



International University, HCMC National University School of
Computer Science and Engineering

FINAL REPORT

Multi-class Classification for Oxford-Pet III dataset

Submitted by.

No.	FULL NAME	STUDENT ID
1	Đặng Quốc Anh Duy	ITDSIU20015
2	Nguyễn Đức Toàn	ITCSIU21112
3	Nguyễn Thái Bình	ITCSIU21043

Course name: **Artificial Intelligence**

Professors: PhD. Le Thanh Sach, Dr. Ly Tu Nga

May 29, 2024

TABLE OF CONTENTS

1. Overview	3
1.1. Introduction	3
1.2. Objectives	3
1.3. Dataset Information	4
2. Data Preprocessing + Exploratory Data Analysis	5
2.1. Data Preprocessing	5
2.2. Exploratory Data Analysis	8
3. Classification Implementation	10
3.1. ResNet50_v2	10
3.2. MobileNet_v2	12
3.3. Inception_V3	14
4. Model Evaluation	17
6. Reference Lists	25

List of Figures

Figure 1.1: Dataset Info	4
Figure 2.1: Data Splitting	5
Figure 2.2: Instances of the dataset	5
Figure 2.3: Resize Image Function	6
Figure 2. 4: One-hot Encoding	6
Figure 2. 5: Image after performing augmentation	7
Figure 2.6: Perform Augmentation with various techniques	8
Figure 2.7: Batching and Prefetching data	8
Figure 2.8: The distribution of species within the Oxford Pet dataset	9
Figure 2.9: Histograms of Image Size	9
Figure 3.1: Architecture of ResNet-V2	11
Figure 3.2: Architecture of MobileNet-V2	13
Figure 3.3: Architecture of Inception-V3	16

1. Overview

1.1. Introduction

Multiclass classification is a crucial task in machine learning, particularly when dealing with datasets that encompass multiple categories within a single domain. In the context of our project, we explore a dataset containing various breeds of pets, specifically focusing on two species: dogs and cats. The primary objective is to develop a model capable of accurately identifying the breed of a given pet from this dataset. This problem presents unique challenges due to the inherent variability in appearance and characteristics among different breeds. Successfully tackling this classification task can lead to valuable applications in areas such as pet identification, health monitoring, and breed-specific care recommendations.

Additionally, understanding breed distinctions through deep learning can aid in advancing research in animal genetics and behavior. By leveraging sophisticated algorithms and computational techniques, this project aims to contribute significantly to the field of pet-related data science.

1.2. Objectives

Here is the list of objectives to be achieved:

- Data Collection: Select the suitable dataset for exploration
- Data Preprocessing: Convert image into expected format of the classification model, handle missing values, encode the target variable, etc.
- Exploratory Data Analysis: Analyze the raw data, visualize some plots to get brief information about the dataset. This is where we can ask questions for exploration
- Implementation: Develop a robust deep learning model (ResNet50_V2 and MobileNet_V2) that can accurately classify the breed of pets with complex architecture
- Model Evaluation: Compare the performance of the defined algorithms, give remarks based on the classification result.

1.3. Dataset Information

- Oxford-IIIT dataset (<https://www.robots.ox.ac.uk/~vgg/data/pets/>)
- The Oxford-IIIT pet dataset has 7349 observations in total with 37 different breeds of two species Dog and Cat. Each breed contains roughly 200 images with large variations in scale, pose and lighting. All images have an associated ground truth annotation of breed
- While there are various sources where the image dataset can be downloaded from, we will use the `tfds` module to load the images to optimize the working storage

Breed	Count
American Bulldog	200
American Pit Bull Terrier	200
Basset Hound	200
Beagle	200
Boxer	199
Chihuahua	200
English Cocker Spaniel	196
English Setter	200
German Shorthaired	200
Great Pyrenees	200
Havanese	200
Japanese Chin	200
Keeshond	199
Leonberger	200
Miniature Pinscher	200
Newfoundland	196
Pomeranian	200
Pug	200
Saint Bernard	200
Samoyed	200
Scottish Terrier	199
Shiba Inu	200
Staffordshire Bull Terrier	189
Wheaton Terrier	200
Yorkshire Terrier	200
Total	4978

1.Dog Breeds

Breed	Count
Abyssinian	198
Bengal	200
Birman	200
Bombay	200
British Shorthair	184
Egyptian Mau	200
Main Coon	190
Persian	200
Ragdoll	200
Russian Blue	200
Siamese	199
Sphynx	200
Total	2371

2.Cat Breeds

Family	Count
Cat	2371
Dog	4978
Total	7349

3.Total Pets

Figure 1.1: Dataset Info

2. Data Preprocessing + Exploratory Data Analysis

2.1. Data Preprocessing

In the preprocessing phase, first we load the data using 'tfds' (tensorflow dataset) and split data into 3 parts: train, validation, and test sets.

```
num_train_examples = tf.data.experimental.cardinality(train_raw).numpy()  
num_val_examples = tf.data.experimental.cardinality(val_raw).numpy()  
num_test_examples = tf.data.experimental.cardinality(test_raw).numpy()  
  
print('Number of training samples:', num_train_examples)  
print('Number of validation samples:', num_val_examples)  
print('Number of test samples:', num_test_examples)
```

Number of training samples: 2944
Number of validation samples: 736
Number of test samples: 3669

Figure 2.1: Data Splitting

For the distribution of three sets, 80% of the raw data will be used as the train set, and the remaining will be applied for validating during the training phase. For the test data, these samples are new unknown records with more than 3600 samples to test the performance of the classification models.

```
# Display several examples of the training set  
examples = tfds.as_dataframe(train_raw.take(5), ds_info)  
examples
```







	file_name	image	label	segmentation_mask	species
0	Sphynx_158.jpg		33 (Sphynx)		0 (Cat)
1	english_cocker_spaniel_135.jpg		12 (english_cocker_spaniel)		1 (Dog)
2	British_Shorthair_181.jpg		9 (British_Shorthair)		0 (Cat)

Figure 2.2: Instances of the dataset

Image Transformation

Since the raw images come in different sizes, we need to resize them to the same size before parsing them into the neural network later. More specifically, the shape of the image must be both 224 pixels in length and width (which is what ResNet50_V2 expects)

```
▶ IMG_SIZE = 224

# Resize images
def resize(image, label):
    image = tf.image.resize(image, (IMG_SIZE, IMG_SIZE))
    return image, label

train_ds = train_raw.map(resize)
val_ds = val_raw.map(resize)
test_ds = test_raw.map(resize)
```

Figure 2.3: Resize Image Function

One-hot-encoding

There are 37 classes (i.e. pet breeds) in our dataset that we are using for multi-class image classification. Therefore, we have to convert the target variable into one-hot encode format with the output vector of length 37 for each observation.

```
# Encoding the target label
def one_hot_encode(image, label):
    label = tf.one_hot(label, num_classes)
    return image, label
```

Figure 2. 4: One-hot Encoding

From the printout, we can see that each dataset object (i.e., `train_ds`, `val_ds`, and `test_ds`) has two components each:

- Images of shape (224,224,3)
- Label vector of shape (37,)

The values in each component have also been casted to a float data type, which is what we want for the deep learning model later.

Data Augmentation

Image augmentation artificially creates training images through different ways of processing or combination of multiple processing, such as random rotation, shifts, shear and flips, etc. The purpose of this technique is to reduce model overfitting by exposing our model to variations and small transformations in the original data. It is useful especially when we do not have a large dataset.

However, the image augmentation needs to be realistic. For example, flipping car images upside down may not be the best choice here since we expect most cars to be photographed with the wheels on the ground (barring severe accidents)

Here are the four newly generated images after we apply the data augmentation in Keras. These new data will be added to the training set to improve the performance of the model.

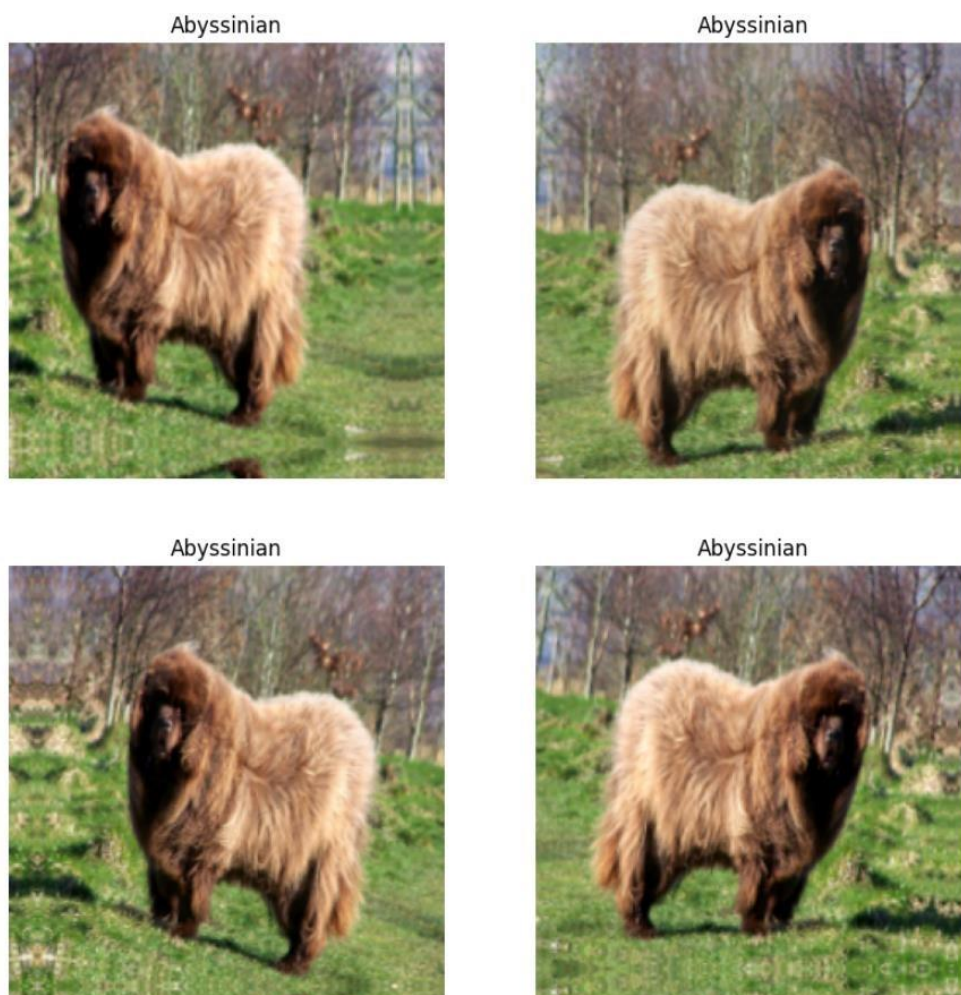


Figure 2. 5: Image after performing augmentation

In our implementation, we use the Keras preprocessing layers API to perform data augmentation. Each type of image augmentation that we want to introduce is defined as a layer within a Keras Sequential class. The generated data are in RandomFlip, RandomRotation, adjust translation, and with RandomContrast.

```
data_augmentation = keras.Sequential(
    [layers.RandomFlip('horizontal'),
     layers.RandomRotation(factor=(-0.025, 0.025)),
     layers.RandomTranslation(height_factor=0.1, width_factor=0.1),
     layers.RandomContrast(factor=0.1),
    ])
```

Figure 2.6: Perform Augmentation with various techniques

Batching and Prefetching

In order to optimize loading speed and model efficiency, we apply batching and prefetching method. A batch size of 32 is a good value to start with. The number of elements to prefetch can be automatically determined by making use of `tf.data.AUTOTUNE`, which prompts the runtime to tune the value dynamically for us.

```
BATCH_SIZE = 32

# Batch the data and use prefetching to optimize loading speed
# AVOID use of caching (Google Colab RAM limits)

train_ds = train_ds.cache().shuffle(1000).batch(batch_size=BATCH_SIZE,
                                                drop_remainder=True).prefetch(tf.data.AUTOTUNE)

val_ds = val_ds.cache().shuffle(1000).batch(batch_size=BATCH_SIZE,
                                             drop_remainder=True).prefetch(tf.data.AUTOTUNE)

test_ds = test_ds.cache().shuffle(1000).batch(batch_size=BATCH_SIZE,
                                              drop_remainder=True).prefetch(tf.data.AUTOTUNE)
```

Figure 2.7: Batching and Prefetching data

2.2. Exploratory Data Analysis

During the exploratory data analysis (EDA) stage, we performed a thorough examination of the pet dataset to understand its underlying structure and characteristics. One of the initial steps was to plot the distribution of the species, which showed that the dataset contains 1,999 instances of dogs and 945 instances of cats. This visualization provided a clear overview of the dataset's composition, enabling us to see the prevalence of each species. Such insights are crucial for

informing subsequent steps in the data preprocessing and model training process, ensuring that our approach is well-suited to the dataset's characteristics.

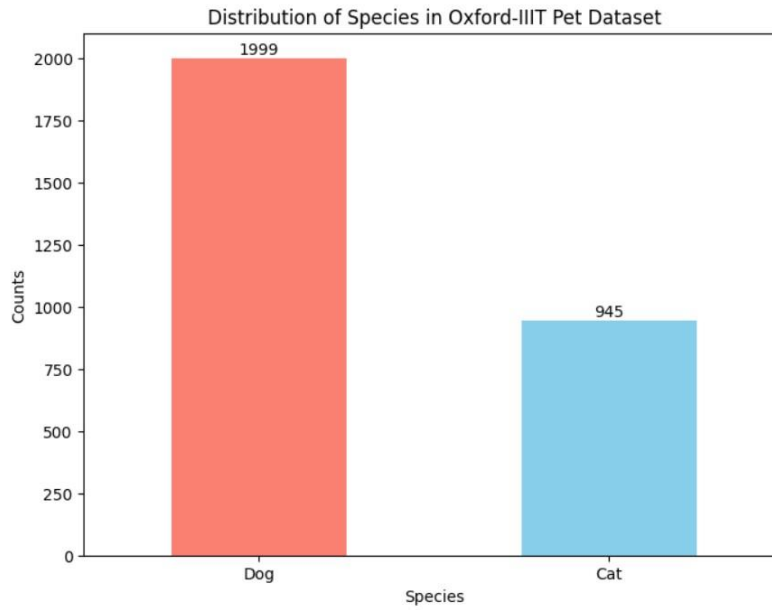


Figure 2.8: The distribution of species within the Oxford Pet dataset

As part of our exploratory data analysis, we generated histograms to examine the distribution of image sizes in terms of height and width. These histograms revealed the range and frequency of different image dimensions within the dataset, providing insights into the variability of image sizes. The image heights are distributed around 300-500 pixels while this figure for the widths are dominant in 500 pixels, which means that the standard width is 500 pixels.

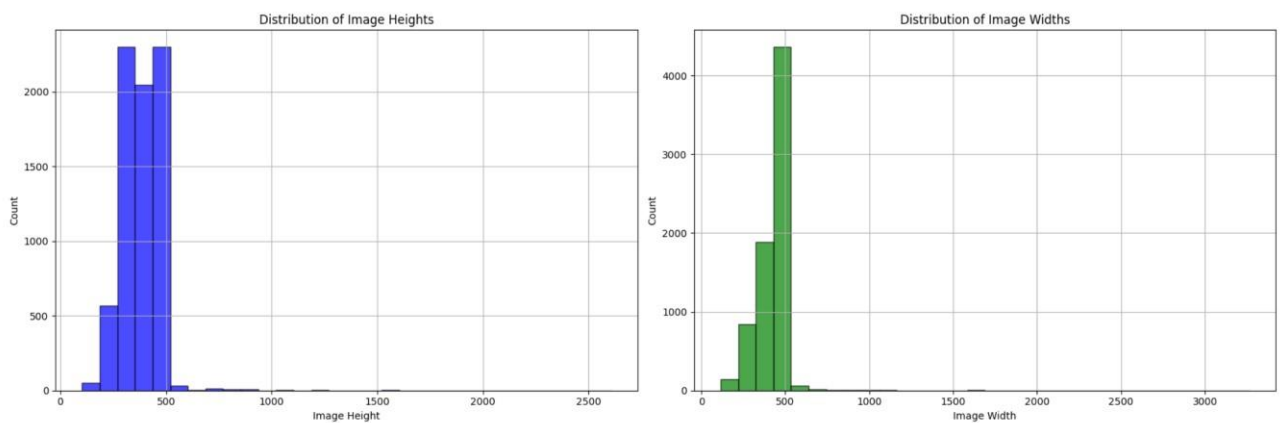


Figure 2.9: Histograms of Image Size

3. Classification Implementation

3.1. ResNet50_v2

The updated model architecture uses MobileNetV2 as its core feature extractor. Here is a detailed explanation of each stage of the model:

- **Input Layer:** the model accepts input images of shape (224, 224, 3), representing 224x224 pixel images with 3 color channels (RGB)
- **Sequential Block:** apply preprocessing steps sequentially to the input images
 - `tf.math.truediv`: This layer performs element-wise division, typically used for normalizing the pixel values
 - `tf.math.subtract`: This layer performs element-wise subtraction, likely as part of the normalization to adjust the mean pixel value
- **MobileNetV2 Layer:** a pre-trained deep convolutional neural network known for its efficiency and effectiveness in feature extraction. This layer processes the normalized images and outputs a feature map of shape (7, 7, 1280)
- **Global Average Pooling Layer:** reduce the spatial dimensions of the feature map by computing the average of each feature map channel, resulting in a 1280-dimensional vector for each image
- **Batch Normalization Layer:** normalize the output from the global average pooling layer to improve training stability and performance
- **Dropout Layer:** apply regularization method to prevent overfitting by randomly setting a fraction of the input units to zero during training
- **Dense (Fully Connected) Layer:** final dense layer, which uses a softmax activation function, outputs a 37-dimensional vector representing the probabilities of the 37 classes (different pet breeds)

Model Architecture

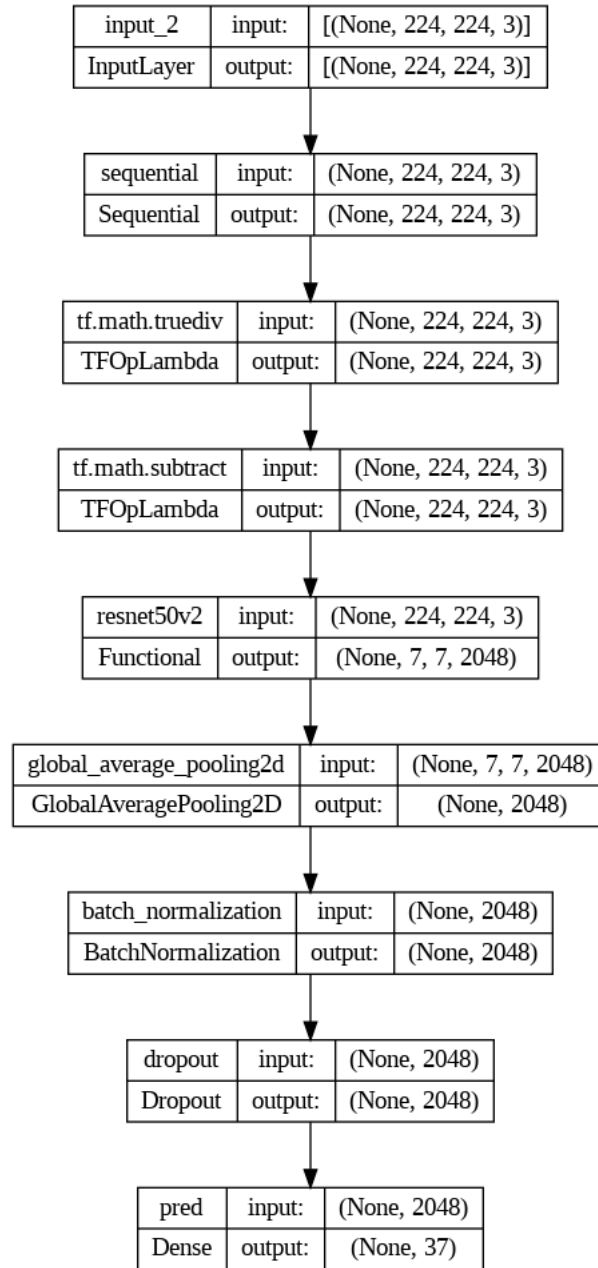


Figure 3.1: Architecture of ResNet-V2

As the input image passes through MobileNetV2, Convolutional and Pooling Layers are used to extract features while progressively reducing the spatial dimensions. Pooling layers and strided convolutions within MobileNetV2 reduce the height and width of the feature maps from 224 to 7. Depthwise Separable Convolutions help in maintaining a high number of channels (1280) for detailed feature extraction. By the time the data reaches the end of MobileNetV2, the spatial dimensions have been reduced from (224, 224) to (7, 7), while the number of channels (depth) has increased to 1280.

3.2. MobileNet_v2

The model architecture depicted in the diagram is designed for image classification and utilizes a combination of a pre-trained convolutional neural network (CNN) and custom layers for the classification task.

- Input Layer: begin with an input layer that accepts images of shape (224, 224, 3), representing 224x224 pixel images with 3 color channels (RGB) that match the expected shape of the MobileNet_V2 model
- Sequential Block:
 - sequential: preprocessing steps that applied sequentially to the input images.
 - `tf.math.truediv`: perform element-wise division for normalizing the pixel values.
 - `tf.math.subtract`: perform element-wise subtraction, which is part of a normalization step to adjust the mean pixel value.
- ResNet50V2 Layer: a pre-trained deep CNN known for its powerful feature extraction capabilities. This layer processes the normalized images before generating the output of a feature map with shape (7, 7, 2048).
- Global Average Pooling Layer:
 - `global_average_pooling2d`: reduce the spatial dimensions of the feature map by computing the average of each feature map channel, resulting in a 2048-dimensional vector for each image.
- Batch Normalization Layer: normalize the output from the global average pooling layer, improving training stability and performance.
- Dropout Layer: regularization method to prevent overfitting by randomly setting a fraction of the input units to zero during training.
- Dense (Fully Connected) Layer: the last dense layer to perform prediction, which uses a softmax activation function, outputs a 37-dimensional vector representing the probabilities of the 37 classes (different pet breeds).

Model Architecture:

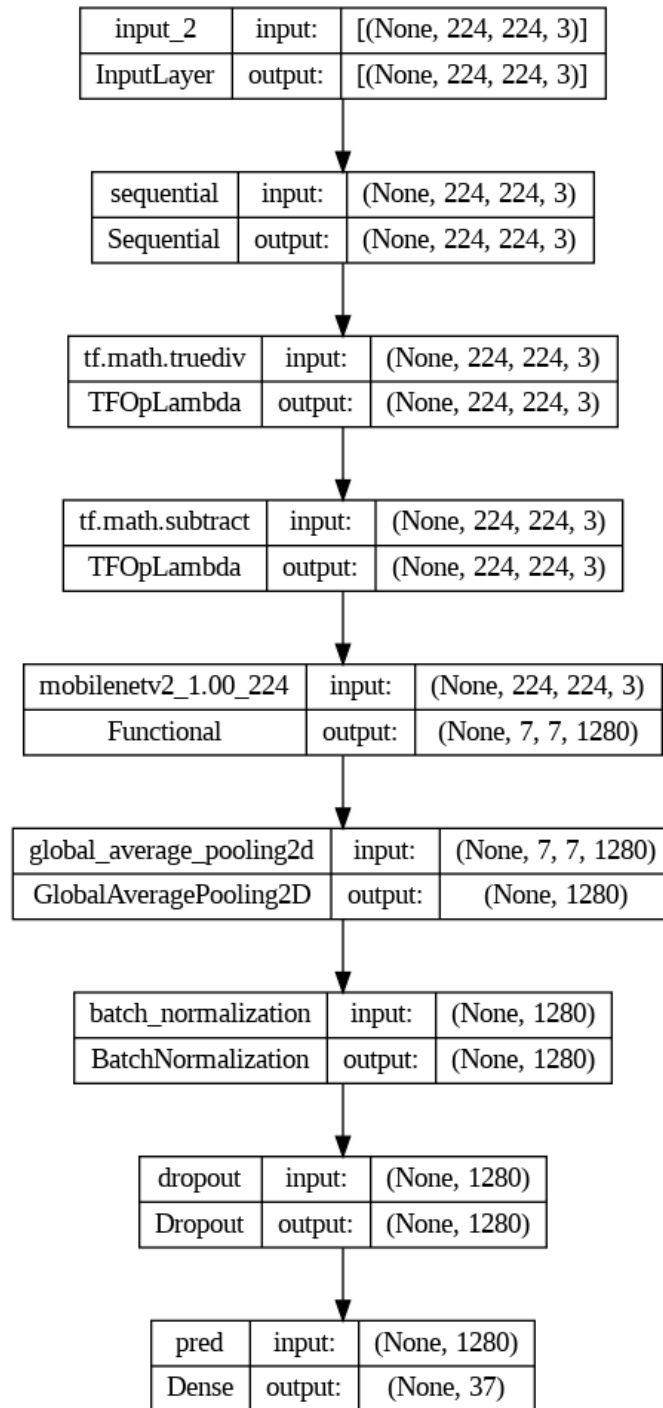


Figure 3.2: Architecture of MobileNet-V2

Overall, this architecture combines powerful feature extraction from ResNet50V2 with custom preprocessing and regularization layers to build a robust image classification model tailored for distinguishing between multiple pet breeds.

3.3. Inception_V3

The Deception V3 Model is a sophisticated machine learning model designed to detect and analyze deceptive behaviors in various contexts, such as text, speech, and possibly multimedia content. This model is the third iteration in the Deception series, building upon the strengths of its predecessors while introducing advanced features and enhancements.

- Input Layer (input_2):
 - Input Shape: (299, 299, 3)
 - Description: This layer specifies the input shape of the model, which is a 299x299 pixel RGB image (3 channels).
- Sequential Block (sequential):
 - Description: This block appears to contain a series of layers or operations that are not detailed in the diagram. It's likely used for preprocessing or additional transformations on the input data.
- Lambda Layers (tf.math.truediv and tf.math.subtract):
 - tf.math.truediv: This layer performs element-wise division of the input by a constant. It's often used for normalization.
 - tf.math.subtract: This layer performs element-wise subtraction of a constant from the input. It is also commonly used in normalization steps.
- Inception V3 Base (inception_v3):
 - Input Shape: (299, 299, 3)
 - Output Shape: (8, 8, 2048)
 - Description: This is the core of the Inception V3 architecture, which is a pre-trained convolutional neural network known for its efficiency and high performance in image recognition tasks. It extracts features from the input image and outputs a feature map of shape (8, 8, 2048).
- Global Average Pooling 2D (global_average_pooling2d):

- Input Shape: (8, 8, 2048)
- Output Shape: (2048)
- Description: This layer reduces each feature map (8x8) to a single number by taking the average of all values in the feature map. This significantly reduces the number of parameters and helps in preventing overfitting.
- Batch Normalization (batch_normalization_94):
 - Input Shape: (2048)
 - Output Shape: (2048)
 - Description: This layer normalizes the output of the previous layer by adjusting and scaling the activations. It helps in speeding up training and improving model stability.
- Dropout (dropout):
 - Input Shape: (2048)
 - Output Shape: (2048)
 - Description: This layer randomly sets a fraction of input units to 0 at each update during training time, which helps prevent overfitting by ensuring that the model doesn't rely too heavily on any one node.
- Dense (pred):
 - Input Shape: (2048)
 - Output Shape: (37)
 - Description: This fully connected layer is the output layer of the model. It maps the 2048 features to the 37 classes (or outputs). The activation function here is likely to be softmax if it is a classification problem, providing the probabilities for each of the 37 classes.

Model Architecture:

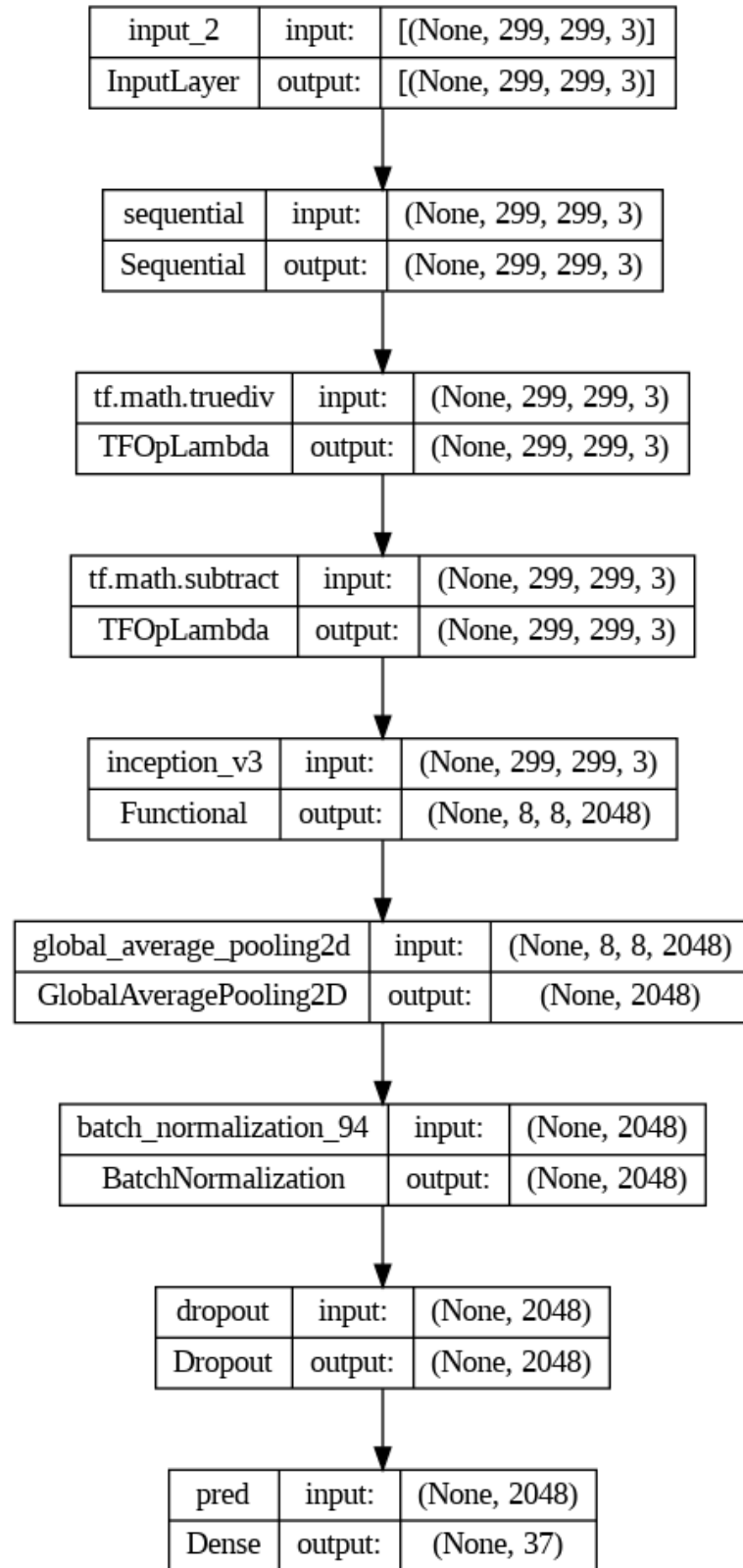


Figure 3.3: Architecture of Inception-V3

This architecture leverages the powerful feature extraction capabilities of the Inception V3 model and adds additional layers for normalization, regularization, and final prediction.

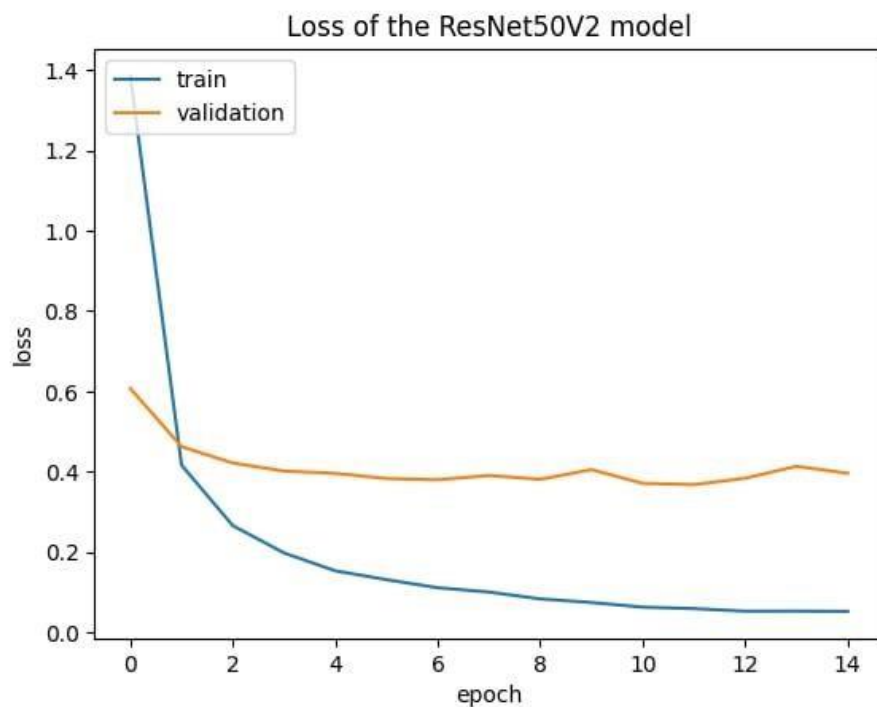
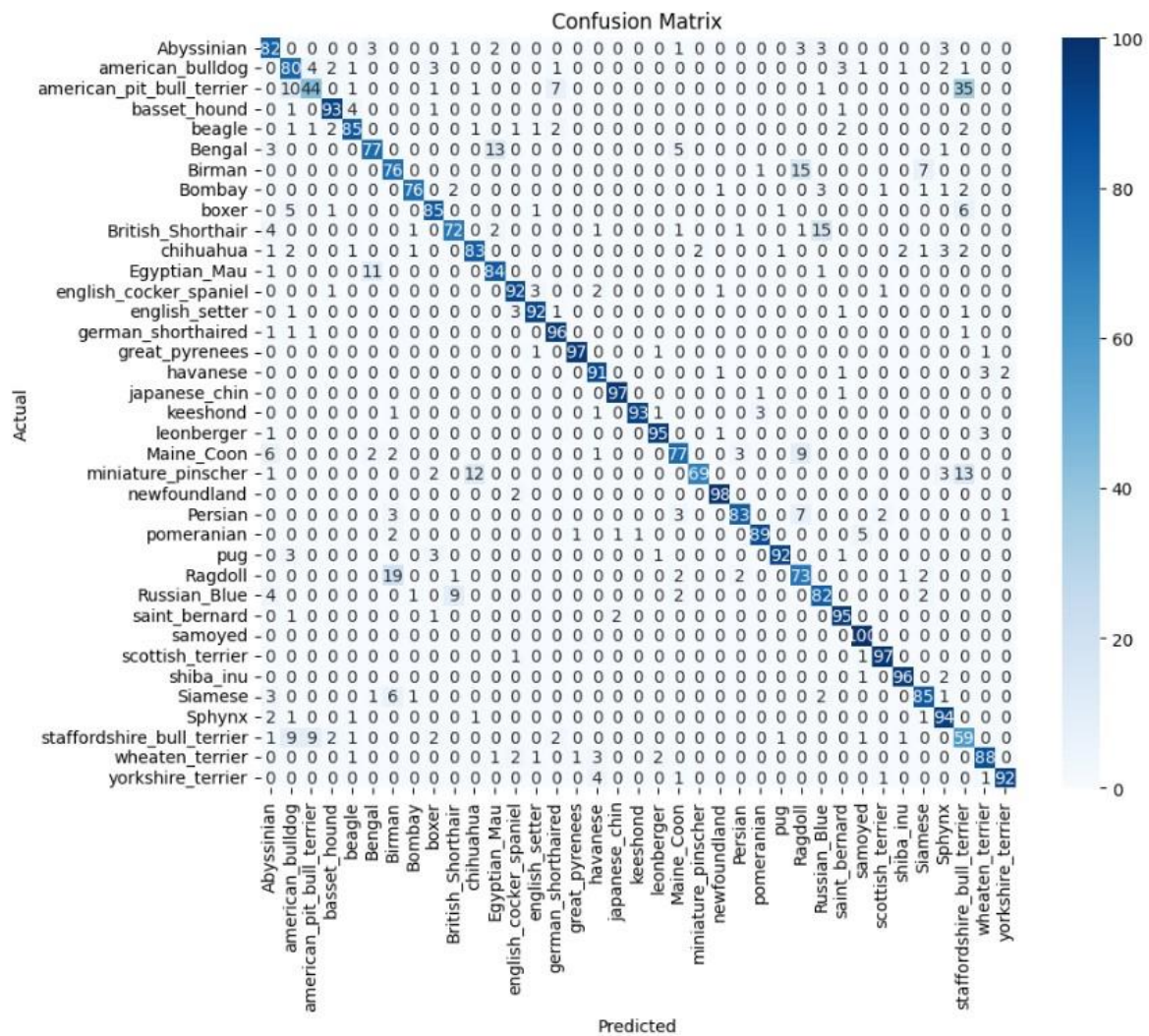
4. Model Evaluation

4.1. Resnet50-V2 Evaluation

The evaluation of the ResNet50_V2 model involved assessing its performance using a range of metrics including accuracy, mean absolute error, root mean squared error, and F1-score, providing comprehensive insights into its predictive capabilities across various tasks and datasets.

Classification Report:				
	precision	recall	f1-score	support
Abyssinian	0.75	0.84	0.79	98
american_bulldog	0.70	0.81	0.75	99
american_pit_bull_terrier	0.75	0.44	0.55	100
basset_hound	0.92	0.93	0.93	100
beagle	0.89	0.87	0.88	98
Bengal	0.82	0.78	0.80	99
Birman	0.70	0.77	0.73	99
Bombay	0.95	0.87	0.91	87
boxer	0.87	0.86	0.86	99
British_Shorthair	0.85	0.73	0.79	98
chihuahua	0.85	0.84	0.84	99
Egyptian_Mau	0.82	0.87	0.84	97
english_cocker_spaniel	0.91	0.92	0.92	100
english_setter	0.93	0.93	0.93	99
german_shorthaired	0.88	0.96	0.92	100
great_pyrenees	0.98	0.97	0.97	100
havanese	0.88	0.93	0.91	98
japanese_chin	0.97	0.98	0.97	99
keeshond	0.99	0.94	0.96	99
leonberger	0.95	0.95	0.95	100
Maine_Coon	0.84	0.77	0.80	100
miniature_pinscher	0.97	0.69	0.81	100
newfoundland	0.96	0.98	0.97	100
Persian	0.93	0.84	0.88	99
pomeranian	0.95	0.90	0.92	99
pug	0.97	0.92	0.94	100
Ragdoll	0.68	0.73	0.70	100
Russian_Blue	0.77	0.82	0.79	100
saint_bernard	0.90	0.96	0.93	99
samoyed	0.92	1.00	0.96	100
scottish_terrier	0.95	0.98	0.97	99
shiba_inu	0.95	0.97	0.96	99
Siamese	0.86	0.86	0.86	99
Sphynx	0.85	0.94	0.90	100
staffordshire_bull_terrier	0.48	0.67	0.56	88
wheaten_terrier	0.92	0.89	0.90	99
yorkshire_terrier	0.97	0.93	0.95	99
accuracy			0.87	3648
macro avg	0.87	0.87	0.87	3648
weighted avg	0.87	0.87	0.87	3648

While the prediction metrics scores exhibit commendable performance, the drawback lies in the model's runtime, suggesting a need for optimization to enhance its efficiency and usability in real-time or resource-constrained environments.

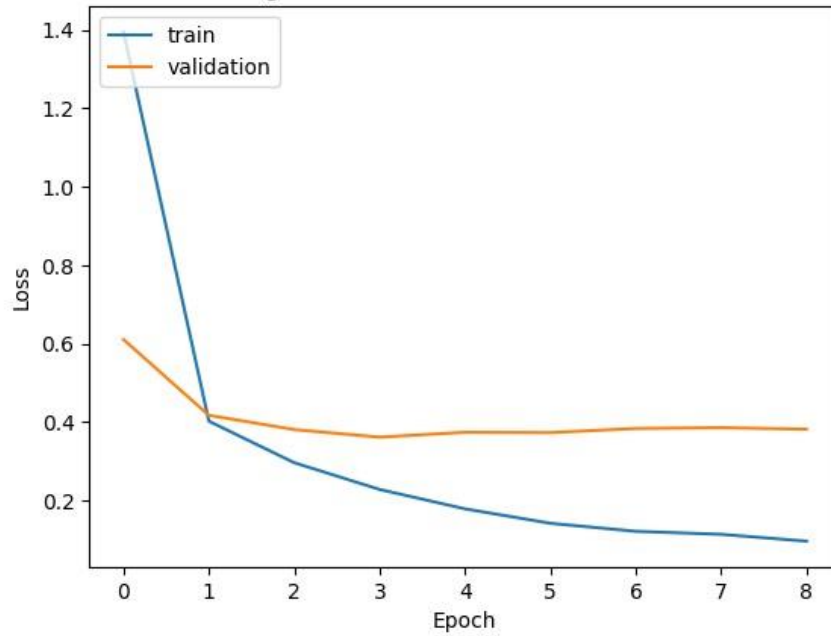


4.2. MobileNet-V2 Evaluation

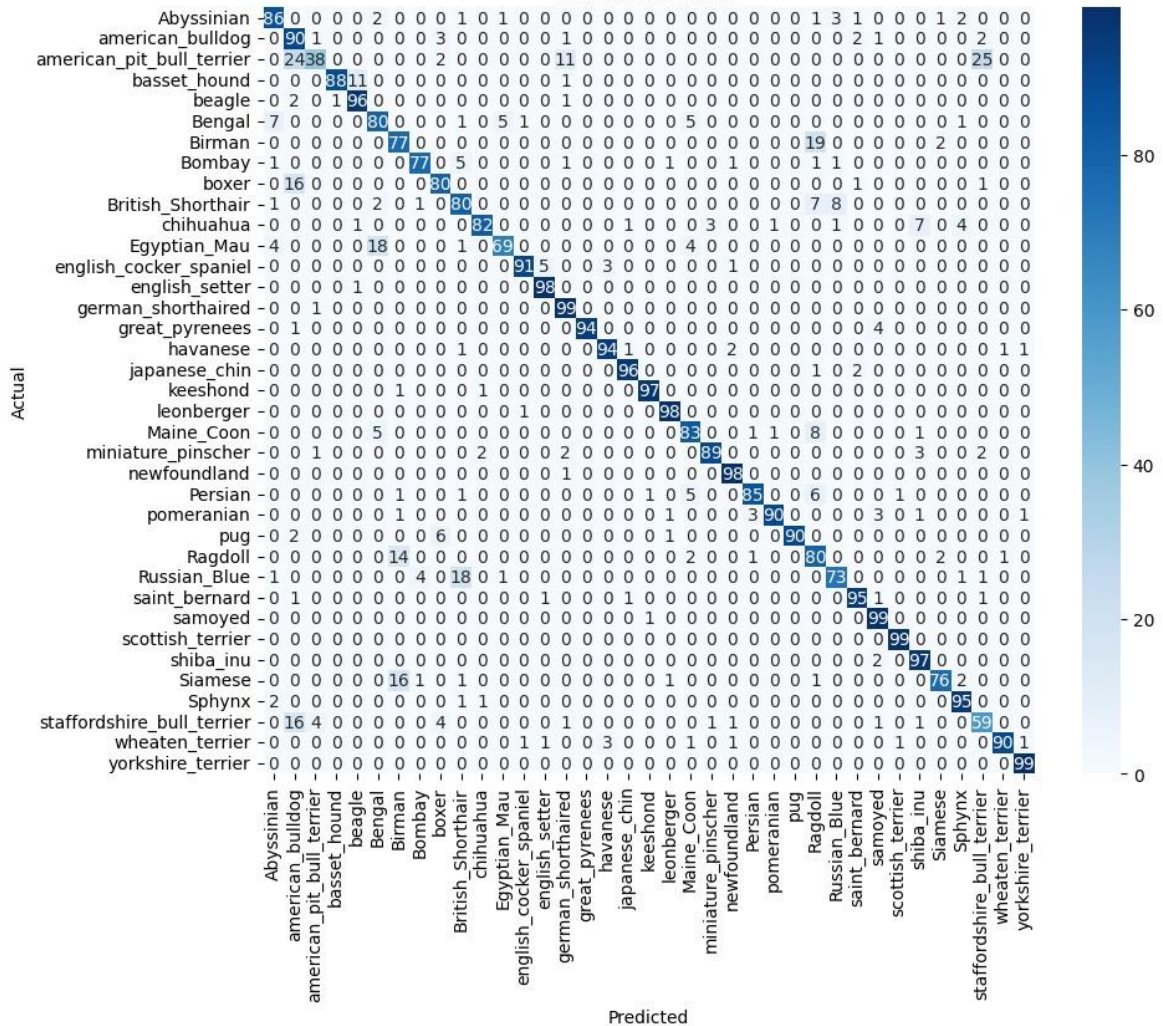
MobileNet, with its streamlined architecture, not only boasts superior performance metrics but also excels in runtime efficiency, making it ideal for deployment on mobile and edge devices. Its remarkable ability to achieve better predictions on test sets underscores its effectiveness in realworld scenarios, offering a compelling solution for tasks requiring rapid inference without compromising accuracy. With MobileNet's balance of performance and efficiency, it emerges as a promising choice for a wide range of applications, from image recognition to object detection in resource-constrained environments.

Classification Report:				
	precision	recall	f1-score	support
Abyssinian	0.84	0.88	0.86	98
american_bulldog	0.59	0.90	0.71	100
american_pit_bull_terrier	0.84	0.38	0.52	100
basset_hound	0.99	0.88	0.93	100
beagle	0.88	0.96	0.92	100
Bengal	0.75	0.80	0.77	100
Birman	0.70	0.79	0.74	98
Bombay	0.93	0.88	0.90	88
boxer	0.84	0.82	0.83	98
British_Shorthair	0.73	0.81	0.77	99
chihuahua	0.95	0.82	0.88	100
Egyptian_Mau	0.91	0.72	0.80	96
english_cocker_spaniel	0.97	0.91	0.94	100
english_setter	0.93	0.99	0.96	99
german_shorthaired	0.84	0.99	0.91	100
great_pyrenees	1.00	0.95	0.97	99
havanese	0.94	0.94	0.94	100
japanese_chin	0.97	0.97	0.97	99
keeshond	0.98	0.98	0.98	99
leonberger	0.96	0.99	0.98	99
Maine_Coon	0.83	0.84	0.83	99
miniature_pinscher	0.96	0.90	0.93	99
newfoundland	0.94	0.99	0.97	99
Persian	0.94	0.85	0.89	100
pomeranian	0.98	0.90	0.94	100
pug	1.00	0.91	0.95	99
Ragdoll	0.65	0.80	0.71	100
Russian_Blue	0.85	0.74	0.79	99
saint_bernard	0.94	0.95	0.95	100
samoyed	0.89	0.99	0.94	100
scottish_terrier	0.98	1.00	0.99	99
shiba_inu	0.88	0.98	0.93	99
Siamese	0.94	0.78	0.85	98
Sphynx	0.90	0.96	0.93	99
staffordshire_bull_terrier	0.65	0.67	0.66	88
wheaten_terrier	0.98	0.91	0.94	99
yorkshire_terrier	0.97	1.00	0.99	99
accuracy			0.88	3648
macro avg	0.89	0.88	0.88	3648
weighted avg	0.89	0.88	0.88	3648

Training and Validation Loss for MobileNet-V2



Confusion Matrix

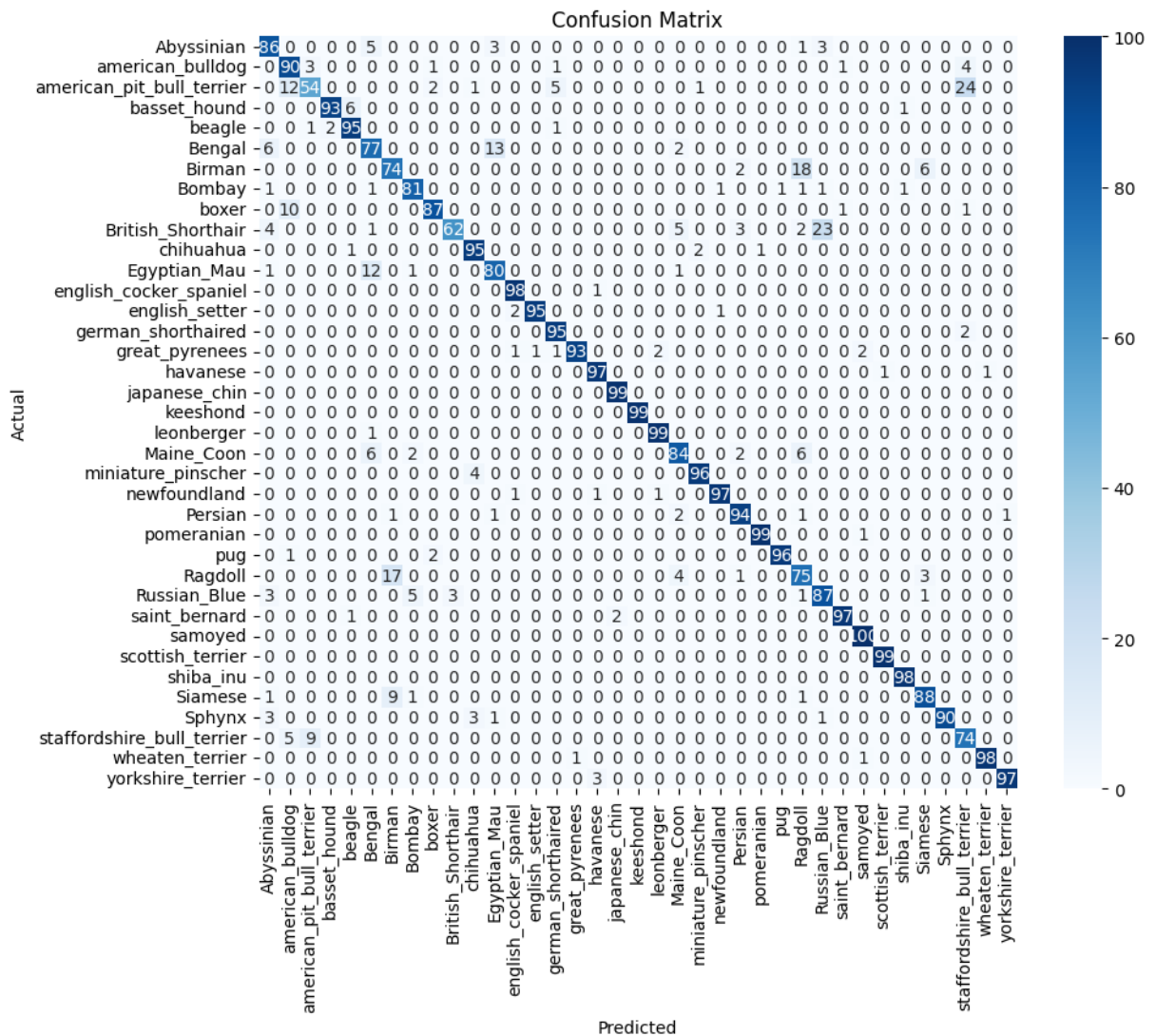
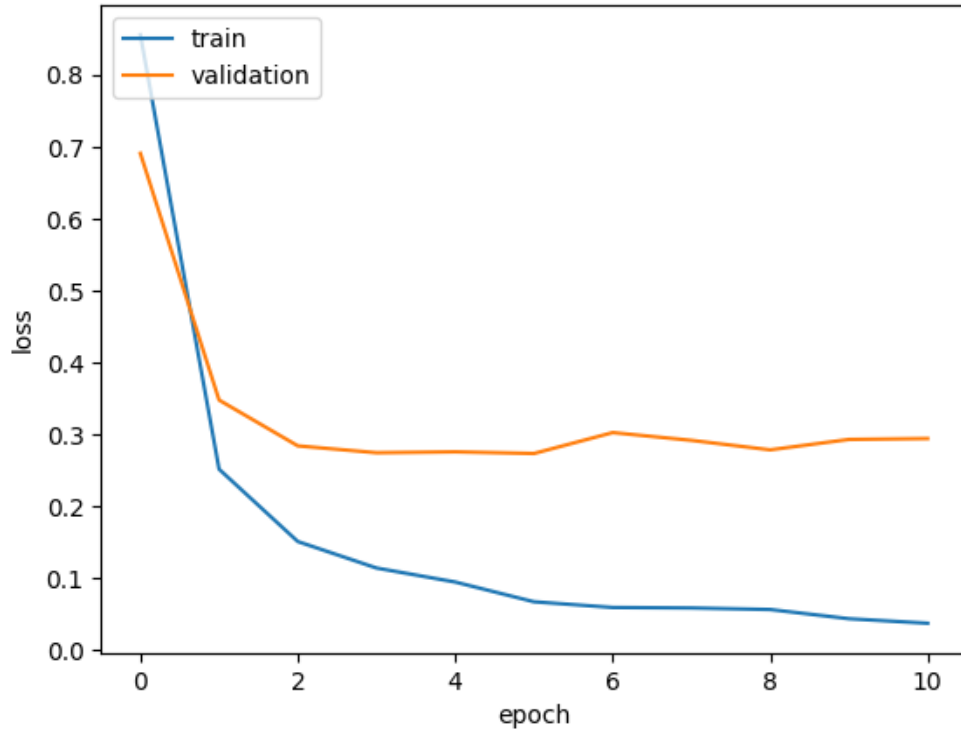


4.3. Inception-V3 Evaluation

Evaluating the Inception V3 model using precision, recall, and F1 score provides a comprehensive understanding of its performance, especially in scenarios with class imbalance. Precision focuses on the accuracy of positive predictions, recall on the completeness of positive predictions, and the F1 score on the balance between the two, offering a single metric that considers both precision and recall.

Classification Report:				
	precision	recall	f1-score	support
Abyssinian	0.82	0.88	0.85	98
american_bulldog	0.76	0.90	0.83	100
american_pit_bull_terrier	0.81	0.55	0.65	99
basset_hound	0.98	0.93	0.95	100
beagle	0.92	0.96	0.94	99
Bengal	0.75	0.79	0.77	98
Birman	0.73	0.74	0.74	100
Bombay	0.90	0.92	0.91	88
boxer	0.95	0.88	0.91	99
British_Shorthair	0.95	0.62	0.75	100
chihuahua	0.92	0.96	0.94	99
Egyptian_Mau	0.82	0.84	0.83	95
english_cocker_spaniel	0.96	0.99	0.98	99
english_setter	0.99	0.97	0.98	98
german_shorthaired	0.92	0.98	0.95	97
great_pyrenees	0.99	0.93	0.96	100
havanese	0.95	0.98	0.97	99
japanese_chin	0.98	1.00	0.99	99
keeshond	1.00	1.00	1.00	99
leonberger	0.97	0.99	0.98	100
Maine_Coon	0.86	0.84	0.85	100
miniature_pinscher	0.97	0.96	0.96	100
newfoundland	0.98	0.97	0.97	100
Persian	0.92	0.94	0.93	100
pomeranian	0.99	0.99	0.99	100
pug	0.99	0.97	0.98	99
Ragdoll	0.71	0.75	0.73	100
Russian_Blue	0.76	0.87	0.81	100
saint_bernard	0.98	0.97	0.97	100
samoyed	0.96	1.00	0.98	100
scottish_terrier	0.99	1.00	0.99	99
shiba_inu	0.98	1.00	0.99	98
Siamese	0.90	0.88	0.89	100
Sphynx	1.00	0.92	0.96	98
staffordshire_bull_terrier	0.70	0.84	0.77	88
wheaten_terrier	0.99	0.98	0.98	100
yorkshire_terrier	0.99	0.97	0.98	100
accuracy			0.91	3648
macro avg	0.91	0.91	0.91	3648
weighted avg	0.91	0.91	0.91	3648

Loss of the Inception V3 model



4.4. Performance comparison between models:

	Accuracy	MAE	RMSE	Runtime
ResNet50_V2	0.8659	2.121	6.884	171s
MobileNet_V2	0.8791	1.723	6.0741	73s
Inception_V3	0.9095	1.395	5.5262	201s

Remarks:

- Accuracy: Inception_V3 outperforms ResNet50_V2, MobileNet_V2 marginally in accuracy.
- MAE and RMSE: Inception_V3 exhibits lowest errors, indicating the best predictive performance. MobileNet_V3 follows up
- Runtime: MobileNet_V2 significantly reduces runtime, enhancing efficiency and suitability for real-time applications. While giving the best predictions, Inception_V3 give the slowest runtime among 3 models

Selection:

Considering the trade-off between performance and runtime, MobileNet_V2 emerges as the preferred choice due to its superior accuracy, lower errors, and significantly reduced runtime, making it well-suited for deployment in scenarios where fast inference is crucial without compromising predictive quality.

5. Conclusion

5.1. Limitation

The project faces several limitations that could impact its overall success and scope. Firstly, the field of multiclass classification for pet breeds is relatively new, and there is limited existing research to draw upon, which may hinder the development of the most effective methodologies. Secondly, the depth of knowledge required to fine-tune complex models like MobileNetV2 or ResNet50V2 can be substantial, necessitating expertise in advanced machine learning techniques and hyperparameter optimization. Finally, time constraints could limit the thoroughness of the exploratory data analysis, model training, and evaluation phases, potentially leading to suboptimal performance or insufficient validation of the model.

5.2. Future Plans

In the future, we plan to enhance our NLP skills to better handle textual data and integrate it with image data for more comprehensive pet classification tasks. Additionally, we aim to fine-tune the current model by experimenting with various hyperparameters and training techniques to improve its accuracy and robustness. Furthermore, we intend to explore more complex architectures, such as combining convolutional neural networks with transformers, to capture more intricate features and relationships within the data. We also would like to provide more data with different classes into the dataset. Therefore, our models can classifcate most of the dogs and cats species.

6. Reference Lists

- [1] [Visual Geometry Group - University of Oxford](#)
- [2] [TensorFlow](#)
- [3] [Keras Applications](#)
- [4] [robots.ox.ac.uk/~vgg/publications/2012/parkhi12a/poster.pdf](#)
- [5] [Resnet Architecture Explained. In their 2015 publication "Deep..." | by Siddhesh Bangar | Medium](#)
- [6] [Review: MobileNetV2 — Light Weight Model \(Image Classification\) | by Sik-Ho Tsang | Towards Data Science](#)
- [7] [Inception V3 CNN Architecture Explained . | by Anas BRITAL | Medium](#)