Dog Breed Identification using Transfer Learning

SRI VASAVI DEGREE COLLEGE

Team ID : LTVIP2026TMIDS46482

Team Leader : Shaik Alekhya Vijaya

REG NO: SBAP0040431

Team member : Bara Lakshmi

REG NO: SBAP0040465

Team member : Pinninti Sridurga

REG NO: SBAP0040422

Team member : Mamidi Lakshmi

REG NO:SBAP0040471

❖ 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.

❖ Scenarios:

- ### Pet Adoption Platform:

  An online pet adoption platform wants to improve the user experience by automatically categorizing the dog breeds of animals available for adoption based on uploaded images. This helps potential adopters quickly find dogs that match their preferences.

- ### Lost Dog Identification:

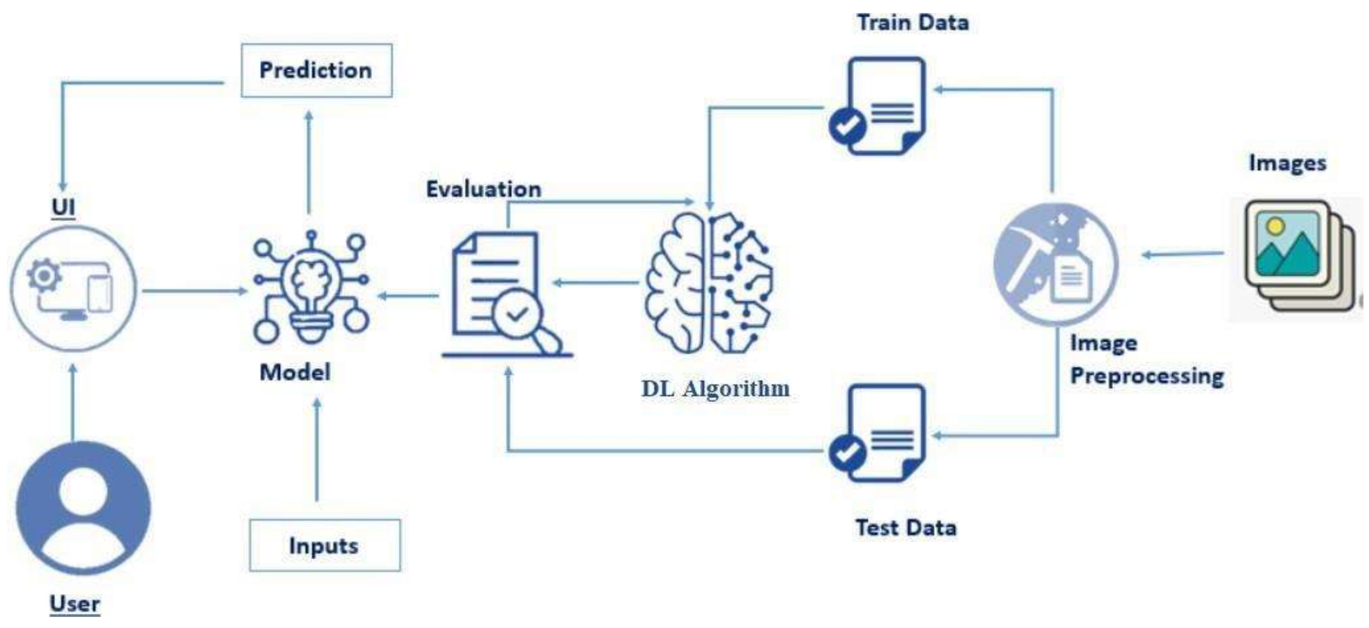
  A person finds a lost dog and wants to help reunite it with its owner. However, they are unsure of the dog's breed, making it challenging to create an accurate description for missing pet posters or online announcements.

- ### Veterinary Diagnosis Support:

  A veterinarian needs assistance in identifying the breed of a mixed-breed dog brought in for a health checkup. Knowing the breed can provide valuable insights into potential genetic predispositions to certain health conditions or behavioral traits.

  Dog breed identification using transfer learning can significantly support veterinary diagnosis by providing accurate and rapid breed classification, which is essential for identifying breed-specific health risks, genetic disorders, and treatment considerations. Many canine diseases, such as hip dysplasia, progressive retinal atrophy, cardiac disorders, and skin conditions, are strongly associated with specific breeds. By automatically determining the breed from an image, the system can assist veterinarians in making faster preliminary diagnoses, suggesting appropriate medical tests, and designing personalized treatment plans. This intelligent support system reduces diagnostic time, improves clinical decision-making, and enhances overall animal healthcare quality, especially in clinics with limited access to breed records or expert knowledge.

❖ Technical Architecture:



❖ Prerequisites:

To complete this project, you must require the following software's, concepts, and packages

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related Applications It can be installed on Windows, Linux, and macOS. . Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupyter notebook and VS code.

To build Machine learning models you must require the following packages

➢ NumPy:

- It is an open-source numerical Python library. It contains a multidimensional array and matrix data structures and can be used to

➢ Scikit-learn:

- It is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbors,
- and it also supports Python numerical and scientific libraries like NumPy and SciPy

➢ Flask:

- Web framework used for building Web applications

➢ Python packages:

- open anaconda prompt as administrator
- Type “pip install numpy” and click enter.
- Type “pip install scikit-learn” and click enter.
- Type “pip install pandas” and click enter.
- Type “pip install tensorflow==2.12.0” and click enter.
- Type “pip install keras==2.12.0” and click enter.
- Type “pip install Flask” and click enter.

➢ DeepLearningConcepts:

- **CNN:** a convolutional neural network is a class of deep neural networks, most commonly applied to analyzing visual imagery.
- **VGG19:** VGG19 is a deep convolutional neural network architecture for image classification, consisting of 19 layers with small convolution filters.
- **Flask:** Flask is a popular Python web framework, meaning it is a third-party Python library used for developing web applications.
- If you are using Pycharm IDE, you can install the packages through the command prompt and follow the same syntax as above

➢ Basic Knowledge Requirements:

- Concepts of classification, training, testing, and validation
- Understanding of Convolutional Neural Networks (CNNs)
- Working knowledge of TensorFlow + Keras or PyTorch
- A labeled dataset containing dog images with breed names
- GPU-enabled system (NVIDIA CUDA support recommended)
- Understanding of supervised learning
- Minimum 8 GB RAM (16 GB preferred)
- Kaggle Dog Breed Identification Dataset

❖ Project Objectives:

➢ By the end of this project you will:

- Know fundamental concepts and techniques of Convolutional Neural Network.
- Gain a broad understanding of image data.
- To optimize model performance by fine-tuning selected layers of the pre-trained network.
- Know how to pre-process/clean the data using different data preprocessing techniques.
- To preprocess and augment the dataset to enhance model generalization and robustness.
- know how to build a web application using the Flask framework.
- To develop a user-friendly interface or application for real-time or offline dog breed prediction.
- To compare the performance of multiple transfer learning models and select the most efficient one.
- To ensure scalability so that additional dog breeds can be added to the system in the future.
- To reduce computational cost and training time by leveraging knowledge from pre-trained deep learning models.
- To demonstrate the practical applicability of the system in areas such as pet adoption services, veterinary assistance, and animal welfare monitoring.

❖ Project Flow:

- The user interacts with the UI (User Interface) to choose the image.
- The chosen image is analysed by the model which is integrated with flask application.
- CNN Models analyse the image, then prediction is showcased on the Flask UI. To accomplish this, we must complete all the activities and tasks listed below.
- Identify the need for an automated system to classify dog breeds from images.
- Apply techniques such as rotation, flipping, zooming, and cropping to improve model robustness.
- Deploy the model as a web or mobile application for real-time dog breed prediction.

➢ Data Collection :

- Create Train and Test Folders.
- **Data Preprocessing** points
- **Training & Evaluation** steps
- **Model Selection & Transfer Learning** flow
- Split the dataset into **training, validation, and testing** sets
- Store data in a format suitable for **transfer learning models**
- Repare dataset to support **data augmentation** during training

➢ Data Preprocessing:

- Import the ImageDataGenerator library
- Encode dog breed labels into **numerical form**
- Handle **class imbalance** using augmentation or resampling techniques
- Shuffle the dataset to **avoid training bias**
- Normalize pixel values to **improve model convergence**
- Configure ImageDataGenerator class
- ApplyImageDataGenerator functionality to Train dataset and Test dataset

➢ Model Building:

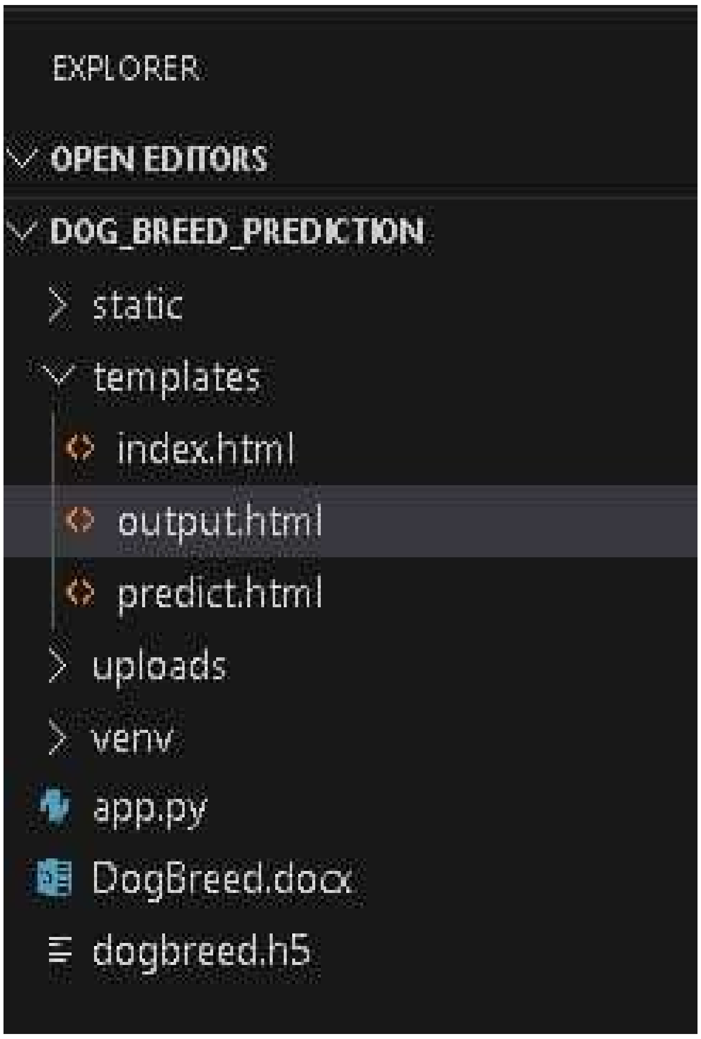
- Import the model building Libraries.
- Importing the VGG19.
- Initializing the model
- Adding Fully connected Layer
- Configure the Learning Process
- Training and Testing the model.
- Save the Model

➢ Application Building:

- Deploy the application using suitable platforms (web/mobile/local)
- Design a **user-friendly interface** for dog breed identification
- Integrate the **trained transfer learning model** into the application
- Implement an **image upload feature** for users to input dog images
- Perform **image preprocessing** on user input before prediction
- Use the model to **predict the dog breed** from the uploaded image
- Handle **invalid or unsupported image inputs** gracefully
- Optimize application performance for **fast inference**

❖ Project Structure:

➢ **Create a Project folder which contains files as shown below.**



```
EXPLORER
∨ OPEN EDITORS
∨ DOG_BREED_PREDICTION
  > static
  ∨ templates
      index.html
      output.html
      predict.html
  > uploads
  > venv
  app.py
  DogBreed.docx
  dogbreed.h5
```

- We are building a Flask Application that needs HTML pages stored in the templates.
- folder and a python script app.py for server-side scripting
- we need the model which is saved and the saved model in this content is a dogbreed.h5
- templates folder contains index.html, predict.html & output.html pages.
- The Static folder contains css file, images.

❖ Data Collection & Image Preprocessing:

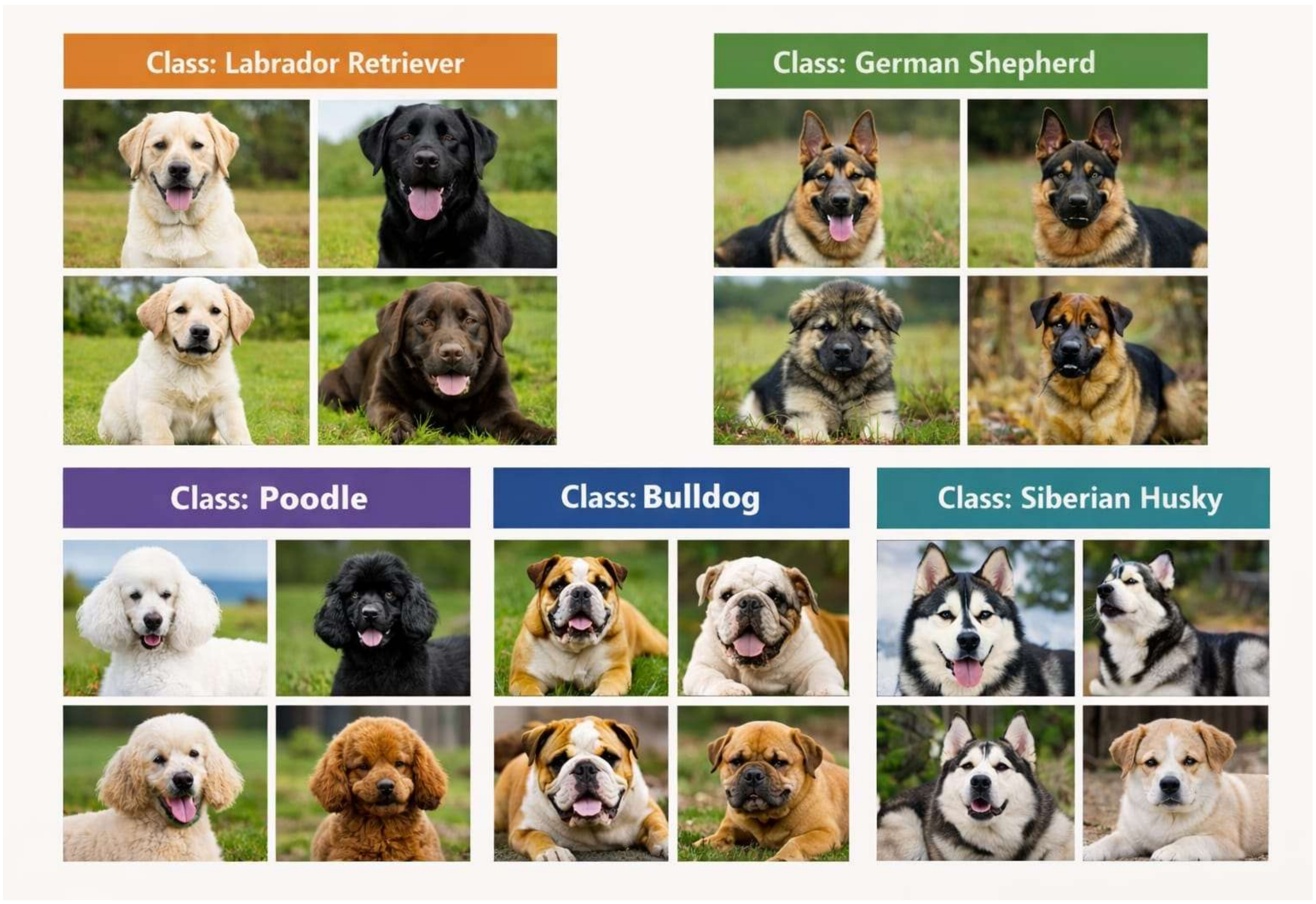
In this milestone First, we will collect images of Dog Breeds then organized into subdirectories based on their respective names as shown in the project structure. Create folders of types of Dog Breeds that need to be recognized. In this project, we have collected images of 20 types of Images like affenpinscher, beagle, appenzeller, basset, bluetick, boxer, cairn, doberman, german_shepherd, golden_retriever, kelpie, komondor, leonberg,
mexican_hairless, pug, redbone, shih-tzu, toy_poodle, vizsla, whippet they are saved in the respective sub directories with their respective names.

Download the Dataset - https://www.kaggle.com/competitions/dog-breed-identification/data?select=train

In Image Processing, we will be improving the image data that suppresses unwilling distortions or enhances some image features important for further processing, although perform some geometric transformations of images like rotation, scaling, translation, etc.

➢ Organizing the Images into Different Classes:

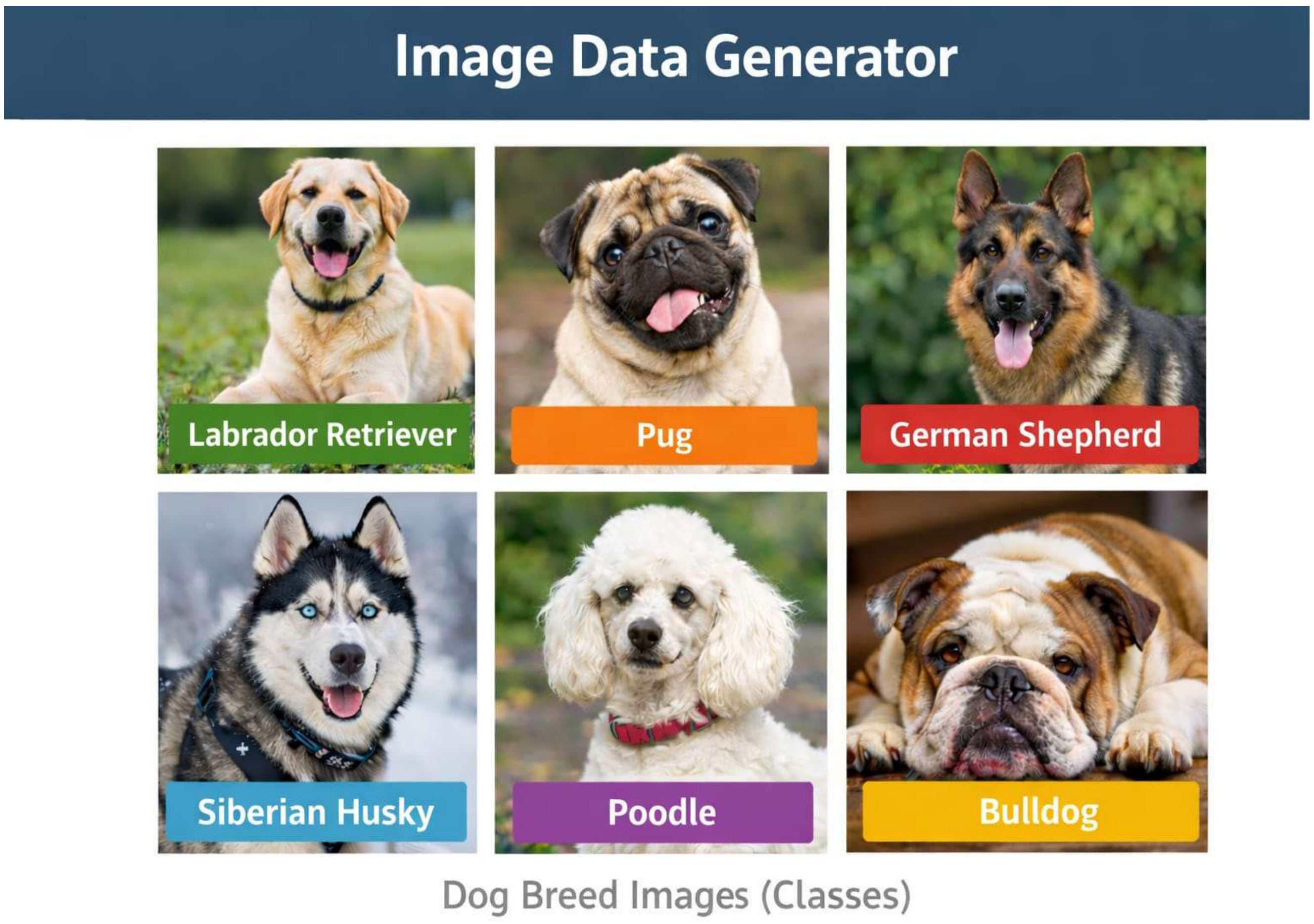
The Images should be organized based on the Image id's. So that Training the Model will be simpler. For each image ID (image_id) in the current breed, the source path is formed by combining the dataset_dir and the image filename (f'{image_id}.jpg'). The destination path is formed by combining the breed_folder and the same image filename. shutil.copyfile() is used to copy the image from the source path to the destination path.



- Store the organized dataset in a structured directory format compatible with deep learning frameworks.
- Create separate folders for each dog breed to represent different classes.
- Apply data augmentation techniques such as rotation, flipping, zooming, and brightness adjustment.
- Normalize pixel values to standardize image input for the model.
- Verify class labels and folder hierarchy before feeding the dataset into the transfer learning model.

➢ Import the ImageDataGenerator library:

- Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.
- The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class.
- Let us import the ImageDataGenerator class from tensorflow Keras.

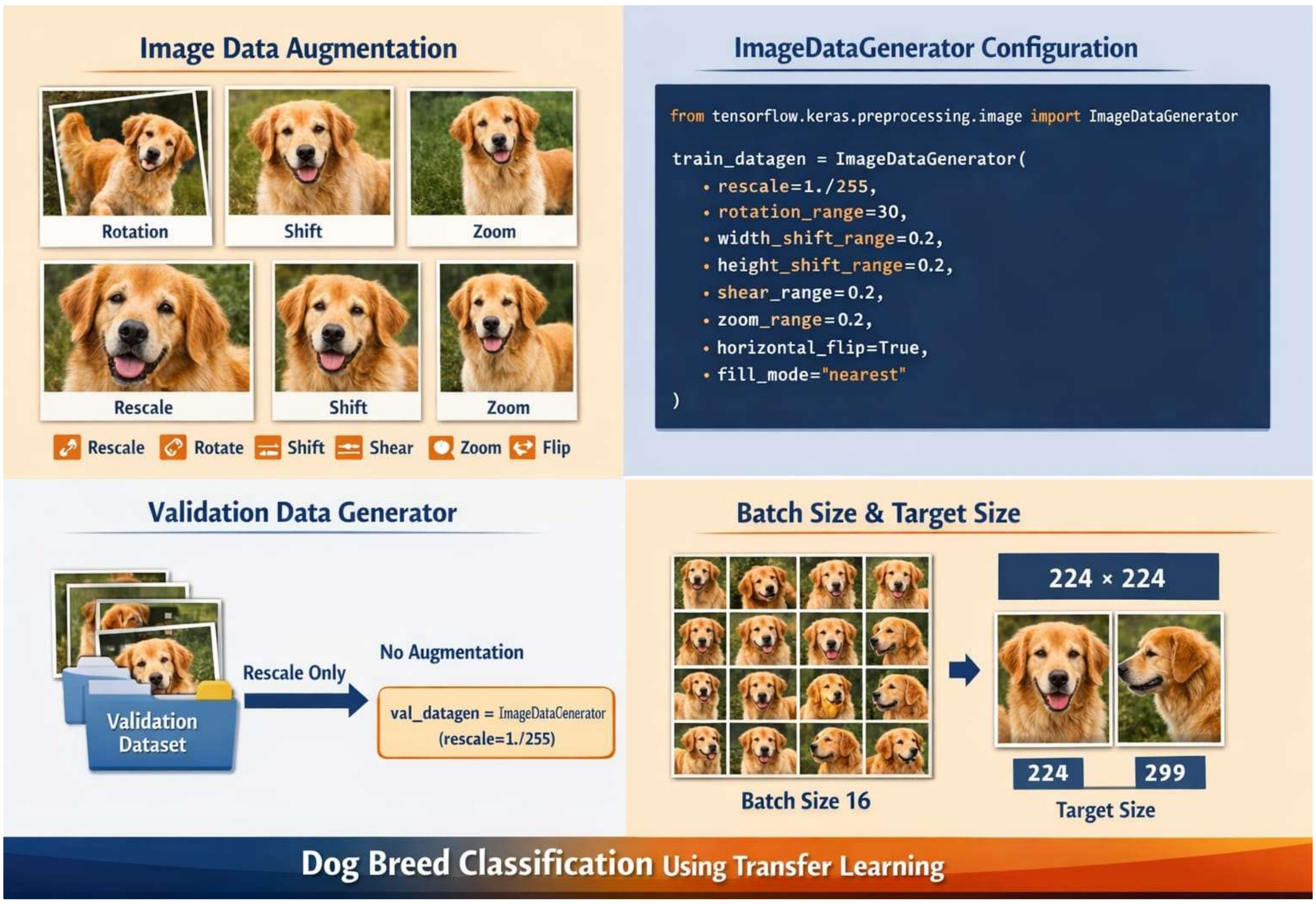


- Apply transformations such as rescaling, rotation, zoom, width & height shift, shear, and horizontal flipping to improve model generalization.
- Prepare validation and test generators without augmentation, using only rescaling.
- Use ImageDataGenerator to perform real-time image augmentation for training data.
- Ensure shuffling is enabled during training to reduce model bias.
- Set suitable batch size and class mode (categorical) for multi-class dog breed classification.

➢ Configure ImageDataGenerator class:

ImageDataGenerator class is instantiated and the configuration for the types of data augmentation. There are five main types of data augmentation techniques for image data; specifically:

- mage shifts via the width_shift_range and height_shift_range arguments.
- The image flips via the horizontal_flip and vertical_flip arguments.
- Image rotations via the rotation_range argument
- Image brightness via the brightness_range argument.
- Image zoom via the zoom_range argument.



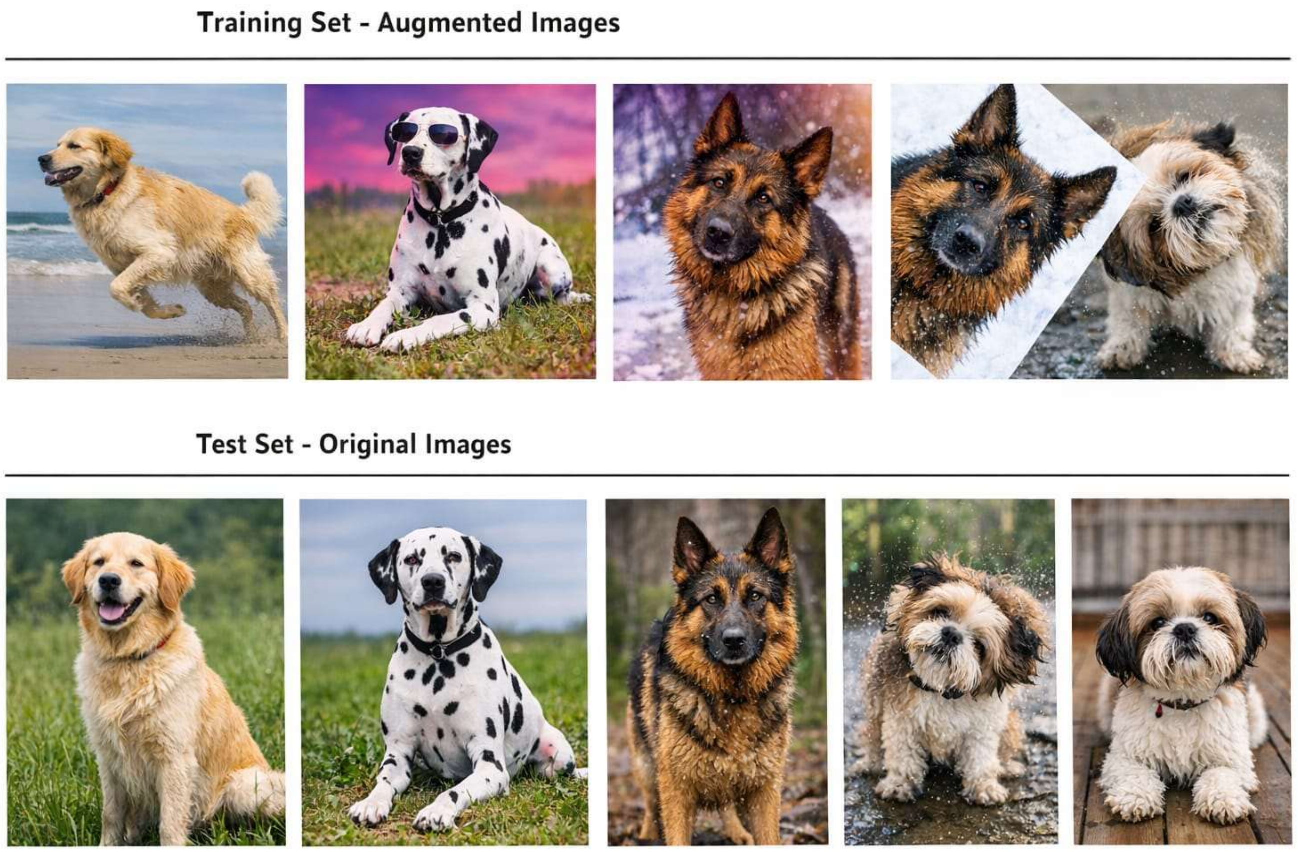
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
    • rescale=1./255,
    • rotation_range=30,
    • width_shift_range=0.2,
    • height_shift_range=0.2,
    • shear_range=0.2,
    • zoom_range=0.2,
    • horizontal_flip=True,
    • fill_mode="nearest"
)
```

```
val_datagen = ImageDataGenerator
(rescale=1./255)
```

- Apply real-time data augmentation to increase dataset diversity and reduce overfitting.
- Apply width_shift_range=0.2 and height_shift_range=0.2 to simulate camera movement and object displacement.
- Configure a separate generator for validation with only rescaling, without augmentation, to ensure unbiased evaluation.
- Set fill_mode='nearest' to fill missing pixels after transformations.
- Set class_mode='categorical' for multi-class dog breed classification.

➢ Apply ImageDataGenerator functionality to Trainset and Test set:

Let us apply ImageDataGenerator functionality to Train set and Test set by using the following code. For Training set using flow_from_directory function.
This function will return batches of images from the subdirectories affenpinscher, beagle, appenzeller, basset, bluetick, boxer, cairn, doberman, german_shepherd, golden_retriever, kelpie, komondor, leonberg, mexican_hairless, pug, redbone, shih-tzu, toy_poodle, vizsla, whippe, together with labels 0 to 19.



Arguments:

- directory: Directory where the data is located. If labels are "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
- batch_size: Size of the batches of data which is 32.
- target_size: Size to resize images after they are read from disk.
- **class_mode:**
- **‘**int': means that the labels are encoded as integers (e.g. for sparse_categorical_crossentropy loss).

❖Model Building:

Now it's time to build our Convolutional Neural Networking using vgg19 which contains an input layer along with the convolution, max-pooling, and finally an output layer.

Link: https://thesmartbridge.com/documents/spsaimldocs/CNNflow.pdf

- Import the required deep learning libraries like TensorFlow, Keras, NumPy, and Matplotlib.
- Freeze the convolutional layers of the base model to preserve learned features.
- Configure the input image shape according to the selected model architecture.
- Apply data augmentation during training using ImageDataGenerator.
- Train the model using the training dataset and validate using the validation dataset.
- Fine-tune the model by unfreezing selected layers of the pre-trained network.
- Recompile the model with a lower learning rate for fine-tuning.
- Retrain the model to improve feature learning and classification accuracy.
- Test the model using new unseen dog images.
- Optimize model performance using hyperparameter tuning.
- Convert the model for deployment platforms such as TensorFlow Lite or ONNX.
- Integrate the trained model into a web or mobile application for real-time dog breed identification.

➢Importing the Model Building Libraries:

o Importing the necessary libraries:

- They provide high-level APIs that make complex neural networks easy to design.
- Libraries like TensorFlow and Keras are widely used for image-based applications.
- Pre-trained CNN architectures are available through these libraries for transfer learning.
- They support GPU and CPU acceleration, improving training speed.
- Optimizers are imported to control learning rate and convergence of the model.
- Loss functions help in calculating the difference between predicted and actual labels.
- Evaluation metrics are used to monitor training progress.
- Libraries ensure model reproducibility and consistency.
- They reduce manual implementation errors and improve code readability.

➢ Importing the VGG19 model:

To initialize the VGG19 model, the weights are usually pre-trained on the ImageNet dataset, which is a large-scale dataset of images belonging to 1,000 different categories. These pre-trained weights can be downloaded from the internet, and they can be used as a starting point to fine-tune the model for a specific task, such as object recognition or classification.

- mport the required deep learning libraries such as TensorFlow and Keras.
- Load the VGG19 pre-trained model with ImageNet weights.
- Exclude the top fully connected layers by setting include_top = False.
- Freeze the convolutional base layers to retain learned features.
- Add custom classification layers suitable for dog breed classification.
- Compile the model using an appropriate optimizer like Adam.
- Use categorical cross-entropy as the loss function for multi-class classification.
- Summarize the model architecture to verify layer configuration.
- Prepare the model for training on the dog breed dataset.

➢ Initializing the model:

- The model will be initialized with the pre-trained weights from the ImageNet dataset, and the last fully connected layer will be excluded from the model architecture.
- The loop that follows freezes the weights of all the layers in the VGG19 model by setting
- `i.trainable=False` for each layer in the model. This is done to prevent the weights from being updated during training, as the model is already pre-trained on a large dataset.
- Finally, a `Flatten()` layer is added to the output of the VGG19 model to convert the output tensor into a 1D tensor.
- The resulting model can be used as a feature extractor for transfer learning or as a starting point for building a new model on top of it.
- Freeze the weights of the base model layers to prevent them from updating during initial training.
- Ensure input image size and preprocessing match the requirements of the chosen pre-trained model.
- Optionally enable **fine-tuning** by unfreezing top layers of the base model after initial convergence.

➢ Adding Fully connected Layers:

A dense layer is a deeply connected neural network layer. It is the most common and frequently used layer. The number of neurons in the Dense layer is the same as the number of classes in the training set.The neurons in the last De nse layer, use softmax activation to convert their outputs into respective probabilities.
Understanding the model is a very important phase to properly use it for training and prediction purposes. Keras provides a simple method, summary to get the full information about the model and its layers.

- For information regarding CNN Layers refer to the link
  Link: https://victorzhou.com/blog/intro-to-cnns-part-1/
- As the input image contains three channels, we are specifying the input shape as (128,128,3).
- We are adding a output layer with activation function as "softmax".
- The number of neurons in dense layers is chosen based on model complexity and dataset size (e.g., 256 or 512 units).
- Ensure the fully connected layers are trainable while the base model remains frozen during initial training.
- This architecture helps the model adapt pre-trained features to the specific task of dog breed identification.

➢ Configure The Learning Process:

- The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find errors or deviations inthe learning process. Keras requires a loss function during the model compilation process.
- Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer.
- Metrics are used to evaluate the performance of your model. It is similar to the loss function, but not used in the training process.
- Proper configuration ensures faster convergence, higher accuracy, and reduced overfitting.
- It is essential for reliable and stable training of the dog breed classification model.
- Configuring the learning process means setting up how the model will learn from data.
- Optimizer selection determines how model weights are updated during training (e.g., Adam, SGD).
- Callbacks like EarlyStopping or ModelCheckpoint can improve training efficiency and save the best model.

➢ Train The model:

Now, let us train our model with our image dataset. The model is trained for 6 epochs and after every epoch, the current model state is saved if the model has the least loss encountered till that time. We can see that the training loss decreases in almost every epoch till 30 epochs and probably there is further scope to improve the model.

fit_generator functions used to train a deep learning neural network.

o Arguments:

- steps_per_epoch: it specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of steps_per_epoch as the total number of samples in your dataset divided by the batch size.
- Epochs: an integer and number of epochs we want to train our model for.
- validation_data can be either:
- an inputs and targets list
- a generator
- an inputs, targets, and sample_weights list which can be used to evaluate the loss and metrics for any model after any epoch has ended.
- validation_steps: only if the validation_data is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is
- stopped at every epoch and its value is calculated as the total number of validation data points in your dataset divided by the validation batch size.
- Evaluate training progress using accuracy and loss curves to ensure proper learning behavior.

➢ Save the Model:

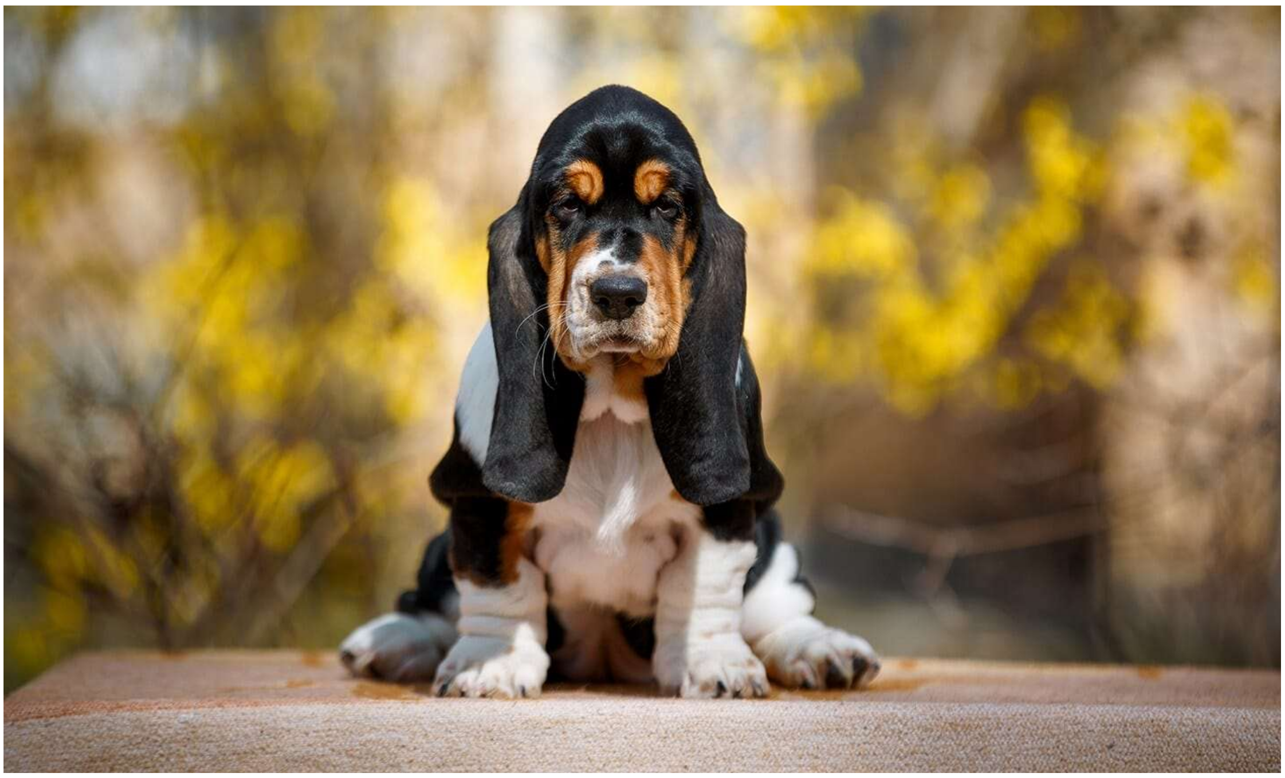
o The model is saved with .h5 extension as follows.

- An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.
- Saving the model means storing the trained neural network for future use.
- It preserves the learned weights and architecture of the model.
- Saved models can be reloaded without retraining, saving time and resources.
- Helps in deploying the model in real-world applications such as web or mobile apps.
- Prevents loss of training progress due to system failure or interruption.
- Allows the best-performing model to be stored during training
- Useful for comparing performance across different experiments.
- Essential for production, inference, and long-term usage.

➢ Test The model:

- Evaluation is a process during the development of the model to check whether the model is the best fit for the given problem and corresponding data.
- esting the model is the process of evaluating its performance on unseen data.
- A separate test dataset is used that was not part of training.
- Test images are only preprocessed, not augmented, to ensure fair evaluation.
- The model predicts the dog breed class for each test image.
- Performance metrics such as accuracy, precision, recall, and loss are measured.
- esting helps identify overfitting or underfitting.
- Ensures the model generalizes well to real-world dog images.
- Provides confidence in the model before deployment.
- Helps compare results with other models or approaches.
- Final test results indicate the reliability and effectiveness of the trained model.

- **Taking an image as input and checking the results**
  By using the model we are predicting the output for the given input image. The predicted class index name will be printed here.



❖ Application Building:

Now that we have trained our model, let us build our flask application which will be running in our local browser with a user interface.
In the flask application, the input parameters are taken from the HTML page These factors are then given to the model to know to predict the type of Colon Diseases and showcased on the HTML page to notify the user. Whenever the user interacts with the UI and selects the "Inspect" button, the next page is opened where the user chooses the image and predicts the output.

o Problem Statement:

- Identifying dog breeds from images is challenging due to similarities between breeds, variations in pose, lighting, and background.
- Manual identification requires expert knowledge and is time-consuming.

o Objective:

- To build an application that automatically identifies the **breed of a dog from an image** using **transfer learning**.
- To achieve high accuracy with reduced training time and limited dataset.

o Applications:

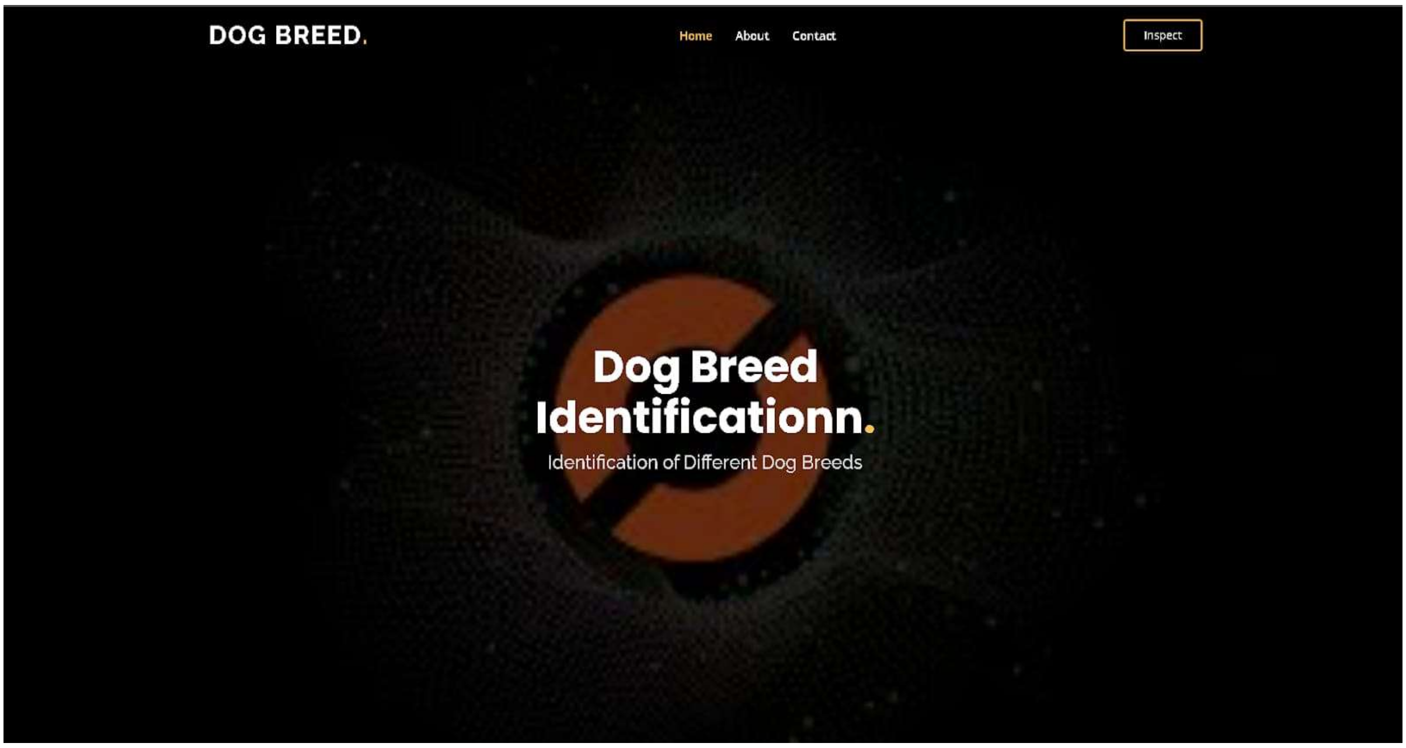
- Pet adoption systems
- Veterinary diagnosis support
- Animal shelters and rescue centers
- Educational tools for animal lovers

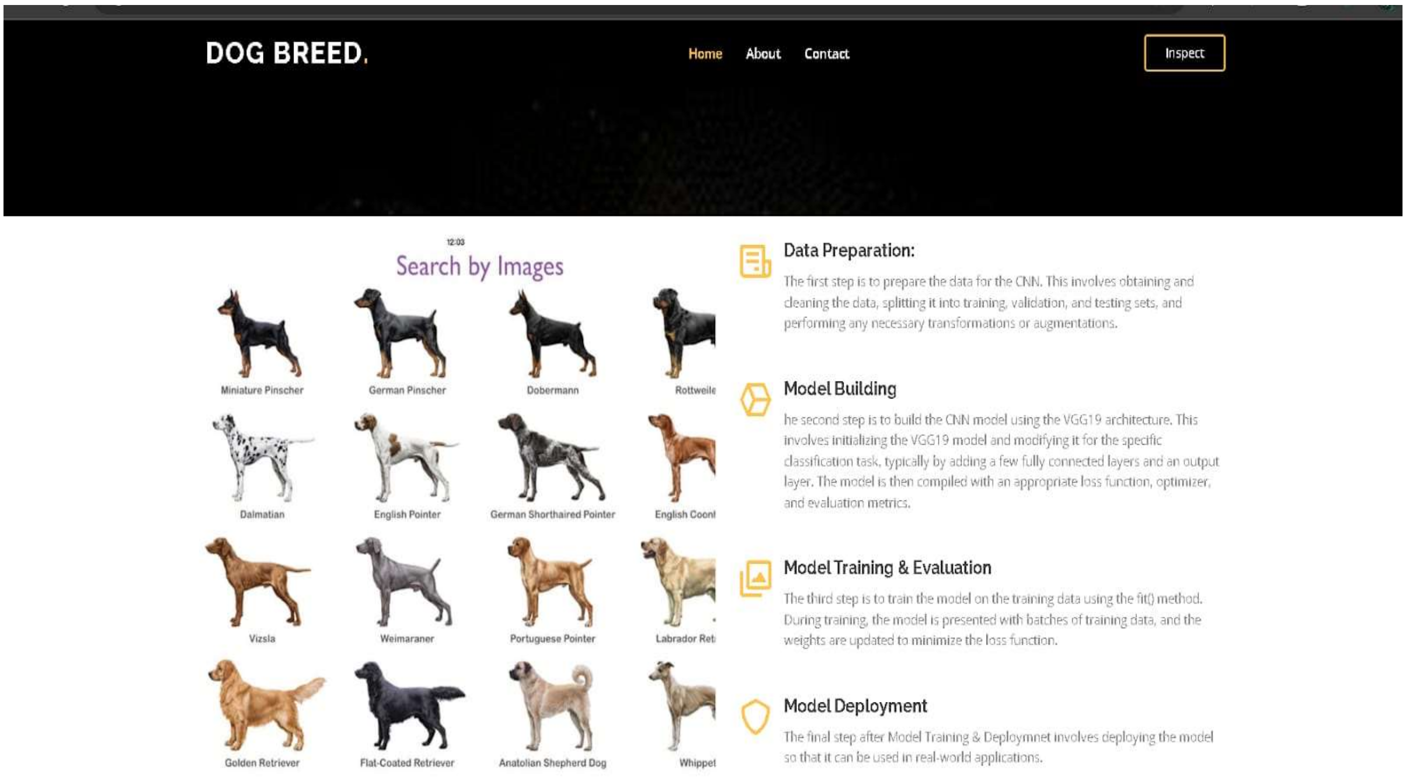
o Application Development:

- Web or mobile application where users upload a dog image.
- Backend processes the image and predicts the breed.
- Display predicted breed with confidence score.

o Future Enhancements:

- Multi-breed detection in a single image.
- Mobile app integration.
- Real-time camera-based detection.
- Breed health and care recommendations.


DOG BREED.
Home
About
Contact
Inspect
Dog Breed Identificationn.
Identification of Different Dog Breeds



Data Preparation:

The first step is to prepare the data for the CNN. This involves obtaining and cleaning the data, splitting it into training, validation, and testing sets, and performing any necessary transformations or augmentations.

Model Building

he second step is to build the CNN model using the VGG19 architecture. This involves initializing the VGG19 model and modifying it for the specific classification task, typically by adding a few fully connected layers and an output layer. The model is then compiled with an appropriate loss function, optimizer, and evaluation metrics.

Model Training & Evaluation

The third step is to train the model on the training data using the fit() method. During training, the model is presented with batches of training data, and the weights are updated to minimize the loss function.

Model Deployment

The final step after Model Training & Deploymnet involves deploying the model so that it can be used in real-world applications.

➢ Create HTML Pages:

- We use HTML to create the front end part of the web page.
- Here, we have created 3 HTML pages- index.html, predict.html, and output.html
- home.html displays the home page.
- index.html displays an introduction about the project
- upload.html gives the emergency alert For more information regarding HTML https://www.w3schools.com/html/
- We also use JavaScript-main.js and CSS-main.css to enhance our functionality and view of HTML pages.

➢ Build python code:

- Collect a dog breed image dataset and organize it into training, validation, and testing folders.
- Install required Python libraries such as TensorFlow, Keras, NumPy, and OpenCV.
- Perform image preprocessing like resizing, normalization, and data augmentation.
- Load a pre-trained convolutional neural network (CNN) model such as MobileNet, VGG16, or ResNet.
- Remove the top layer of the pre-trained model and freeze the base layers.
- Add custom fully connected layers for dog breed classification.
- Compile the model using categorical cross-entropy loss and Adam optimizer.
- Train the model on the dog breed dataset using transfer learning.
- Evaluate the model using accuracy and validation loss.

➢ Importing Libraries:

- The first step is usually importing the libraries that will be needed in the program.
- Importing the flask module in the project is mandatory. An object of the Flask class is our WSGI application. Flask constructor takes the name of the current module (name) as argument Pickle library to load the model file.
- Import **OS** library for handling directory paths and dataset management.
- “Required Python libraries such as TensorFlow, Keras, NumPy, OpenCV, and Matplotlib are imported to support model training, image processing, and performance evaluation.

➢ Creating our flask application and loading our model by using load_model method:

- "A Flask web application is created and the trained transfer learning model is loaded using the load_model() method to perform real-time dog breed identification."
- Import the **Flask framework** to create a web-based application for dog breed identification.
- Initialize the Flask application using the `Flask(__name__)` method.
- Design routes to handle the home page and image upload requests.
- Use the **`load_model()`** method from Keras to load the pre-trained dog breed classification model.
- Load the saved transfer learning model (`.h5` file) only once at application startup to improve performance.
- Preprocess the uploaded dog image (resize, normalize, and convert to array) before prediction.
- Run the Flask application in debug mode during development.

➢ Routing to the html Page:

Here, the declared constructor is used to route to the HTML page created earlier.

In the above example, '/' URL is bound with index.html function. Hence, when the home page of a web server is opened in the browser, the html page will be rendered. Whenever you browse an image from the html page this photo can be accessed through POST or GET Method.

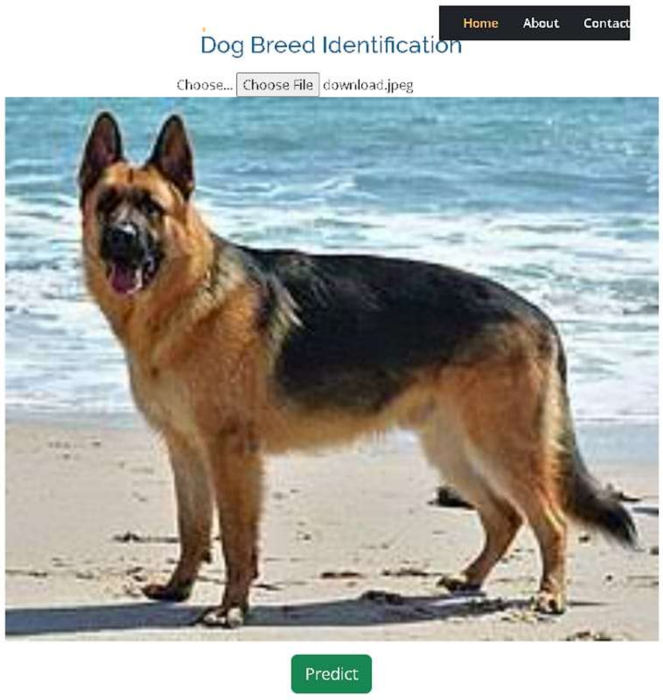
Here we are defining a function which requests the browsed file from the html page using the post method. The requested picture file is then saved to the uploads folder in this same directory using OS library. Using the load image class from Keras library we are retrieving the saved picture from the path declared. We are applying some image processing techniques and then sending that preprocessed image to the model for predicting the class. This returns the numerical value of a class (like 0 to 19.) which lies in the 0th index of the variable preds. This numerical value is passed to the index variable declared. This returns the name of the class. This name is rendered to the prediction variable used in the html page.

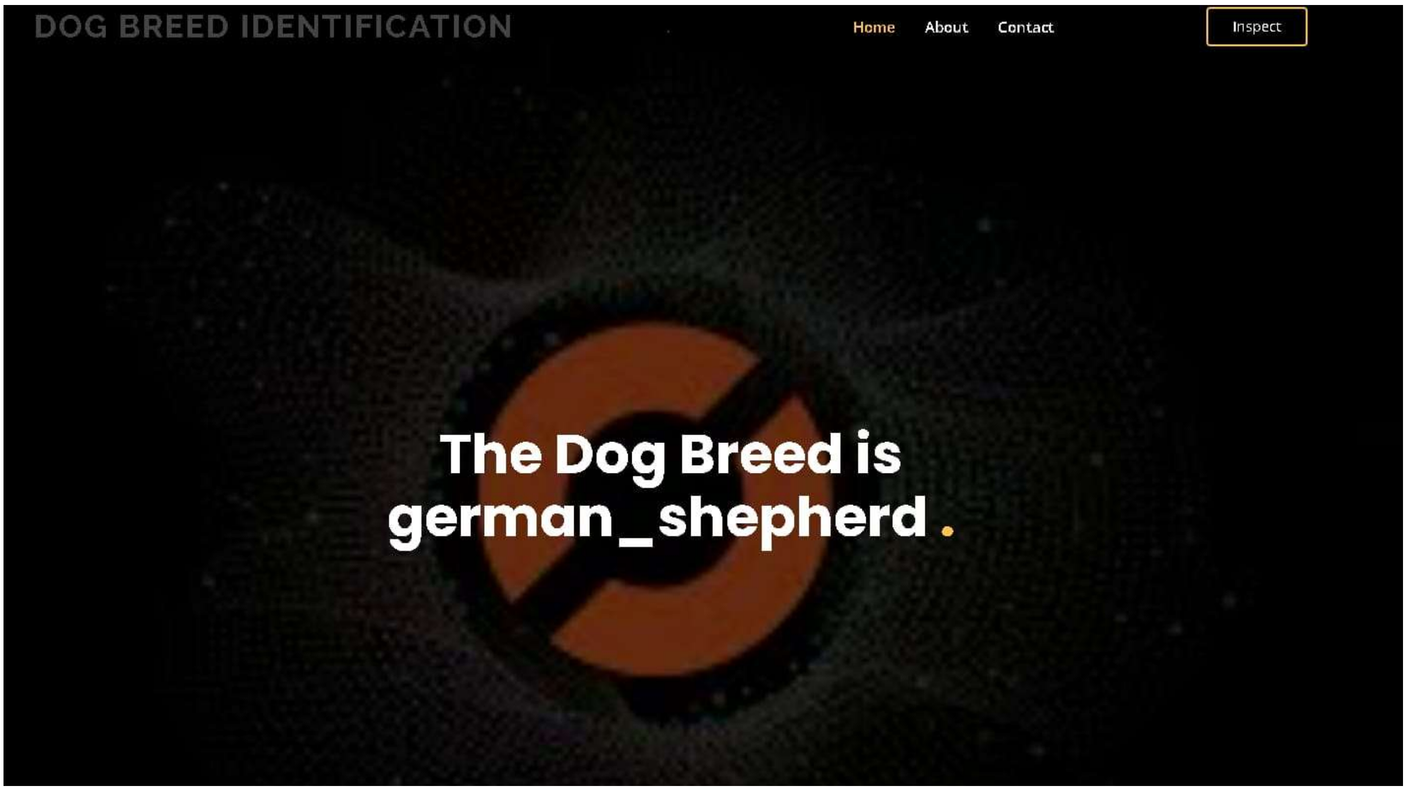
➢ Run the application:

- Open the anaconda prompt from the start menu.
- Navigate to the folder where your app.py resides.
- Now type “python app.py” command.
- It will show the local host where your app is running on http://127.0.0.1.5000/
- Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.
- Enter the values, click on the predict button and see the result/prediction on the web page.
- Ensure all required Python libraries such as TensorFlow, Keras, Flask, and OpenCV are installed.
- Place the trained dog breed identification model file (`dog_breed_model.h5`) in the project directory.
- Open the project folder in the command prompt or terminal.
- Activate the virtual environment (if used).
- Wait until the Flask server starts successfully without errors.
- Upload a dog image through the application interface.
- Click the predict button to start breed identification.
- View the predicted dog breed along with the confidence score on the screen.
- Stop the application by pressing `Ctrl + C` in the terminal.
- “The Flask application is executed locally to identify the dog breed from an uploaded image using a transfer learning model.”

❖ FINAL OUTPUT:

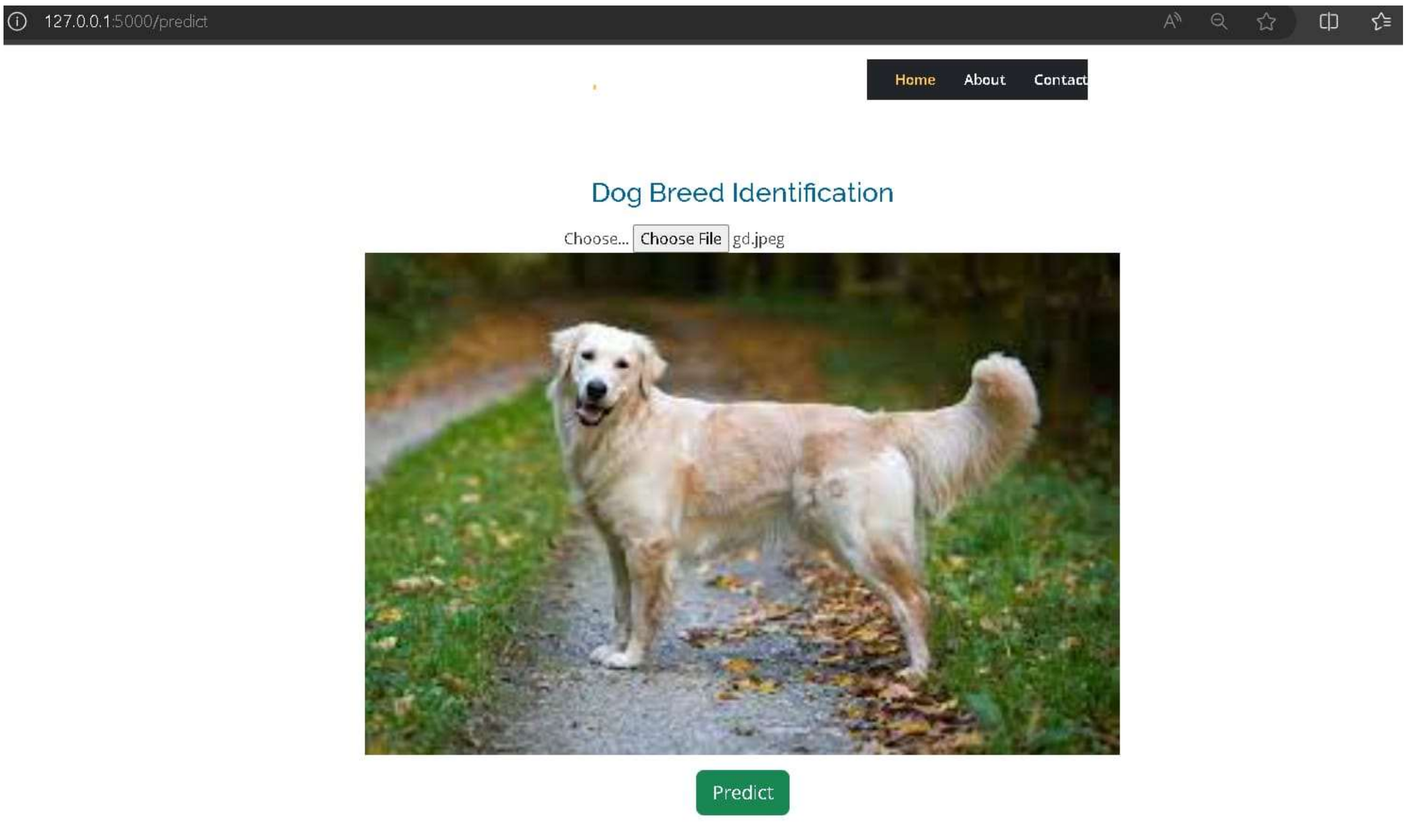
➢ Output 1:





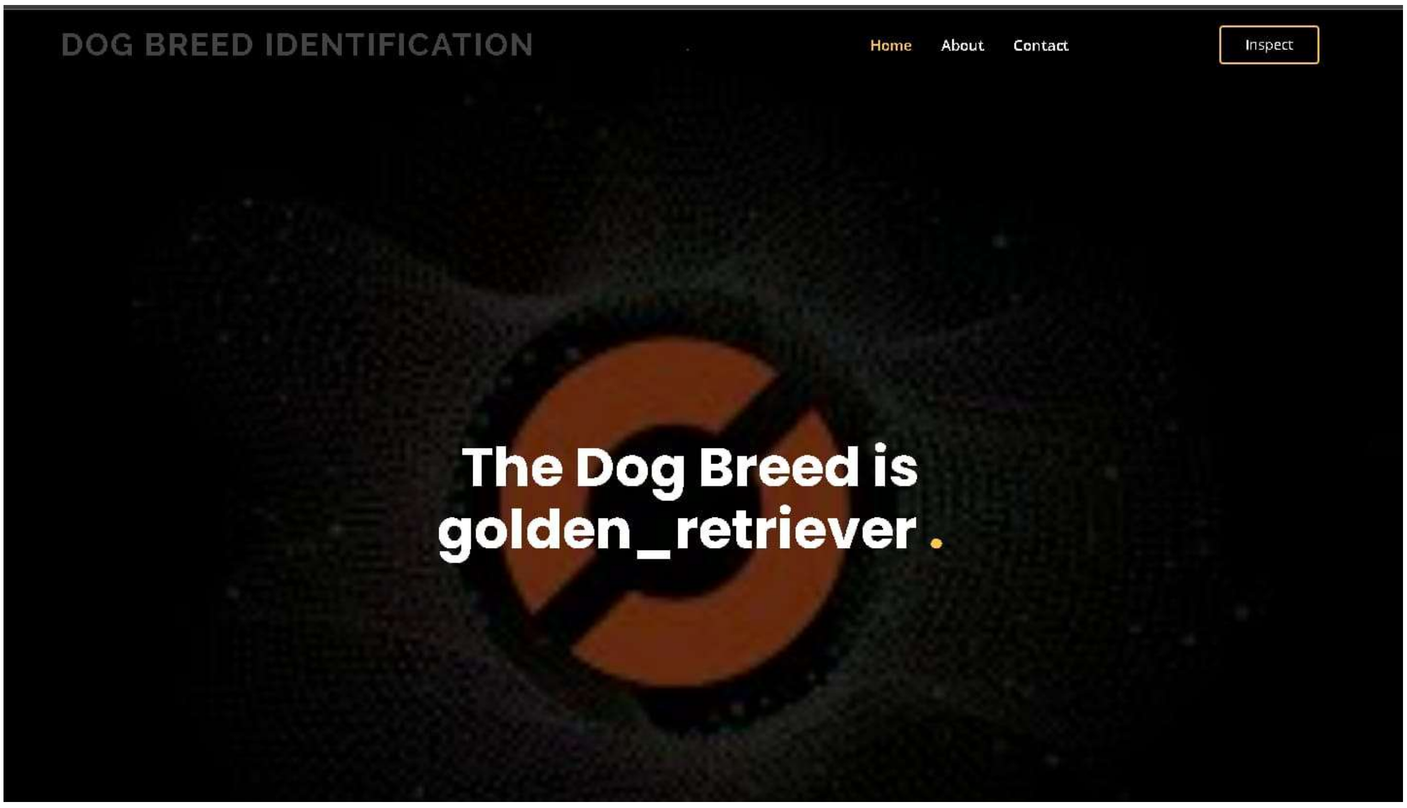
The Dog Breed is german_shepherd .

➢ Output 2:


127.0.0.1:5000/predict

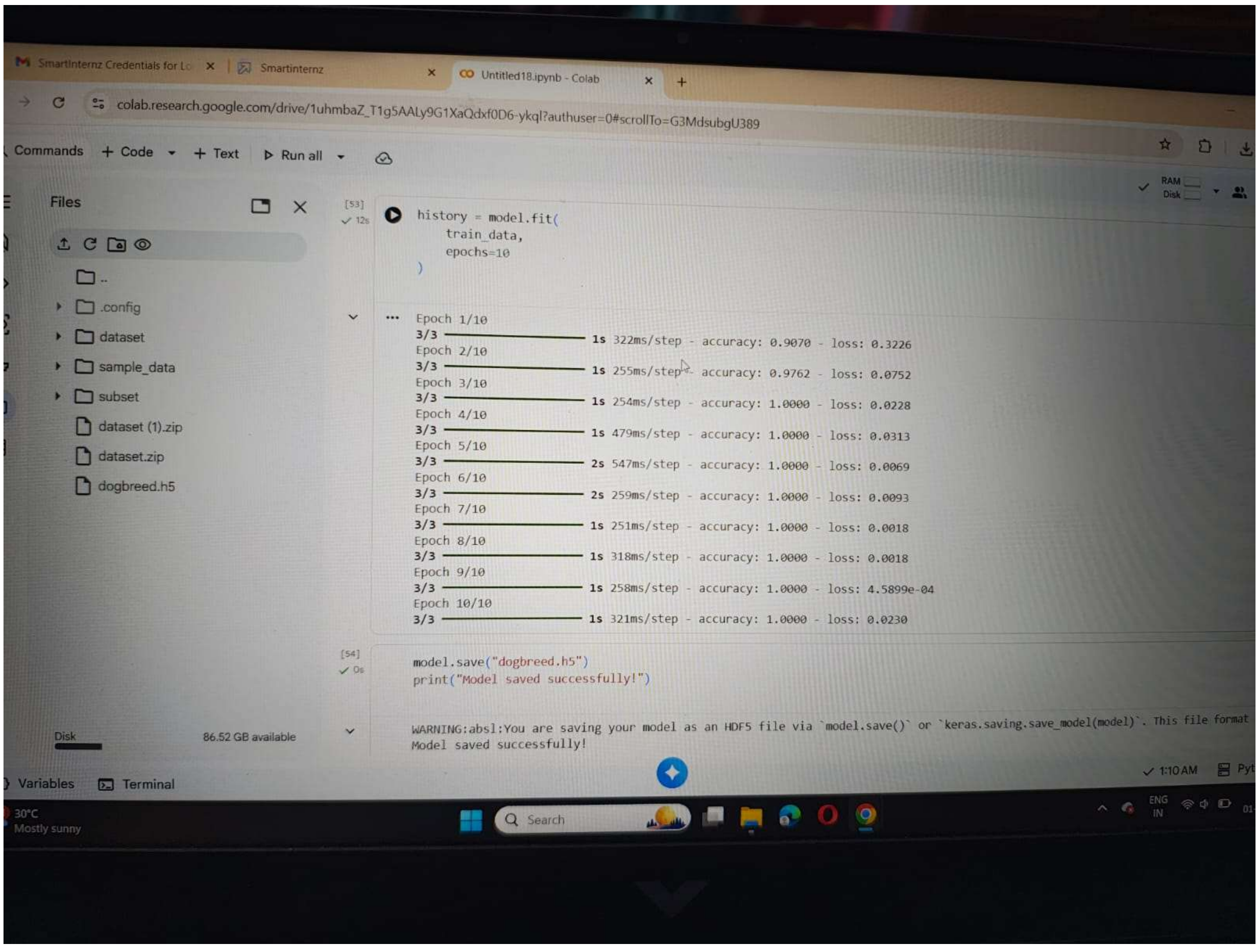
Dog Breed Identification

Choose... Choose File gd.jpeg



The Dog Breed is golden_retriever .

❖ I have done my coding in Google colab :

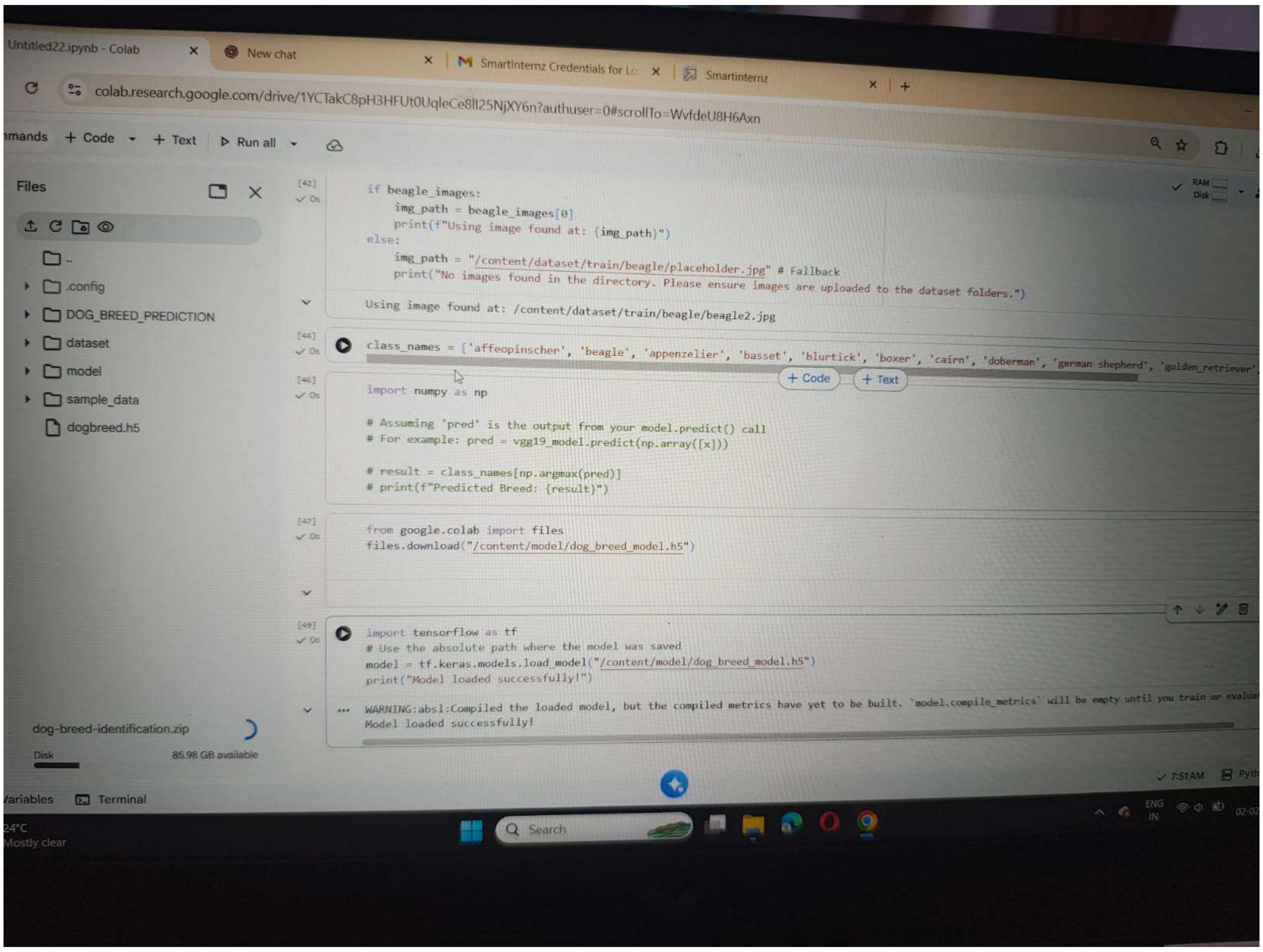


```
history = model.fit(
    train_data,
    epochs=10
)
```

```
Epoch 1/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 322ms/step - accuracy: 0.9070 - loss: 0.3226
Epoch 2/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 255ms/step - accuracy: 0.9762 - loss: 0.0752
Epoch 3/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 254ms/step - accuracy: 1.0000 - loss: 0.0228
Epoch 4/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 479ms/step - accuracy: 1.0000 - loss: 0.0313
Epoch 5/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 547ms/step - accuracy: 1.0000 - loss: 0.0069
Epoch 6/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 2s 259ms/step - accuracy: 1.0000 - loss: 0.0093
Epoch 7/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 251ms/step - accuracy: 1.0000 - loss: 0.0018
Epoch 8/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 318ms/step - accuracy: 1.0000 - loss: 0.0018
Epoch 9/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 258ms/step - accuracy: 1.0000 - loss: 4.5899e-04
Epoch 10/10
3/3 ━━━━━━━━━━━━━━━━━━━━ 1s 321ms/step - accuracy: 1.0000 - loss: 0.0230
```

```
model.save("dogbreed.h5")
print("Model saved successfully!")
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format
Model saved successfully!
```



```
if beagle_images:
    img_path = beagle_images[0]
    print(f"Using image found at: {img_path}")
else:
    img_path = "/content/dataset/train/beagle/placeholder.jpg" # Fallback
    print("No images found in the directory. Please ensure images are uploaded to the dataset folders.")
```

```
Using image found at: /content/dataset/train/beagle/beagle2.jpg
```

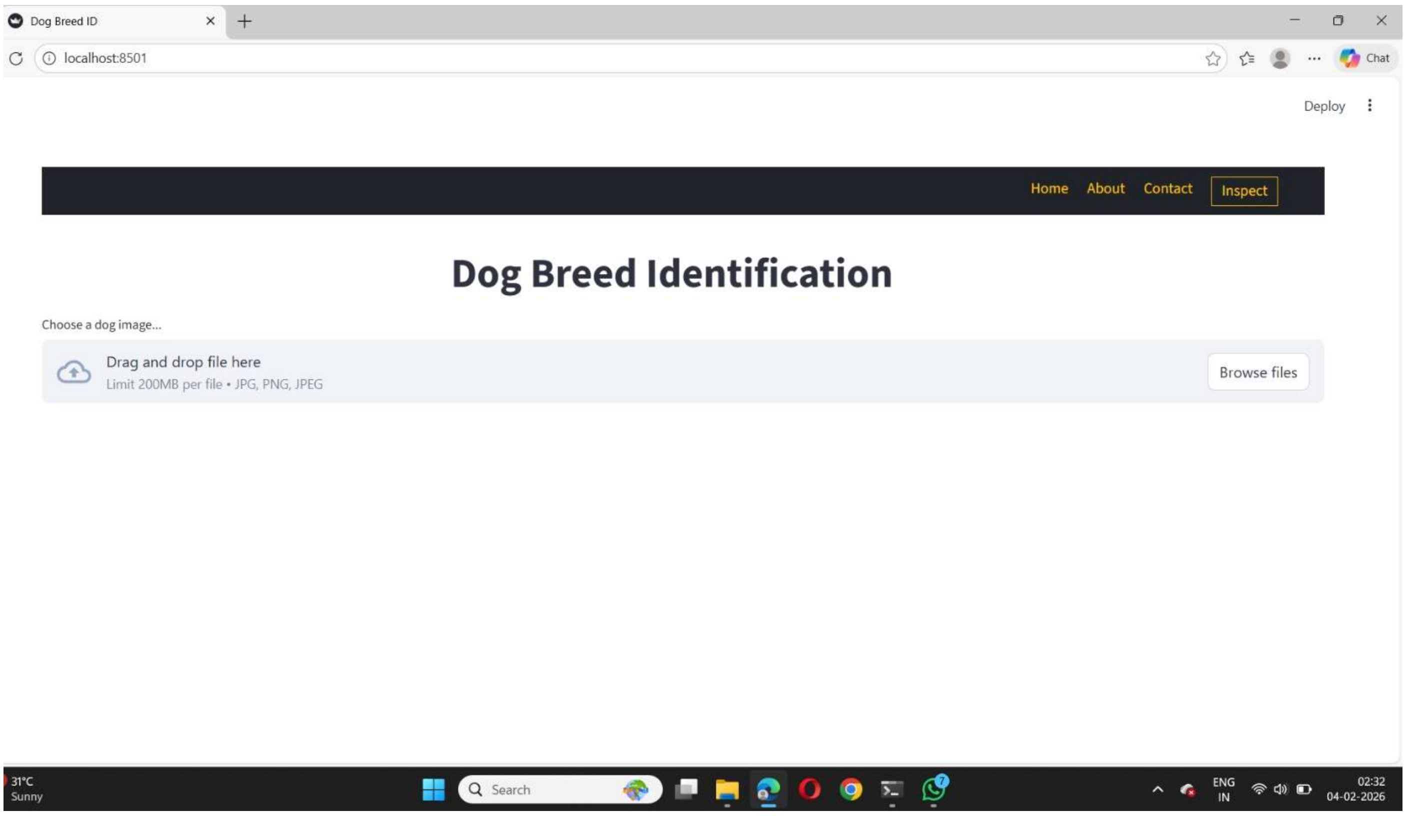
```
class_names = ['affeopinscher', 'beagle', 'appenzelier', 'basset', 'blurtick', 'boxer', 'cairn', 'doberman', 'german shepherd', 'golden_retriever',
```

```
import numpy as np

# Assuming 'pred' is the output from your model.predict() call
# For example: pred = vgg19_model.predict(np.array([x]))

# result = class_names[np.argmax(pred)]
# print(f"Predicted Breed: {result}")
```

```
from google.colab import files
files.download("/content/model/dog_breed_model.h5")
```

```
import tensorflow as tf
# Use the absolute path where the model was saved
model = tf.keras.models.load_model("/content/model/dog_breed_model.h5")
print("Model loaded successfully!")
```

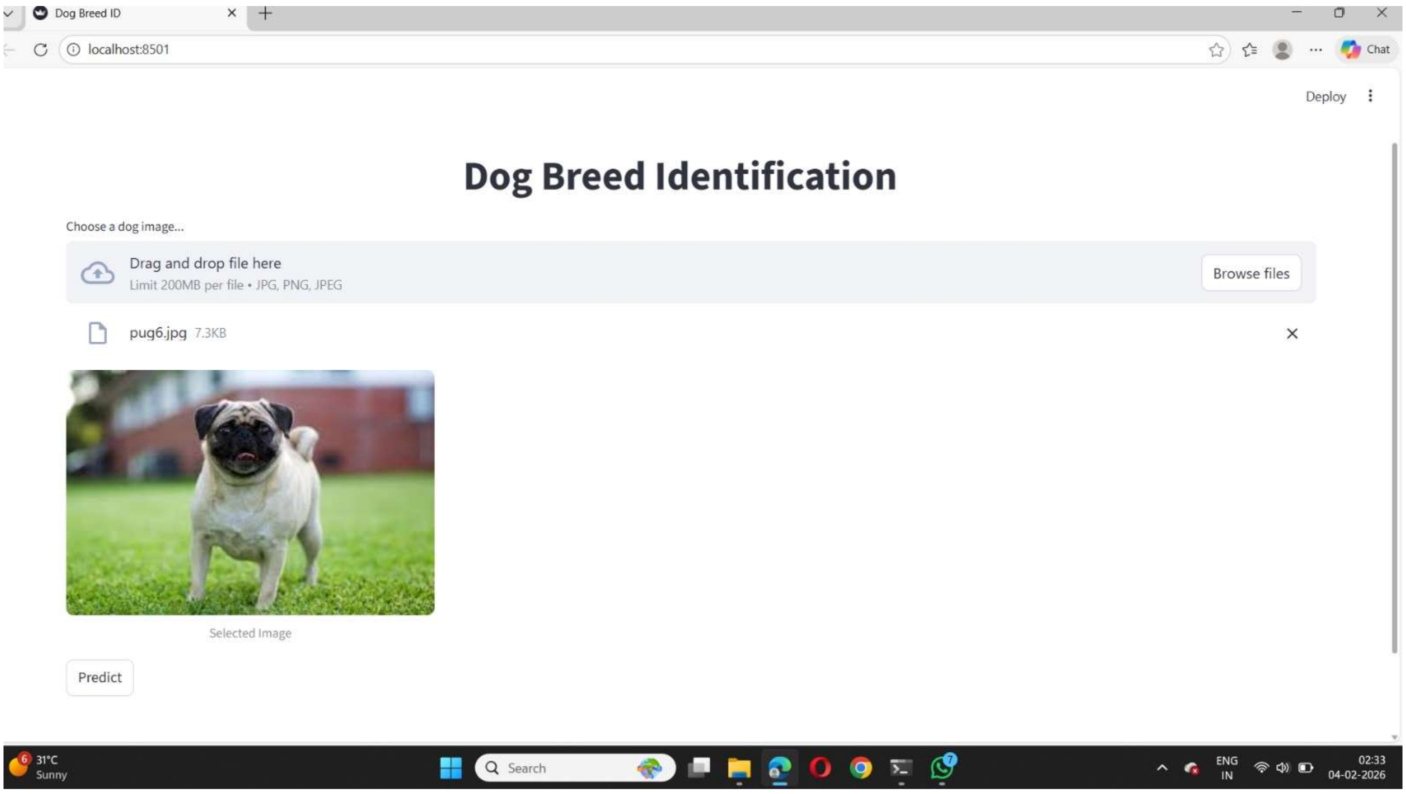
```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evalua
Model loaded successfully!
```



Dog Breed Identification

Choose a dog image...

Drag and drop file here
Limit 200MB per file • JPG, PNG, JPEG



Dog Breed Identification

Choose a dog image...

Drag and drop file here
Limit 200MB per file • JPG, PNG, JPEG

❖ Conclusion:

The **Dog Breed Identification using Transfer Learning** in this project demonstrates the effective use of transfer learning for image classification tasks. By leveraging pre-trained convolutional neural network models, the system is able to recognize and classify dog breeds from images with high accuracy. Transfer learning significantly reduces training time and computational requirements while maintaining strong performance, even with a limited dataset.

Image preprocessing techniques such as resizing, normalization, and data augmentation played a crucial role in improving the robustness of the model. These steps helped the system handle variations in lighting, background, and dog poses. Fine-tuning the final layers of the pre-trained model allowed the network to adapt learned features specifically for dog breed classification.

The trained model was successfully integrated into a Flask-based web application, enabling real-time predictions through a user-friendly interface. Users can upload dog images and receive the predicted breed along with a confidence score, making the system practical and easy to use. This application highlights how deep learning models can be effectively deployed in real-world scenarios.

In conclusion, the project confirms that transfer learning is a powerful approach for building accurate and efficient image recognition systems. The Dog Breed Identification application has potential uses in pet adoption platforms, veterinary services, and animal shelters. Future improvements may include adding more breeds, improving accuracy, and extending the system to mobile and real-time applications.

❖ Acknowledgement:

FIRST AND FOREMOST, I SINCERELY GRATITUDE TO OUR ESTEEMED INSTITUTE SRI VASAVI DEGREE COLLEGE FOR GIVING ME THIS OPPORTUNITY TO FULFILL OUR WARM DREAM TO BECOME A GRADUATE. OUR SINCERE GRADITUDE TO OUR LONG TERM INTERNSHIP GUIDE SRI L LAKSHMI NARAYANA, LECTURER DEPARTMENT OF COMPUTER SCIENCE FOR TIMELY COOPERATION AND VALUABLE SUGGESTIONS WHILE CARRYING OUT THIS INTERNSHIP.

I EXPRESS MY SINCERE THANKS AND HEARTFUL GRATITUDE TO OUR MENTOR SRI **L LAKSHMI NARAYANA**, HOD IN COMPUTER SCIENCE FOR PERMITTING ME TO DO MY PROJECT INTERNSHIP.

I EXPRESS MY SINCERE THANKS AND HEARTFUL GRATITUDE TO SRI **M RAMA KRISHNA**, PRINCIPAL OF SRI VASAVI DEGREE COLLEGE FOR PROVIDING A FAVOURABLE ENVIRONMENT AND SUPPORTING ME DURING THE DEVELOPMENT OF THIS INTERNSHIP.

THANK YOU,
SMART BRIDGE

SHAIK ALEKHYA VIJAYA

TEAM LEADER

SIGNATURE OF THE HOD SIGNATURE OF THE PRINCIPAL