**FINAL PROJECT REPORT**

1.Introduction

1.1. Project overview

1.2.Objectives

2. Project Initialization and Planning Phase

2.1 Define Problem Statement

2.2 Project proposal

2.3 Initial Project Planning

3. Data Collection and Preprocessing phase

3.1 Data Collection plan and Raw Data sources identified

3.2. Data Quality Report

3.3.Data Processing

4.Model Development Phase

4.1.Model Selection Report

4.2.Initial Model Training Code, Model Validation and Evaluation Report

5.Model Optimization and Tuning Phase

5.1.Tuning Documentation

5.2.Final Model Selection Justification

6.Result

6.1.Output Screenshots

7.Advantages and Disadvantages

8.Conclusion

9.Future Scope

10.Appendix

10.1 Source Code

10.2 Github and Project Demo link

1.INTRODUCTION

**1.1. PROJECT OVERVIEW:**

|  |  |
| --- | --- |
| Objective | To develop a high-accuracy image classification system capable of identifying dog breeds, even with subtle visual differences, using transfer learning. |
| Scope | The system will classify dog breeds from images and be deployable for use in applications such as pet registration, veterinary support, animal shelter systems, and lost pet recovery. |
| **Problem Statement** | | |
| Description | | Accurate breed identification is challenging due to the high degree of visual similarity among breeds. This project utilizes pre-trained convolutional neural networks (e.g., ResNet, EfficientNet) fine-tuned on a curated dataset of labeled dog breed images. The goal is to enable reliable, real-time classification by learning fine-grained visual features. |
| Impact | | **Pet owners:** Simplified registration and breed recognition.  **Veterinarians:** Tailored health guidance based on breed-specific traits.  **Animal shelters:** Faster breed identification for adoption/rescue.  **Lost pet services:** Visual matching of found animals with missing pet databases. |
| **Proposed Solution** | | |
| Approach | | 1. **Dataset Collection & Curation** (from open sources like Kaggle/Stanford Dogs) 2. **Data Augmentation** for better generalization 3. **Transfer Learning** using models like ResNet50 or EfficientNetB0 4. **Fine-tuning** with breed-specific layers 5. **Evaluation** using metrics like accuracy, precision, and recall 6. **Deployment** via a web/mobile interface |
| Key Features | | * High-accuracy classification of over 100 dog breeds * Fine-grained feature recognition (e.g., snout shape, coat texture) * Lightweight, fast inference for mobile/web deployment * Scalable for future addition of new breeds * API support for integration into external applications |
|  |  |

**1.2.OBJECTIVE:**

To develop a high-accuracy image classification system capable of identifying dog breeds, even with subtle visual differences, using transfer learning. It will give High-accuracy classification of over 100 dog breeds, Fine-grained feature recognition (e.g., snout shape, coat texture),Lightweight, fast inference for mobile/web deployment, Scalable for future addition of new breeds and API support for integration into external applications.

2.PROJECT INITIALIZATION AND PLANNING PHASE

**2.1.Define Problem Statement:**

Accurate identification of dog breeds from images presents a significant challenge due to the high visual similarity between many breeds. Traditional image recognition systems often fail to distinguish between breeds with subtle differences in features such as coat colour, size, or facial structure. This leads to misidentification, which can have practical implications for pet owners, veterinarians, animal shelters, and researchers.

To address this, the project proposes a transfer learning-based image classification model trained on a diverse set of dog breed images. By leveraging pre-trained deep learning architectures and fine-tuning them on a curated dataset, the model aims to learn fine-grained visual patterns that differentiate breeds more effectively. This system can serve multiple real-world use cases:

* Pet registration systems for breed documentation.
* Veterinary clinics for breed-specific medical advice.
* Shelters for faster breed identification and rehoming.
* Lost pet identification through visual search apps.

**Visual Similarity confusion between breeds:**

|  |  |  |  |
| --- | --- | --- | --- |
| Image | Actual Breed | Predicted Breed | Issue |
| Labrador Retriever - Wikipedia | Labrador Retriever | Golden Retriever | Similar coat color and size |

**Goal:**

To build an intelligent breed classification system using transfer learning that consistently outperforms traditional methods by accurately identifying dog breeds, even among visually similar categories

**2.2 Project Proposal (Proposed Solution)**

The proposal report aims to build an intelligent breed classification system using transfer learning that consistently outperforms traditional methods by accurately identifying dog breeds, even among visually similar categories.

|  |  |
| --- | --- |
| Approach | 1. **Dataset Collection & Curation** (from open sources like Kaggle/Stanford Dogs) 2. **Data Augmentation** for better generalization 3. **Transfer Learning** using models like ResNet50 or EfficientNetB0 4. **Fine-tuning** with breed-specific layers 5. **Evaluation** using metrics like accuracy, precision, and recall 6. **Deployment** via a web/mobile interface |
| Key Features | * High-accuracy classification of over 100 dog breeds * Fine-grained feature recognition (e.g., snout shape, coat texture) * Lightweight, fast inference for mobile/web deployment * Scalable for future addition of new breeds * API support for integration into external applications |

**2.3 Initial Project Planning:**

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Sprint Task Table

| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Priority | Team Members | Sprint Start Date | Sprint End Date (Planned) |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 | Data Collection and Preprocessing | SL-3 | Understanding & loading data | Low | Dhiraj | 13/06/25 | 14/06/25 |
| Sprint- | Data Collection and Preprocessing | SL-4 | Data cleaning | High | Dhiraj | 13/06/25 | 14/06/25 |
| Sprint-2 | Model Development | SL-8 | Training the model | Medium | Nandita | 14/06/25 | 16/06/25 |
| Sprint-2 | Model tuning and testing | SL-14 | Model testing | Medium | Amrita | 17/06/25 | 17/06/25 |
| Sprint-3 | Web integration and Deployment | SL-16 | Building HTML templates | Low | Aditi | 17/06/25 | 18/06/25 |
| Sprint-3 | Web integration and Deployment | SL-17 | Local deployment | Medium | Aditi | 17/06/25 | 18/06/25 |
| Sprint-4 | Project Report | SL-20 | Report | Medium | Amrita | 18/06/25 | 19/06/25 |

**3. Data Collection and Preprocessing phase**

**3.1 Data Collection plan and Raw Data sources identified**

Elevate your data strategy with the Data Collection plan and the Raw Data Sources report, ensuring meticulous data curation and integrity for informed decision-making in every analysis and decision-making endeavor.

Data Collection Plan :

|  |  |
| --- | --- |
| Section | Description |
| Project Overview | Dog Breed Identification using Transfer Learning" aims to develop a robust machine learning model for accurately classifying dog breeds from images. The project leverages transfer learning, a technique that utilizes pre-trained deep learning models as feature extractors, to overcome the challenges of limited training data and computational resources. By fine-tuning a pre-trained convolutional neural network (CNN) on a dataset of dog images, the model learns to distinguish between different breeds with high accuracy. The resulting system provides a valuable tool for dog breed recognition in various applications, including pet care, veterinary medicine, and animal welfare. |
| Data Collection Plan | The datasets are acquired from Kaggle |

Raw Data Sources :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Source Name | Description | Location/URL | Format | Size | Access Permissions |
| Kaggle Dataset 1 | The training dataset consists of folders corresponding to each class containing image data. | https://www.kaggle.com/competitions/dog-breed-identification/ | Image | 344 MB | Public |
| Label | Contains the label for all the images in the dataset | https://www.kaggle.com/competitions/dog-breed-identification/ | CSV | 471 KB | Public |
| Kaggle Dataset 2 | The testing dataset consists of folders corresponding to each class containing image data. | https://www.kaggle.com/datasets/gpiosenka/70-dog-breedsimage-data-set | Image | 344 MB | Public |
| Dogs | Contains the label for all the images in the dataset | https://www.kaggle.com/datasets/gpiosenka/70-dog-breedsimage-data-set | CSV | 471 KB | Public |

**3.2 Data Quality Report:**

The Data Quality Report Template will summarize data quality issues from the selected source, including severity levels and resolution plans. It will aid in systematically identifying and rectifying data discrepancies.

|  |  |  |  |
| --- | --- | --- | --- |
| Data Source | Data Quality Issue | Severity | Resolution Plan |
| Kaggle Dataset 1 | The testing data was unlabelled. | Moderate | Separate data were used to verify the accuracy of the model |
| Kaggle Dataset 1 | The dataset had more than required classes | Low | The data of required classes were filtered |
| Kaggle Dataset 1 | The data were not classified into folders | Low | The data were filtered into folders using os module functions |
| Kaggle Dataset 2 | The labels were not matching with training dataset | Low | Used string modification functions to normalize the labels |

**3.3. Data Processing:**

The images will be preprocessed by resizing, normalizing, converting color space and batch normalizing. These steps will enhance data quality, promote model generalization, and improve convergence during neural network training, ensuring robust and efficient performance across various computer vision tasks.

|  |  |
| --- | --- |
| Section | Description |
| Data Overview | The dataset contains 10,222 images categorized into 120 classes. For training purposes, only 20 of these classes were selected, comprising a total of 1,683 images. |
| Resizing | The images were resized to 224 × 224 pixels to ensure uniformity. |
| Normalization | Pixel values were normalized to a target range of 0 to 1. |
| Color Space Conversion | The images are taken BGR format |
| Batch Normalization | A batch size of 32 was used during model training. |
| Data Preprocessing Code Screenshots |  |
| Loading Data | A black and white background with lines  AI-generated content may be incorrect. |
| Resizing | A screen shot of a computer program  AI-generated content may be incorrect. |
| Normalization | A screenshot of a computer  AI-generated content may be incorrect. |
| Batch Normalization | A screen shot of a computer code  AI-generated content may be incorrect. |

**4.Model Development Phase**

**4.1.Model Selection Report:**

|  |  |
| --- | --- |
| Model | Description |
| VGG19 | VGG19 is a deep CNN with 19 layers, pre-trained on ImageNet. It performs well for image classification tasks and can capture detailed spatial features. However, due to its large size and high number of parameters, it can be slow to train and prone to overfitting on smaller datasets. |
| MobileNetV2 | MobileNetV2 is a lightweight model designed for mobile and low-resource environments. It uses depth-wise separable convolutions and performs well even with limited data. Its smaller size makes it faster to train and less prone to overfitting, |
| EfficientNetB0 | Balances performance and efficiency using a compound scaling method. Offers better accuracy and lower computational cost compared to VGG19 and MobileNetV2. Suitable for most tasks, especially when both accuracy and speed are important. |

**4.2.Initial Model Training Code, Model Validation and Evaluation Report**

A screenshot of a computer program

AI-generated content may be incorrect.

Model Validation and Evaluation Report (5 marks):

|  |  |  |
| --- | --- | --- |
| Model | Summary | Training and Validation Performance Metrics |
| VGG19 | A screenshot of a computer program  AI-generated content may be incorrect. | A screenshot of a computer program  AI-generated content may be incorrect. |
| MobileNetV2 | A screenshot of a computer program  AI-generated content may be incorrect. | A screen shot of a computer  AI-generated content may be incorrect. |
| EfficientNetB0 | A screenshot of a computer program  AI-generated content may be incorrect. | A screenshot of a computer screen  AI-generated content may be incorrect. |

**5.Model Optimization and Tuning Phase**

**5.1.Tuning Documentation:**

|  |  |
| --- | --- |
| Model | Tuned Hyperparameters |
| VGG19 | A screen shot of a computer screen  AI-generated content may be incorrect.   1. Batch Size = 32 2. Epochs = 50 3. Learning Rate = 0.0001 4. Optimizer: Adam optimizer 5. Loss Function: categorical crossentropy 6. Image Size = (224 × 224). |
| MobileNetV2 | A screen shot of a computer code  AI-generated content may be incorrect.   1. Batch Size = 32 2. Epochs = 10 3. Learning Rate = 0.0001 4. Optimizer: Adam optimizer 5. Loss Function: categorical crossentropy 6. Image Size = (224 × 224). |
| EfficientNetB0 | A screen shot of a computer code  AI-generated content may be incorrect.   1. Batch Size = 32 2. Epochs = 10 3. Learning Rate = 0.0001 4. Optimizer: Adam optimizer 5. Loss Function: categorical crossentropy 6. Image Size = (224 × 224). |

**5.2.Final Model Selection Justification:**

|  |  |
| --- | --- |
| Final Model | Reasoning |
| MobileNetV2 | For this project, MobileNetV2 was chosen as the final model for dog breed classification using transfer learning. This decision was based on a combination of empirical performance and architectural advantages. Among the evaluated models, MobileNetV2 achieved the highest training and testing accuracy, demonstrating strong generalization capabilities even on a relatively small dataset of 1683 images across 20 classes. Its lightweight architecture, which uses depth-wise separable convolutions, allows it to maintain high efficiency while reducing the number of trainable parameters compared to larger models like VGG19. This made it particularly well-suited for our limited data scenario, where larger models tend to overfit. Additionally, MobileNetV2’s pretrained weights on ImageNet allow it to transfer rich feature representations, significantly improving convergence speed and accuracy. The model is also highly compatible with deployment environments due to its low computational footprint, making it an ideal choice for real-time applications on mobile or web platforms. |

**6.Result**

**6.1.Output Screenshots:**

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

**7.Advantages and Disadvantages**

Advantages

1. **High Accuracy on Similar Breeds:**  
   Transfer learning enables the model to recognize fine-grained visual features, improving accuracy even for look-alike breeds.
2. **Reduced Training Time and Data Requirement:**  
   Using pre-trained models drastically cuts down the need for massive datasets and training time.
3. **Scalability:**  
   New breeds can be added to the system by fine-tuning with additional images, making the system adaptable.
4. **Real-World Utility:**  
   Helps in pet registration, lost pet recovery, breed-specific health recommendations, and adoption processes.
5. **Cross-Platform Deployment:**  
   Can be integrated into web/mobile apps, veterinary software, and shelter databases.
6. **Cost-Effective:**  
   Eliminates the need for manual breed identification by experts, reducing time and resources.

DISADVANTAGES

1. **Bias from Pre-trained Models:**  
   If the pre-trained model was trained on biased data (e.g., underrepresented breeds), it may lead to misclassifications.
2. **Dependence on Image Quality:**  
   Poor lighting, angles, or occlusions (e.g., accessories or dirt) can affect the model’s performance.
3. **Limited to Visual Features:**  
   The system can’t account for genetic or behavioral traits that may also define a breed.
4. **Maintenance Overhead:**  
   As new breeds emerge or breed standards evolve, the model will require updates and re-training.
5. **Overfitting Risk:**  
   Without proper data augmentation and regularization, the model may overfit to specific features in the training dataset.
6. **Computation Requirements:**  
   Training and even inference may need powerful hardware, especially for high-resolution images or real-time use.

**8.Conclusion**

The proposed dog breed identification system leverages transfer learning to accurately classify dog breeds from images, addressing the limitations of traditional image recognition methods. By fine-tuning pre-trained deep learning models on a curated dataset, the system can detect subtle visual distinctions between breeds, offering a practical solution for pet owners, veterinarians, shelters, and animal welfare applications. While challenges such as image quality and model bias exist, the advantages in terms of accuracy, efficiency, and real-world impact make this a valuable and scalable tool. With proper data management and model updates, the system holds strong potential for broad adoption and continued improvement.

**9.Future Scope**

1. **Real-Time Mobile Applications:**  
   Deploy the model in lightweight mobile apps for instant breed recognition through smartphone cameras.
2. **Mixed Breed Detection:**  
   Enhance the model to identify and estimate breed composition in mixed-breed dogs using ensemble or hybrid models.
3. **Integration with Veterinary Systems:**  
   Link with vet databases to provide instant breed-specific health tips and vaccination schedules.
4. **Lost & Found Pet Platforms:**  
   Implement visual matching systems in pet recovery platforms to compare found pet images with missing pet reports.
5. **Global Breed Expansion:**  
   Expand the dataset to include lesser-known and region-specific dog breeds for broader international application.
6. **Explainable AI Features:**  
   Add interpretability to the model using Grad-CAM or similar methods to highlight the features that influenced breed predictions.
7. **Voice and Behavior Analysis Integration:**  
   Combine visual identification with audio cues or behavior data for a more holistic breed assessment.
8. **Cloud-Based API Services:**  
   Offer the model as a cloud API that can be integrated into third-party apps and services like pet adoption platforms and smart pet cameras.

**10.Appendix**

**10.1 Source Code**

Data collection and Preprocessing[¶](file:///C:\Users\NANDITA\AppData\Local\Microsoft\Windows\INetCache\IE\6R0L6K02\model_creation%5b1%5d.docx#Data-collection-and-Preprocessing)

In [ ]:

!rm -rf /content/subset

In [ ]:

!pip install -q kaggle

In [ ]:

!mkdir ~/.kaggle

In [ ]:

!cp kaggle.json ~/.kaggle

In [ ]:

!kaggle competitions download -c dog-breed-identification

Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /root/.kaggle/kaggle.json'  
Downloading dog-breed-identification.zip to /content  
 96% 665M/691M [00:01<00:00, 262MB/s]  
100% 691M/691M [00:01<00:00, 379MB/s]

In [ ]:

!unzip /content/dog-breed-identification.zip

Streaming output truncated to the last 5000 lines.  
 inflating: train/83bc62b0fffa99a9c94ba0b67a5f7395.jpg   
 inflating: train/83bcff6b55ee179a7c123fa6103c377a.jpg   
 inflating: train/83be6d622ab74a5e7e08b53eb8fd566a.jpg   
 inflating: train/83c2d7419b0429b9fe953bc1b6cddbec.jpg   
 inflating: train/83cf7d7cd2a759a93e2ffd95bea9c6fb.jpg   
 inflating: train/83d405858f0931722ef21e8ac0adee4d.jpg   
 inflating: train/83d4125a4c3c7dc5956563276cb1cd74.jpg   
 inflating: train/83f0bb565b2186dbcc6a9d009cb26ff2.jpg   
 inflating: train/83fad0718581a696132c96c166472627.jpg   
 inflating: train/83fbbcc9a612e3f712b1ba199da61f20.jpg   
 inflating: train/8403d8936430c2f05ab7d74d23c2c0cb.jpg   
 inflating: train/8406d837b2d7fac1c3cd621abb4c4f9e.jpg   
 inflating: train/840b67d26e5e43f8eb6430f62d4ba1ac.jpg   
 inflating: train/840db91ba4600148f3dcb06ec419b421.jpg   
 inflating: train/840dbad5a691c22611d85b2488bf4cbb.jpg   
   
 inflating: train/ffe2ca6c940cddfee68fa3cc6c63213f.jpg   
 inflating: train/ffe5f6d8e2bff356e9482a80a6e29aac.jpg   
 inflating: train/fff43b07992508bc822f33d8ffd902ae.jpg

In [ ]:

import os  
import shutil  
import sys  
import pandas as pd

In [ ]:

dataset\_dir = '/content/train'  
test\_data\_dir='/content/test'  
labels = pd.read\_csv('/content/labels.csv')

In [ ]:

def make\_dir(x):  
 if os.path.exists(x) == False:  
 os.makedirs (x)  
  
base\_dir = './subset'  
make\_dir (base\_dir)

In [ ]:

train\_dir = os.path. join(base\_dir, 'train')  
test\_dir=os.path.join(base\_dir,'test')  
make\_dir(train\_dir)  
make\_dir(test\_dir)

In [ ]:

selected\_classes = [ 'affenpinscher', 'beagle', 'appenzeller', 'basset', 'bluetick', 'boxer', 'cairn', 'doberman', 'german\_shepherd', 'golden\_retriever', 'kelpie', 'komondor', 'leonberg',  
'mexican\_hairless', 'pug', 'redbone', 'shih-tzu', 'toy\_poodle', 'vizsla', 'whippet']  
len(selected\_classes)

Out[ ]:

20

In [ ]:

breeds = labels.breed. unique()  
for breed in selected\_classes:  
 # Make folder for each breed  
 \_= os.path.join(train\_dir, breed)  
 make\_dir(\_)  
  
 # Copy images to the corresponding folders  
 images = labels [labels.breed == breed] ['id' ]  
 for image in images:  
 source = os.path.join(dataset\_dir, f'{image}.jpg')  
 destination = os.path.join(train\_dir, breed, f'{image}.jpg')  
 shutil.copyfile(source, destination)

In [ ]:

#import image datagenerator library  
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
train\_datagen=ImageDataGenerator(rescale=1./255,shear\_range=0.2,zoom\_range=0.2,horizontal\_flip=True)

In [ ]:

# datagen = ImageDataGenerator ()  
generator = train\_datagen.flow\_from\_directory(  
 '/content/subset/train',  
 target\_size=(224, 224), # Adjust target size as needed  
 batch\_size=32,  
 class\_mode='categorical',  
 shuffle=False, # Ensure order is maintained for class indices  
 classes=selected\_classes # Specify the selected classes  
)  
# test\_datagen=ImageDataGenerator(rescale=1./255)  
# test\_generator = test\_datagen.flow\_from\_directory(  
# '/content/subset/test',  
# target\_size=(224, 224), # Adjust target size as needed  
# batch\_size=32,  
# class\_mode='categorical',  
# shuffle=False, # Ensure order is maintained for class indices  
# classes=selected\_classes # Specify the selected classes  
# )

Found 1683 images belonging to 20 classes.  
Found 0 images belonging to 20 classes.

Model Building[¶](file:///C:\Users\NANDITA\AppData\Local\Microsoft\Windows\INetCache\IE\6R0L6K02\model_creation%5b1%5d.docx#Model-Building)

MobileNetV2[¶](file:///C:\Users\NANDITA\AppData\Local\Microsoft\Windows\INetCache\IE\6R0L6K02\model_creation%5b1%5d.docx#MobileNetV2)

In [ ]:

import tensorflow as tf  
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
from tensorflow.keras.applications import MobileNetV2  
from tensorflow.keras.applications.mobilenet\_v2 import preprocess\_input  
from tensorflow.keras.layers import GlobalAveragePooling2D, Dropout, Dense  
from tensorflow.keras.models import Model  
from tensorflow.keras.callbacks import EarlyStopping

In [ ]:

from tensorflow. keras. layers import Dense, Flatten  
from tensorflow. keras.models import Model  
from tensorflow.keras.optimizers import Adam

In [ ]:

Image\_size=[224,224]

In [ ]:

base\_model=MobileNetV2(input\_shape=Image\_size + [3] , weights='imagenet' , include\_top = False)

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet\_v2/mobilenet\_v2\_weights\_tf\_dim\_ordering\_tf\_kernels\_1.0\_224\_no\_top.h5  
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step

In [ ]:

for i in base\_model.layers:  
 i.trainable = False

In [ ]:

y=Flatten()(base\_model.output)

In [ ]:

output=Dense(20,activation='softmax')(y)

In [ ]:

mn=Model(inputs=base\_model.input,outputs=output)

In [ ]:

mn.summary()

Model: "functional"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┓  
┃ Layer (type) ┃ Output Shape ┃ Param # ┃ Connected to ┃  
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━┩  
│ input\_layer (InputLayer) │ (None, 224, 224, 3) │ 0 │ - │  
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤  
│ Conv1 (Conv2D) │ (None, 112, 112, 32) │ 864 │ input\_layer[0][0] │  
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤  
│ bn\_Conv1 │ (None, 112, 112, 32) │ 128 │ Conv1[0][0] │  
│ (BatchNormalization) │ │ │ │  
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤  
│ Conv1\_relu (ReLU) │ (None, 112, 112, 32) │ 0 │ bn\_Conv1[0][0] │  
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤  
│ expanded\_conv\_depthwise │ (None, 112, 112, 32) │ 288 │ Conv1\_relu[0][0] │  
│ (DepthwiseConv2D) │ │ │ │  
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤  
│ expanded\_conv\_depthwise\_… │ (None, 112, 112, 32) │ 128 │ expanded\_conv\_depthwi… │  
│ (BatchNormalization) │ │ │ │  
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤  
│ expanded\_conv\_depthwise\_… │ (None, 112, 112, 32) │ 0 │ expanded\_conv\_depthwi… │  
│ (ReLU) │ │ │ │  
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤  
│ expanded\_conv\_project │ (None, 112, 112, 16) │ 512 │ expanded\_conv\_depthwi… │  
│ (Conv2D) │ │ │ │  
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤  
│ expanded\_conv\_project\_BN │ (None, 112, 112, 16) │ 64 │ expanded\_conv\_project… │  
│ (BatchNormalization) │ │ │ │  
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤  
│ block\_1\_expand (Conv2D) │ (None, 112, 112, 96) │ 1,536 │   
│ flatten (Flatten) │ (None, 62720) │ 0 │ out\_relu[0][0] │  
├───────────────────────────┼────────────────────────┼────────────────┼────────────────────────┤  
│ dense (Dense) │ (None, 20) │ 1,254,420 │ flatten[0][0] │  
└───────────────────────────┴────────────────────────┴────────────────┴────────────────────────┘

Total params: 3,512,404 (13.40 MB)

Trainable params: 1,254,420 (4.79 MB)

Non-trainable params: 2,257,984 (8.61 MB)

In [ ]:

from keras.callbacks import EarlyStopping  
from keras.optimizers import Adam  
opt = Adam(learning\_rate=0.0001)  
  
# Define Early Stopping callback  
early\_stopping = EarlyStopping(monitor='accuracy', patience=3, restore\_best\_weights=True)  
  
# Compile the model (you may have already done this)  
mn.compile(optimizer=opt , loss='categorical\_crossentropy', metrics=['accuracy'])  
  
# Train the model with early stopping callback  
history = mn.fit(train\_generator, epochs=10,callbacks=[early\_stopping])

/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data\_adapters/py\_dataset\_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().\_\_init\_\_(\*\*kwargs)` in its constructor. `\*\*kwargs` can include `workers`, `use\_multiprocessing`, `max\_queue\_size`. Do not pass these arguments to `fit()`, as they will be ignored.  
 self.\_warn\_if\_super\_not\_called()

Epoch 1/10  
53/53 ━━━━━━━━━━━━━━━━━━━━ 37s 487ms/step - accuracy: 0.4396 - loss: 2.2295  
Epoch 2/10  
53/53 ━━━━━━━━━━━━━━━━━━━━ 23s 427ms/step - accuracy: 0.8593 - loss: 0.4512  
Epoch 3/10  
53/53 ━━━━━━━━━━━━━━━━━━━━ 23s 430ms/step - accuracy: 0.8943 - loss: 0.3610  
Epoch 4/10  
53/53 ━━━━━━━━━━━━━━━━━━━━ 22s 409ms/step - accuracy: 0.9083 - loss: 0.2969  
Epoch 5/10  
53/53 ━━━━━━━━━━━━━━━━━━━━ 22s 417ms/step - accuracy: 0.9211 - loss: 0.2452  
Epoch 6/10  
53/53 ━━━━━━━━━━━━━━━━━━━━ 22s 424ms/step - accuracy: 0.9331 - loss: 0.1990  
Epoch 7/10  
53/53 ━━━━━━━━━━━━━━━━━━━━ 23s 429ms/step - accuracy: 0.9091 - loss: 0.2969  
Epoch 8/10  
53/53 ━━━━━━━━━━━━━━━━━━━━ 21s 401ms/step - accuracy: 0.9403 - loss: 0.2139  
Epoch 9/10  
53/53 ━━━━━━━━━━━━━━━━━━━━ 42s 426ms/step - accuracy: 0.9530 - loss: 0.1671  
Epoch 10/10  
53/53 ━━━━━━━━━━━━━━━━━━━━ 22s 413ms/step - accuracy: 0.9558 - loss: 0.1240

In [ ]:

mn.save("mn.h5")

Testing[¶](file:///C:\Users\NANDITA\AppData\Local\Microsoft\Windows\INetCache\IE\6R0L6K02\model_creation%5b1%5d.docx#Testing)

In [ ]:

!pip install -q kaggle

In [ ]:

!mkdir ~/.kaggle

In [ ]:

!cp kaggle.json ~/.kaggle

In [ ]:

!kaggle datasets download gpiosenka/70-dog-breedsimage-data-set

Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /root/.kaggle/kaggle.json'  
Dataset URL: https://www.kaggle.com/datasets/gpiosenka/70-dog-breedsimage-data-set  
License(s): CC0-1.0  
Downloading 70-dog-breedsimage-data-set.zip to /content  
 81% 174M/215M [00:00<00:00, 656MB/s]   
100% 215M/215M [00:00<00:00, 542MB/s]

In [ ]:

!unzip /content/70-dog-breedsimage-data-set.zip

Streaming output truncated to the last 5000 lines.  
 inflating: train/Dhole/085.jpg   
 inflating: train/Dhole/086.jpg   
 inflating: train/Dhole/087.jpg   
 inflating: train/Dhole/088.jpg   
 inflating: train/Dhole/089.jpg   
 inflating: train/Dhole/090.jpg   
 inflating: train/Dhole/091.jpg   
 inflating: train/Dhole/092.jpg   
 inflating: train/Dhole/093.jpg   
 inflating: train/Dhole/094.jpg   
 inflating: train/Dhole/095.jpg   
 inflating: train/Dhole/096.jpg   
 inflating: train/Dhole/097.jpg   
 inflating: train/Dhole/098.jpg   
 inflating: train/Dhole/099.jpg   
 inflating: train/Dhole/100.jpg   
 inflating: train/Dhole/101.jpg   
 inflating: train/Dhole/102.jpg   
 inflating:   
 inflating: valid/Yorkie/09.jpg   
 inflating: valid/Yorkie/10.jpg

In [ ]:

import os  
import shutil  
import sys  
import pandas as pd

In [ ]:

!rm -rf /content/filtered\_test

In [ ]:

import os  
import shutil  
  
# Path to the original test folder  
source\_dir = '/content/test'  
  
# Path where the filtered classes will be copied  
target\_dir = '/content/filtered\_test'  
  
# List of selected class names to keep  
selected\_classes = [ 'affenpinscher', 'beagle', 'appenzeller', 'basset', 'bluetick', 'boxer', 'cairn', 'doberman', 'german\_shepherd', 'golden\_retriever', 'kelpie', 'komondor', 'leonberg',  
'mexican\_hairless', 'pug', 'redbone', 'shih-tzu', 'toy\_poodle', 'vizsla', 'whippet']  
  
# Create target directory if it doesn't exist  
os.makedirs(target\_dir, exist\_ok=True)  
  
for class\_name in selected\_classes:  
 print("Name:",class\_name.replace('\_',' ').title())  
 src\_class\_path = os.path.join(source\_dir, class\_name.replace('\_',' ').title())  
 dst\_class\_path = os.path.join(target\_dir, class\_name)  
  
 # Check if the class folder exists in source  
 if os.path.exists(src\_class\_path):  
 shutil.copytree(src\_class\_path, dst\_class\_path)  
 # print(f"Copied: {class\_name}")  
 else:  
 os.makedirs(dst\_class\_path, exist\_ok=True)  
 # print(f"Created empty folder: {class\_name}")

Testing MobileNetV5[¶](file:///C:\Users\NANDITA\AppData\Local\Microsoft\Windows\INetCache\IE\6R0L6K02\model_creation%5b1%5d.docx#Testing-MobileNetV5)

In [ ]:

import tensorflow as tf  
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
from tensorflow.keras.models import load\_model  
  
# Load the trained model  
model = load\_model('mn.h5')  
  
# Image size and batch settings  
image\_size = (224, 224)  
batch\_size = 32  
  
#import image datagenerator library  
from tensorflow.keras.applications.mobilenet\_v2 import preprocess\_input  
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
datagen=ImageDataGenerator(  
 preprocessing\_function=preprocess\_input,  
 rotation\_range=30,  
 zoom\_range=0.2,  
 width\_shift\_range=0.2,  
 height\_shift\_range=0.2,  
 shear\_range=0.2,  
 horizontal\_flip=True,  
 fill\_mode='nearest',  
 validation\_split=0.2  
)  
  
# Testing data  
test\_generator = datagen.flow\_from\_directory(  
 '/content/filtered\_test',  
 target\_size=(224, 224),  
 batch\_size=32,  
 class\_mode='categorical',  
 shuffle=True  
)  
# Evaluate the model  
loss, accuracy = model.evaluate(test\_generator)  
  
print(f"Test Loss: {loss:.4f}")  
print(f"Test Accuracy: {accuracy:.4f}")

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile\_metrics` will be empty until you train or evaluate the model.

4/4 ━━━━━━━━━━━━━━━━━━━━ 7s 677ms/step - accuracy: 0.9797 - loss: 0.0551  
Test Loss: 0.0815  
Test Accuracy: 0.9727

**10.2 Github and Project Demo link:**

**Github:**

[**https://github.com/dhirajskitchen/Dog-Breed-Identification-using-Transfer-Learning**](https://github.com/dhirajskitchen/Dog-Breed-Identification-using-Transfer-Learning)

**Project Demo Link:**

[**https://drive.google.com/file/d/1FMe\_he8XyxUlTZLFeM\_gNfz7G\_cw8atH/view**](https://drive.google.com/file/d/1FMe_he8XyxUlTZLFeM_gNfz7G_cw8atH/view)