Dog Breed Classification

INTRODUCTION :

1.1 Overview :

The "Dog Breed Classification" project is a website developed using HTML, CSS, and JavaScript. Its primary objective is to identify the breed of a dog based on its photograph. The project utilises machine learning, CNN, and transfer learning techniques to analyse and classify dog images into specific breeds.

1.2 Purpose :

The purpose of this project is to provide a user-friendly platform where users can upload a photo of a dog and obtain accurate information about its breed. By leveraging advanced image classification algorithms, the project aims to automate the process of identifying dog breeds, eliminating the need for manual identification and potential human errors. This project can be beneficial for dog enthusiasts, veterinarians, or anyone interested in learning more about dog breeds.

2 LITERATURE SURVEY :

2.1 Existing problem :

Prior to this project, the identification of dog breeds typically relied on human expertise, which could be time-consuming and subjective. Traditional methods involved manual inspection of physical attributes and comparing them to breed standards. While experienced individuals could make accurate identifications, the process was not efficient and prone to errors, especially for less familiar or mixed breeds.

2.2 Proposed solution :

To address the existing problem, the proposed solution in this project involves leveraging machine learning techniques, particularly image classification algorithms. By training a deep learning model on a large dataset of labeled dog images, the model can learn to recognize patterns and features specific to different dog breeds. Once trained, the model can be deployed on the website, where users can upload a dog photo. The uploaded image is then processed by the model, which predicts the most likely breed based on the learned patterns and classifications. The proposed solution aims to automate and enhance the accuracy of dog breed identification using advanced technology.

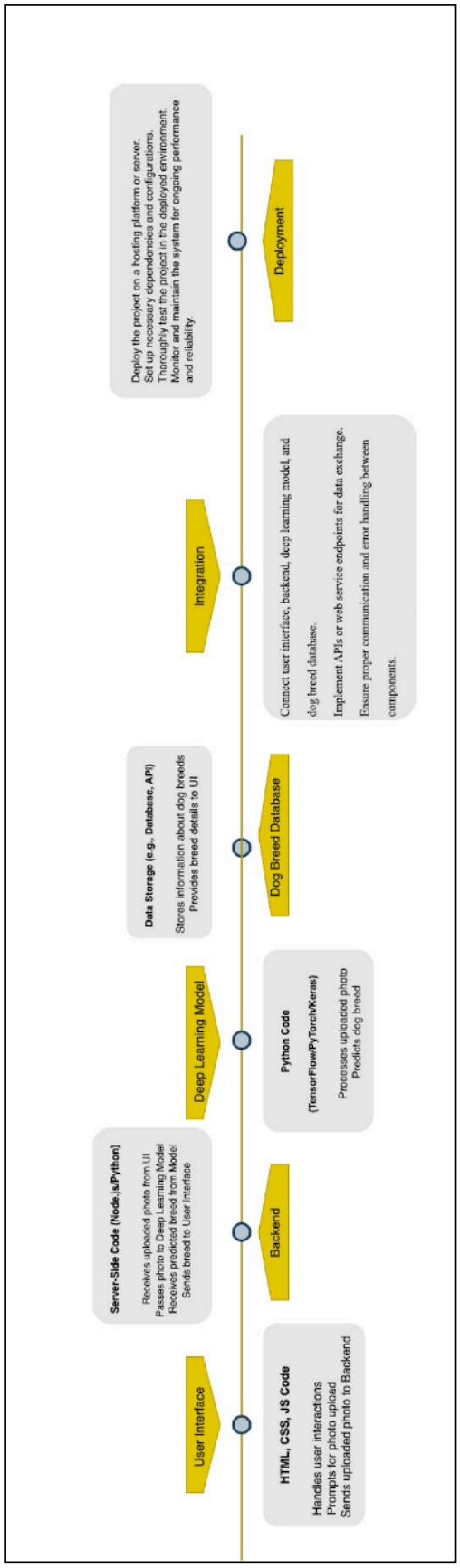
3 THEORETICAL ANALYSIS:

3.1 Block diagram :

The Block diagram provides a diagrammatic overview of the project's components and their interactions.

- User Interface (HTML/CSS): Represents the website's interface where users can upload dog photos.
- JavaScript: Handles the interactions between the user interface and the backend.
- Backend: Receives the uploaded dog photo, processes it using a deep learning model, and returns the predicted breed.
- Deep Learning Model: Trained on a dataset of labeled dog images to classify the breeds.
- Dog Breed Database: Contains information about various dog breeds to provide additional details to the user.
- Integration : Connect user interface, backend, deep learning model, and dog breed database using Flask.
- Deployment : Deploy the project on a hosting platform or server. Set up necessary dependencies and configurations. Thoroughly test the project in the

deployed environment. Monitor and maintain the system for ongoing performance and reliability.



3.2 Hardware/Software Designing :

The hardware and software requirements for this project typically include :

Hardware :

- Computer or server for hosting the website and running the backend.
- Sufficient storage space for storing the deep learning model and the dog breed database.
- Internet connection for hosting the website and allowing users to access it.

Software :

- Web development tools: HTML, CSS, and JavaScript for creating the website's frontend.
- Backend development: A server-side language such as Node.js or Python for handling the image processing and classification.
- Deep learning framework: Libraries like TensorFlow, PyTorch, or Keras, with InceptionV3 as the pre-trained model, for training and deploying the deep learning model.
- InceptionV3 is a convolutional neural network architecture that has been pre-trained on a large dataset, including the ImageNet dataset. It is commonly used for image classification tasks and can be integrated into deep learning frameworks such as TensorFlow, PyTorch, or Keras. By utilizing InceptionV3, you can enhance the accuracy and efficiency of your image processing and classification tasks in the backend development of your web application.
- Database management system: Optional if you plan to store dog breed information in a database.

4 EXPERIMENTAL INVESTIGATIONS :

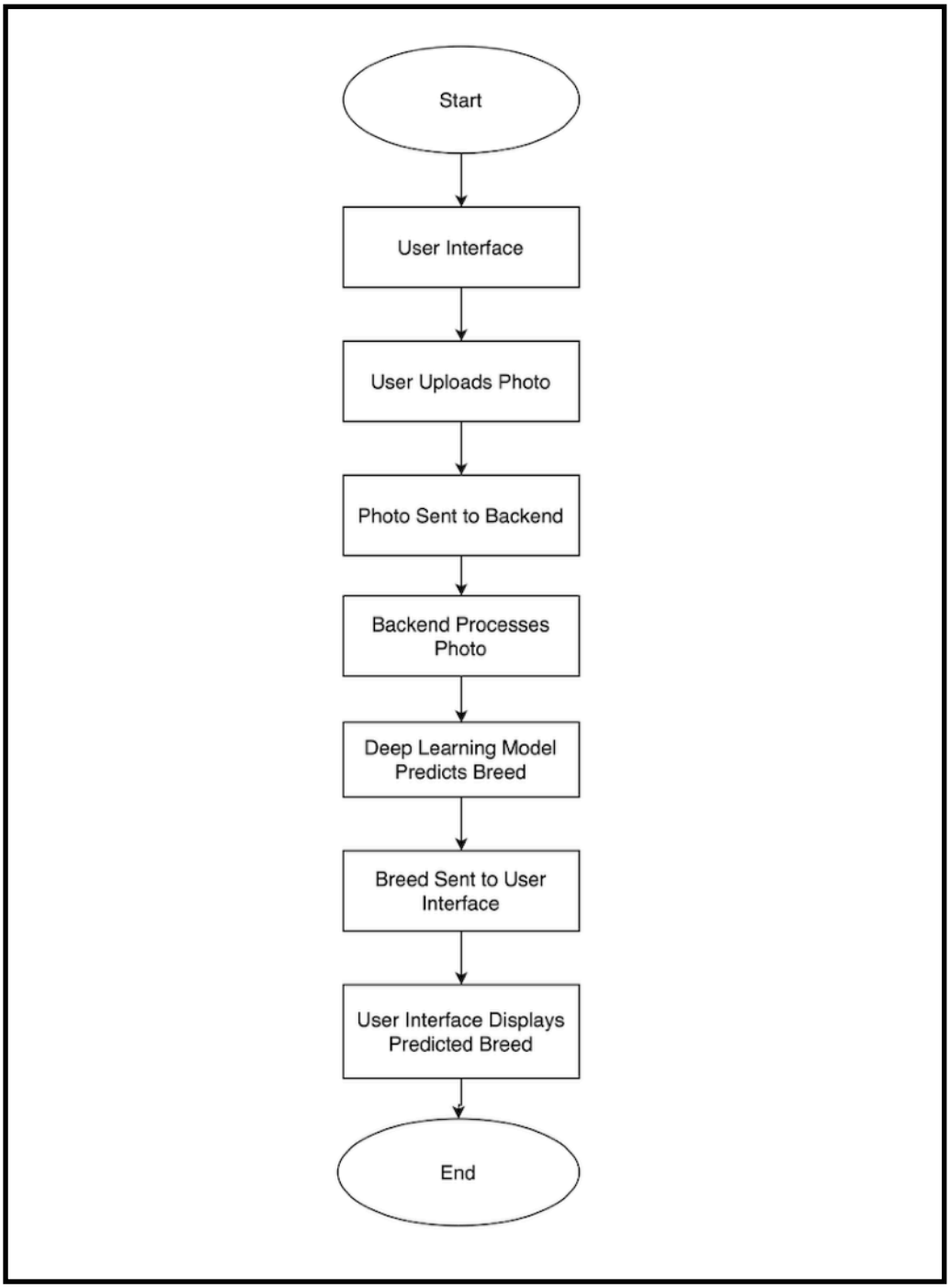
The experimental investigations involve analyzing and fine-tuning the solution during its development. This could include :

- Collecting and preprocessing a large dataset of dog images.
- Training and optimizing the deep learning model with different architectures and hyperparameters.
- Evaluating the model's performance using metrics like accuracy, precision, recall, and F1 score.
- Testing the website's functionality by uploading various dog images and verifying the predicted breed against ground truth information.
- Gathering user feedback and making improvements based on their experience with the website.

5 FLOWCHART :

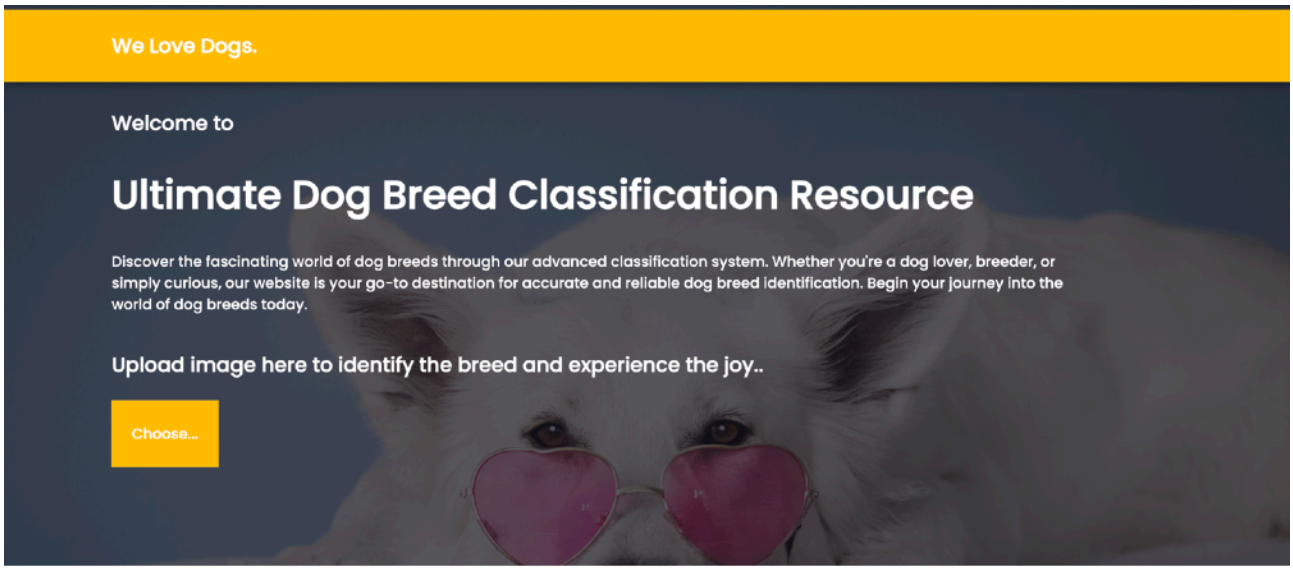
The flowchart illustrates the control flow of the solution.

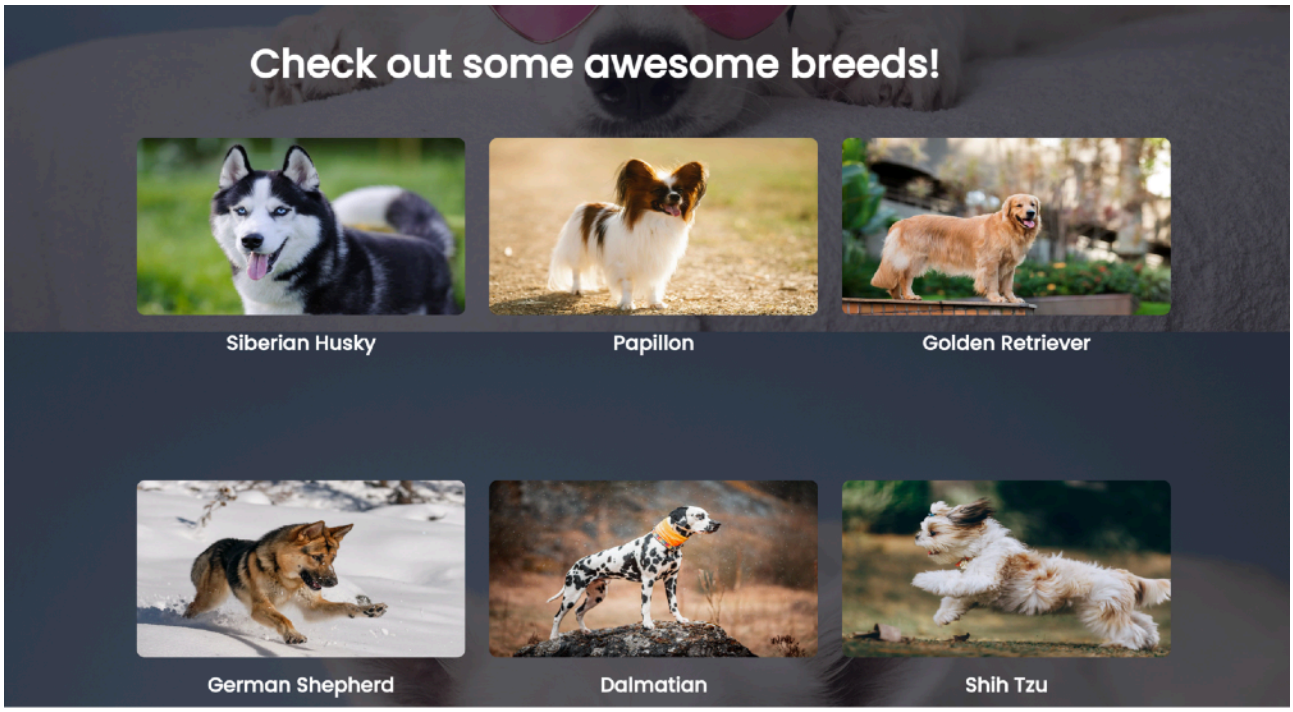
- User accesses the website.
- User interface prompts the user to upload a dog photo.
- User selects and uploads the photo.
- The uploaded photo is sent to the backend.
- Backend receives the photo and passes it to the deep learning model.
- Deep learning model processes the image and predicts the dog breed.
- Backend sends the predicted breed back to the user interface.
- User interface displays the predicted breed to the user.

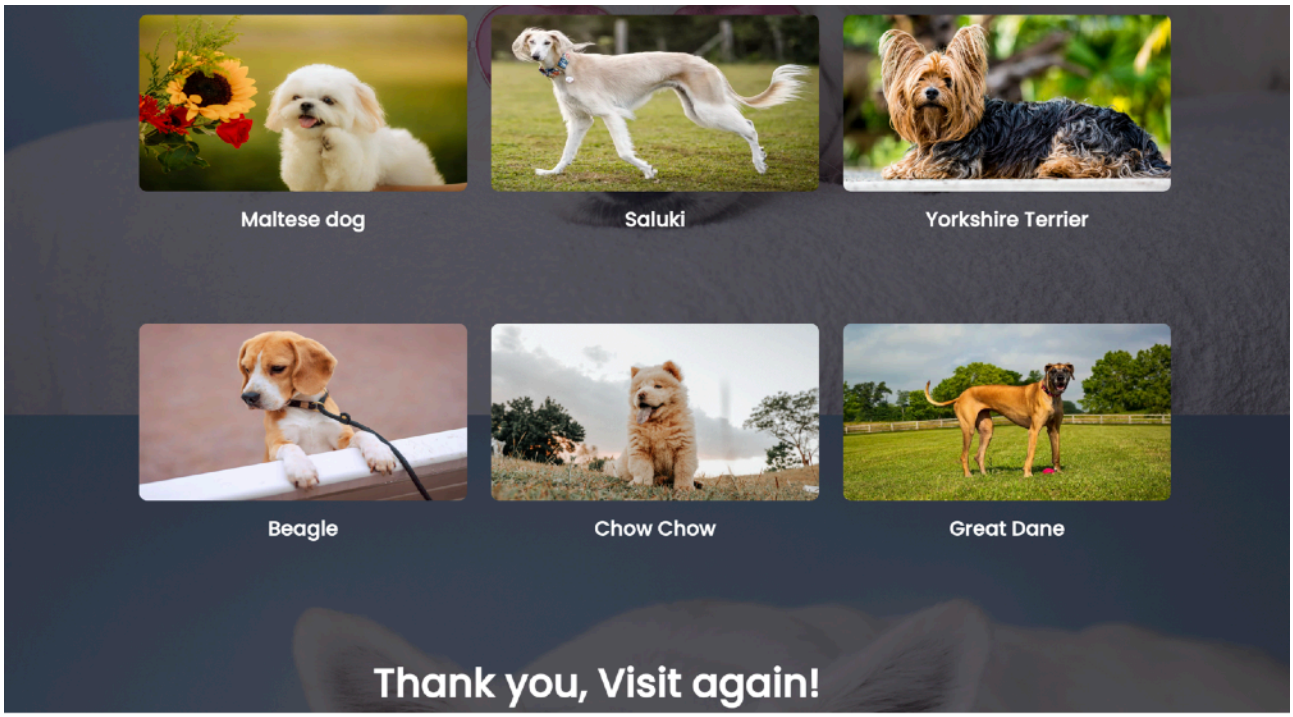


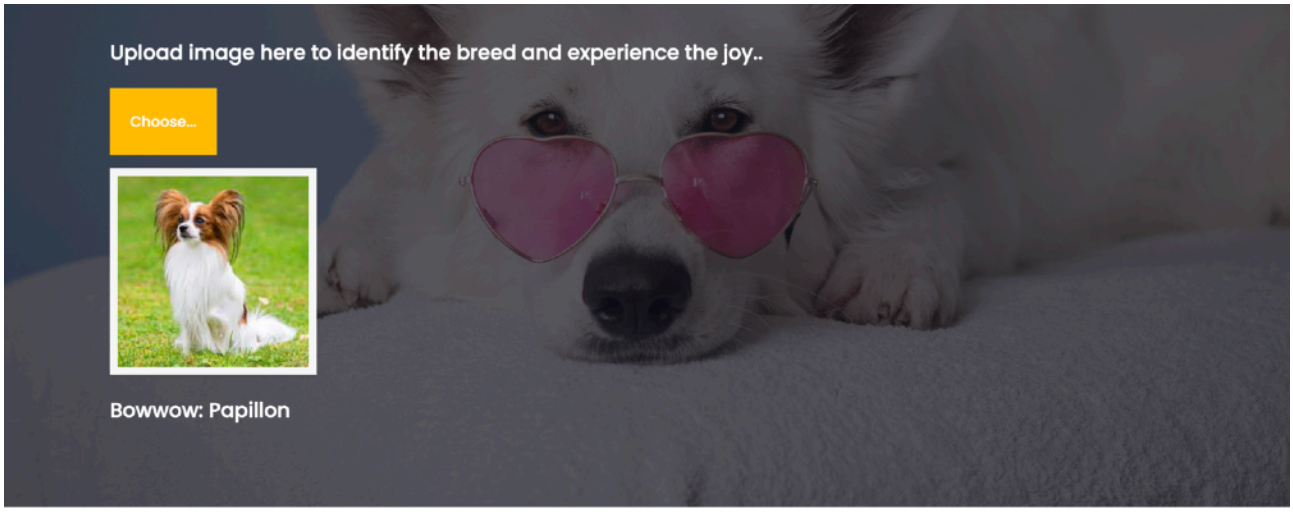
6 RESULT :

The final findings of the project include the output or screenshots of the website displaying the predicted dog breed for uploaded photos.


We Love Dogs.
Welcome to
Ultimate Dog Breed Classification Resource
Discover the fascinating world of dog breeds through our advanced classification system. Whether you're a dog lover, breeder, or simply curious, our website is your go-to destination for accurate and reliable dog breed identification. Begin your journey into the world of dog breeds today.
Upload image here to identify the breed and experience the joy..
Choose...


Check out some awesome breeds!
Siberian Husky
Papillon
Golden Retriever
German Shepherd
Dalmatian
Shih Tzu


Maltese dog
Saluki
Yorkshire Terrier
Beagle
Chow Chow
Great Dane
Thank you, Visit again!


Upload image here to identify the breed and experience the joy..
Choose...
Bowwow: Papillon

7 ADVANTAGES & DISADVANTAGES :

Advantages :

- Automated and accurate dog breed identification.
- Eliminates the need for manual inspection and potential human errors.
- Provides a user-friendly platform for dog breed identification.
- Can handle a large number of dog photos and provide quick results.
- Can be accessible to anyone with an internet connection.

Disadvantages :

- Reliance on the quality and diversity of the training dataset for accurate predictions.
- May struggle with identifying mixed breeds or dogs with uncommon or rare characteristics.
- Potential biases in the training data could result in biased predictions.
- Requires continuous updates and maintenance to keep the deep learning model up to date.
- The accuracy of the predictions may not be 100% and may vary depending on the complexity of the image.
- The model, trained only with 99 breeds, can't predict other than these trained breeds.

8 APPLICATIONS :

The "Dog Breed Classification" project has various applications in different domains. Here are some areas where this solution can be applied:

- Pet Adoption Platforms: The solution can be integrated into pet adoption websites or platforms. It can help potential adopters identify the breed of a dog based on its photo, making the adoption process more informative and efficient.

- Veterinary Clinics: Veterinary clinics can use this solution to quickly identify the breed of a dog based on a photo. It can aid in providing better healthcare and tailored treatments based on specific breed characteristics.
- Dog Shows and Competitions: In dog shows and competitions, the solution can be used to validate and verify the breed of participating dogs. It can assist judges and organizers in accurately categorizing dogs into their respective breed classes.
- Animal Shelters and Rescue Centers: Animal shelters and rescue centers can benefit from this solution by quickly identifying the breed of a stray or abandoned dog. This information can be helpful for proper care, feeding, and potential adoption of the dog.
- Dog Breeders and Kennels: Breeders and kennels can utilize this solution to identify and verify the breed of their dogs. It can assist them in maintaining accurate breed records and pedigrees.
- Educational and Research Purposes: The project can be used as an educational tool to help students learn about different dog breeds. It can also be utilized in research studies focused on dog genetics, behavior, and breed characteristics.

Overall, the "Dog Breed Classification" solution can be applied in various areas related to dog identification, adoption, healthcare, research, and education. Its ability to accurately classify dog breeds based on photos can provide valuable information and streamline processes in these domains.

9 CONCLUSION :

In conclusion, the "Dog Breed Classification" project successfully developed a website that utilizes machine learning techniques to accurately identify dog breeds based on uploaded photos. By training a deep learning model on a large dataset of labeled dog images, the project achieved automated and reliable breed classification. The website provides a user-friendly interface, allowing users to easily upload photos and obtain the predicted breed. The project addresses the limitations of manual identification methods and improves the efficiency and accuracy of dog breed classification.

10 FUTURE SCOPE :

There are several potential enhancements that can be made to the "Dog Breed Classification" project in the future:

- Improved Accuracy: Continuously updating and expanding the training dataset can enhance the model's accuracy, especially for identifying mixed breeds or rare characteristics.
- User Feedback Integration: Implementing a feedback system where users can provide feedback on the predicted breeds can help improve the model and make it more robust over time.
- Mobile Application: Developing a mobile application version of the project can provide greater accessibility and convenience for users who prefer to use their smartphones or tablets.
- Real-Time Classification: Integrating real-time image classification capabilities can enable users to capture photos using their device's camera and instantly receive the predicted breed.
- Additional Information: Including more comprehensive information about each dog breed, such as temperament, size, lifespan, and care requirements, can enhance the user experience and provide valuable insights.
- Breed Recognition from Videos: Expanding the project to include video classification, allowing users to upload videos and receive breed predictions from the video frames.
- Multi-Class Classification: Extending the project to handle multiple dogs in a single photo and accurately identifying the breeds of each dog present.
- Social Features: Incorporating social features such as sharing the predicted breed on social media platforms or creating a community where users can discuss and share their love for different dog breeds.

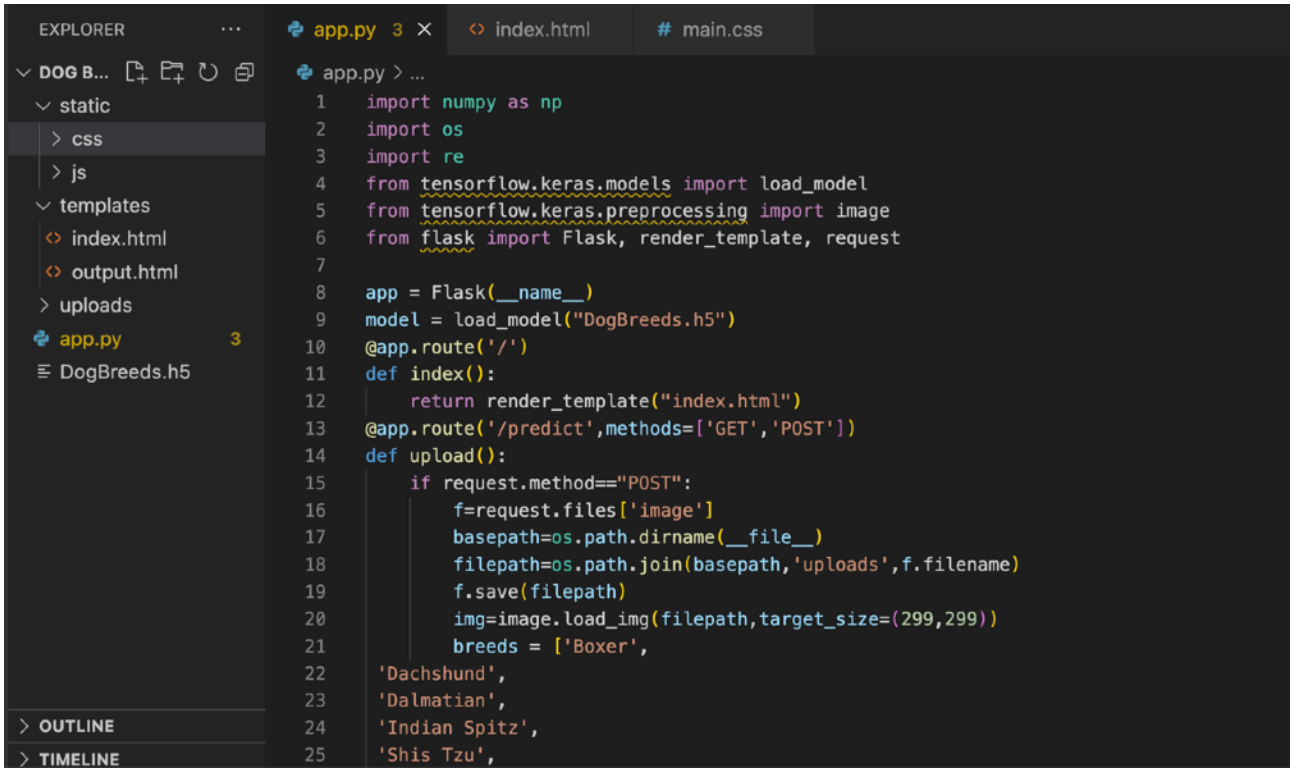
By implementing these enhancements, the "Dog Breed Classification" project can provide an even more comprehensive and user-friendly experience for dog enthusiasts, veterinarians, and anyone interested in exploring and learning about various dog breeds.

11 BIBILOGRAPHY :

https://www.kaggle.com/
https://www.dogspot.in/dog-breeds/

APPENDIX:

App.py:



Ipynb file:

jupyter Dog_Breed_Classification_Inception Last Checkpoint: Last Friday at 15:15 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
# Importing the required libraries
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Dense,GlobalAveragePooling2D, Dropout
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.models import Model, Sequential
import numpy as np
from tensorflow.keras.preprocessing import image
```

```
/opt/anaconda3/lib/python3.9/site-packages/scipy/__init__.py:155: UserWarning: A NumPy version >=1.18.5 and <1.25.0 i
s required for this version of SciPy (detected version 1.25.0
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [2]:
```
train_path = "Dog Breed Classification/train"
test_path = "Dog Breed Classification/test"
```

In [ ]:
```
# Create data generators for training and testing data
train_gen = ImageDataGenerator(rescale=1./255,
                               shear_range=0.2,
                               zoom_range=0.2,
                               horizontal_flip=True)
test_gen = ImageDataGenerator(rescale=1./255)
```

In [4]:
```
# Load training and testing data using the data generator
train = train_gen.flow_from_directory(train_path,
                                      target_size=(299,299),
                                      batch_size=32,
                                      class_mode='categorical')

test = test_gen.flow_from_directory(test_path,
                                    target_size=(299,299),
                                    batch_size=32,
                                    class_mode='categorical')
```

```
Found 6726 images belonging to 99 classes.
Found 928 images belonging to 99 classes.
```

In [5]:
```
# Visualizing sample images from the dataset

# Retrieve a batch of images and their labels
sample_images, labels = next(train)

# Get the mapping of class indices to class names
class_names = train.class_indices

plt.figure(figsize=(12, 6))

# Display the first 12 images
for i in range(12):
    plt.subplot(3, 4, i+1)
    plt.imshow(sample_images[i])
    plt.title(list(class_names.keys())[list(class_names.values()).index(labels[i].argmax())])
    plt.axis('off')

plt.tight_layout()
plt.show()
```

In [6]:
```
# Load InceptionV3 pre trained model
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(299,299,3))
```

In [7]:
```
# Build Model using pre trained model
model = Sequential()
model.add(base_model)
model.add(GlobalAveragePooling2D())
model.add(Dropout(0.3))
model.add(Dense(512, activation = 'relu'))
model.add(Dense(512, activation = 'relu'))
model.add(Dense(99, activation='softmax'))
```

In [8]:
```
# Train model with existing weights
for layer in base_model.layers:
  print(layer)
```

```
<keras.src.engine.input_layer.InputLayer object at 0x16a3bd0d0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x16a7d3520>
<keras.src.layers.normalization.batch_normalization.BatchNormalization object at 0x16a7d3df0>
<keras.src.layers.core.activation.Activation object at 0x16b046190>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x16b0460d0>
<keras.src.layers.normalization.batch_normalization.BatchNormalization object at 0x16b0b5130>
<keras.src.layers.core.activation.Activation object at 0x16b0c34f0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x16b0c3a90>
<keras.src.layers.normalization.batch_normalization.BatchNormalization object at 0x16b0cb4c0>
<keras.src.layers.core.activation.Activation object at 0x16b0d2190>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x16b0d2a30>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x16b0d2f10>
<keras.src.layers.normalization.batch_normalization.BatchNormalization object at 0x16b0dbc40>
<keras.src.layers.core.activation.Activation object at 0x16b0dbeb0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x16b0e48b0>
<keras.src.layers.normalization.batch_normalization.BatchNormalization object at 0x16b0e8790>
<keras.src.layers.core.activation.Activation object at 0x16b0f0550>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x16b0f0ca0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x16b21bc70>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x16b22b910>
<keras.src.layers.normalization.batch_normalization.BatchNormalization object at 0x16b206a30>
```

In [9]:
```
# Freeze the base model layers
print("Number of trainable weights before freezing the base layer: ", len(model.trainable_weights))
model.layers[0].trainable = False
print("Number of trainable weights after freezing the base layer: ", len(model.trainable_weights))
```

```
Number of trainable weights before freezing the base layer:  194
Number of trainable weights after freezing the base layer:  6
```

In [10]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 8, 8, 2048)        21802784

 global_average_pooling2d (  (None, 2048)              0
 GlobalAveragePooling2D)

 dropout (Dropout)           (None, 2048)              0

 dense (Dense)               (None, 512)               1049088

 dense_1 (Dense)             (None, 512)               262656

 dense_2 (Dense)             (None, 99)                50787

=================================================================
Total params: 23165315 (88.37 MB)
Trainable params: 1362531 (5.20 MB)
Non-trainable params: 21802784 (83.17 MB)
_________________________________________________________________
```

In [11]:
```
# Compile the model
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

In [12]:
```
# Train the model
model.fit(train,epochs=10,validation_data=test,steps_per_epoch=len(train),validation_steps=len(test))
```

```
Epoch 1/10
211/211 [==============================] - 621s 3s/step - loss: 1.1741 - accuracy: 0.7202 - val_loss: 0.3584 - val_ac
curacy: 0.8976
Epoch 2/10
211/211 [==============================] - 616s 3s/step - loss: 0.4300 - accuracy: 0.8712 - val_loss: 0.3465 - val_ac
curacy: 0.9030
Epoch 3/10
211/211 [==============================] - 611s 3s/step - loss: 0.3812 - accuracy: 0.8864 - val_loss: 0.3240 - val_ac
curacy: 0.9084
Epoch 4/10
211/211 [==============================] - 611s 3s/step - loss: 0.3131 - accuracy: 0.9041 - val_loss: 0.3423 - val_ac
curacy: 0.9170
Epoch 5/10
211/211 [==============================] - 608s 3s/step - loss: 0.3016 - accuracy: 0.9093 - val_loss: 0.3432 - val_ac
curacy: 0.9009
Epoch 6/10
211/211 [==============================] - 576s 3s/step - loss: 0.2747 - accuracy: 0.9130 - val_loss: 0.3115 - val_ac
curacy: 0.9149
Epoch 7/10
211/211 [==============================] - 577s 3s/step - loss: 0.2457 - accuracy: 0.9214 - val_loss: 0.3264 - val_ac
curacy: 0.9030
Epoch 8/10
211/211 [==============================] - 576s 3s/step - loss: 0.2132 - accuracy: 0.9300 - val_loss: 0.3559 - val_ac
curacy: 0.9062
Epoch 9/10
211/211 [==============================] - 418s 2s/step - loss: 0.2109 - accuracy: 0.9321 - val_loss: 0.3078 - val_ac
curacy: 0.9062
Epoch 10/10
211/211 [==============================] - 531s 3s/step - loss: 0.1962 - accuracy: 0.9367 - val_loss: 0.3221 - val_ac
curacy: 0.9019
```

Out[12]:
```
<keras.src.callbacks.History at 0x16c0d8f70>
```

In [13]:
```
model.save('DogBreeds.h5')
```

```
/opt/anaconda3/lib/python3.9/site-packages/keras/src/engine/training.py:3000: UserWarning: You are saving your model
```

In [14]:
```
# Evaluating the model
loss, accuracy = model.evaluate(test)
print("Test Loss:", loss)
print("Test Accuracy:", accuracy)
```

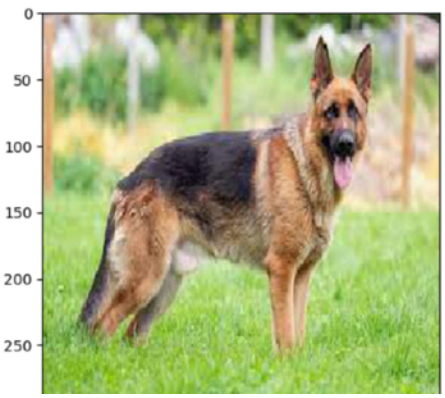
```
29/29 [==============================] - 68s 2s/step - loss: 0.3221 - accuracy: 0.9019
Test Loss: 0.3220541477203369
Test Accuracy: 0.9019396305084229
```

In [15]:
```
# List of trained breeds
breeds = list(train.class_indices.keys())
```

In [16]:
```
# Testing 1
img = image.load_img('German Shepard.jpeg',target_size = (299,299))
# Display image
plt.imshow(img)
img = image.img_to_array(np.squeeze(img))
# Add an extra dimension to match the model's input shape
img = np.expand_dims(img, axis=0)
#Normalize the image
img /= 255.0
pred = np.argmax(model.predict(img))
print(pred)
breeds[pred]
```

```
1/1 [==============================] - 1s 752ms/step
40
```

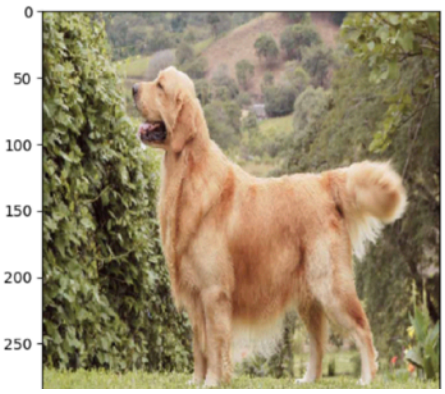
Out[16]:
```
'german_shepherd'
```



In [17]:
```
# Testing 2
img = image.load_img('golden-retriever 4.jpeg',target_size = (299,299))
# Display image
plt.imshow(img)
img = image.img_to_array(np.squeeze(img))
# Add an extra dimension to match the model's input shape
img = np.expand_dims(img, axis=0)
#Normalize the image
img /= 255.0
pred = np.argmax(model.predict(img))
print(pred)
breeds[pred]
```

```
1/1 [==============================] - 0s 103ms/step
42
```

Out[17]: 'golden_retriever'



In [27]:
```
# Testing 12
img = image.load_img('Pomeranian.jpeg',target_size = (299,299))
# Display image
plt.imshow(img)
img = image.img_to_array(np.squeeze(img))
# Add an extra dimension to match the model's input shape
img = np.expand_dims(img, axis=0)
#Normalize the image
img /= 255.0
pred = np.argmax(model.predict(img))
print(pred)
breeds[pred]
```

```
1/1 [==============================] - 0s 107ms/step
75
```

Out[27]: 'pomeranian'