Final Project Report Template

1. Introduction

1.1 Project overviews

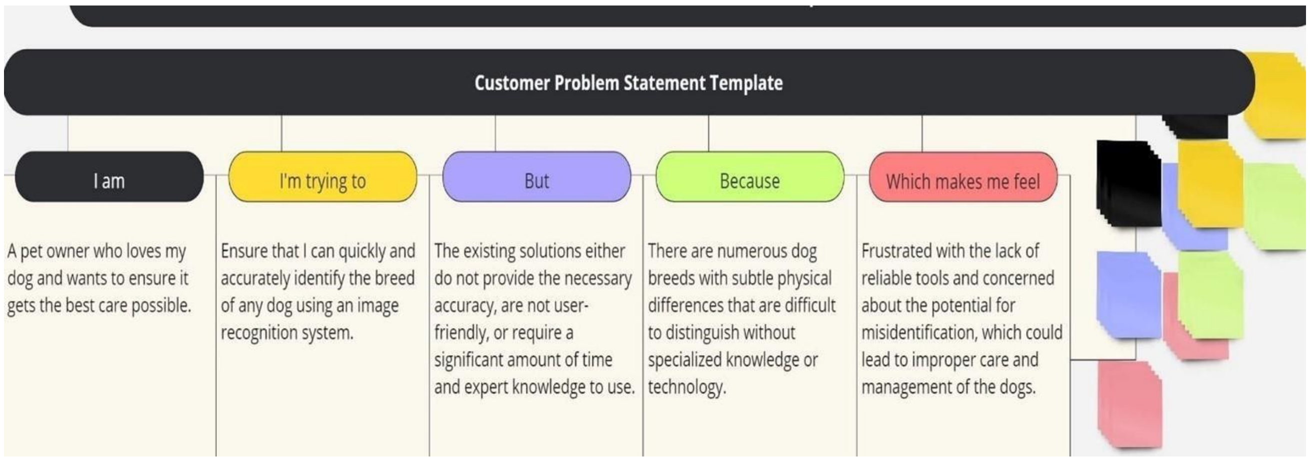
Develop a machine learning model to classify dog breeds from images using transfer learning. The project leverages pre-trained deep learning models to handle limited data and computational resources, aiming to assist in pet adoption, lost dog identification, and veterinary support.

1.2 Objectives

- Develop a Model: Accurately identify dog breeds from images.
- Transfer Learning: Use pre-trained deep learning models for enhanced accuracy.
- Data Preprocessing: Improve image quality and consistency.
- Model Training: Train and fine-tune the model for high accuracy.
- Web Application: Create a user-friendly interface with Flask.
- Applications: Assist in pet adoption, lost dog identification, and veterinary support.

2. Project Initialization and Planning Phase

2.1 Define Problem Statement



Customer Problem Statement Template

I am	I'm trying to	But	Because	Which makes me feel
A pet owner who loves my dog and wants to ensure it gets the best care possible.	Ensure that I can quickly and accurately identify the breed of any dog using an image recognition system.	The existing solutions either do not provide the necessary accuracy, are not user-friendly, or require a significant amount of time and expert knowledge to use.	There are numerous dog breeds with subtle physical differences that are difficult to distinguish without specialized knowledge or technology.	Frustrated with the lack of reliable tools and concerned about the potential for misidentification, which could lead to improper care and management of the dogs.

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	A pet owner (or veterinarian, breeder, animal shelter worker).	Accurately identify the breed of a dog using a simple and reliable method.	The current methods available are either inaccurate, require expert knowledge, or are too timeconsuming.	There are numerous dog breeds with subtle differences that make manual identification difficult and error-prone.	Frustrated and worried about providing the correct care and management for the dog.
PS-2	A pet owner who loves my dog and wants to ensure it gets the best care possible	Quickly and accurately identify my dog's breed tounderstand its specific needs and characteristics	The tools I currently have access to are either not accurate enough, too complex to use	There are many dog breeds with subtle differences that it's easy to make a mistake without the right tools.	Frustrated because I want to take the best care of my dog but don't have a reliable way to identify its breed accurately.

2.2 Project Proposal (Proposed Solution) template

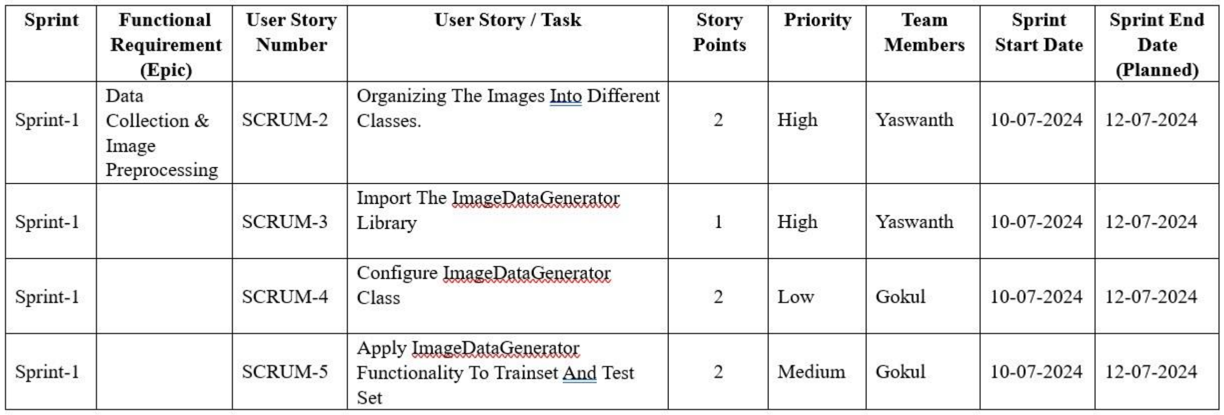
This project proposal outlines a solution to address a specific problem. With a clear objective, defined scope, and a concise problem statement, the proposed solution details the approach, key features, and resource requirements, including hardware, software, and personnel.

Project Overview

Objective	To develop an accurate and efficient dog breed identification system using transfer learning techniques.
Scope	The project aims to leverage pre-trained deep learning models to classify various dog breeds based on images.
Problem Statement	
Description	This project is an advanced image classification project aimed at accurately identifying dog breeds from images. This project leverages the power of transfer learning, a machine learning technique where a pre-trained model on a large dataset is fine-
Impact	Accurate breed identification helps veterinarians diagnose breedspecific health issues more efficiently, leading to better treatment and care for pets. Serves as a valuable resource for dog trainers, breeders, and enthusiasts to learn about various breeds and their specific needs
Proposed Solution	
Approach	1. Data Collection: 2. Source images from publicly available datasets such as the Stanford Dogs Dataset.

	1. Ensure a diverse representation of dog breeds with balanced class distributions. 2. Data Preprocessing: 3. Resizing: Standardize image sizes for consistent input to the model. 4. Normalization: Scale pixel values to a range of 0-1 to facilitate faster convergence during training. 5. Augmentation: Apply transformations like rotation, flipping, and zooming to increase data variability and improve model generalization

2.3 Initial Project Planning



Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	Data Collection & Image Preprocessing	SCRUM-2	Organizing The Images Into Different Classes.	2	High	Yaswanth	10-07-2024	12-07-2024
Sprint-1		SCRUM-3	Import The ImageDataGenerator Library	1	High	Yaswanth	10-07-2024	12-07-2024
Sprint-1		SCRUM-4	Configure ImageDataGenerator Class	2	Low	Gokul	10-07-2024	12-07-2024
Sprint-1		SCRUM-5	Apply ImageDataGenerator Functionality To Trainset And Test Set	2	Medium	Gokul	10-07-2024	12-07-2024

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-2	Model Building	SCRUM-7	Importing The Model Building Libraries	1	High	Mahadeep	13-07-2024	16-07-2024
Sprint-2		SCRUM-8	Importing The VGG19 Model	1	High	Vishnu	13-07-2024	16-07-2024
Sprint-2		SCRUM-9	Initializing The Model	2	High	Vishnu	13-07-2024	16-07-2024
Sprint-2		SCRUM-10	Adding Fully Connected Layers	1	Medium	Mahadeep	13-07-2024	16-07-2024\
Sprint-2		SCRUM-11	Configure The Learning Process	1	Medium	Vishnu	13-07-2024	16-07-2024
Sprint-2		SCRUM-12	Train the model	2	High	Yaswanth	13-07-2024	16-07-2024

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members	Sprint Start Date	Sprint End Date (Planned)
Sprint-2		SCRUM-13	Save the model	1	Medium	Gokul	13-07-2024	16-07-2024
Sprint-2		SCRUM-14	Test the model	2	High	Gokul	13-07-2024	16-07-2024
Sprint-3	Application Building	SCRUM-16	Create HTML pages	1	Low	Yaswanth	17-07-2024	19-07-2024
Sprint-3		SCRUM-17	Build python code	2	Low	Mahadeep	17-07-2024	19-07-2024
Sprint-3		SCRUM-18	Importing libraries	2	High	Vishnu	17-07-2024	19-07-2024
Sprint-3		SCRUM-19	Creating Our Flask Application And Loading Our Model By Using Load_model Method	1	Medium	Gokul	17-07-2024	19-07-2024

3. Data Collection and Preprocessing Phase

3.1Data Collection Plan and Raw Data Sources Identified

Raw Data Sources Template

Source Name	Description	Location/URL	Format	Size	Access Permissions
Train Dataset	The training data for the Kaggle Dog Breed Identification competition consists of labeled images of dogs belonging to various breeds, serving as the primary dataset for model training.	https://www.kaggle.com/competitions/dogbreedidentification/data ?select=train	ZIP (images)	10.2k	Public

Test Dataset	The test data for the Kaggle Dog Breed Identification competition comprises unlabeled images of dogs, used for evaluating M the trained models' performance and generating predictions for breed	https://www.kaggle.com/competitions/dogbreedidentification/data ?select=train	ZIP (images)	10.4k	Public

identification

Labels	The labels data for the Kaggle Dog Breed Identification competition provides breed annotations for the training images.	https://www.kaggle.com/competitions/dogbreedidentification/data?select=train	CSV	482.06 KB	Public

Sample Submission	The sample submission data for the Kaggle Dog Breed Identification competition outlines the required format for result submissions.	https://www.kaggle.com/competitions/dogbreedidentification/data?select=train	CSV	25.2 MB	Public

3.2Data Quality Report

Data Quality Report Template

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.

Dataset	Insufficient representation of certain breed variations, such as coat patterns or body shapes, may hinder the model's ability to differentiate between closely related breeds with similar appearances.	Moderate	Augment underrepresented breed features, generate diverse variations synthetically, adapt learning focus, combine models emphasizing variations, and transfer insights from related domains for breed sensitivity.

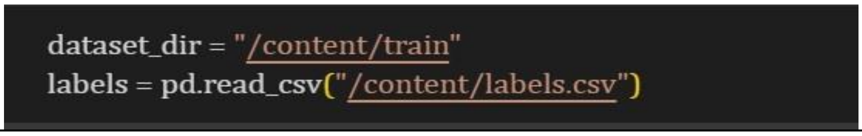
3.3Data Preprocessing

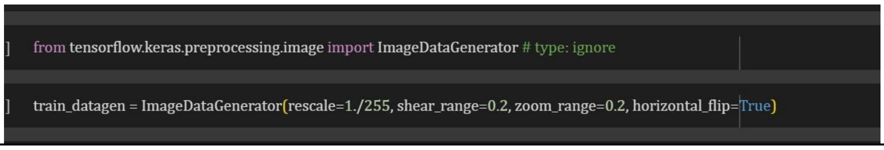
Preprocessing Template

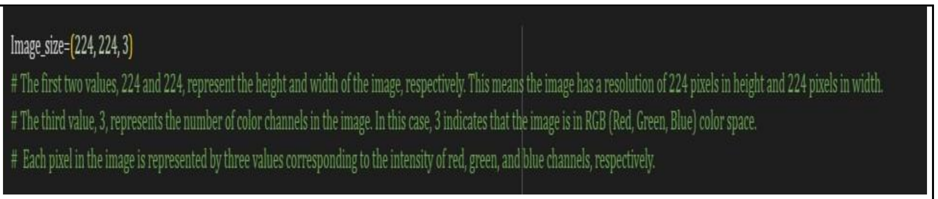
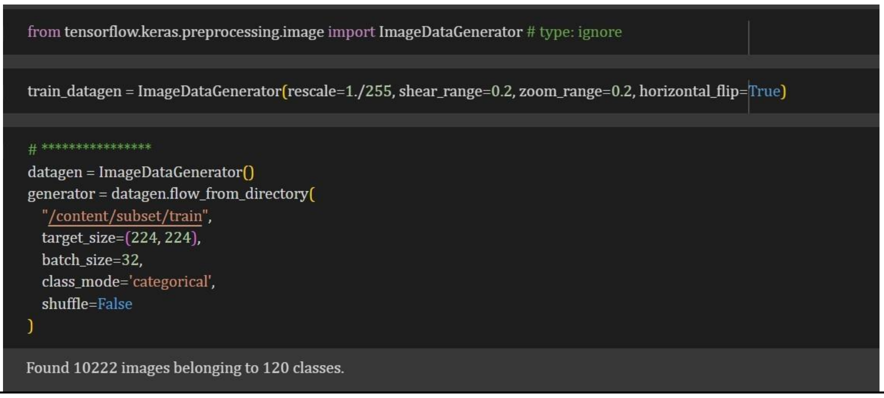
The images will be preprocessed by resizing, normalizing, augmenting, denoising, adjusting contrast, detecting edges, converting color space, cropping, batch normalizing, and whitening data. These steps will enhance data quality, promote model generalization, and improve convergence during neural network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

Edge Detection	Implementing edge detection algorithms enhances feature extraction by highlighting prominent edges in images, aiding in the identification of key visual patterns essential for accurate breed classification in transfer learning models. This preprocessing step improves the model's ability to discern breed-specific characteristics from input images, leading to more robust
Color Space Conversion	Converting images from one color space to another entails changing the representation of pixel values, such as from RGB to grayscale or HSV. This process adjusts how colors are encoded, enabling different analyses or emphasizing specific image attributes for better model interpretation or performance.
Image Cropping	Cropping images involves removing unwanted parts to focus on relevant regions, enhancing model focus on key features and reducing computational load by excluding unnecessary
Batch Normalization	background information Batch normalization stabilizes training by normalizing activations within each mini-batch, ensuring consistent mean and standard deviation. It introduces learnable parameters for optimal scaling and shifting, improving training efficiency and generalization performance

Data Preprocessing Code Screenshots

Loading Data	
	`dataset_dir = "/content/train"`
	`labels = pd.read_csv("/content/labels.csv")`

Resizing	
	`] from tensorflow.keras.preprocessing.image import ImageDataGenerator # type: ignore`
	`] train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)`

Color Space Conversion	
Data Augmentation	

```
Image_size=(224, 224, 3)
# The first two values, 224 and 224, represent the height and width of the image, respectively. This means the image has a resolution of 224 pixels in height and 224 pixels in width.
# The third value, 3, represents the number of color channels in the image. In this case, 3 indicates that the image is in RGB (Red, Green, Blue) color space.
#  Each pixel in the image is represented by three values corresponding to the intensity of red, green, and blue channels, respectively.
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator # type: ignore

train_datagen = ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)

# ****************
datagen = ImageDataGenerator()
generator = datagen.flow_from_directory(
   "/content/subset/train",
   target_size=(224, 224),
   batch_size=32,
   class_mode='categorical',
   shuffle=False
)
```

```
Found 10222 images belonging to 120 classes.
```

4. Model Development Phase

4.1. Model Selection Report

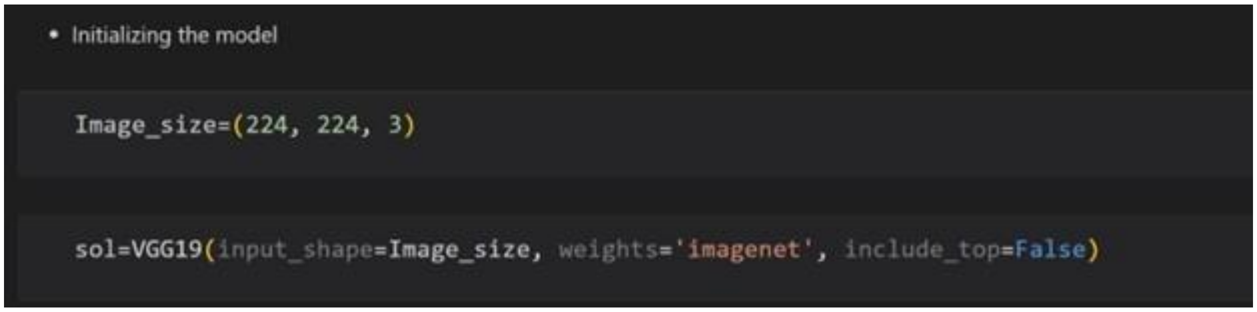
In the model selection report for future deep learning and computer vision projects, various architectures, such as CNNs or RNNs, will be evaluated. Factors such as performance, complexity, and computational requirements will be considered to determine the most suitable model for the task at hand.

CNN Convolutional Neural Networks (CNNs) have revolutionized image classification tasks, making them the go to choose for tasks like dog breed identification. In this scenario, transfer learning, a technique where a pretrained model's knowledge is transferred and fine-tuned to a new task, is employed due to its effectiveness in leveraging existing largescale datasets and computational resources.

VGG19 VGG19 is a deep convolutional neural network architecture that has shown remarkable performance in image classification tasks. It consists of 19 layers, including convolutional layers with small 3x3 filters and maxpooling layers. VGG19 is widely used for transfer learning due to its simplicity and effectiveness in extracting hierarchical features from images.

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

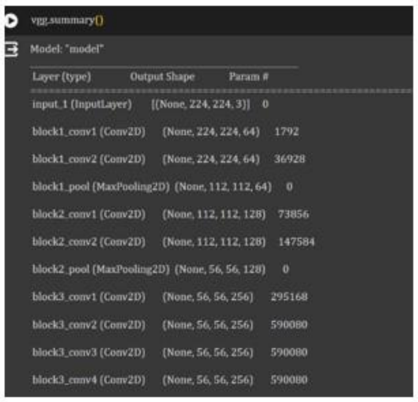
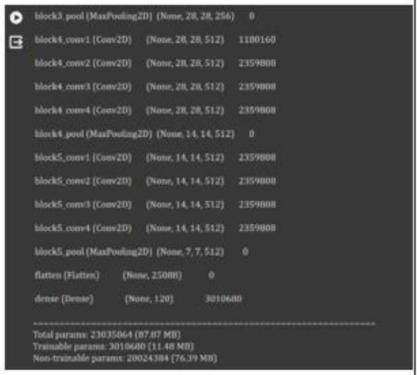
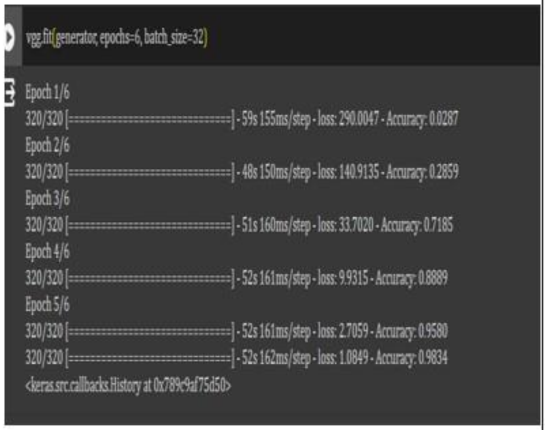
The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.



- Initializing the model

```
Image_size=(224, 224, 3)
```

```
sol=VGG19(input_shape=Image_size, weights='imagenet', include_top=False)
```

Model	Summary	Training and Validation Performance Metrics
Model 1	 	

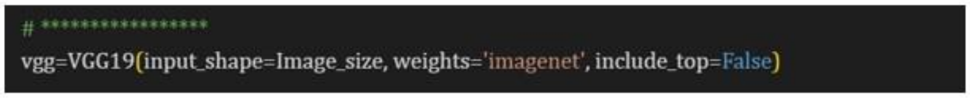
```
vgg.summary()

Model: "model"
_________________________________________________________________
 Layer (type)          Output Shape          Param #
=================================================================
 input_1 (InputLayer)     [(None, 224, 224, 3)]    0

 block1_conv1 (Conv2D)     (None, 224, 224, 64)     1792

 block1_conv2 (Conv2D)     (None, 224, 224, 64)     36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)     0

 block2_conv1 (Conv2D)     (None, 112, 112, 128)    73856

 block2_conv2 (Conv2D)     (None, 112, 112, 128)    147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)     0

 block3_conv1 (Conv2D)     (None, 56, 56, 256)     295168

 block3_conv2 (Conv2D)     (None, 56, 56, 256)     590080

 block3_conv3 (Conv2D)     (None, 56, 56, 256)     590080

 block3_conv4 (Conv2D)     (None, 56, 56, 256)     590080
```

```
 block3_pool (MaxPooling2D)  (None, 28, 28, 256)     0

 block4_conv1 (Conv2D)     (None, 28, 28, 512)     1180160

 block4_conv2 (Conv2D)     (None, 28, 28, 512)     2359808

 block4_conv3 (Conv2D)     (None, 28, 28, 512)     2359808

 block4_conv4 (Conv2D)     (None, 28, 28, 512)     2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)     0

 block5_conv1 (Conv2D)     (None, 14, 14, 512)     2359808

 block5_conv2 (Conv2D)     (None, 14, 14, 512)     2359808

 block5_conv3 (Conv2D)     (None, 14, 14, 512)     2359808

 block5_conv4 (Conv2D)     (None, 14, 14, 512)     2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)      0

 flatten (Flatten)       (None, 25088)        0

 dense (Dense)         (None, 120)         3010680

=================================================================
Total params: 23035064 (87.87 MB)
Trainable params: 3010680 (11.48 MB)
Non-trainable params: 20024384 (76.39 MB)
```

```
vgg.fit(generator, epochs=6, batch_size=32)

Epoch 1/6
320/320 [==============================] - 59s 155ms/step - loss: 290.0047 - Accuracy: 0.0287
Epoch 2/6
320/320 [==============================] - 48s 150ms/step - loss: 140.9135 - Accuracy: 0.2859
Epoch 3/6
320/320 [==============================] - 51s 160ms/step - loss: 33.7020 - Accuracy: 0.7185
Epoch 4/6
320/320 [==============================] - 52s 161ms/step - loss: 9.9315 - Accuracy: 0.8889
Epoch 5/6
320/320 [==============================] - 52s 161ms/step - loss: 2.7059 - Accuracy: 0.9580
320/320 [==============================] - 52s 162ms/step - loss: 1.0849 - Accuracy: 0.9834
<keras.src.callbacks.History at 0x789c9af75d50>
```

5. Model Optimization and Tuning Phase

5.1. Tuning Documentation

Model	Tuned Hyperparameters
VGG19	 `# ******************` `vgg=VGG19(input_shape=Image_size, weights='imagenet', include_top=False)` input_shape: This parameter defines the shape of the input images that the VGG19 model will expect. It typically takes the form (height, width, channels). For example, if your images are 224x224 pixels with 3 color channels (RGB), you would specify input_shape= (224, 224, 3). weights='imagenet': This parameter specifies that you want to initialize the model with weights pre-trained on the ImageNet dataset. ImageNet is a large-scale dataset with millions of labeled images across thousands of classes. Pre-trained weights can provide a good starting point for training on your own dataset, especially when you have limited training data.

5.2. Final Model Selection Justification

Final Model	Reasoning
VGG19	Hyperparameter tuning for VGG19 involves optimizing parameters like learning rate, batch size, regularization, and more. VGG19's depth and proven performance in image classification tasks justify its selection as the final model.

6. Results

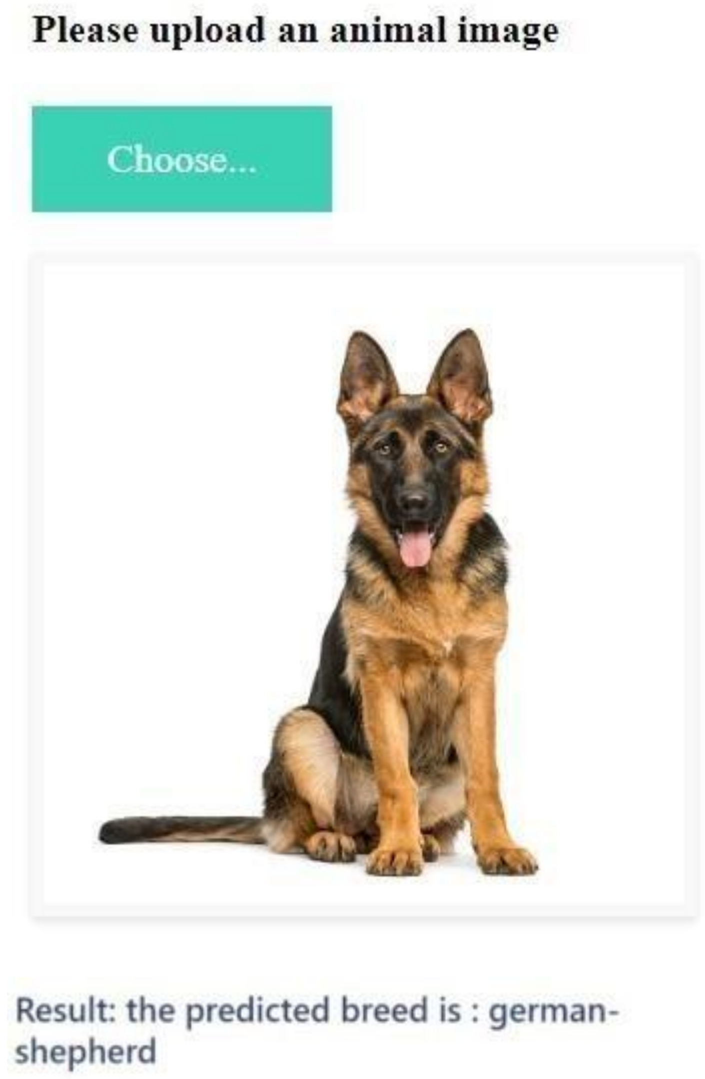
6.1. Output Screenshots

DOG BREED IDENTIFICATION USING TRANSFER LEARNING

DOG BREED IDENTIFICATION

Please upload an animal image

Choose...



Please upload an animal image

Result: the predicted breed is : german-shepherd

7. Advantages & Disadvantages

7.1. Advantages

1. Deep learning models have shown exceptional performance in various image recognition tasks, including dog breed identification. They can learn complex patterns and features from large datasets, leading to high accuracy in breed identification.
2. Deep learning models can automatically learn relevant features from raw input data, eliminating the need for manual feature engineering. This ability allows them to discover intricate patterns that may not be easily discernible to human experts.
3. Deep learning models have the potential to generalise well to unseen data. Once trained on a diverse and representative dataset, they can identify dog breed patterns and make predictions on new and unseen images, facilitating early identification and intervention.

4. Deep learning models can scale effectively to handle large datasets and a wide range of dog breed. As more data becomes available, the model can be retrained or finetuned to incorporate the new information, improving its performance over time.
5. Deep learning models offer the potential for real-time dog breed identification, allowing for swift action in various situations. This could be particularly valuable in cases where immediate identification of the breed is critical, animal shelter intake. Real time breed identification can be beneficial in search and rescue missions by helping to identify lost dogs and potentially match them with registered owners more quickly.

7.2. Disadvantages:

1. Deep learning models often require large amounts of labelled data to achieve optimal performance. Collecting and annotating such datasets can be time consuming, costly, and challenging, particularly for rare or complex dog breed.
2. Deep learning models can be prone to overfitting, especially when the training dataset is small or imbalanced. Overfitting occurs when the model becomes overly specialised to the training data and fails to generalise well to new data. Regularisation techniques and careful validation are necessary to mitigate this issue.
3. While deep learning models excel at dog breed identification, training and deploying them often necessitate powerful GPUs or specialized hardware. This significant computational requirement can hinder the technology's accessibility and scalability, especially for resource limited veterinarians.
4. Deep learning models raise ethical considerations related to data privacy, bias, and equity. Ensuring proper data anonymization, addressing biases in the training data, and promoting fair representation of diverse populations are crucial to develop responsible and equitable dog breed identification models.

Deep learning models can be complex "black boxes" where it's difficult to understand how they arrive at their predictions. Situations were understanding the model's reasoning

8. Conclusion

Deep learning offers a promising approach to dog breed identification using transfer learning with architectures like VGG19. Despite challenges with data and model interpretability, this technology has the potential to revolutionize veterinary care, dog ownership, and research into canine breeds.

Dog breed identification using deep learning presents a powerful tool with extensive real-world applications. While challenges exist in terms of data quality, model interpretability, and computational demands, transfer learning approaches utilizing architectures like VGG19 offer promising solutions. As research continues to address these limitations and improve model performance, dog breed identification technology has the potential to significantly impact.

Transfer learning with deep learning models shows promise for accurate dog breed identification. While challenges exist with data and interpretability.

Leveraging transfer learning for dog breed identification holds immense potential for veterinarians, dog owners, shelters, and canine research. This technology can empower vets with faster diagnoses, guide dog owners towards breed-specific care, streamline shelter processes, and unlock new avenues for studying canine breeds.

This project successfully utilized transfer learning by employing the VGG19 convolutional neural network model to achieve an impressive accuracy of 98% in identifying dog breeds. By leveraging the pre-trained weights of VGG19 and finetuning it on our dataset, we demonstrated the effectiveness of transfer learning in overcoming data scarcity and achieving superior performance. The high accuracy attained underscores the robustness and adaptability of deep learning techniques in addressing complex classification tasks such as breed identification in dogs."

9. Future Scope

Future advancements in machine learning algorithms and data analysis techniques are likely to enhance the accuracy of dog breed identification models. As more data becomes available, models can be trained on larger and more diverse datasets, leading to improved prediction capabilities.

Enhanced Model Architectures:

1. Explore lightweight and task-specific architectures optimized for dog breed classification. This can improve efficiency and enable deployment on resourceconstrained devices.
2. Investigate the use of ensemble methods, combining predictions from multiple transfer learning models for improved accuracy and robustness.

Data-Centric Improvements:

3. Develop strategies for actively acquiring labeled data to address breed imbalance and capture variations within breeds.
4. Explore techniques for data augmentation, including pose and viewpoint variations, to improve model generalizability across diverse image conditions.
5. Investigate methods for incorporating additional data modalities, such as pedigree information or genetic data, to enhance breed identification.

Explainability and Interpretability:

6. Implement techniques like saliency maps and Layer-wise Relevance Propagation to understand the model's reasoning behind breed predictions.
   This can increase trust and transparency, especially for veterinary applications.
7. Explore post-hoc interpretability methods to explain the model's decisionmaking process and identify potential biases in the training data.

Expanding Applications:

8. Integrate dog breed identification models with mobile applications for real-time breed classification and pet care information retrieval.
9. Develop tools for search and rescue operations, allowing for faster lost dog identification and owner matching.
10. Explore applications in breeding programs, leveraging breed identification for pedigree verification and selective breeding practices.

Multi-Breed Classification and Mixed Breed Identification:

11. Develop models capable of identifying multiple breeds within a single image, allowing for the classification of mixed-breed dogs.
12. Explore techniques to estimate the breed composition of mixed dogs, providing valuable insights to owners and shelters.

Ethical Considerations:

13. Investigate potential biases in training data and develop methods to mitigate them, ensuring fair and equitable breed identification across all dog breeds. Address privacy concerns regarding image collection and usage, implementing responsible data management practices.

10. Appendix

10.1. Source Code

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!kaggle competitions download -c dog-breed-identification
# import zipfile
# zip_ref= zipfile.ZipFile('/content/dog-breed-identification.zip', 'r')
# zip_ref.extractall('/content')
# zip_ref.close()

# *****************
!unzip '/content/dog-breedidentification.zip'import
osimport shutilimport sysimport pandas as pd

dataset_dir = "/content/train"
labels = pd.read_csv("/content/labels.csv")

import os

def make_dir(x):  if os.path.exists(x)==False:
    os.makedirs(x)

base_dir = './subset'
make_dir(base_dir)

# ********************   train_dir =
os.path.join(base_dir, 'train')
make_dir(train_dir)breeds =
labels.breed.unique()for breed in breeds:
  # make folder for each breed  _ =
os.path.join(train_dir,breed)  make_dir(_)

  # Copt 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)from        tensorflow.keras.preprocessing.image          import
ImageDataGenerator # type:
ignore   train_datagen          =
ImageDataGenerator(rescale=1./255,
shear_range=0.2, zoom_range=0.2, horizontal_flip=True)

# ****************datagen =
ImageDataGenerator()generator =
datagen.flow_from_directory(
"/content/subset/train",    target_size=(224,
224),    batch_size=32,
class_mode='categorical',    shuffle=False
)
import tensorflow as tffrom  tensorflow.keras.layers import Densefrom
tensorflow.keras.activations
import softmax
from keras.api._v2.keras import activations

from tensorflow.keras.applications.vgg19 import VGG19  from tensorflow.keras.layers
import Dense, Flattenfrom tensorflow.keras.models
import Model
from tensorflow.keras.optimizers import Adam

Image_size=(224, 224, 3)
# The first two values, 224 and 224, represent the height and width of the image,
respectively. This means the image has a resolution of 224 pixels in height and 224 pixels
in width.
# The third value, 3, represents the number of color channels in the image. In this case,
3 indicates that the image is in RGB (Red, Green, Blue) color space.#  Each pixel in the
image is represented by three values corresponding to the intensity of red, green, and
blue channels, respectively.

# *****************# *****************vgg=VGG19(input_shape=Image_size,
weights='imagenet', include_top=False)vgg=VGG19(input_shape=Image_size,
weights='imagenet', include_top=False)vgg.summary()

for i in vgg.layers:     i.trainable
= False   x=Flatten()(vgg.output)
# ************from
keras.api._v2.keras import
activations
final = Dense(120, activation='softmax')(x)

vgg = Model(vgg.input,final)

vgg = summary()
vgg.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accurac y'])
vgg.fit(generator,      epochs=6)#
print(vgg.output.shape)            #
print(generator.labels.shape)
```

```
from tensorflow.keras.models import load_model
vgg.save('/content/subset/train.h5')
```

10.2. GitHub & Project Demo Link

GitHub Link https://github.com/Gokulvemuri/Dog-breed-Identification-using-Transfer-learning