Abdullah Kamal

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

Table of Contents

1. Introduction

Deep learning continues to revolutionize computer vision, enabling highly accurate image classification tasks. This report analyzes a notebook that extends the previous MobileNetV2-based pet classification model by incorporating **data augmentation techniques**. The primary objective is to enhance model generalization by artificially expanding the dataset using transformations such as rotation, zooming, flipping, and shifting.

This project builds upon the **Oxford-IIIT Pet Dataset**, which contains images of **37 pet breeds**. By applying **data augmentation**, the model aims to reduce overfitting and improve classification performance on unseen data. The workflow follows a structured approach, including data preprocessing, model development, training, evaluation, and inference.

Dataset Information

The dataset used in this project is the **Oxford-IIIT Pet Dataset**, which consists of **37 different pet breeds**, covering both cats and dogs. The dataset presents challenges such as class imbalance, background variations, and similarity among certain breeds, making it an excellent benchmark for testing model robustness.

Key Dataset Characteristics:

- **Total Classes**: 37 (various breeds of cats and dogs).
- **Total Images**: ~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 misclassification.

Objectives of the Notebook

The primary goals of this notebook are:

1. **To apply data augmentation techniques** to increase dataset diversity.
2. **To fine-tune MobileNetV2** and compare its performance with and without augmentation.
3. **To evaluate model performance** using metrics such as accuracy, precision, recall, and F1-score.
4. **To visualize classification errors** and analyze which breeds are commonly misclassified.

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.

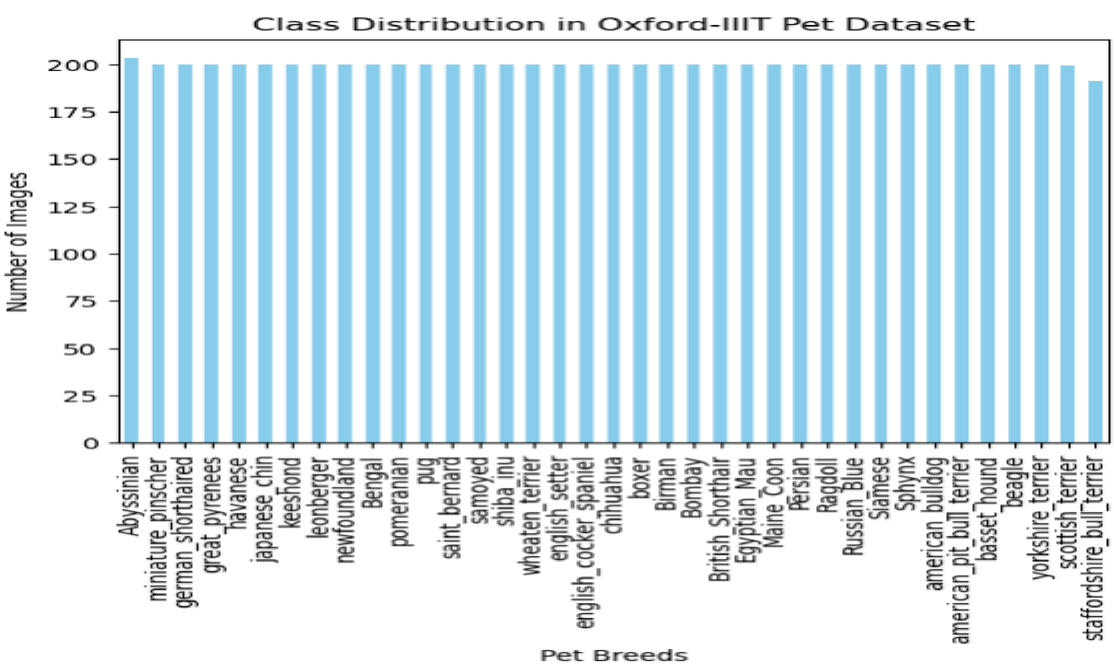
3. Methodology and Implementation

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)
```

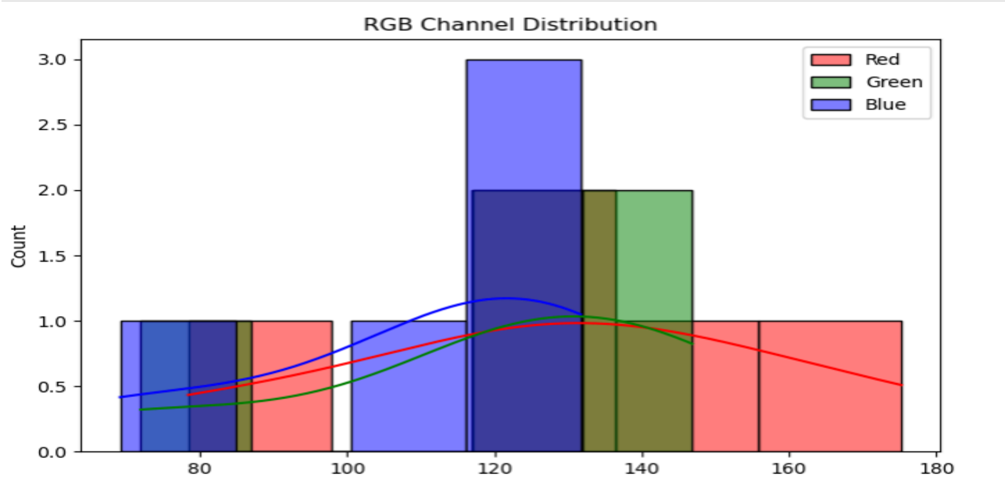
```
Total Classes: 37
Class Names: ['Abyssinian', 'Bengal', 'Birman', 'Bombay', 'British_Shorthair', 'Egyptian_Mau', 'Maine_Coon', 'Persian', 'Ragdoll', 'Russian_Blue', 'Siamese', 'Sphynx', 'american_bulldog', 'american_pit_bull_terrier', 'basset_hound', 'beagle', 'boxer', 'chihuahua', 'english_cocker_spaniel', 'english_setter', '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`.

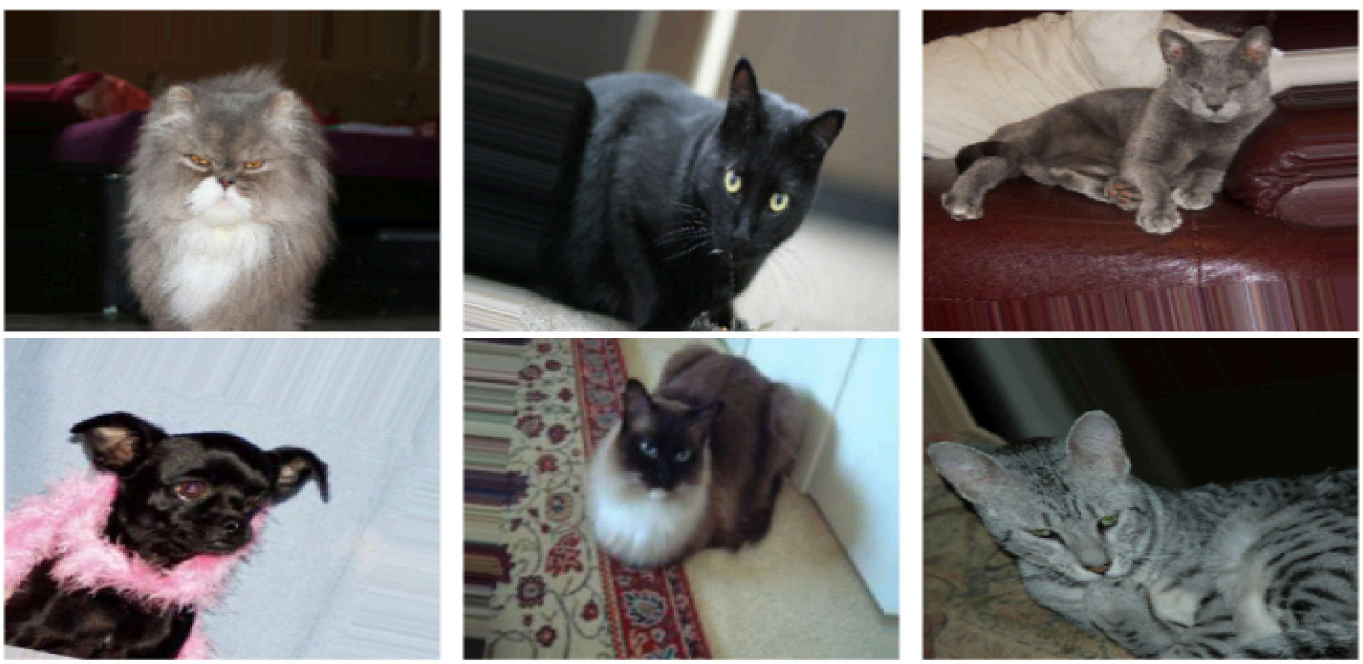
```
Data Split:
Training: 5174 images
Validation: 1108 images
Test: 1108 images
```

Data Augmentation

To prevent overfitting and improve model robustness, **ImageDataGenerator** is used to apply the following transformations:

- **Rotation Range**: Up to **30 degrees** rotation.
- **Width & Height Shift**: Random horizontal/vertical shifts up to **20%** of the image size.
- **Shear & Zoom Transformations**: Adds perspective distortion and scale variations.
- **Horizontal Flipping**: Introduces mirrored versions of images to simulate real-world diversity.

The augmented images are visualized to confirm the correct application of transformations.



3.2 Model Development

The deep learning model is built upon **MobileNetV2**, leveraging **transfer learning** to accelerate training and improve classification accuracy.

Feature Extraction (Using MobileNetV2)

- The base **MobileNetV2 model is pre-trained on ImageNet** and serves as the feature extractor.
- The **top fully connected layers are removed** to allow customization for pet classification.
- The base model layers are **frozen initially** to retain pre-trained knowledge.

```
Total params: 2,426,467 (9.26 MB)
Trainable params: 168,483 (658.14 KB)
Non-trainable params: 2,257,984 (8.61 MB)
```

Custom Classification Layers

To adapt MobileNetV2 for the **Oxford-IIIT Pet Dataset**, additional layers are added:

- **Global Average Pooling Layer** – Reduces feature maps to a single vector per image.
- **Fully Connected Dense Layer** – Enables complex feature learning.
- **Dropout Layer** – Prevents overfitting by randomly disabling neurons during training.
- **Output Layer** – Uses **35 neurons** (one per class) with **softmax activation** for multi-class classification.

Compilation Settings

- **Loss Function**: **Categorical Cross-Entropy** (suitable for multi-class classification).
- **Optimizer**: **Adam**, chosen for its adaptive learning rate.
- **Evaluation Metric**: **Accuracy** to track classification performance.

3.3 Model Training

The training phase involves **fine-tuning MobileNetV2** using the augmented dataset.

Hyperparameters

- **Initial Learning Rate**: **0.001**
- **Batch Size**: **32**
- **Number of Epochs**: **10**

Training Strategy

- The **MobileNetV2 base layers remain frozen initially**, training only the newly added layers.
- The model is trained on the **augmented training set** while monitoring validation accuracy.
- **Early stopping and model checkpointing** are applied to save the best-performing model.

Fine-Tuning Phase

- Some **layers of the base MobileNetV2 model are unfrozen** to enable further learning.
- A **lower learning rate (1e-5)** is applied to fine-tune the model carefully.
- Additional training is performed for **10 more epochs** while tracking validation performance.

Training history is visualized, showing **loss and accuracy trends** over epochs.

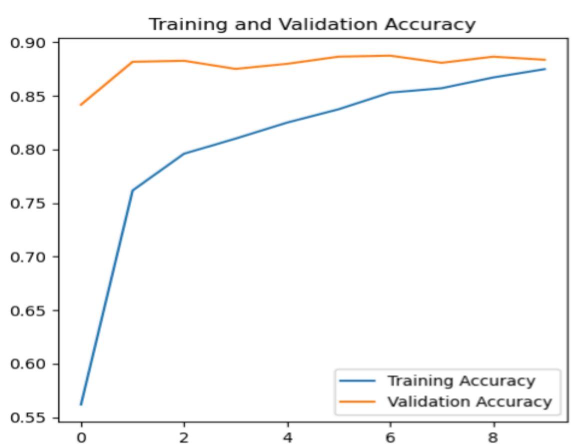
3.4 Model Evaluation

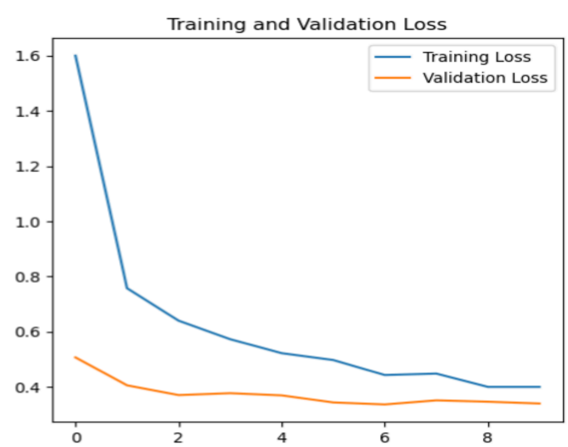
The trained model is evaluated using multiple performance metrics:

Accuracy and Loss

- **Test Accuracy**: **89%**
- **Test Loss**: Analyzed to ensure stable convergence.

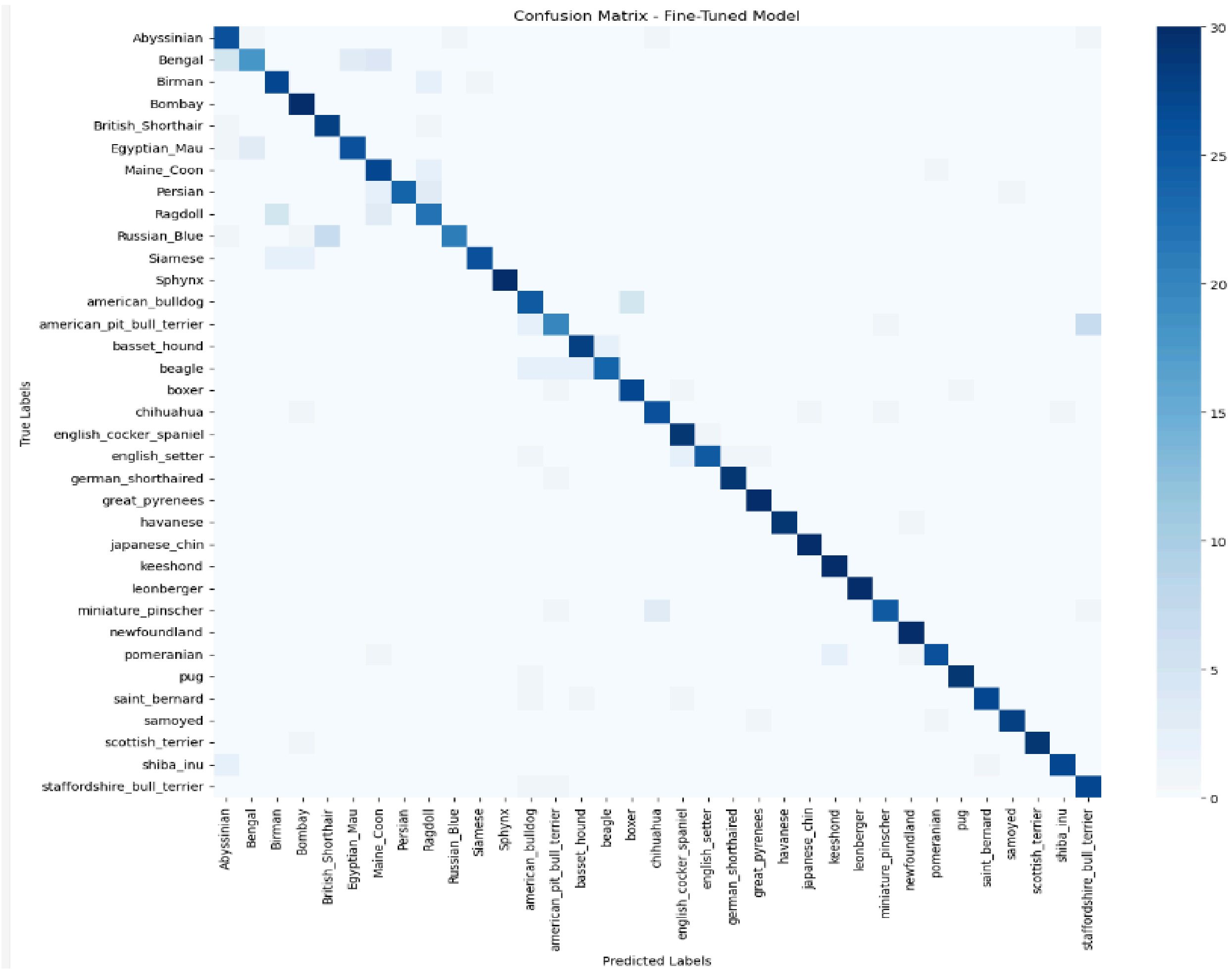
```
33/33 ━━━━━━━━━━━━━━━━━━━━
Test Accuracy: 0.8913
Test Loss: 0.3353
```





Confusion Matrix Analysis

- The **diagonal represents correct classifications**, while off-diagonal values indicate errors.
- Certain breeds like **Sphynx and Siamese** exhibit near-perfect classification, while **Bengal cats** show some confusion.



Precision, Recall, and F1-Score

- **Classification reports** provide detailed metrics per class.
- Breeds with lower recall scores indicate potential **class imbalance** or **difficult-to-distinguish patterns**.

accuracy			0.89	1049
macro avg	0.90	0.89	0.89	1049
weighted avg	0.90	0.89	0.89	1049

3.5 Predictions and Visualization

The model's real-world applicability is demonstrated through test image predictions:

Generating Predictions

- The model assigns **probability scores** to each class.
- The predicted class is selected based on the **highest probability**.

Visualizing Results

- Randomly selected **test images** are displayed with their **predicted labels**.
- Misclassified samples are analyzed to identify common errors.

A subset of **misclassified breeds** is examined to highlight the impact of **data augmentation on reducing errors**.

4. Challenges Encountered

During the implementation, the following challenges were noted:

1. **Increased Training Time**
    - **Data augmentation increases dataset complexity**, requiring more training time.
    - Fix: Used **Google Colab's GPU acceleration** to speed up training.

To ensure reproducibility, the trained model was saved and successfully reloaded. The reloaded model produced consistent predictions on unseen images, confirming that the training process was correctly preserved.

5. Conclusion and Future Work

This notebook successfully implemented **data augmentation** to improve the classification performance of **MobileNetV2** on the **Oxford-IIIT Pet Dataset**. The use of augmentation techniques such as **rotation, flipping, zooming, and shifting** helped the model generalize better, reducing overfitting and improving test accuracy.

Future Work

Although data augmentation significantly improved the model, several areas can still be explored:

1. **Advanced Augmentation Techniques**
    - Applying **GAN-based augmentation** to generate synthetic images of rare breeds.
2. **Hyperparameter Tuning**
    - Exploring **learning rate scheduling** for more stable convergence.
    - Optimizing the **batch size and number of fine-tuning epochs**.
3. **Exploring Other Architectures**
    - Comparing **MobileNetV2** with more advanced architectures like **EfficientNet or Vision Transformers (ViTs)**.
    - Evaluating **ensemble models** that combine predictions from multiple architectures.