Department of Computer Science and Engineering

Classification of Cat and Dog Breeds Using Machine Learning Techniques

Submitted By: Radhika Rani
Roll no.-102203775

Course Code- UML501

Supervisor: Dr. Jyoti Maggu

A report submitted in partial fulfilment of the requirements of
Thapar University for the degree of
Bachelor of Engineering in *Computer Engineering*
(August 2024-December 2024)

November 18, 2024

Declaration

I, Radhika Rani, a student of Computer Engineering at Thapar Institute of Engineering and Technology, hereby declare that the work presented in this project report, titled "Classification of Cat and Dog Breeds Using Machine Learning Techniques", is an original piece of research conducted solely by myself, under the guidance of Dr. Jyoti Maggu.

I confirm that this work is my own, and that all sources and references have been duly acknowledged. I affirm that no part of this report has been submitted elsewhere for any other degree, diploma, or qualification. Furthermore, I assure that all data, results, and insights shared in this report are genuine, and any assistance received has been acknowledged in the report

I declare that the data, code, and methods used in this project were selected, designed, and implemented by me. The findings and conclusions expressed in this report reflect my own understanding and analysis, conducted with integrity and adherence to academic standards.

Radhika Rani
November 18, 2024
Thapar Institute of Engineering and Technology (Patiala)

Abstract

The accurate classification of cat and dog breeds is essential for various applications in animal care, particularly kennels and catteries where identifying breeds quickly and efficiently can help provide the necessary care. This project explores the use of deep learning techniques to classify images of cats and dogs into 37 distinct breeds. Using a dataset with over 7,000 images, I employed Convolutional Neural Networks (CNNs) and utilized transfer learning with the pre-trained InceptionV3 model to enhance performance. Data augmentation techniques were also incorporated to improve the model's generalization and accuracy. The final model achieved a classification accuracy exceeding 95%, demonstrating the potential of deep learning models in real-world applications where quick and accurate breed identification is required. This approach not only improves efficiency but also has the potential to reduce the manual effort involved in breed identification, benefiting both animal care facilities and pet owners.

Keywords: breed classification, deep learning, image classification, neural networks, transfer learning, InceptionV3

Acknowledgements

I would like to express my sincere gratitude to Dr. Jyoti Maggu for her invaluable guidance and support throughout this project. Her expertise in machine learning and encouragement were essential to the completion of this work. I would also like to thank my college for providing the resources and environment conducive to research and learning.

Special thanks go to the creators of the Oxford Cats and Dogs Dataset for providing such a comprehensive dataset, which was crucial for the success of our project. I also appreciate the assistance of my peers who offered their insights and feedback during the course of this project.

Finally, I would like to acknowledge the support of my family and friends for their constant encouragement and understanding.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The growth in machine learning and image processing has opened doors for tools that can help with a variety of tasks, including identifying and classifying animal breeds. Knowing a pet's breed can be helpful in veterinary clinics, shelters, and pet care businesses where accurate records are crucial. However, breed identification isn't always easy, especially with images taken in an everyday setting that vary in quality, lighting, or angle. In this project, we focus on building a tool that uses machine learning to help recognize different breeds of cats and dogs. The goal is to simplify the process, save time, and improve accuracy in breed record-keeping.(1)(2)(3)(4)

1.2 Problem statement

The challenge we're tackling is to create an algorithm that can correctly identify the breed of a cat or dog from an image. Current solutions struggle with accuracy, especially when dealing with breeds that look similar or when images are less than perfect. We aim to address these gaps by building a model that's capable of learning from a limited amount of data, while still providing reliable results.

1.3 Aims and Objectives

Aim: To develop a model that can accurately identify the breed of a cat or dog based on images, helping to streamline animal identification for real-world applications.

Objectives: Tasks undertaken to achieve the above aim :

- Preparing images by making them consistent in size, removing background elements, and creating variations of each image to help the model learn better.
- Designing a neural network to handle the task of breed classification.
- Using transfer learning to make the model learn faster and more accurately by building on the knowledge from other pre-trained models.
- Training the model using techniques like 3-Fold Cross-Validation to find the best settings and achieve reliable results.
- Measuring how well the model performs in terms of accuracy and speed.

1.4 Solution approach

This section outlines the general approach used in developing our model. The methodology was designed to handle the classification of cats and dogs across 37 breeds with accuracy and efficiency. The main steps employed are as follows:

1. **Data Preparation:** The images were resized to a standard format, unnecessary background elements were removed to help focus on the animal, and data augmentation techniques were applied. This included rotating and flipping images to make the model robust and less prone to overfitting.

2. **Model Design:** A Convolutional Neural Network (CNN) was chosen for this project, as CNNs are well-suited for image classification along with applying transfer learning , using the InceptionV3 model as a base to use prior knowledge from a large dataset.

3. **Training and Optimization:** The model was trained using 3-Fold Cross-Validation for optimization. Hyperparameters, mainly learning rate and dropout percentage were adjusted to find the best possible configuration for high accuracy.

4. **Evaluation:** The model's performance was evaluated using confusion metrics such as accuracy, precision, and recall.

1.4.1 Data Preprocessing

Data preprocessing included steps to prepare images for the model. All images were resized to ensure uniformity, and data augmentation (such as flipping, rotating, and scaling) was applied to increase diversity in the training data. Also, backgrounds were removed from images to allow model to focus only on the animals.

1.4.2 Model Testing and Comparison

A basic CNN model and a more complex model using transfer learning were tested. Their performance was compared based on speed, accuracy, and computational efficiency, allowing for a clear evaluation of which approach was most effective.

1.5 Summary of contributions and achievements

This project contributes a refined approach to breed identification for cats and dogs, utilizing a combination of transfer learning and data pre-processing. The final model achieves high classification accuracy and robustness, suitable for practical application in animal care facilities. Testing showed that pre-processing steps like background removal and data augmentation can significantly increase model accuracy, making this project valuable for better image classification methods in animal breed identification.

The developed model can serve as a tool in real-world applications where quick and reliable breed identification is required, reducing the need for manual identification and providing an accessible solution for animal shelters, veterinary clinics, and other related sectors.

Chapter 2

Data Analysis

The dataset used in this project is publicly available and can be accessed at Oxford Cats and Dogs Dataset. It consists of 2 parts: images and annotations:

2.1 Images

There are a total of 7384 images of pets, with 37 breeds present (both cats and dogs) which implies that there are 37 different classes in which the model can classify an image. These images are set in a variety of environments and locations, and are very different in style. Most importantly, they are casual pet pictures, which means that this solution can be used in real world applications where people with no experience and with no special equipment can use it. The following figure (Fig. 2.1) presents one image from breed present in the data set. These have been re shaped to be the same size of 299 x 299 pixels.

It is important to guarantee that all classes have roughly the same representation in the

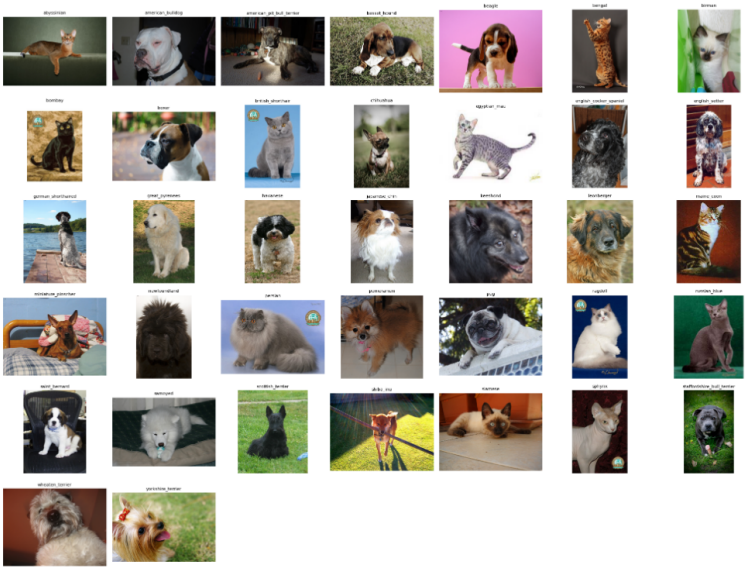

Figure 2.1: One example image of each breed present in the dataset (resized to 299 x 299 pixels)

data set to prevent the model from not learning all breeds equally. Thus it is important to understand the distribution of all classes. Breed-wise information can be seen in Fig. 2.2. For better understanding, I plotted the chart as seen in Fig 2.3.

```
Breed Information:
Breed: abyssinian, Species: cat, Count: 199
Breed: american_bulldog, Species: dog, Count: 200
Breed: american_pit_bull_terrier, Species: dog, Count: 200
Breed: basset_hound, Species: dog, Count: 200
Breed: beagle, Species: dog, Count: 200
Breed: bengal, Species: cat, Count: 200
Breed: birman, Species: cat, Count: 200
Breed: bombay, Species: cat, Count: 200
Breed: boxer, Species: dog, Count: 200
Breed: british_shorthair, Species: cat, Count: 200
Breed: chihuahua, Species: dog, Count: 200
Breed: egyptian_mau, Species: cat, Count: 195
Breed: english_cocker_spaniel, Species: dog, Count: 200
Breed: english_setter, Species: dog, Count: 200
Breed: german_shorthaired, Species: dog, Count: 200
Breed: great_pyrenees, Species: dog, Count: 200
Breed: havanese, Species: dog, Count: 200
Breed: japanese_chin, Species: dog, Count: 200
Breed: keeshond, Species: dog, Count: 200
Breed: leonberger, Species: dog, Count: 200
Breed: maine_coon, Species: cat, Count: 200
Breed: miniature_pinscher, Species: dog, Count: 200
Breed: newfoundland, Species: dog, Count: 200
Breed: persian, Species: cat, Count: 200
Breed: pomeranian, Species: dog, Count: 200
Breed: pug, Species: dog, Count: 200
Breed: ragdoll, Species: cat, Count: 200
Breed: russian_blue, Species: cat, Count: 200
Breed: saint_bernard, Species: dog, Count: 200
Breed: samoyed, Species: dog, Count: 200
Breed: scottish_terrier, Species: dog, Count: 199
Breed: shiba_inu, Species: dog, Count: 200
Breed: siamese, Species: cat, Count: 200
Breed: sphynx, Species: cat, Count: 200
Breed: staffordshire_bull_terrier, Species: dog, Count: 191
Breed: wheaten_terrier, Species: dog, Count: 200
Breed: yorkshire_terrier, Species: dog, Count: 200
```

Figure 2.2: Images distribution over breeds

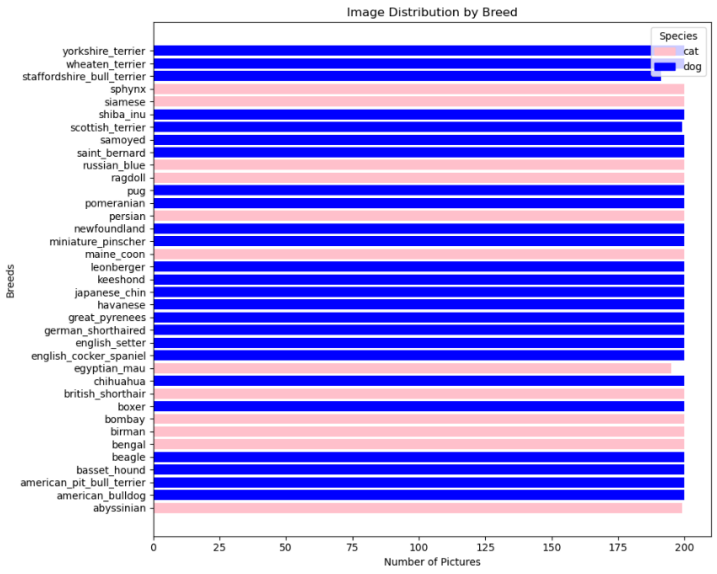

Figure 2.3: Graph representation of distribution

It can be deduced that all breeds have similar representation in the data set. There is around 200 images of pets from each breed. It can also be seen that dogs have almost twice the number of images than cats. However, this won't affect the model as the classifier isn't trying to distinguish between those two classes, but between the 37 different breeds.

2.2 Annotations

The dataset also includes extra information, including:

- **Trimap Annotations:** There is one trimap annotation for each picture indicating the pixels that are foreground and background. Background removal was done using this annotations.
- **Head Bounding Box Annotations:** Annotations in PASCAL VOC Format (not utilized in this project).
- **Other Documents:** Files such as a text file containing information for each image (class ID, species, etc), and two files describing possible train and test image splits.(not used in this project)

2.3 Summary

This chapter outlined the Oxford Cats and Dogs Breeds Classification Dataset, consisting of 7,384 pet images across 37 breeds in diverse settings. Each breed is well-represented, making the dataset ideal for a robust classification model. The dataset also includes trimap annotations for background removal and other documents with metadata, laying the groundwork for accurate breed classification.

Chapter 3

Fundamentals

In this chapter, I explore the foundational concepts that are critical to addressing fine-grained image classification problem.

3.1 General Aspects

This problem is considered one of fine-grained image classification, because it focuses on distinguishing between classes with very similar features. I was aware that with classic methods of machine learning it would be impossible to develop a good solution.

Most documents available online explored the use of convolutional neural networks (CNN), since they are the most viable for problems which involve large quantities of complex data from which features like form and color must be obtained to get accurate results.

In both the models I created, the loss function used is Categorical Cross-Entropy (Softmax loss) which is common for multi-class classification. The optimizer is Adam, an implementation of stochastic gradient descent, used with a default learning rate of 0.001. This parameter is one of the hyper-parameters varied during the hyper-parameter tuning phase. I chose accuracy as the metric to evaluate the model performance during training and tuning.

3.2 Simple Neural Network Architecture

I decided to try a simple architecture, not applying the concept of transfer learning, but that still could be considered deep learning.
It is composed of the following layers (in order):

- **Flatten Layer:** This layer flattens the input of shape (299, 299, 3) into a single-dimensional vector with 268,203 elements, preparing the data for the fully connected layers.
- **Dense Layer:** A fully connected layer with 256 neurons and **ReLU** activation. This activation function maps negative inputs to 0 and leaves positive inputs unchanged, making it suitable for deep learning.
- **Dropout Layer:** A dropout layer with a rate of 0.25, which randomly ignores a fraction of the outputs from the previous layer to prevent overfitting.

- **Batch Normalization Layer:** This layer normalizes the output of the previous layer to have a mean of 0 and a standard deviation of 1, stabilizing and accelerating the training process.
- **Second Dense Layer:** A fully connected layer with 128 neurons and **ReLU** activation. Similar to the first dense layer, it helps learn complex features.
- **Dropout Layer:** Another dropout layer with a rate of 0.35, further regularizing the model and reducing overfitting.
- **Output Dense Layer:** A fully connected layer with 37 neurons, each representing a classification class. A **SoftMax** activation function is applied to output a probability distribution over the classes.

Fig. 3.1 shows an automatically generated diagram of the architecture of this network.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_1 (Flatten)         (None, 268203)            0

 dense_5 (Dense)             (None, 256)               68660224

 dropout_3 (Dropout)         (None, 256)               0

 batch_normalization_96 (Ba  (None, 256)               1024
 tchNormalization)

 dense_6 (Dense)             (None, 128)               32896

 dropout_4 (Dropout)         (None, 128)               0

 dense_7 (Dense)             (None, 37)                4773

=================================================================
```

Figure 3.1: Automatically generated diagram of the architecture of the simple neural network.

3.3 Complex Neural Network Architecture

As previously mentioned, transfer learning plays a significant role in optimizing new models. In this, pre-trained models are used as the foundation, improving performance for the new task. To achieve this, additional layers are added after the frozen layers of the base model, preventing the destruction of the learned features. These new layers focus on training with the new dataset features and making predictions.

The **InceptionV3** architecture, pre-trained on the ImageNet dataset, was chosen as the base model because it has been proven effective for similar problems in multiple studies.

This Neural Network combines the following layers:

- **Input Layer:** The first layer that processes the input image with a shape of $(299, 299, 3)$, ensuring it is compatible with the base model. It preprocesses the image by rescaling pixel values to a range between -1 and 1.
- **InceptionV3 Base Model:** The pre-trained InceptionV3 network is used without its top layers. All its layers are frozen to preserve the pre-trained features.

- **Global Average Pooling Layer:** This layer reduces the dimensions of the feature maps from the base model by averaging their values, resulting in a 1D vector representation.
- **Dense Layer (256 units):** A fully connected layer with 256 neurons and ReLU activation to learn new patterns from the extracted features.
- **Dropout Layer:** Dropout is applied to randomly ignore 35% of the neurons, helping prevent overfitting.
- **Batch Normalization Layer:** Normalizes the output of the previous layer, stabilizing training and speeding up convergence.
- **Output Dense Layer (37 units):** The final layer with 37 neurons, one for each class, using a SoftMax activation function to produce a probability distribution over the classes.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 299, 299, 3)]     0

 tf.math.truediv (TFOpLambd  (None, 299, 299, 3)       0
 a)

 tf.math.subtract (TFOpLamb  (None, 299, 299, 3)       0
 da)

 inception_v3 (Functional)   (None, 8, 8, 2048)        21802784

 global_average_pooling2d (  (None, 2048)              0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 256)               524544

 dropout (Dropout)           (None, 256)               0

 batch_normalization_94 (Ba  (None, 256)               1024
 tchNormalization)

 dense_1 (Dense)             (None, 37)                9509

=================================================================
```

Figure 3.2: Automatically generated diagram of the architecture of the complex neural network.

3.4 Summary

In this chapter, I explained the key concepts and methods used for solving the fine-grained image classification problem. I discussed the use of convolutional neural networks (CNNs) for handling complex image data and described two architectures: a simple neural network and a more advanced model using transfer learning with the InceptionV3 architecture. Important components like pre-processing, dropout, batch normalization, and activation functions (ReLU and SoftMax) were also highlighted.

Chapter 4

PRE-PROCESSING AND HYPER-PARAMETER TUNING

This chapter focuses on image pre-processing and hyper-parameter tuning, two critical steps in optimizing machine learning models. I begin with image pre-processing, which involves reshaping and splitting the dataset to prepare it for model input. Next, I explore hyper-parameter tuning, where I adjust parameters like learning rate and dropout rate to improve model performance. These experiments were conducted using **3-Fold Cross-Validation** to ensure robust results.

4.1 Image Pre-processing

To prepare images for the model, they had to go through a pre-processing phase. This included reshaping them to (299, 299, 3) tensors (recommended for InceptionV3 model input). Changing the color map of the images to black and white, which is a common pre-processing process, was not considered because for this problem the colours of the animals have an importance to distinguish them.
Then, the data was split in 2 parts: train data and test data. The train data corresponds to 70% of the total images, while the test data corresponds to the remaining 30%. Test data was left aside for this section of project.
In order to find the best model configuration I experimented with each one of the models described while varying two hyper-parameters (learning rate and dropout rate). This way I was able to select the best values for the parameters to design the efficient model. The selection was based on the values of accuracy and loss, both for the training and validation data.

4.2 Hyper-parameter Optimization

With the goal of optimizing the model's configuration, both of them went through an iterative process which included varying hyper-parameters, namely the learning rate and the dropout percentage. The method applied was 3-Fold Cross-Validation.

Two values were experimented with for each of the two hyper-parameters. This is not ideal, but justified by the scarcity of time and low computational resources. Then, for both parameters, 3-Fold Cross-Validation was performed, and the history information was saved. This contains data about the progression of the loss function and accuracy over each epoch.

K-Fold Cross-Validation works by dividing the training data in K parts (or folds). Then the model will be trained on K-1 parts, and validated on the remaining part. This is done K times, ensuring that every fold will have an opportunity to have the role of validation. Then an average is made on the results. This solves the problem that the validation data used can give a poor representation of the model performance.
For this project, the data was divided in 3 folds mainly because of low resources and lack of available time, since the most common number of folds seem to be 5 or 10. Each 3-Fold Cross-Validation process took between 30 minutes and 1 hour for simple neural network and about 5 hours (because of limited computation resources available) for complex neural network.

1. **Simple Neural Net:** The images were resized to a standard format, unnecessary background elements were removed to help focus on the animal, and data augmentation techniques were applied to make the model less prone to overfitting.

    - *Learning Rate:* Two values for the learning rate were used: 0.01 and 0.001. Through the analysis of the charts present in Fig 4.1, it can be noted that having a higher learning rate helped reducing the train loss more quickly, even though the model didn't converge in either of the charts for 8 epochs. As the lower learning rate had less discrepancy between the train and validation loss, I decided to use it. None of the settings show good results, though.

    - *Dropout Rate:*Another parameter observed was the dropout value. As we can see in Fig. 4.2, having a higher value lead to a lower slope of the train loss, while the other metrics stayed roughly the same. Even though neither of the two configurations showed good results, the one with the higher dropout rate was the chosen because of less difference between final values of train and validation loss.

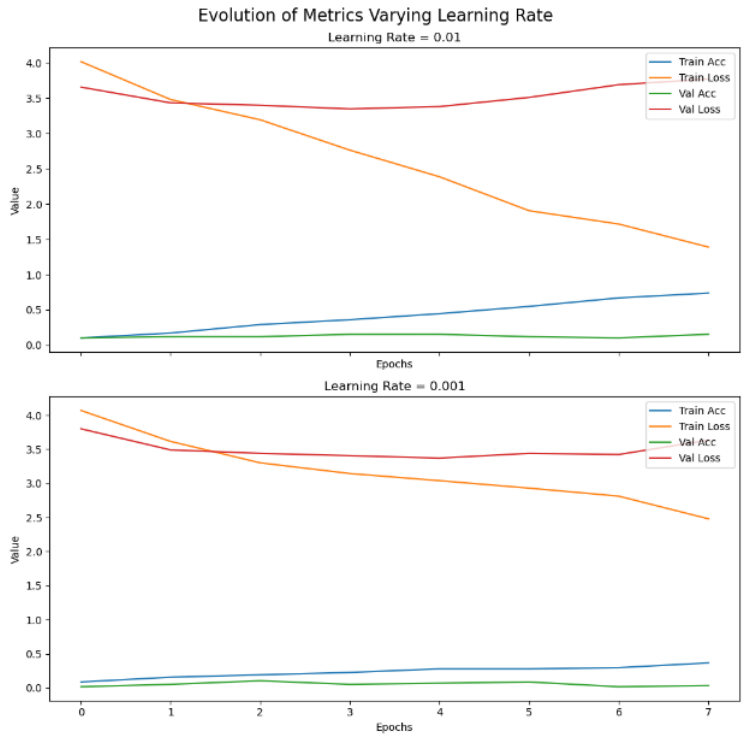

Figure 4.1: For learning rates 0.01 and 0.001(in simple neural network)

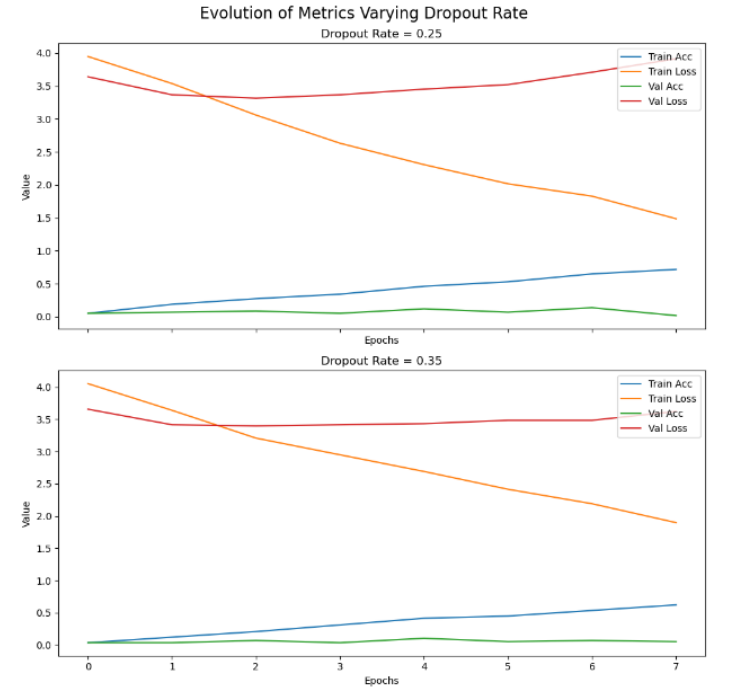

Figure 4.2: For dropout rates 0.25 and 0.35 (in simple neural network)

2. **Complex Neural Network:** For the more complex neural network too, I used the same parameters. I set the number of epochs to 8, and a batch size of 32. These are hyper parameters that should've also been through experimentation to understand the best value. But our resources didn't allow us to perform such tasks (especially for the complex model).

   - *Learning Rate:*The values tested were 0.01 and 0.001. The graphs in Fig. 4.3 show the evolution of the accuracy and loss function, both for the train data and the validation data. As referred above, these are the average of the results from all the iterations in the 3-Fold Cross Validation.

     From the graphs, it is clear how the lower learning rate benefits the model. The accuracy's of both the training and validation show similar curves and the train loss is also similar. The biggest difference is in the validation loss. With the larger learning rate, this curve did not converge to a value, probably because it was "hopping" through the local minimum, due to the larger step. Thus, I concluded that the 0.001 is the best learning rate of all compared.

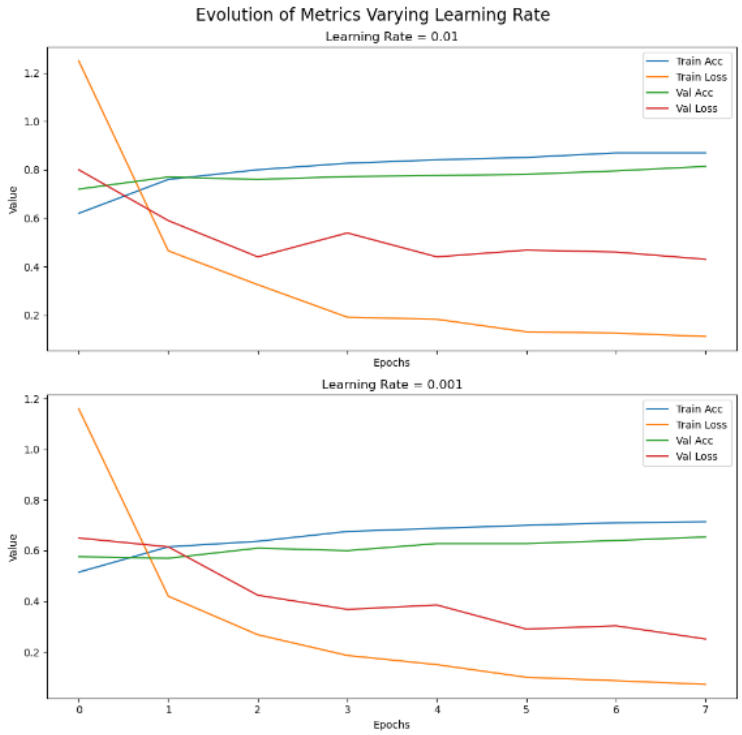

Figure 4.3: For learning rates 0.01 and 0.001 (in complex neural network)

- *Dropout:*The values 0.25 and 0.35 were experimented with. Similar to the learning rate, the charts in Fig. 4.4 show evolution of the metrics along the 8 epochs.

  In this case, the differences are less noticeable. The train accuracy with 0.25 dropout is higher only by a small value in the last epoch, and also has a lower train loss by approximately 0.037. Whereas for the validation data, the accuracy is lower by a small value than with a 0.35 dropout, and has a larger validation loss by 0.025. So, there isn't a clear "winner" here, but it will be assumed that 0.35 is better value because of the slightly better average results in the validation data.

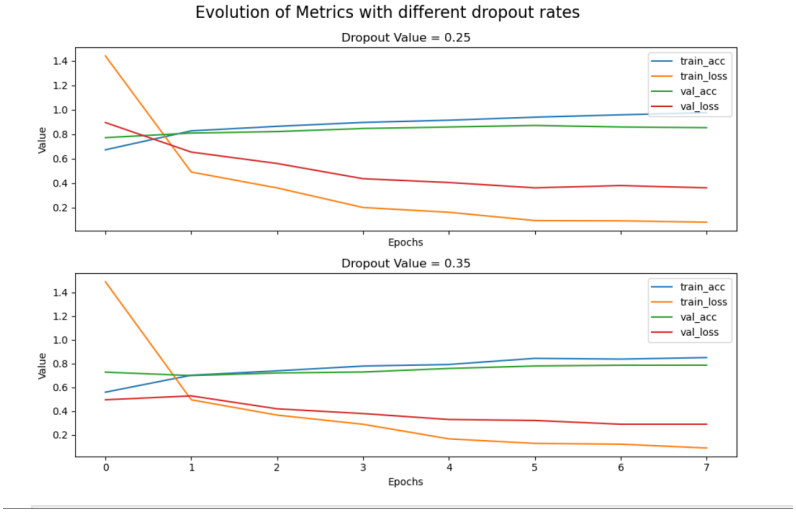

Figure 4.4: For dropout rates 0.25 and 0.35 (in complex neural network)

4.3 Summary

In this chapter, I covered the essential steps of image pre-processing and hyper-parameter tuning to optimize the model. Image pre-processing included reshaping images to fit the InceptionV3 model and splitting the data into training and test sets, ensuring consistent input and retaining crucial color information. Hyper-parameter tuning involved adjusting key settings, such as learning rate and dropout rate, through 3-Fold Cross-Validation to improve accuracy and reduce loss. These steps helped achieve a more balanced model with better generalization.

Chapter 5

MODEL EVALUATION

This chapter evaluates how well the trained models perform by testing them on a designated dataset. The models are assessed using the best hyper-parameters from the previous chapter. Metrics like precision, recall, and F1-score are used, along with visual tools like confusion matrices. The results show how well each model handles new data.

5.1 Methodology

The train set constitutes 30% of the total data set. This corresponds to 2216 images. I used a stratified split, which guarantees that the proportions of number of images in each class is maintained in the test set. The confusion matrix and other metrics presented in this section were made with the model predictions of this test set. At this point the model is already using the most adequate hyper-parameters found in the previous hyper-parameter tuning phase.

5.2 Simple Neural Network Evaluation

In Fig. 5.1 there is a heat map representation of the obtained confusion matrix for the simple neural network model. It can be observed that the model has problems predicting most of the classes. Instead, it seemed to favor few breeds, choosing them most of the time. Other metrics were also calculated and are summarized in the table 5.1.

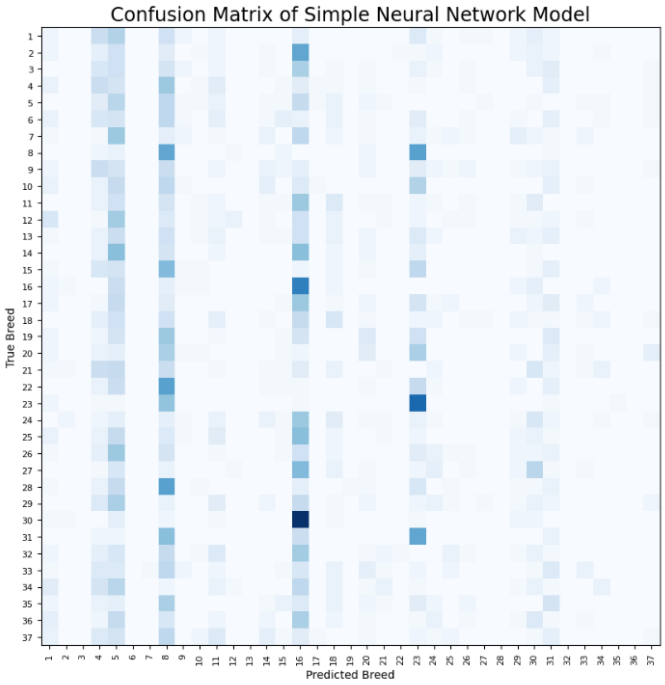

Figure 5.1: Confusion matrix heat map for simple neural network

Table 5.1: Metric Evaluation for Simple Model

Metric	Precision	Recall	F1-score
Macro Average	0.05	0.07	0.06
Weighted Average	0.06	0.066	0.06

5.3 Complex Neural Network Evaluation

Fig. 5.2 is a heat map representation of the obtained confusion matrix. It can be seen that the model had some trouble breed no. 27 (i.e ragdoll cat). It often predicted the breed birman instead of it. When looking at the images shown in Fig 5.3, it can be seen why this happens, since they look really similar to one another.Other metrics were also calculated and are summarized in the table 5.2.

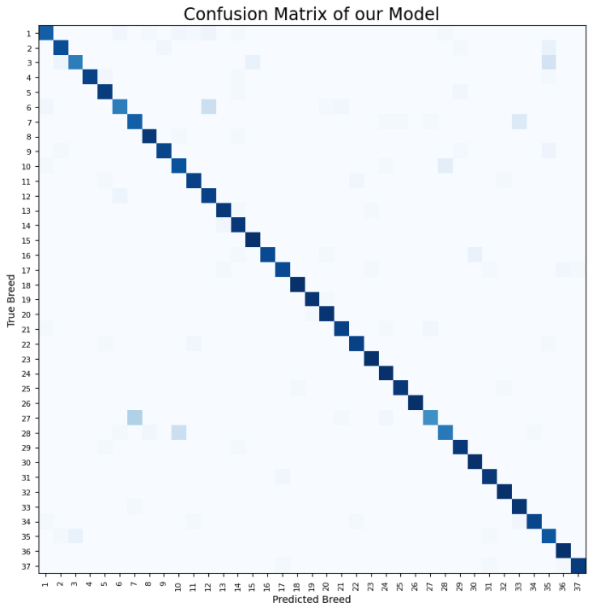

Figure 5.2: Confusion matrix heat map for complex neural network

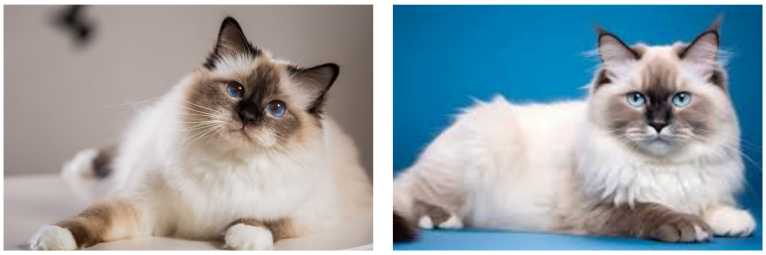

Figure 5.3: Ragdoll Cat (class no. 27) and Birman Cat (class no. 7) look almost identical

Table 5.2: Metric Evaluation for Complex Model

Metric	Precision	Recall	F1-score
Macro Average	0.94	0.95	0.945
Weighted Average	0.96	0.955	0.955

5.4 Comparison

From the results of the two tables (table II and table I), it can be concluded that the use of transfer learning techniques employed in the more complex model lead to better results in all measures when compared to the simple model, even after choosing the best hyper-parameters. The difference between both of them is giant with the first one barely managing to classify

an image correctly (average accuracy = 0.066), and the other doing that almost every time (average accuracy = 0.955).

5.5 Summary

In this chapter, I evaluated the simple and complex neural network models using a stratified test set and key performance metrics. The simple model struggled with classification accuracy, consistently favoring a few specific breeds, as reflected in its confusion matrix and low metric scores. In contrast, the complex model, utilizing transfer learning, performed significantly better across all metrics, demonstrating the effectiveness of this approach. This evaluation confirms that transfer learning substantially improves the model's ability to generalize and accurately classify images.

Chapter 6

Conclusion

In conclusion, after experimenting with two distinct model architectures, tuning hyper-parameters, and utilizing data augmentation techniques, I was able to develop a model that achieved an average accuracy of 95.5% for classifying 37 different categories. This performance exceeds that of several similar studies I reviewed. While the model has proven effective, there are clear areas for improvement. Limited time and computational resources prevented me from testing a wider range of hyper-parameters, such as the number of epochs, batch size, and the optimal layer unfreezing during fine-tuning. Further exploration of advanced data augmentation techniques could also enhance model performance.

References

[1] P. Borwarnginn, W. Kusakunniran, S. Karnjanapreechakorn, and K. Thongkanchorn, "Knowing your dog breed: Identifying a dog breed with deep learning," *International Journal of Automation and Computing*, vol. 18, no. 1, pp. 45–54, Feb 2021. [Online]. Available: https://doi.org/10.1007/s11633-020-1261-0

[2] A. Varshney, A. Katiyar, A. Singh, and S. Chauhan, "Dog breed classification using deep learning," 06 2021, pp. 1–5.

[3] Z. Ráduly, C. Sulyok, Z. Vadászi, and A. Zölde, "Dog breed identification using deep learning," in *2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY)*. IEEE, 2018, pp. 000 271–000 276.

[4] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A survey of deep learning-based object detection," 07 2019.