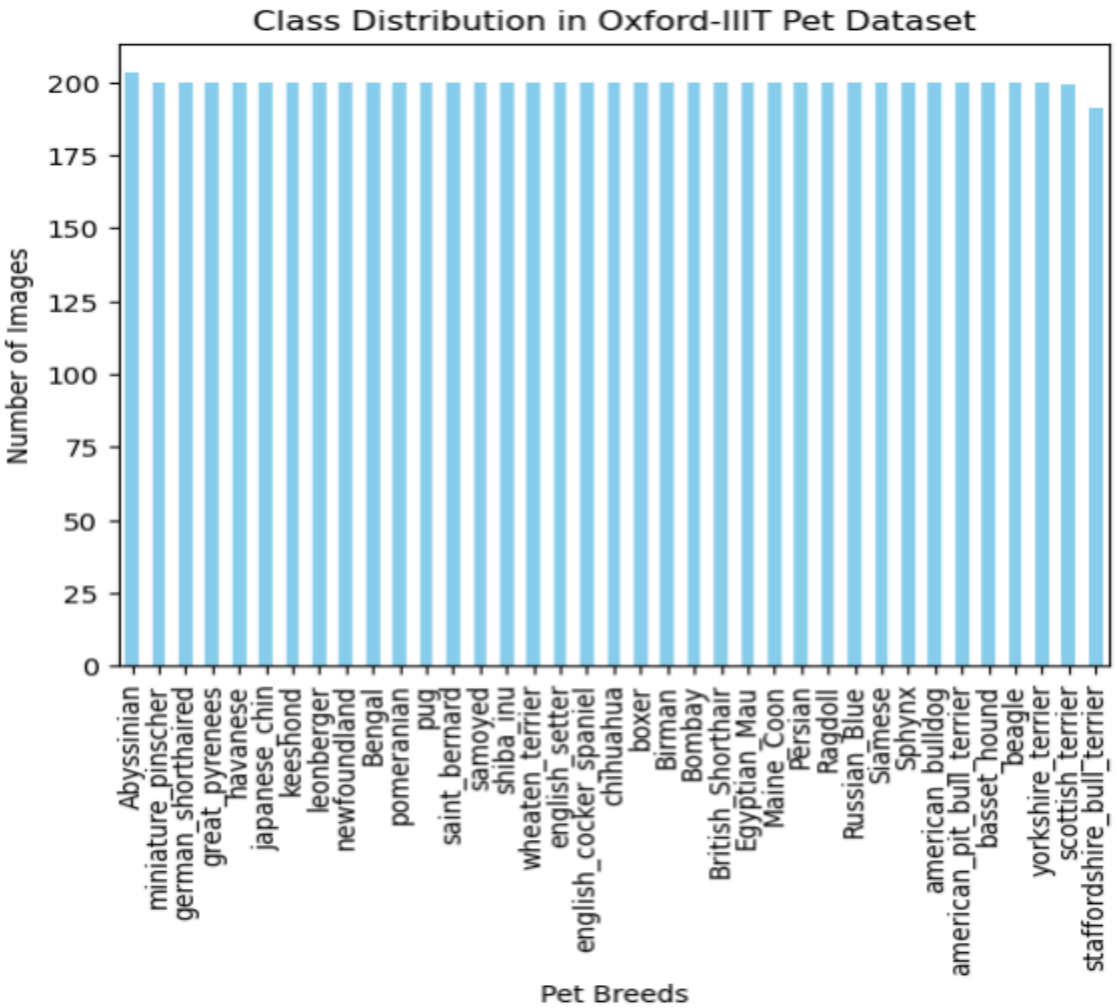
3. Methodology and Implementation

3.1 Data Preprocessing and Preparation

The dataset used in this notebook is the **Oxford-IIIT Pet Dataset**, which consists of images of **37 different pet breeds**.

```
print("Class Names:", class_names)
```

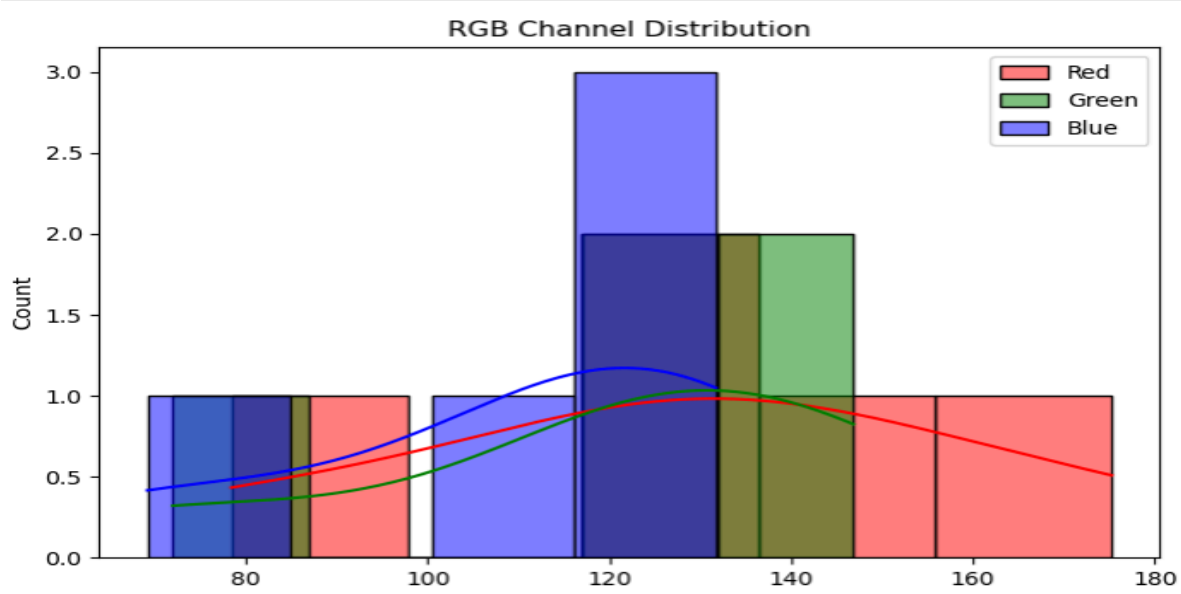
```
Total Classes: 37
Class Names: ['Abyssinian', 'Bengal', 'Birman', 'Bombay', 'British_Shorthair', 'Egyptian_Mau', 'Maine_Coon', 'Persian', 'Ragdoll', 'Russian_Blue', 'Siam
ese', 'Sphynx', 'american_bulldog', 'american_pit_bull_terrier', 'basset_hound', 'beagle', 'boxer', 'chihuahua', 'english_cocker_spaniel', 'english_sett
er', 'german_shorthaired', 'great_pyrenees', 'havanese', 'japanese_chin', 'keeshond', 'leonberger', 'miniature_pinscher', 'newfoundland', 'pomeranian',
'pug', 'saint_bernard', 'samoyed', 'scottish_terrier', 'shiba_inu', 'staffordshire_bull_terrier', 'wheaten_terrier', 'yorkshire_terrier']
```



The dataset is loaded and processed using the following steps:

- **Image Preprocessing**:
  - All images are resized to **224x224 pixels** to match the MobileNetV2 input format.
  - Pixel values are normalized to the range **[0, 1]** to ensure numerical stability during training.

With some data analysis of the images:




```
Class: Abyssinian, Image: Abyssinian_190.jpg, Shape: (300, 297, 3)
Class: Bengal, Image: Bengal_41.jpg, Shape: (500, 500, 3)
Class: Birman, Image: Birman_126.jpg, Shape: (500, 334, 3)
Class: Bombay, Image: Bombay_103.jpg, Shape: (143, 114, 3)
Class: British_Shorthair, Image: British_Shorthair_185.jpg, Shape: (334, 500, 3)
```

```
]: compute_mean_std(dataset_path=images_path_on_Abdullah_Laptop)

Mean pixel values: [0.4281809  0.47315615 0.52750456]
Standard deviation: [0.28521737 0.27700776 0.27655622]
```

- **Data Splitting**:
    - **Training Set (70%)**
    - **Validation Set (15%)**
    - **Test Set (15%)**
    - Implemented using **ImageDataGenerator** with `validation_split=0.3`.

```
Data Split:
Training: 5174 images
Validation: 1108 images
Test: 1108 images
```

3.2 Model Development

The notebook utilizes **MobileNetV2**, a pre-trained deep learning model optimized for efficient image classification. The model is modified as follows:

- **Feature Extraction**:
    - The base **MobileNetV2 model is loaded with pre-trained ImageNet weights**.
    - The **top fully connected layers are removed**.
    - The convolutional base is **frozen**, preventing its weights from being updated during initial training.

```
Total params: 2,257,984 (8.61 MB)
Trainable params: 0 (0.00 B)
Non-trainable params: 2,257,984 (8.61 MB)
```

- **Custom Classification Layers**:
    - **Global Average Pooling Layer** – Reduces the feature maps to a single vector per image.
    - **Fully Connected Dense Layer** – Allows the model to learn complex relationships.

- **Dropout Layer (to prevent overfitting)** – Randomly disables some neurons during training.
- **Output Layer** – Uses **37 neurons (one per class)** with a **softmax activation function**.

```
Total params: 2,426,725 (9.26 MB)
Trainable params: 168,741 (659.14 KB)
Non-trainable params: 2,257,984 (8.61 MB)
```

- **Compilation Settings**:
  - **Loss Function**: Categorical Cross-Entropy (since it's a multi-class classification problem).
  - **Optimizer**: Adam (with an adaptive learning rate).
  - **Evaluation Metric**: Accuracy.

3.3 Model Training

The training phase includes:

- **Hyperparameters**:
  - Initial **learning rate = 0.001**.
  - **Batch size** defined for optimal performance.
  - **Number of epochs = 10** (initial training).
- **Training Strategy**:
  - The frozen **MobileNetV2 base layers** remain unchanged.
  - Only the newly added layers are trained initially.
  - **Early stopping and model checkpointing** are implemented to save the best model based on validation accuracy.
- **Fine-Tuning Phase**:
  - After initial training, **some layers of the base model are unfrozen** to allow further learning.
  - A **lower learning rate (1e-5) is used** to ensure fine-tuning does not disrupt pre-learned features.
  - The model is trained for additional epochs with validation monitoring.

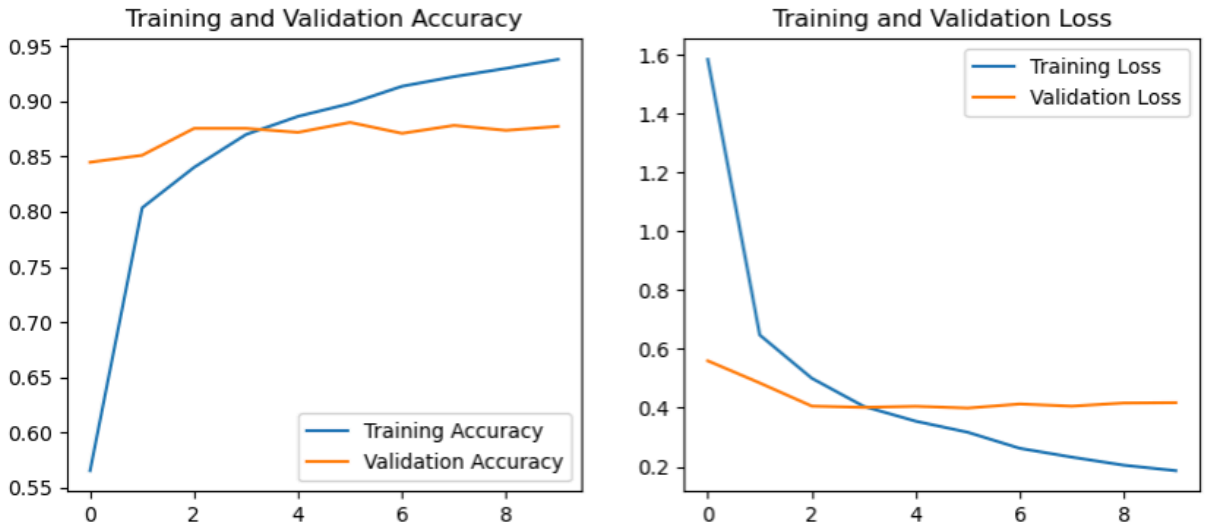
3.4 Model Evaluation

The trained model is evaluated on the test set using multiple performance metrics:

- **Accuracy and Loss**: Measured on the test dataset.
- **Confusion Matrix Analysis**:
    - Used to visualize **misclassified samples** and understand which breeds are commonly confused.
    - Heatmap representation of the confusion matrix is generated for better interpretability.
- **Precision, Recall, and F1-Score**:
    - Classification reports are generated for **each breed** to assess performance across different classes.

Results Before Fine-Tuning:

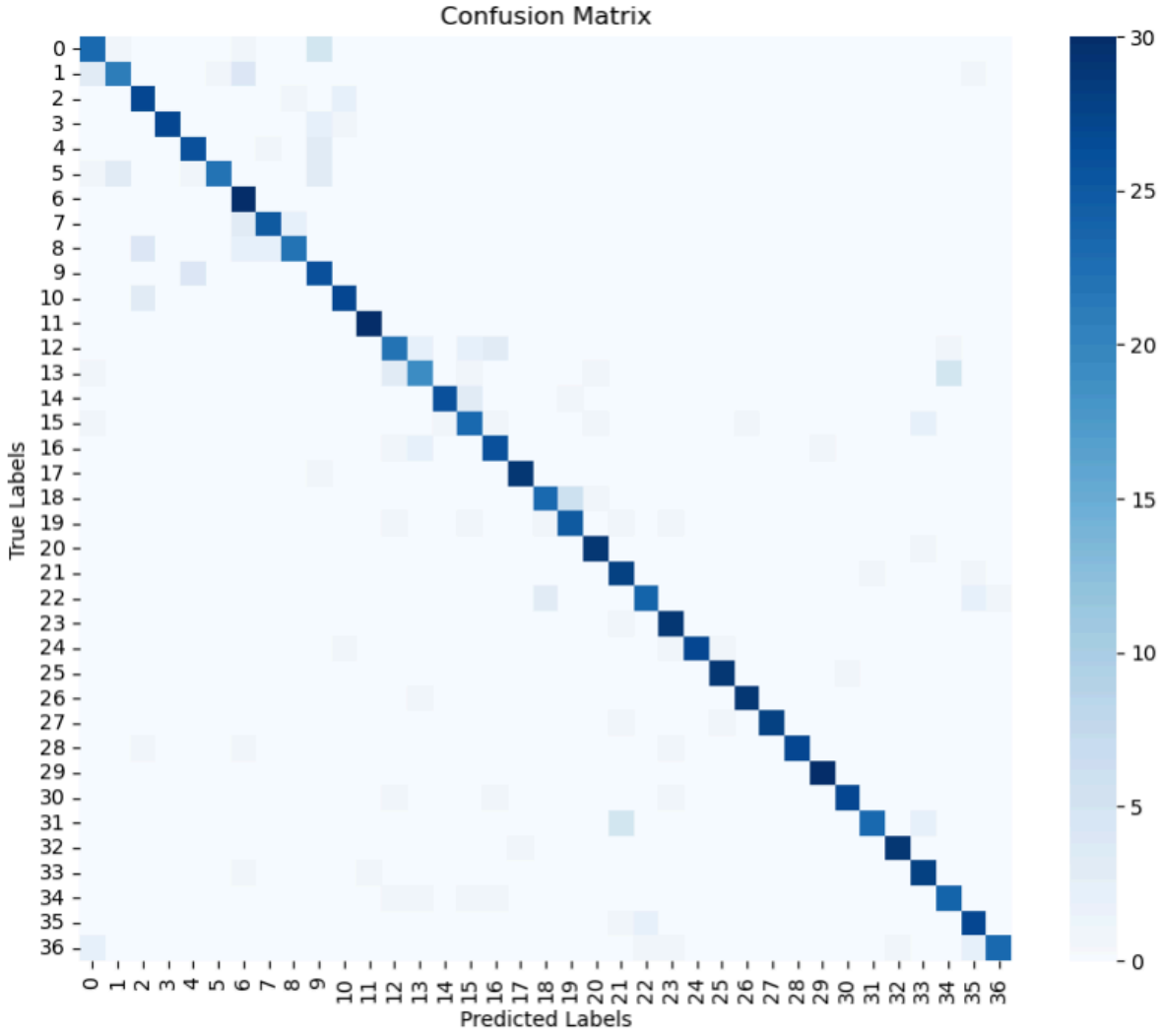
- **Test Accuracy**: **86.64%**
- **Test Loss**: **0.4210**




```
35/35 ━━━━━━━━━━━━━━━━━━━━ 14s 388ms/step - accuracy: 0.8552 - loss: 0.4277
Test Accuracy: 0.8664
Test Loss: 0.4210
```

Classification Report:

accuracy			0.87	1108
macro avg	0.87	0.87	0.87	1108
weighted avg	0.87	0.87	0.87	1108



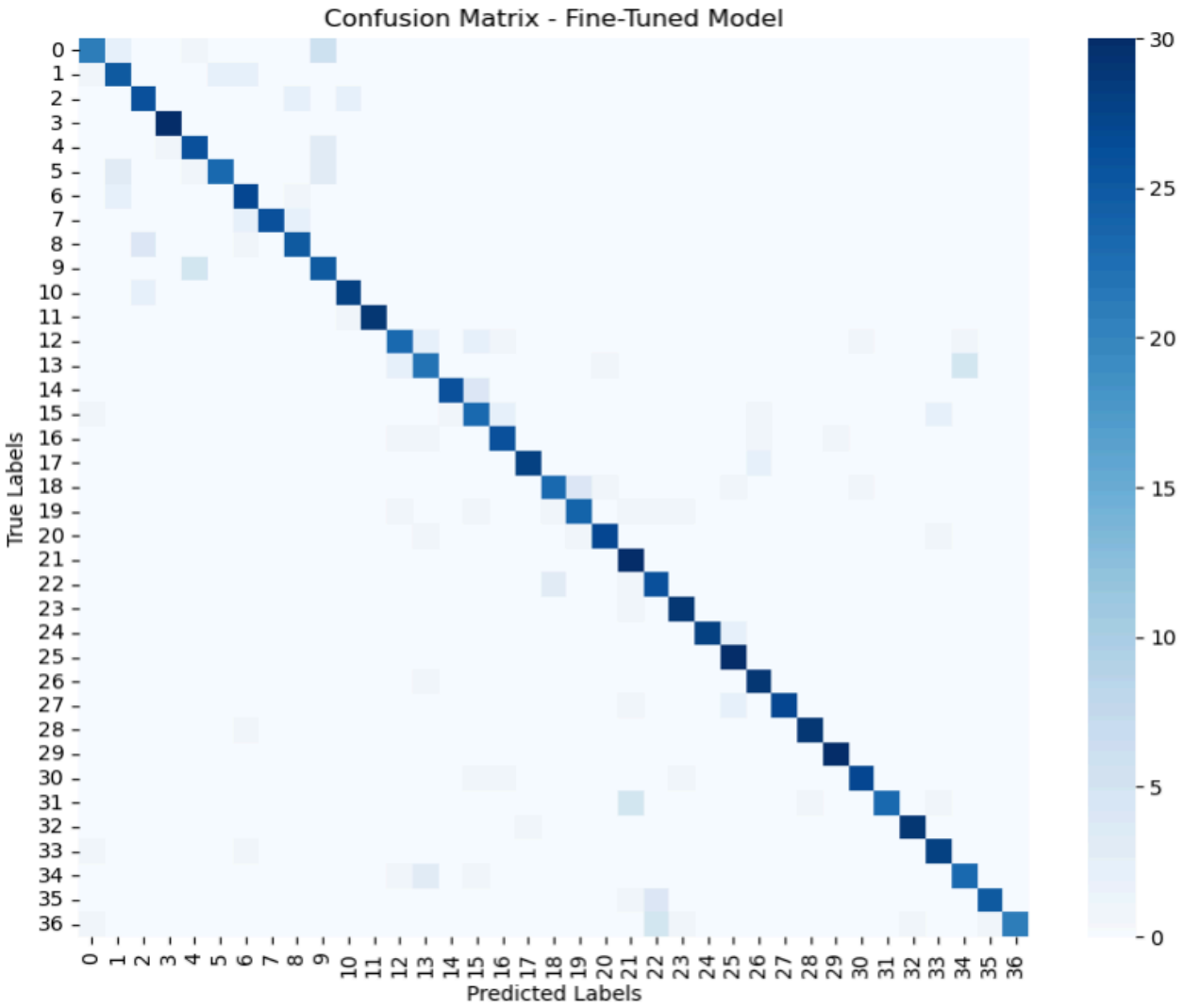
Note that the index is mapped for the class name for now, I will remap it again.

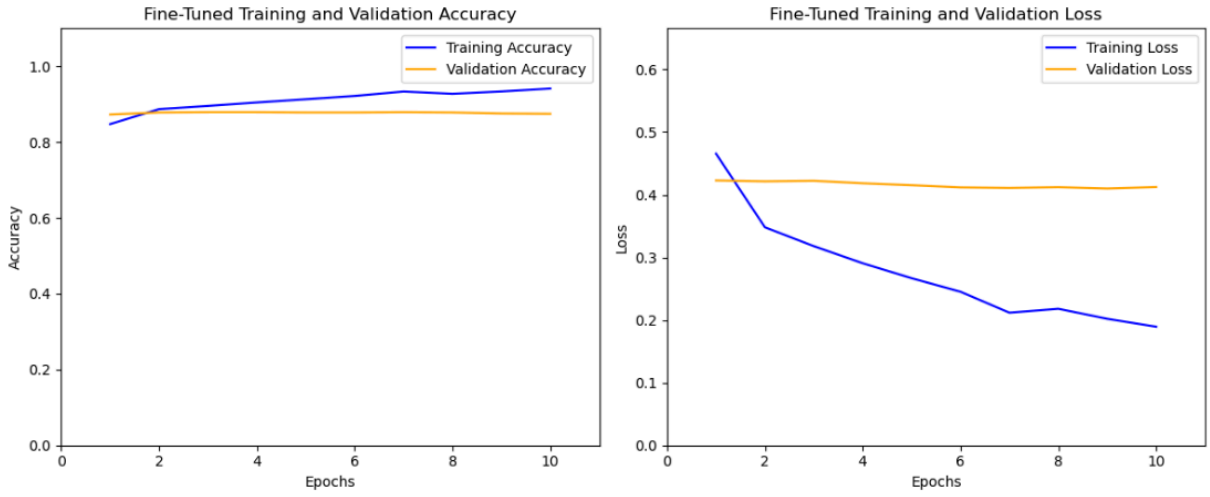
Results After Fine-Tuning:

- **Fine-Tuned Test Accuracy**: **87.27%**
- **Fine-Tuned Test Loss**: **0.3977**

Classification Report:

accuracy			0.87	1108
macro avg	0.88	0.87	0.87	1108
weighted avg	0.88	0.87	0.87	1108





Fine-tuning slightly improved test accuracy and generalization.

3.5 Predictions and Visualization

To demonstrate the model's classification capabilities, predictions are made on **new images**. The following steps are performed:

- **Loading Test Images**: Randomly selected images are passed through the trained model.
- **Generating Predictions**: The model assigns a class label to each test image.

- **Visualizing Results**:
  - Correct and **misclassified images** are displayed with **true and predicted labels**.

```
35/35 ━━━━━━━━━━━━━━━━━━━━ 14s 394ms/step
Total test samples: 1108
Misclassified samples: 148
Misclassification Rate: 13.36%
File: C:\Users\HP Zbook 17 G6\Desktop\RD\oxford-iiit-pet\images\samoyed\samoyed_147.jpg, True: 31, Predicted: 21
File: C:\Users\HP Zbook 17 G6\Desktop\RD\oxford-iiit-pet\images\Abyssinian\Abyssinian_136.jpg, True: 0, Predicted: 9
File: C:\Users\HP Zbook 17 G6\Desktop\RD\oxford-iiit-pet\images\Abyssinian\Abyssinian_126.jpg, True: 0, Predicted: 9
File: C:\Users\HP Zbook 17 G6\Desktop\RD\oxford-iiit-pet\images\Abyssinian\Abyssinian_130.jpg, True: 0, Predicted: 9
File: C:\Users\HP Zbook 17 G6\Desktop\RD\oxford-iiit-pet\images\american_pit_bull_terrier\american_pit_bull_terrier_146.jpg, True: 13, Predicted: 12
File: C:\Users\HP Zbook 17 G6\Desktop\RD\oxford-iiit-pet\images\British_Shorthair\British_Shorthair_129.jpg, True: 4, Predicted: 9
File: C:\Users\HP Zbook 17 G6\Desktop\RD\oxford-iiit-pet\images\Persian\Persian_149.jpg, True: 7, Predicted: 6
File: C:\Users\HP Zbook 17 G6\Desktop\RD\oxford-iiit-pet\images\beagle\beagle_118.jpg, True: 15, Predicted: 33
File: C:\Users\HP Zbook 17 G6\Desktop\RD\oxford-iiit-pet\images\beagle\beagle_134.jpg, True: 15, Predicted: 16
File: C:\Users\HP Zbook 17 G6\Desktop\RD\oxford-iiit-pet\images\Persian\Persian_153.jpg, True: 7, Predicted: 6
```

  - A **subset of misclassified samples** is analyzed to identify common errors.

I searched for some examples to see the similarity among those animals in the real world. For example, Bengal cat predicted and the Egyptian Mau which both have spotted or striped fur patterns.

Challenges Encountered

During the model development and evaluation process, the following challenges were identified:

1. **Model Performance & Accuracy**
    - **High misclassification rate (~17%)** before fine-tuning.
    - Some breeds had **very similar visual patterns**, making classification difficult.
    - Fix: Fine-tuned deeper layers of MobileNetV2 with a lower learning rate.
2. **Class Imbalance**
    - Some classes were predicted more frequently than others.
    - Fix: Applied **data augmentation** and adjusted class weighting.

The next notebook and report contains the same model with augmented data