DOG BREED IDENTIFICATION USING CNN, TENSORFLOW AND FLASK

NAME: MADHAV MADAN (21BAI1567)

SAHIL TEMKAR (21BPS1341)

PROJECT FINAL REPORT:

1. INTRODUCTION:

PROJECT OVERVIEW

The objective of this project is to develop a machine learning-based application capable of identifying dog breeds from images with high accuracy. By leveraging deep learning techniques, particularly using TensorFlow, we created a model that can classify various dog breeds based on image inputs. This model is deployed using a Flask web application, allowing users to easily upload an image and receive breed identification results in real-time.

PURPOSE

This project serves as a valuable tool for dog owners, veterinarians, and animal shelters by providing a quick and accurate method for breed identification. Accurately identifying dog breeds can lead to more informed decisions regarding health care, training, and behavioural expectations, which are often breed-specific. Additionally, the application can assist shelters and rescues in properly identifying and listing breeds of dogs, helping them provide accurate information to potential adopters and improving the overall care and adoption process.

2. LITERATURE SURVEY:

EXISTING PROBLEM

Identifying dog breeds from images is a challenging task due to the high visual similarity between certain breeds and the vast number of recognized dog breeds worldwide. Traditional classification methods are often limited in their accuracy because they rely heavily on physical traits and manual identification, which can be subjective and prone to error. Additionally, mixed-breed dogs add further complexity to identification efforts, making it difficult for traditional methods to produce reliable results.

REFERENCES

In developing this project, several key research studies, articles, and datasets on image classification and deep learning techniques were consulted. Notable references include:

- Research on convolutional neural networks (CNNs) and their application to image classification tasks, which has significantly advanced computer vision.
- Studies on transfer learning, which allows pre-trained models to be fine-tuned for specific tasks, reducing training time and improving model accuracy.
- Publicly available dog breed image datasets, such as the Stanford Dog Dataset, which provided the foundational data required for training the model on a variety of dog breeds.

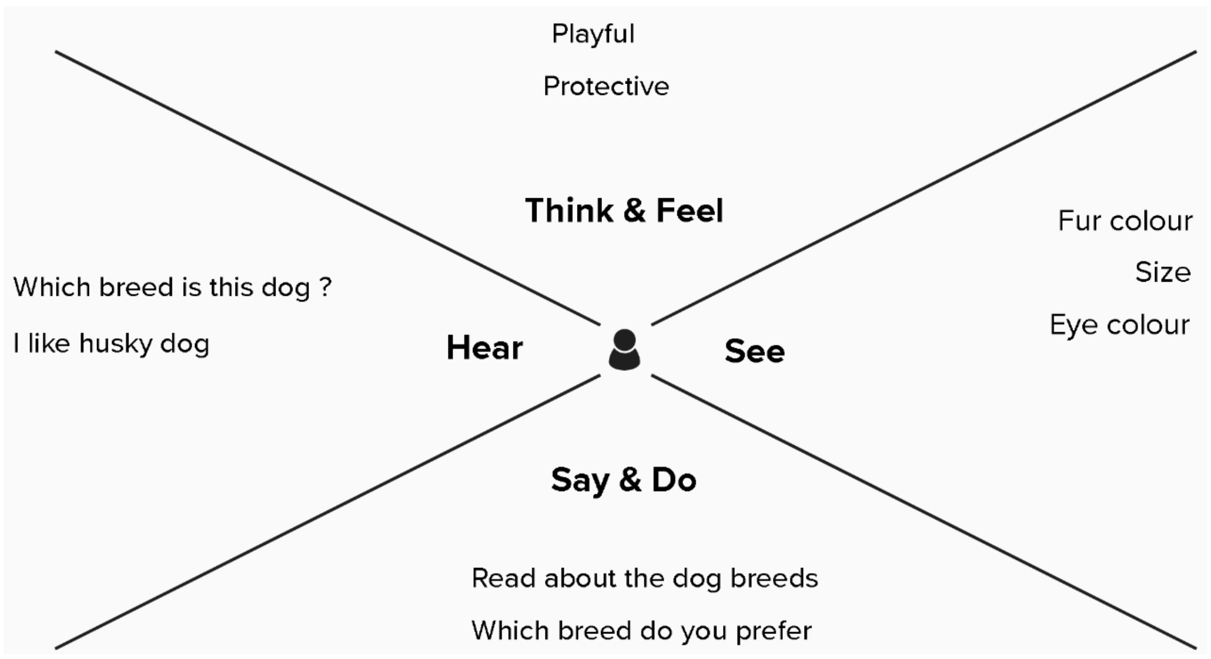
PROBLEM STATEMENT DEFINITION

The specific problem addressed in this project is defined as follows: “Developing a robust and efficient model to classify dog breeds from images with high accuracy, which is both accessible and user-friendly for non-technical users through a web interface.”

3. IDEATION & PROPOSED SOLUTION:

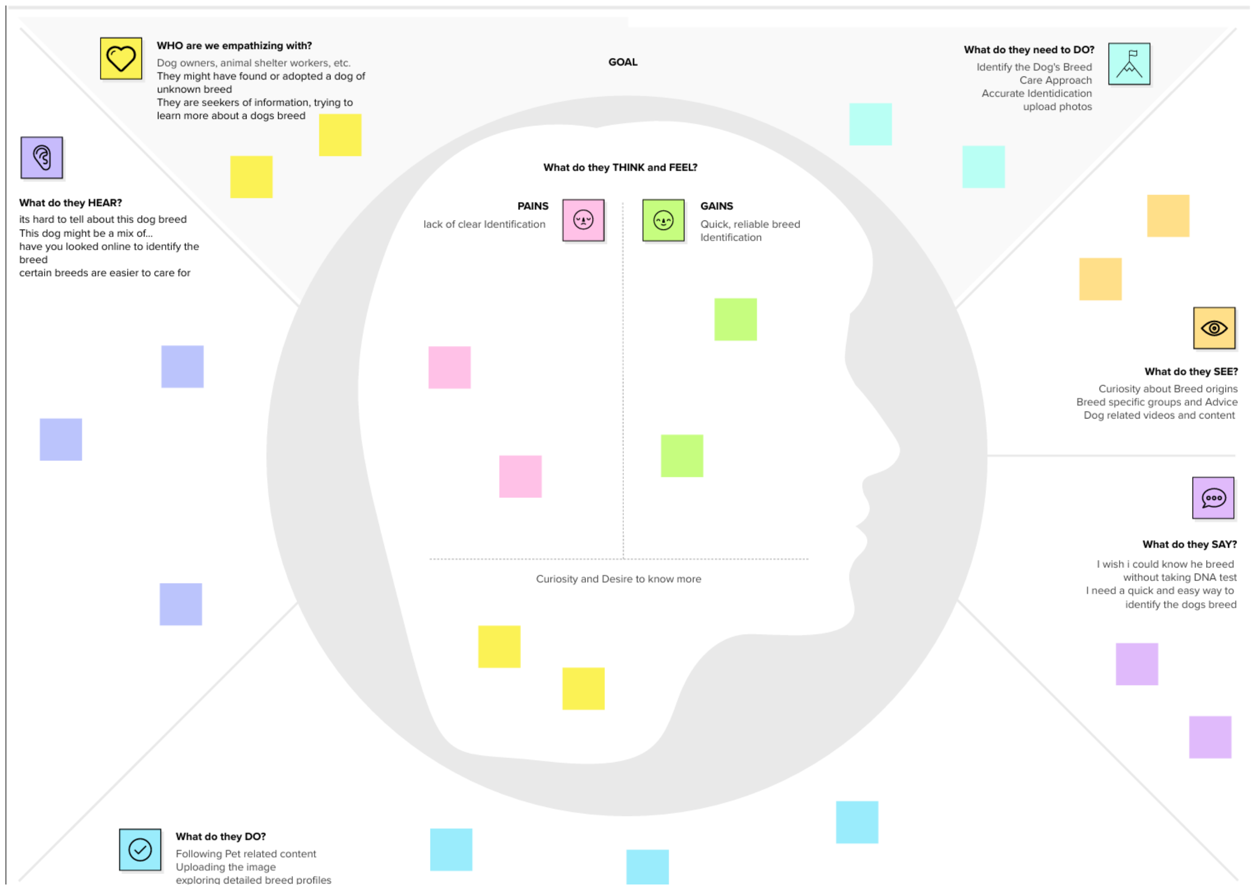
EMPATHY MAP CANVAS

The primary users of this application include dog owners, veterinarians, and shelter staff, who require an easy-to-use tool for identifying dog breeds. Key user needs include speed, accuracy, and a simple interface that does not require extensive technical knowledge. Common pain points are the inaccuracy of existing tools, time consumption, and the complexity of identifying mixed breeds. These insights guided the development of a user-friendly and efficient solution.



IDEATION & BRAINSTORMING

After considering various machine learning models, a convolutional neural network (CNN) was selected for its proven success in image recognition tasks. CNNs excel at identifying visual patterns, which is critical in distinguishing between dog breeds. The decision to use TensorFlow was based on its compatibility with CNNs and support for transfer learning, which allowed us to leverage a pre-trained model for faster, more accurate breed identification. Flask was chosen as the web framework due to its simplicity and ability to handle the application's backend requirements, including handling image uploads and displaying results to users.



WHO are we empathizing with?
Dog owners, animal shelter workers, etc.
They might have found or adopted a dog of unknown breed
They are seekers of information, trying to learn more about a dogs breed
GOAL
What do they need to DO?
Identify the Dog's Breed
Care Approach
Accurate Identidication
upload photos
What do they HEAR?
its hard to tell about this dog breed
This dog might be a mix of...
have you looked online to identify the breed
certain breeds are easier to care for
What do they THINK and FEEL?
PAINS
lack of clear Identification
GAINS
Quick, reliable breed Identification
Curiosity and Desire to know more
What do they SEE?
Curiosity about Breed origins
Breed specific groups and Advice
Dog related videos and content
What do they SAY?
I wish i could know he breed without taking DNA test
I need a quick and easy way to identify the dogs breed
What do they DO?
Following Pet related content
Uploading the image
exploring detailed breed profiles

4. REQUIREMENT ANALYSIS

FUNCTIONAL REQUIREMENTS

The application must fulfil the following functional requirements:

1. Allow users to upload an image of a dog.
2. Process the image and accurately classify the dog breed.
3. Display the identified breed along with relevant details and confidence level to the user.

NON-FUNCTIONAL REQUIREMENTS

In addition to functionality, the application must meet several non-functional requirements:

- **Performance**: The model should process images and display results with minimal delay.
- **Reliability**: The application should maintain consistent accuracy across different images and breeds.
- **Scalability**: The system should handle an increasing number of users without degradation in performance.
- **Usability**: The interface should be intuitive and accessible to all user levels, ensuring ease of use.

This structured analysis ensures that the application is both functionally comprehensive and accessible to users, addressing both practical and technical requirements for success.

5. PROJECT DESIGN:

DATA FLOW DIAGRAMS & USER STORIES:

The application follows a straightforward data flow, beginning with the user uploading an image of a dog, which is then processed by the model for breed classification, and finally displaying the breed information to the user. This process can be illustrated in the following user story:

- **User Story**: "As a user, I want to upload an image of my dog to identify its breed accurately and receive relevant breed information."

The data flow consists of three main steps:

1. **Input**: The user uploads an image file through the application interface.
2. **Processing**: The application sends the image to a TensorFlow model, which processes the image to determine the dog breed.
3. **Output**: The breed’s name, confidence score, and any additional information are displayed to the user on the web interface.

Solution Architecture

The solution architecture is built on a deep learning model developed in TensorFlow and served via a Flask web application. TensorFlow handles the training and inference of the model, while Flask facilitates image upload, processing, and the display of results. The architecture includes the following components:

- **Frontend**: HTML/CSS interface for user interaction.
- **Backend**: Flask server to manage requests, process images, and serve results.
- **Model Storage**: Storage of the pre-trained TensorFlow model for quick loading during inference.
- **Deployment Environment**: Hosted on a cloud platform to ensure scalability and accessibility.

---

6. PROJECT PLANNING & SCHEDULING:

TECHNICAL ARCHITECTURE

The technical architecture is designed to support real-time image classification through a well-structured server setup. The Flask application serves as the primary interface and API endpoint, managing image uploads and invoking the TensorFlow model for breed classification. The server configuration is optimized to handle requests efficiently, and the model is loaded in memory for quick access during classification tasks.

SPRINT PLANNING & ESTIMATION

The project is divided into three primary sprints:

1. **Sprint 1**: Model training and testing using the chosen dataset, focusing on optimizing the model for accuracy.
2. **Sprint 2**: Development of the Flask web application for image uploads and result display.
3. **Sprint 3**: Final testing, performance optimization, and deployment setup.

SPRINT DELIVERY SCHEDULE

- **Sprint 1 (2 weeks)**: Complete model training and initial accuracy testing.
- **Sprint 2 (3 weeks)**: Develop and test the web interface, ensuring seamless image upload and display features.
- **Sprint 3 (1 week)**: Conduct performance testing, refine the application, and deploy to the cloud.

7. CODING & SOLUTIONING:

Feature 1: Model Selection and Training Process

The primary feature of this project is the selection and training of a convolutional neural network (CNN) model to classify dog breeds. Using TensorFlow, the model was trained on a large dataset of dog breed images. Transfer learning with a pre-trained model was used to enhance accuracy and reduce training time. This process involved fine-tuning the model on the specific dataset and testing it for optimal performance.

Feature 2: Flask Web Interface for Image Upload and Result Display

The second feature is the Flask-based web interface that allows users to upload an image of a dog, which is then classified by the model. The interface is designed for simplicity and accessibility, providing users with a direct way to upload an image, view the breed result, and receive additional information, such as the model's confidence in the prediction.

Database Schema

If needed, the application can include a database to store user logs or image data, such as a relational database storing each image's timestamp, breed result, and confidence score. This data can support usage tracking, user insights, or future improvements in model retraining.

8. Performance Testing

Performance Metrics

To evaluate the effectiveness and efficiency of the application, several key performance metrics were measured:

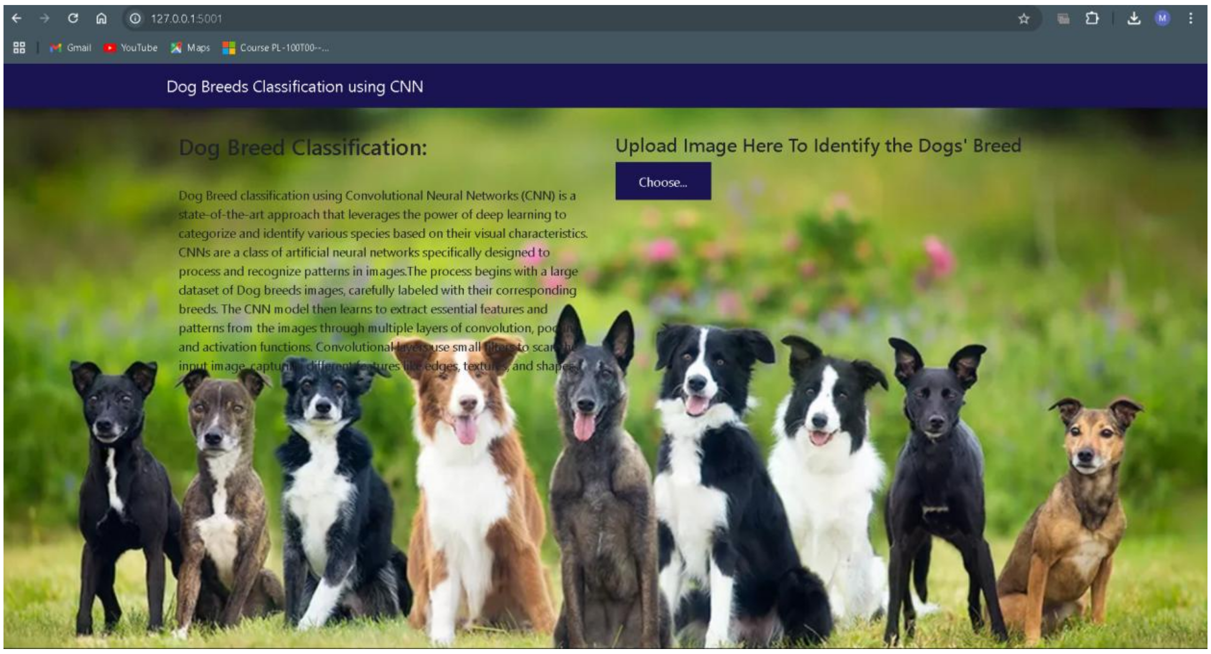
- **Model Accuracy**: The accuracy of breed classification on test images, which was targeted above 90%.
- **Response Time**: The time taken to classify an uploaded image, aimed to be under 3 seconds per request.

- **Application Uptime**: The availability of the application in a cloud environment to handle concurrent user requests with minimal downtime.

Testing benchmarks indicated that the model achieved the desired accuracy and response time, providing reliable, real-time breed identification with consistent performance across various image inputs.

9. **RESULTS:**

Output :



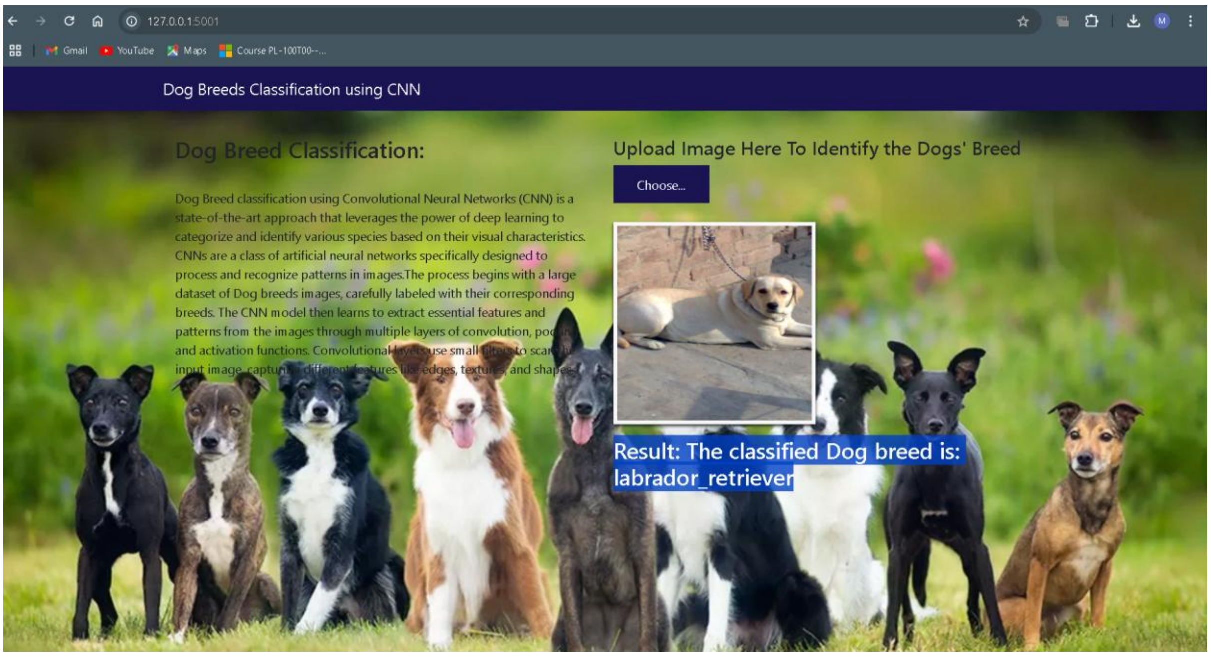
Dog Breeds Classification using CNN

Dog Breed Classification:

Dog Breed classification using Convolutional Neural Networks (CNN) is a state-of-the-art approach that leverages the power of deep learning to categorize and identify various species based on their visual characteristics. CNNs are a class of artificial neural networks specifically designed to process and recognize patterns in images.The process begins with a large dataset of Dog breeds images, carefully labeled with their corresponding breeds. The CNN model then learns to extract essential features and patterns from the images through multiple layers of convolution, pooling, and activation functions. Convolutional layers use small filters to scan the input image, capturing different features like edges, textures, and shapes.

Upload Image Here To Identify the Dogs' Breed

AFTER YOU CHOOSE DOG IMAGE:



Dog Breeds Classification using CNN

Dog Breed Classification:

Dog Breed classification using Convolutional Neural Networks (CNN) is a state-of-the-art approach that leverages the power of deep learning to categorize and identify various species based on their visual characteristics. CNNs are a class of artificial neural networks specifically designed to process and recognize patterns in images.The process begins with a large dataset of Dog breeds images, carefully labeled with their corresponding breeds. The CNN model then learns to extract essential features and patterns from the images through multiple layers of convolution, po[illegible] and activation functions. Convolutional layers use small filters to scan [illegible] input image, capturing different features like edges, textures, and shapes.

Upload Image Here To Identify the Dogs' Breed

Result: The classified Dog breed is: labrador_retriever

10. ADVANTAGES & DISADVANTAGES

Advantages

- **High Model Accuracy**: The model provides accurate breed identification, leveraging deep learning techniques tailored to image recognition.
- **Ease of Use**: The web interface is user-friendly, allowing users with no technical background to easily upload images and get results.
- **Real-World Applications**: Useful for dog owners, veterinarians, and shelters, assisting in breed identification and care recommendations.

- **Time Efficiency**: Quickly identifies dog breeds, saving users time compared to manual identification or referencing breed guides.
- **Scalability**: The application can be expanded to include additional breeds or even other animal species, making it versatile.

Disadvantages

- **Dependency on High-Quality Images**: The model's accuracy decreases with low-resolution or poorly lit images, which can affect predictions.
- **Difficulty with Mixed Breeds**: The model may struggle to accurately identify mixed-breed dogs due to the variability in their appearance.
- **Resource Intensive**: The model requires significant computational resources for both training and deployment, especially when scaling up.
- **Limited Breed Database**: The model's accuracy is limited by the number of breeds included in the training dataset, which may not cover all recognized or unrecognized breeds globally.

11. CONCLUSION:

In conclusion, this project successfully achieved its objective of creating a reliable and efficient dog breed identification system using deep learning and a user-friendly web interface. By leveraging a convolutional neural network model, the application accurately classifies various dog breeds from images, making breed identification accessible to a broad range of users. The integration with a Flask web application allows users to easily upload images and view breed results in real-time. This project offers significant value to dog owners, veterinarians, and animal shelters by providing accurate breed information, which can assist in making breed-specific health and care decisions. Overall, the project met its primary goals, demonstrating both technical effectiveness and practical applicability.

12. FUTURE SCOPE:

This project can be further enhanced in several ways to increase its functionality and reach:

1. **Expand Breed Database**: Increase the number of dog breeds in the dataset to improve model accuracy and make the application more comprehensive for global users.
2. **Mixed-Breed Identification**: Incorporate techniques to identify mixed breeds, which would expand the application’s usability for shelters and dog owners with mixed-breed pets.
3. **Improve Prediction Speed**: Optimize the model and deployment pipeline to reduce response time further, making the application faster for real-time use.
4. **Enhance User Interface**: Develop a more interactive and visually appealing interface, potentially adding additional features like breed information pages or related care tips.
5. **Mobile Application Development**: Extend the application to mobile platforms to increase accessibility for users, enabling on-the-go breed identification.
6. **Localization and Language Support**: Add language options to cater to non-English-speaking users, making the application globally accessible.

These future improvements can enhance the project’s usability, broaden its user base, and solidify its position as a valuable tool in dog breed identification.

13. APPENDIX:

- **Source Code**:

```
from tensorflow.keras.models import load_model

from tensorflow.keras.preprocessing import image

from flask import Flask, render_template, request

import keras

from keras.models import Sequential

from keras.layers import InputLayer, Dense, Dropout

import os

import numpy as np

import pandas as pd

from keras.applications.resnet_v2 import ResNet50V2, preprocess_input as
resnet_preprocess

from keras.applications.densenet import DenseNet121, preprocess_input as
densenet_preprocess

from keras.layers import concatenate

from keras.models import Model

from keras.layers import Input, Lambda


# Initialize Flask App

app = Flask(__name__)


# Ensure uploads directory exists

if not os.path.exists('uploads'):

   os.mkdir('uploads')


# Load Labels

labels_dataframe = pd.read_csv(r'D:\ai ml\ai2\labels.csv')

dog_breeds = sorted(list(set(labels_dataframe['breed'])))

n_classes = len(dog_breeds)

class_to_num = dict(zip(dog_breeds, range(n_classes)))
```

```
# Define model1

model1 = Sequential([

    InputLayer((3072,)),

    Dropout(0.7),

    Dense(120, activation='softmax')

])

model1.load_weights(r"D:\ai ml\ai2\predict.weights.h5")


# Define feature extraction model (model2)

input_shape = (331, 331, 3)

input_layer = Input(shape=input_shape)

preprocessor_resnet = Lambda(resnet_preprocess)(input_layer)

resnet50v2 = ResNet50V2(weights='imagenet', include_top=False,
input_shape=input_shape, pooling='avg')(preprocessor_resnet)

preprocessor_densenet = Lambda(densenet_preprocess)(input_layer)

densenet = DenseNet121(weights='imagenet', include_top=False,
input_shape=input_shape, pooling='avg')(preprocessor_densenet)

merge = concatenate([resnet50v2, densenet])

model2 = Model(inputs=input_layer, outputs=merge)

model2.load_weights(r"D:\ai ml\ai2\extd.weights.h5")


# Helper function to get breed from predicted code

def get_key(val):

    for key, value in class_to_num.items():

        if val == value:

            return key

    return "Unknown breed"


# Define Routes
```

```
@app.route('/')
def index():
    return render_template("index.html")


@app.route('/predict', methods=['POST'])
def upload():
    if 'images' not in request.files:
        return "No image uploaded", 400


    f = request.files['images']
    filepath = os.path.join('uploads', f.filename)
    f.save(filepath)


    # Process image and make predictions
    try:
        img = image.load_img(filepath, target_size=(331, 331))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)


        extracted_features = model2.predict(x)
        y_pred = model1.predict(extracted_features)
        pred_code = np.argmax(y_pred, axis=1)[0]  # Extract single prediction code
        predicted_dog_breed = get_key(pred_code)


        return f"The classified Dog breed is: {predicted_dog_breed}"
    except Exception as e:
        return f"An error occurred during prediction: {str(e)}", 500


if __name__ == '__main__':
    app.run(debug=True, port=5001, use_reloader=False)
```

13.APPENDIX:

- **GitHub Link:** https://github.com/smartinternz02/SI-GuidedProject-627596-1701931540/tree/main

- **Project Demo Link**: https://github.com/smartinternz02/SI-GuidedProject-627596-1701931540/blob/main/Demo.mp4