PROJECT REPORT

ON

"Dog Bread Recognition Model"

Submitted in partial fulfilment of the requirements for the

award of the degree of

MASTER OF COMPUTER APPLICATION

(2023-2025)

SUBMITTED TO:	**SUBMITTED BY:**
Mr. Deepak Dangwal	Nitin Rana (230090600025)
Assistant Professor	Vaibhav Kumar Dabral (230090600035)
Department of Computer Application	MCA III Semester

CANDIDATE'S DECLARATION

We hereby declare that this report is a result of our own work and effort. No part of the report has been copied or plagiarized from any other source, and all external references or information have been duly acknowledged. We have followed ethical guidelines and academic principles while preparing this report.

We affirm that the content is entirely original and has not been submitted elsewhere for any purpose. We understand the significance of maintaining integrity and accountability in academic work, and We take full responsibility if any part of this report is found to violate originality or ethical standards.

Nitin Rana & Vaibhav Kumar Dabral
230090600025 230090600035

Place: Pauri
Date: 17/12/2024

CERTIFICATE

This is to certify that the project report entitled "DOG BREAD RECOGNITION MODEL" submitted to Govind Ballabh Pant Institute of Engineering and Technology, Pauri, is a bonafide record of the work carried out by Nitin Rana & Vaibhav Kumar Dabral under my supervision from 10/09/2024 to 10/12/2024.

The project has been completed to my satisfaction and fulfils the academic requirements as prescribed by the institute.

Mr. Deepak Dangwal
Assistant Professor
Place: Pauri
Date: 05/12/2024

Table of Contents

ABSTRACT

The Dog Breed Recognition Model leverages Convolutional Neural Networks (CNNs) and advanced deep learning techniques to classify dog breeds accurately. This model is designed to assist pet owners, veterinarians, and canine enthusiasts by providing instant and reliable breed identification from images. Utilizing a comprehensive dataset of dog breeds, the model ensures high accuracy and robust performance in diverse lighting and background conditions.

The system employs TensorFlow for building and training the neural network and integrates seamlessly with a user-friendly interface, making it accessible for non-technical users. The modular architecture supports scalability, enabling future enhancements like real-time video recognition. Key features include image preprocessing, breed probability predictions, and feedback mechanisms to improve model accuracy over time.

The Dog Breed Recognition Model serves as a valuable tool for educational, personal, and professional purposes, bridging the gap between technology and canine care.

Keywords

Dog Breed Classification, Convolutional Neural Networks, Image Recognition, TensorFlow, Deep Learning

1. INTRODUCTION

The Dog Breed Recognition Model is an AI-based system designed to identify dog breeds accurately from images. By leveraging Convolutional Neural Networks (CNNs) and advanced image recognition techniques, this model ensures high precision and reliability in breed classification. Aimed at dog owners, veterinarians, breeders, and canine enthusiasts, the system bridges the gap between technology and animal care, providing users with a practical tool for immediate breed identification.

1.1 Objective:

- The primary objective of the project is to develop a robust and user-friendly model capable of recognizing various dog breeds with high accuracy. Key features include image preprocessing, probability-based predictions.

1.2 Scope:

The Dog Breed Recognition Model focuses on the following areas:

- Image-based breed identification.
- Integration with web or mobile platforms for accessibility.
- Scalability for adding more breeds or adapting to real-time video analysis.
- Educational and professional applications for dog enthusiasts and organizations.

1.3 Technology Stack:

- **Model Development:** Python, TensorFlow, Keras.
- **Frontend:** Streamlit
- **Backend:** Python
- **Database:** Kaggle, Bing Image.
- **Other Tools:** Keras

1.4 Methodology:

The project will be developed using an agile methodology, allowing for iterative development and feedback. The phases include:

- **Requirement Analysis:** Identifying core functionalities and datasets.
- **System Design:** Outlining architecture, workflow, and database schema.
- **Model Development:** Training CNN using labeled datasets for breed recognition.
- **Integration:** Connecting the trained model to a user interface.
- **Deployment:** Hosting the application for private use.

1.5 Modules:

- **Image Preprocessing Module:** Prepares input images for the model using resizing, scaling, and augmentation techniques.
- **Classification Module:** Performs breed identification using the trained CNN model.
- **Feedback Module:** Allows users to validate or correct breed predictions to enhance the model's future accuracy.

1.6 Functionalities:

- Real-time or batch image analysis.
- Confidence scores for predictions.
- Support for additional datasets to improve model robustness.

1.7 Challenges:

- **Dataset Quality:** Ensuring access to a diverse and well-annotated dataset covering a broad range of breeds.
- **Model Overfitting:** Balancing model performance to avoid overfitting to specific data.
- **Image Quality Variation:** Managing issues arising from poor-quality, low-light, or cluttered images.
- **Cross-Browser Compatibility:** Ensuring the system works seamlessly across different browsers.

1.8 Conclusion:

- This Dog Breed Recognition Model provides a cutting-edge solution for identifying dog breeds. By harnessing the power of CNNs and deep learning, the project delivers a practical, scalable, and impactful application in both personal and professional domains.

1.9 Future Enhancements:

- Real-time video analysis for dynamic identification.
- Addition of mixed-breed identification capabilities.
- Mobile application development for on-the-go access.
- Incorporation of augmented reality (AR) for educational purposes.
- Multilingual user support for broader accessibility.

2. SRS DOCUMENT

I. Introduction

1.1 Purpose

The purpose of this document is to specify the requirements for the **Dog Breed Recognition Model**. This system will allow users to upload an image of a dog, and the application will identify the breed of the dog using a Convolutional Neural Network (CNN). The system will provide real-time breed recognition results, with a focus on accuracy and user-friendly interaction.

1.2 Scope

The Dog Breed Recognition Model will support functionalities such as image uploading, real-time breed prediction, and confidence scoring for the identified breed. The project will be implemented using the following technologies:

- **TensorFlow** and **Keras** for building and training the CNN model.
- The web application will be deployed on **Streamlit**, and the user interface will allow easy image uploads.

1.3 Definitions, Acronyms, and Abbreviations

- **CNN (Convolutional Neural Network)**: A type of deep learning algorithm for image classification
- **TensorFlow**: An open-source machine learning library used to train the CNN model.
- **Keras**: A high-level neural network API, used for building the model architecture.
- **UI (User Interface)**: The interface through which users interact with the application, allowing them to upload images and view predictions.

II. Overall Description

2.1 Product Overview

The **Dog Breed Recognition Model** is a machine learning-based application designed to identify the breed of a dog from an uploaded image. It leverages a Convolutional Neural Network (CNN) trained on a large dataset of dog images to perform accurate breed classification. The primary goal of this system is to provide real-time breed recognition with high accuracy. Users can interact with the application through a simple interface where they upload an image of a dog, and the system responds with the identified breed and a confidence score.

2.2 Key Features

- **Image Upload**: Users can upload an image of a dog in common formats like JPG, PNG, etc.
- **Breed Recognition**: The application uses a pre-trained CNN model to classify the breed of the dog in the image.
- **User-Friendly Interface**: The system offers a simple and intuitive interface for uploading images and viewing results.
- **Real-Time Processing**: The application delivers predictions in real-time, ensuring that users receive results without significant delay.

2.3 Intended Audience and Users

This system is designed for:

- **Dog Owners**: Individuals curious about the breed of their dog, especially mixed or rescue dogs.
- **Veterinarians**: Professionals who may use the tool as a supplementary resource in identifying dog breeds.
- **Animal Shelters/Rescue Organizations**: Staff who may need to classify the breed of dogs they are caring for.
- **General Public**: Any individual interested in identifying dog breeds.

2.4 Assumptions and Dependencies

- **Assumptions**:
  - The user will upload clear, high-resolution images where the dog is the main subject.
  - The system assumes that the dog belongs to a breed present in the training dataset.
  - Users will have a stable internet connection for uploading images and receiving results.
- **Dependencies**:
  - The application depends on a pre-trained CNN model (developed using **TensorFlow** and **Keras**) for breed recognition.
  - **Flask** will serve as the backend for image processing and API handling.

- The system is dependent on **Streamlit** for hosting the web application and ensuring uptime.
- **Image data generator** will be used for image pre-processing, such as resizing and normalization, before feeding the image into the model.
- The system relies on the availability of the dog breed dataset used to train the model.

III. System Features

3.1 Image Input and Processing

- **Description**: This feature allows users to upload an image of a dog through the web interface. The system accepts images in common formats such as JPG, PNG, JPEG.
- **Functional Requirements**:
  - The system should allow users to select and upload an image from their device.
  - The uploaded image must be pre-processed before feeding it into the CNN model, which includes resizing, normalizing, and possibly augmenting the image for better results.
- **Preconditions**: The user must have an image of a dog ready for upload.
- **Postconditions**: The system will preprocess the image and prepare it for the recognition model.

3.2 Breed Recognition Algorithm (CNN)

- **Description**: The heart of the system is a Convolutional Neural Network (CNN) trained on a dataset of dog breeds. Once the image is processed, it is passed through the CNN to predict the dog's breed.
- **Functional Requirements**:
  - The prediction process must be efficient to ensure real-time results.
  - The system should be able to handle various dog breeds that the model has been trained on.
- **Preconditions**: A preprocessed image is fed into the CNN model.
- **Postconditions**: The system will output the predicted dog breed.

3.3 Accuracy Metrics and Performance

- **Description**: This feature focuses on the accuracy and performance of the breed recognition model. It ensures that the system provides reliable predictions with minimal errors.
- **Functional Requirements**:
  - Accuracy metrics should be tracked during model training and evaluation to ensure optimal performance.

3.4 Error Handling and Notifications

- **Description**: The system must handle errors gracefully and provide meaningful feedback to the user in case something goes wrong (e.g., invalid image upload, prediction failure).
- **Functional Requirements**:
    - The system should detect and report errors such as unsupported image formats, upload issues, or server downtime.
    - Users should be notified of errors via appropriate messages (e.g., "Image upload failed. Please try again.").
- **Preconditions**: An issue occurs during image upload, processing, or prediction.
- **Postconditions**: The system provides clear error messages or suggestions for resolution.

IV. External Interface Requirements

4.1 User Interface

- Simple and intuitive interface for uploading images.
- Display breed prediction clearly.

4.2 Hardware Interface

- Standard hardware: Any device (PC, smartphone) capable of running a web browser.
- image files required for uploading.

4.3 Software Interface

- Frontend: Interaction with Flask-based API to send images and receive breed predictions.
- Backend: Integration with the CNN model through TensorFlow/Keras for processing.

V. Other Specific Requirements

5.1 Constraints

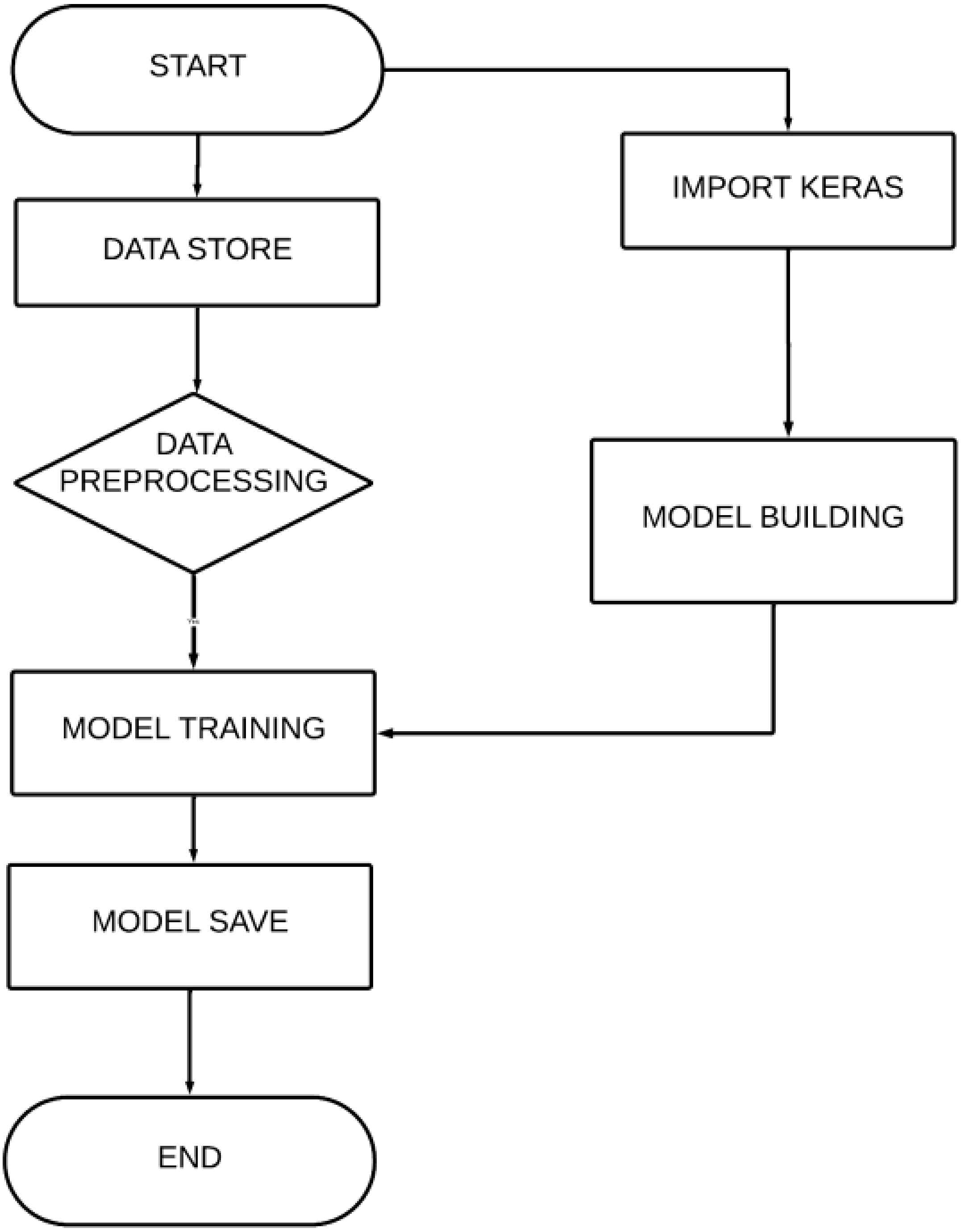
- Requires a stable internet connection for interaction with the backend.
- Limited to the dog breeds the CNN model is trained on.
- Image quality and clarity impact accuracy.

5.2 Business Rules

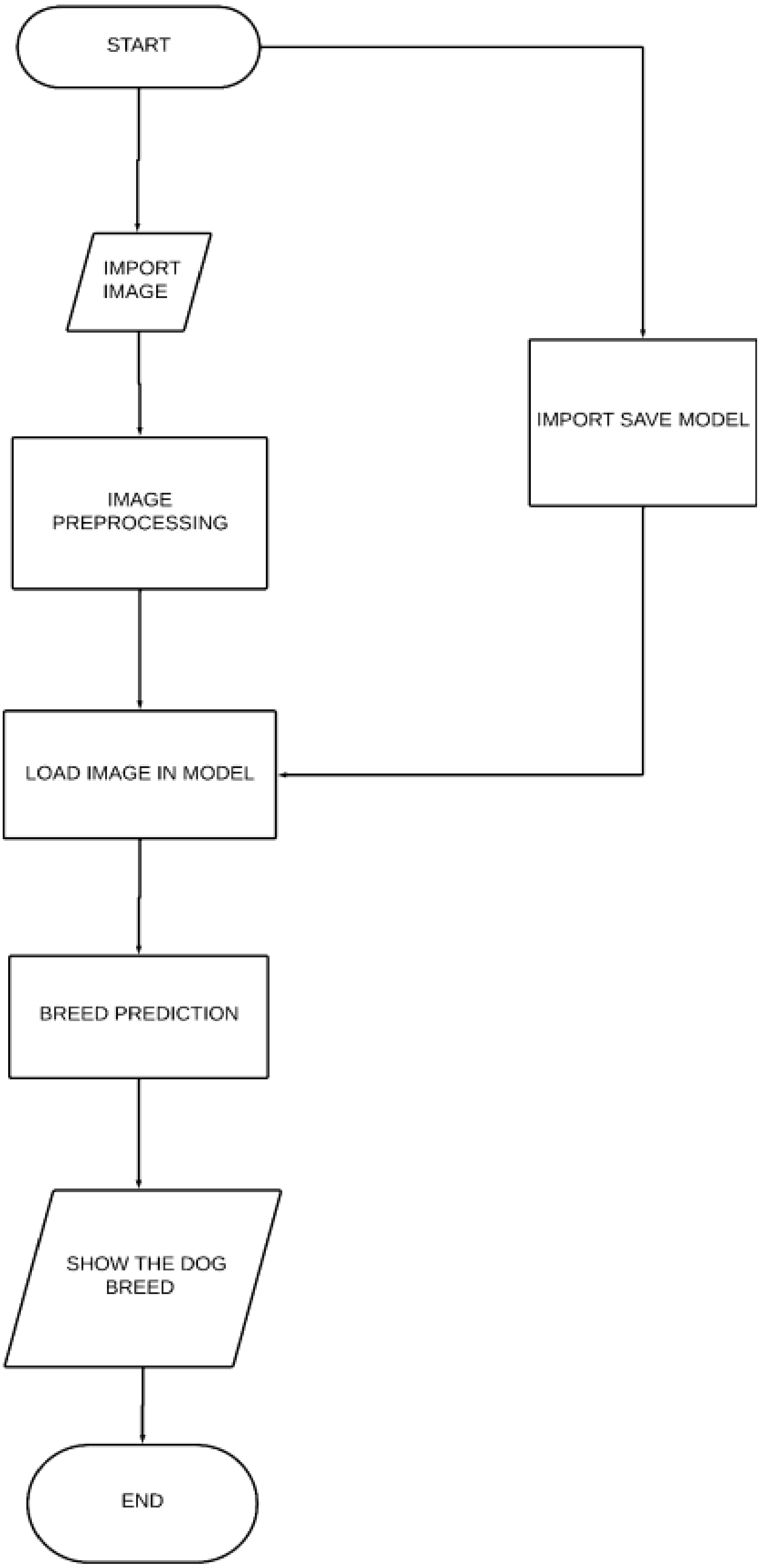
- The system must not store user images after processing.
- User interactions should comply with data privacy policies.

7 Data Flow Diagram

Level 0



Level 1


START
IMPORT IMAGE
IMAGE PREPROCESSING
IMPORT SAVE MODEL
LOAD IMAGE IN MODEL
BREED PREDICTION
SHOW THE DOG BREED
END

3. WORK DONE

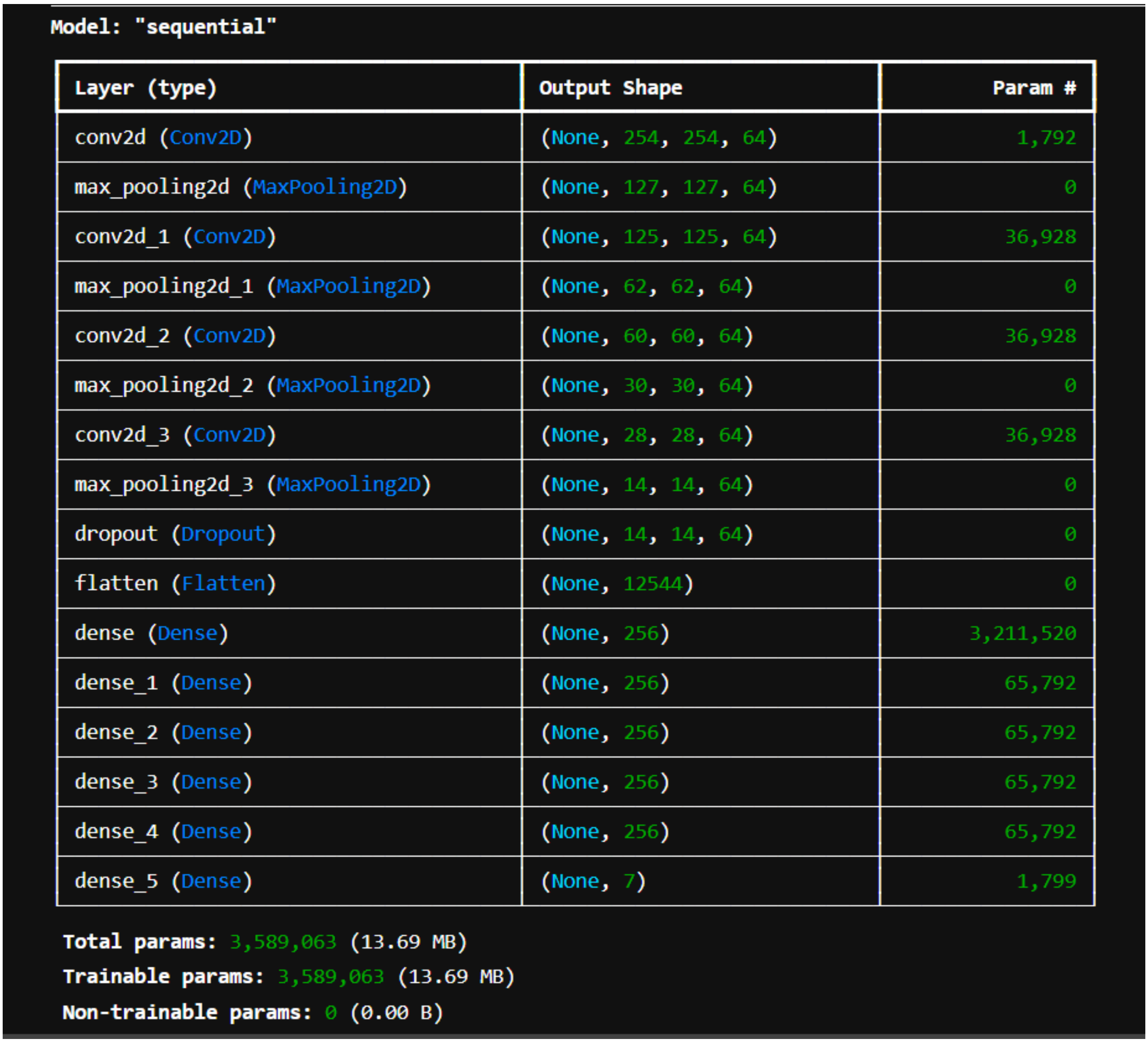
3.1 Data Preparation Module

- Collected and curated a labeled dataset comprising images of various dog breeds with proper annotations.
- Conducted data cleaning and preprocessing, including resizing, normalization, and augmentation techniques such as rotation, flipping, and scaling to enhance dataset quality and diversity.
- Split the dataset into training (70%), validation (15%), and testing (15%) sets for systematic evaluation of the model's performance.

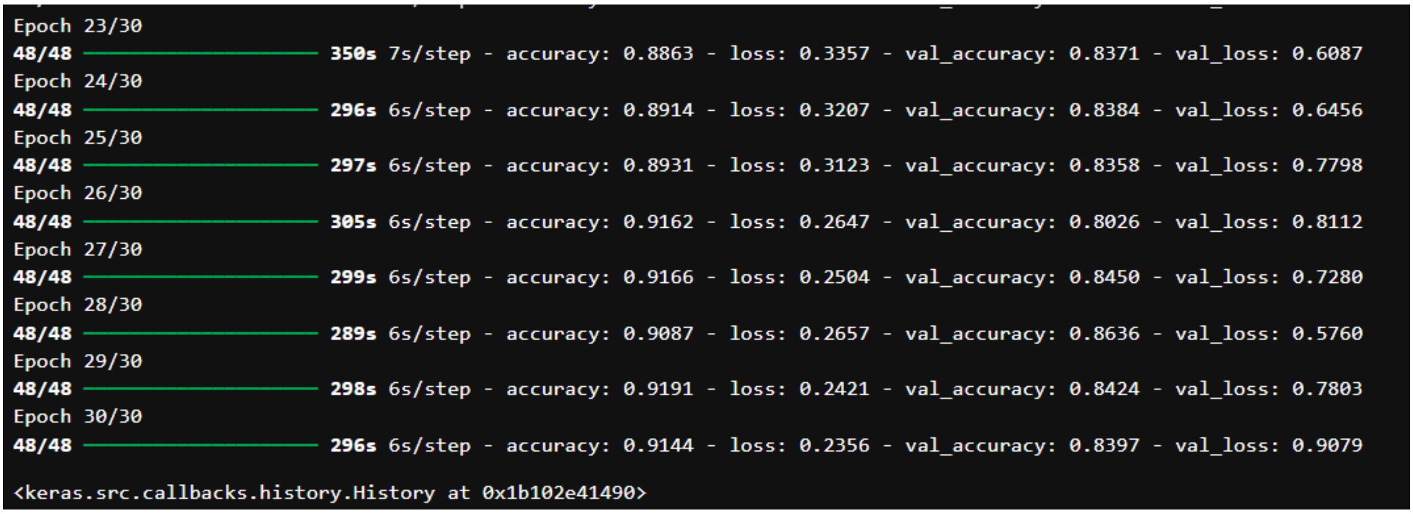
3.2 Model Development Module

- Designed and trained a Convolutional Neural Network (CNN) using TensorFlow and Keras for dog breed classification.
- Configured hyperparameters including:
  - **Epochs:** 30
  - **Batch Size:** 64
  - **Number of Batches:** 48
- Achieved a final model accuracy of **83%** on the testing set.
- Applied early stopping to prevent overfitting and incorporated dropout layers for better generalization.

3.3 SCREENSHOTS

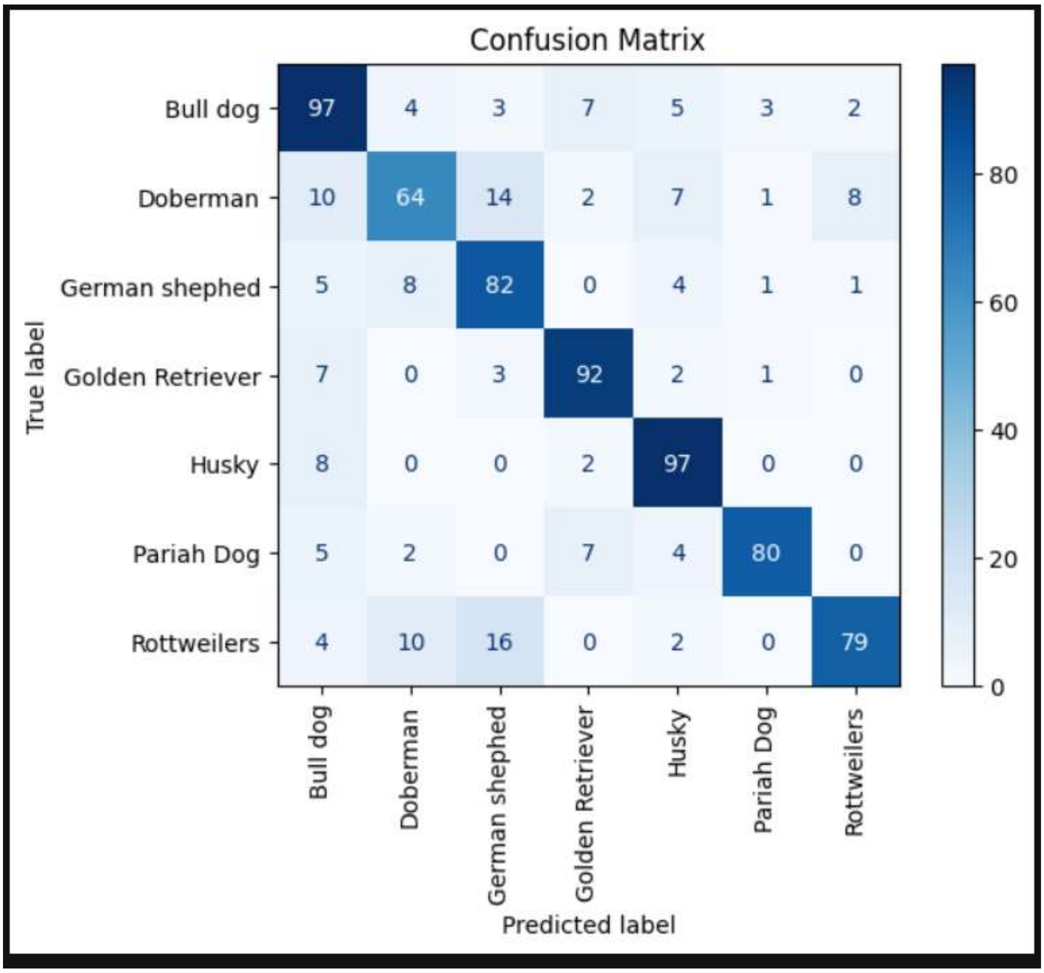


```
Model: "sequential"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃         Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                      │ (None, 254, 254, 64)        │           1,792 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d (MaxPooling2D)         │ (None, 127, 127, 64)        │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_1 (Conv2D)                    │ (None, 125, 125, 64)        │          36,928 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_1 (MaxPooling2D)       │ (None, 62, 62, 64)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_2 (Conv2D)                    │ (None, 60, 60, 64)          │          36,928 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_2 (MaxPooling2D)       │ (None, 30, 30, 64)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ conv2d_3 (Conv2D)                    │ (None, 28, 28, 64)          │          36,928 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ max_pooling2d_3 (MaxPooling2D)       │ (None, 14, 14, 64)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dropout (Dropout)                    │ (None, 14, 14, 64)          │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ flatten (Flatten)                    │ (None, 12544)               │               0 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense (Dense)                        │ (None, 256)                 │       3,211,520 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_1 (Dense)                      │ (None, 256)                 │          65,792 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_2 (Dense)                      │ (None, 256)                 │          65,792 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_3 (Dense)                      │ (None, 256)                 │          65,792 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_4 (Dense)                      │ (None, 256)                 │          65,792 │
├──────────────────────────────────────┼─────────────────────────────┼─────────────────┤
│ dense_5 (Dense)                      │ (None, 7)                   │           1,799 │
└──────────────────────────────────────┴─────────────────────────────┴─────────────────┘
 Total params: 3,589,063 (13.69 MB)
 Trainable params: 3,589,063 (13.69 MB)
 Non-trainable params: 0 (0.00 B)
```

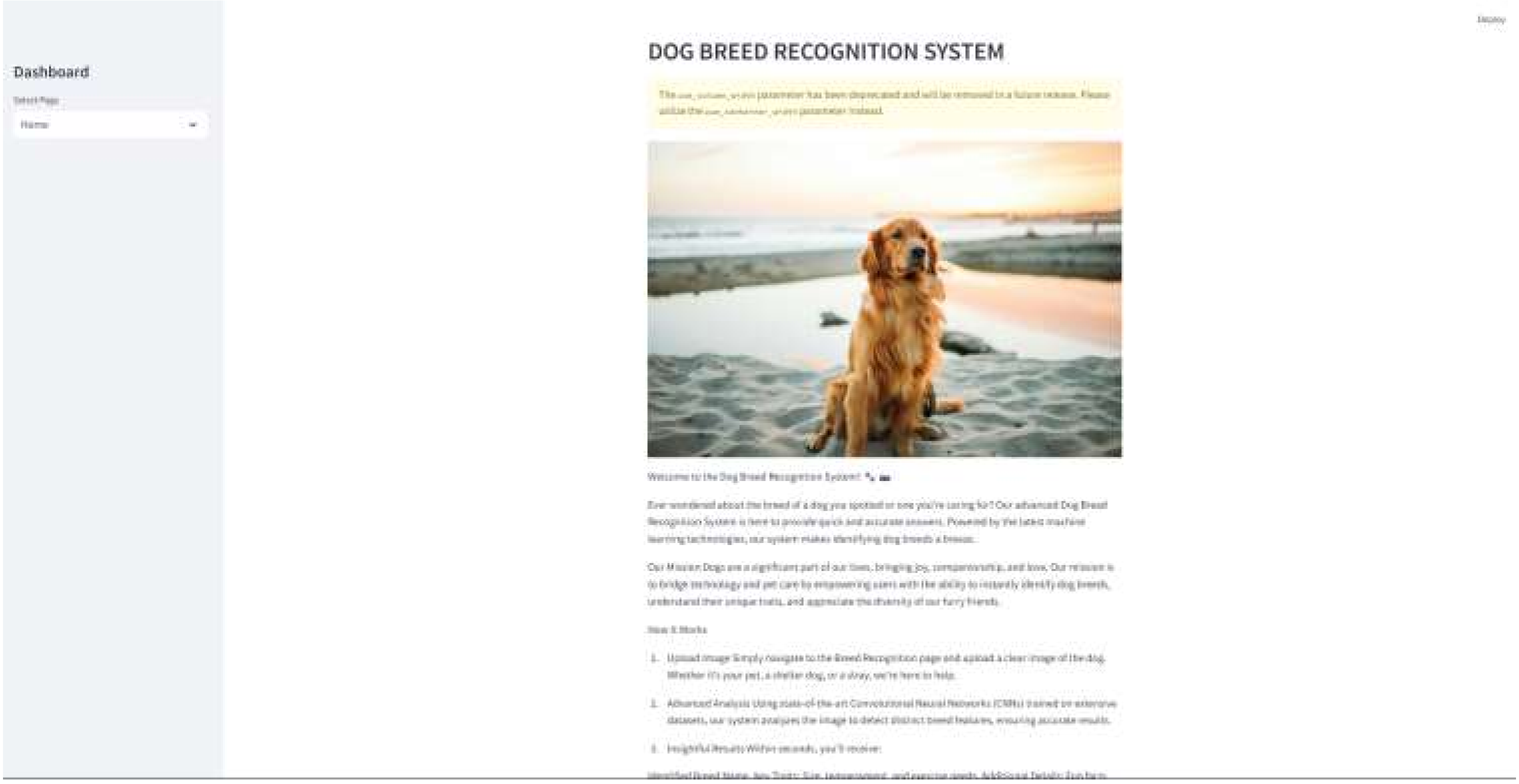


```
Epoch 23/30
48/48 ──────────────────── 350s 7s/step - accuracy: 0.8863 - loss: 0.3357 - val_accuracy: 0.8371 - val_loss: 0.6087
Epoch 24/30
48/48 ──────────────────── 296s 6s/step - accuracy: 0.8914 - loss: 0.3207 - val_accuracy: 0.8384 - val_loss: 0.6456
Epoch 25/30
48/48 ──────────────────── 297s 6s/step - accuracy: 0.8931 - loss: 0.3123 - val_accuracy: 0.8358 - val_loss: 0.7798
Epoch 26/30
48/48 ──────────────────── 305s 6s/step - accuracy: 0.9162 - loss: 0.2647 - val_accuracy: 0.8026 - val_loss: 0.8112
Epoch 27/30
48/48 ──────────────────── 299s 6s/step - accuracy: 0.9166 - loss: 0.2504 - val_accuracy: 0.8450 - val_loss: 0.7280
Epoch 28/30
48/48 ──────────────────── 289s 6s/step - accuracy: 0.9087 - loss: 0.2657 - val_accuracy: 0.8636 - val_loss: 0.5760
Epoch 29/30
48/48 ──────────────────── 298s 6s/step - accuracy: 0.9191 - loss: 0.2421 - val_accuracy: 0.8424 - val_loss: 0.7803
Epoch 30/30
48/48 ──────────────────── 296s 6s/step - accuracy: 0.9144 - loss: 0.2356 - val_accuracy: 0.8397 - val_loss: 0.9079

<keras.src.callbacks.history.History at 0x1b102e41490>
```



	Bull dog	Doberman	German shephed	Golden Retriever	Husky	Pariah Dog	Rottweilers
Bull dog	97	4	3	7	5	3	2
Doberman	10	64	14	2	7	1	8
German shephed	5	8	82	0	4	1	1
Golden Retriever	7	0	3	92	2	1	0
Husky	8	0	0	2	97	0	0
Pariah Dog	5	2	0	7	4	80	0
Rottweilers	4	10	16	0	2	0	79



DOG BREED RECOGNITION SYSTEM

The use_column_width parameter has been deprecated and will be removed in a future release. Please utilize the use_container_width parameter instead.

Welcome to the Dog Breed Recognition System! 🐾 📷

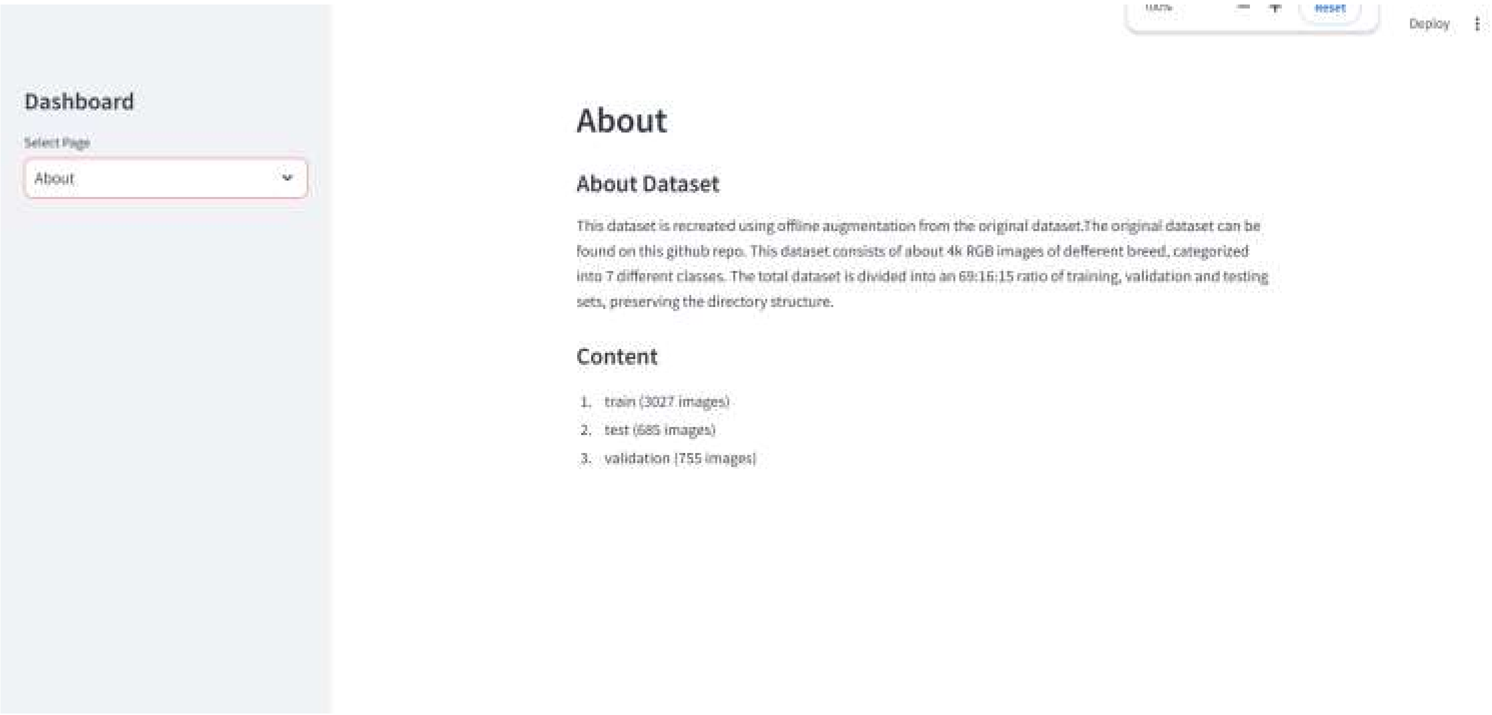
Ever wondered about the breed of a dog you spotted or one you're caring for? Our advanced Dog Breed Recognition System is here to provide quick and accurate answers. Powered by the latest machine learning technologies, our system makes identifying dog breeds a breeze.

Our Mission Dogs are a significant part of our lives, bringing joy, companionship, and love. Our mission is to bridge technology and pet care by empowering users with the ability to instantly identify dog breeds, understand their unique traits, and appreciate the diversity of our furry friends.

How It Works

1. Upload Image Simply navigate to the Breed Recognition page and upload a clear image of the dog. Whether it's your pet, a shelter dog, or a stray, we're here to help.
2. Advanced Analysis Using state-of-the-art Convolutional Neural Networks (CNNs) trained on extensive datasets, our system analyzes the image to detect distinct breed features, ensuring accurate results.
3. Insightful Results Within seconds, you'll receive:

Identified Breed Name. Key Traits: Size, temperament, and exercise needs. Additional Details: Fun facts



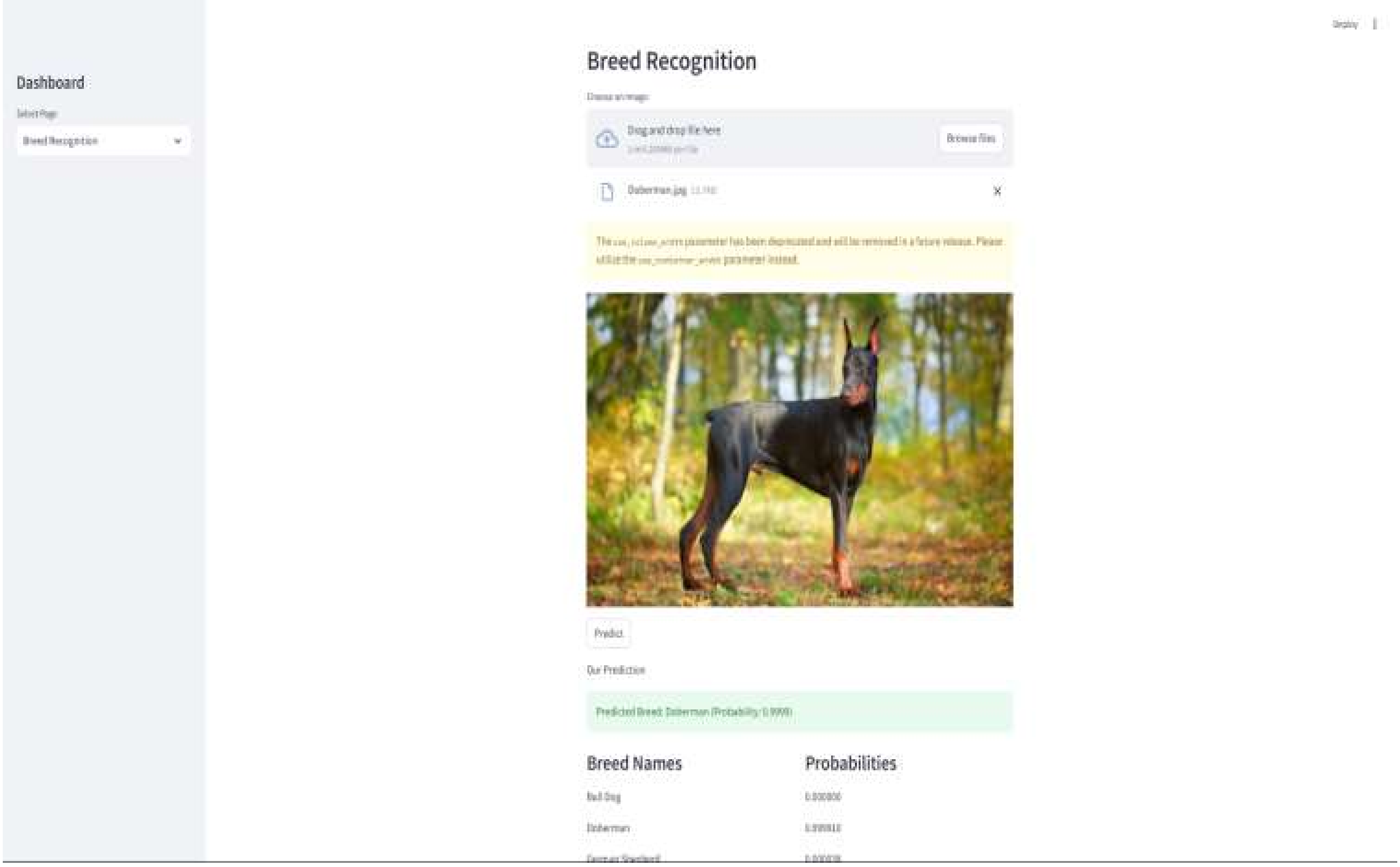
About

About Dataset

This dataset is recreated using offline augmentation from the original dataset.The original dataset can be found on this github repo. This dataset consists of about 4k RGB images of defferent breed, categorized into 7 different classes. The total dataset is divided into an 69:16:15 ratio of training, validation and testing sets, preserving the directory structure.

Content

1. train (3027 images)
2. test (685 images)
3. validation (755 images)



Dashboard

Select Page

Breed Recognition

Breed Recognition

Choose an image

Drag and drop file here

Limit 200MB per file

Doberman.jpg 11.7KB

The use_column_width parameter has been deprecated and will be removed in a future release. Please utilize the use_container_width parameter instead.

Our Prediction

Predicted Breed: Doberman (Probability: 0.9998)

Breed Names	Probabilities
Bull Dog	0.000000
Doberman	0.999810
German Shepherd	0.000078

4. CONCLUSION

The Dog Breed Recognition Model leverages advanced deep learning techniques, specifically a Convolutional Neural Network (CNN), to accurately classify various dog breeds. By employing robust data preprocessing methods and an optimized training pipeline, the model achieves a commendable accuracy of **83%**, demonstrating its effectiveness in real-world applications.

This project highlights the potential of integrating modern machine learning frameworks such as TensorFlow and Keras with preprocessing tools to build an intelligent image recognition system. The structured modular approach, encompassing data preparation, model development, and image recognition, ensures scalability and adaptability for future improvements.

The Dog Breed Recognition Model successfully meets its objective of providing an automated and efficient solution for breed identification, paving the way for advancements in pet care, veterinary diagnostics, and animal research. It stands as an example of how technology can transform traditional processes into precise and innovative solutions.

5. Future Scope

I. Model Accuracy Enhancement:

- Increase model accuracy beyond the current 83% by incorporating larger and more diverse datasets.
- Employ advanced architectures like Transfer Learning using pre-trained models such as ResNet or EfficientNet.

II. Real-Time Recognition on Devices:

- Optimize the model for deployment on mobile and embedded devices, enabling real-time breed recognition directly through smartphone cameras or edge devices.

III. Multi-Breed Detection:

- Expand the system's capabilities to identify multiple dog breeds within a single image, making it applicable to group scenarios like shelters or parks.

IV. Integration with Veterinary Systems:

- Collaborate with veterinary platforms to use breed identification for improving diagnostics and personalized pet care recommendations.

V. User-Friendly Application Development:

- Develop a mobile or web application for end-users to upload images and receive breed predictions, ensuring a seamless and intuitive experience.

VI. Cloud Integration for Scalability:

- Integrate cloud-based services for processing and storing large volumes of data, ensuring scalability and faster response times for real-time requests.

VII. Interactive Education Tools:

- Create an interactive feature to educate users about different breeds, their characteristics, and care requirements, enhancing pet owners' knowledge.

VIII. Continuous Model Training:

- Implement a feature to update and retrain the model periodically using newly collected data to maintain performance and adapt to emerging needs.

BIBLOGRAPHY

- TensorFlow Documentation [Online] // TensorFlow Developers. – Google. – https://www.tensorflow.org/
- Keras API Reference [Online] // – https://keras.io/api/data_loading/image/
- Dataset Source [Online] // Bing Image – https://pypi.org/project/bing-image-downloader/
- Python Official Documentation [Online] // Python Software Foundation. – https://docs.python.org