Starting date	30/01/2026
Team id	LTVIP2026TMIDS87688
Project Name	Dog Breed Identification Using Transfer Learning

Dog Breed Identification using Transfer Learning

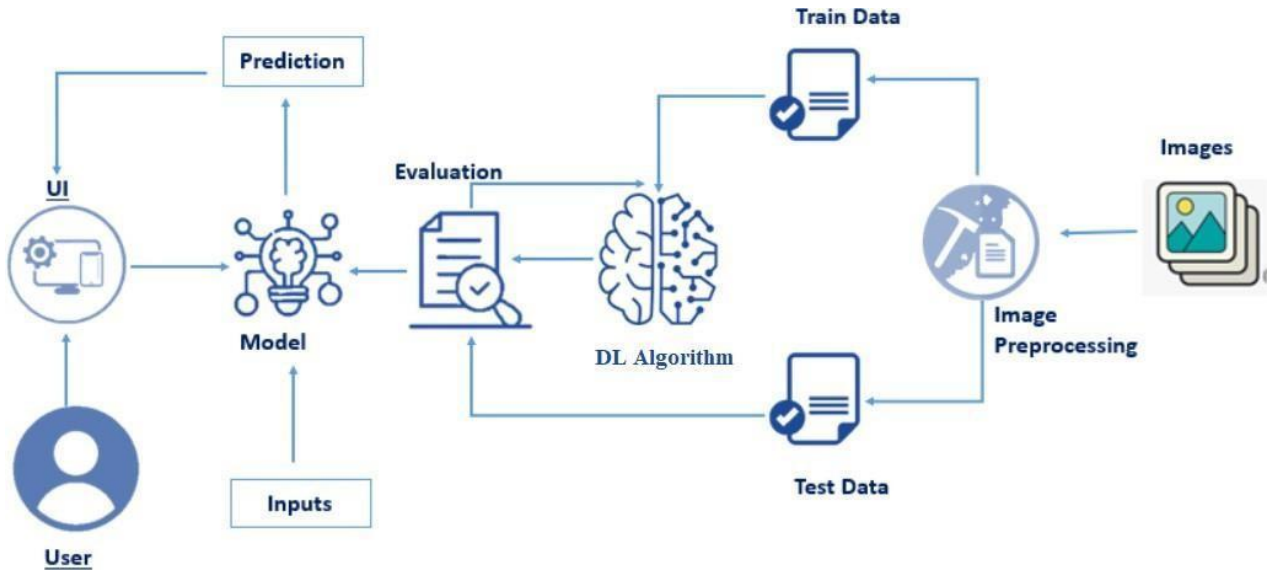
Project Description:

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.

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.

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.

Technical Architecture:



Pre requisites:

To develop and deploy the Dog Breed Classification system, the following prerequisites were required:

1. Technical Knowledge :

   Basic understanding of **Python programming**

   Knowledge of **Machine Learning fundamentals**

   Understanding of:

- Convolutional Neural Networks (CNNs)
- Transfer Learning
- Image preprocessing techniques

   Basic familiarity with **model training and evaluation**

2. Software Requirements :

- Python
- NumPy
- Matplotlib
- TensorFlow & Keras
- Streamlit
- Google Colab

3. Python Libraries :

```
import streamlit as st
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.applications.vgg19 import preprocess_input
from PIL import Image
```

```
import numpy as np
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg19 import preprocess_input
from PIL import Image
import matplotlib.pyplot as plt
```

Project objective :

The primary objective of this project is to develop a deep learning-based image classification system capable of accurately identifying dog breeds from input images and deploy it as an interactive web application.

Develop a Convolutional Neural Network (CNN) using transfer learning with the pre-trained VGG19 architecture to classify images into 120 different dog breeds.

Implement image resizing, normalization, and data augmentation to enhance model performance and generalization.

Measure training and validation accuracy using appropriate metrics to assess the model's effectiveness.

Integrate the trained model into a Streamlit-based interface that allows users to upload an image and receive real-time dog breed predictions.

Ensure that the application is simple, interactive, and usable by non-technical users.

Project Flow

Data Collection

↓

Data Preprocessing

↓

Transfer Learning (VGG19 Model)

↓

Model Training

↓

Model Evaluation

↓

Model Deployment (Streamlit)

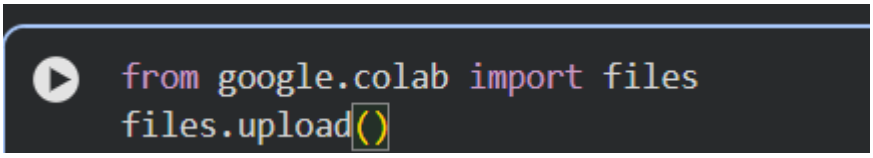
↓

User Upload → Prediction Output

Project Structure :

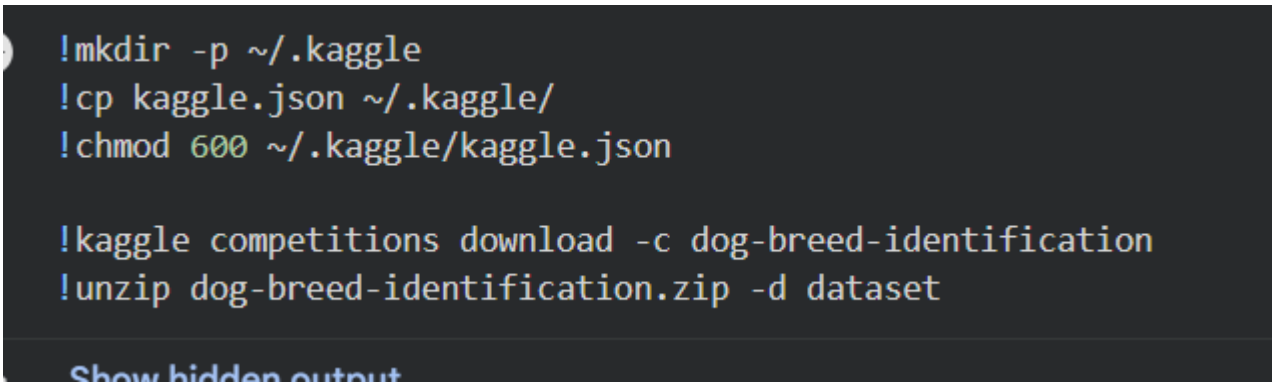
Milestone 1 — Setup and Dataset Preparation

Before training a machine learning model, you must configure the runtime environment (Google Colab), upload credentials, authenticate Kaggle API, and fetch the dataset.



```
from google.colab import files
files.upload()
```

Uploads kaggle.json containing Kaggle API credentials.



```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json

!kaggle competitions download -c dog-breed-identification
!unzip dog-breed-identification.zip -d dataset
```

Show hidden output

Configures Kaggle API folder and permissions. Downloads the Stanford Dogs dataset. Extracts the dataset so TensorFlow can read it.

Milestone 2 — Data Preprocessing

Convert raw images into a format suitable for deep learning training using TensorFlow's ImageDataGenerator.

The model takes 224×224 normalized images. Data is split into training and validation sets, with optional data augmentation to improve generalization.

Setup Data Generators

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.vgg19 import preprocess_input

train_gen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

test_gen = ImageDataGenerator(preprocessing_function=preprocess_input)

train_data = train_gen.flow_from_directory(
    "/content/train_data",
    target_size=(128,128),
    batch_size=32,
    class_mode='categorical'
)

test_data = test_gen.flow_from_directory(
    "/content/test_data",
    target_size=(128,128),
    batch_size=32,
    class_mode='categorical'
)
```

rescale=1./255: Normalizes pixel values

validation_split=0.2: 80/20 split

flow_from_directory: Reads folder structure and maps classes

Milestone 3 — Model Building with Transfer Learning

Create a deep learning model using pretrained VGG19 architecture and custom classification layers.

VGG19 is used as a feature extractor. Only the added classifier layers are trained (initially).

Import Model Libraries

```python
import tensorflow as tf
from tensorflow.keras.applications import VGG19
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.optimizers import Adam
```

Load Pretrained VGG19

```
vgg = VGG19(
    weights='imagenet',
    include_top=False,
    input_shape=(224, 224, 3)
)
```

Freeze Base Layers

```
from tensorflow.keras.models import Model

base_model = VGG19(weights='imagenet', includ

for layer in base_model.layers:
    layer.trainable = False

x = Flatten()(base_model.output)
```

Purpose: Speed training and prevent overfitting on small dataset.

Milestone 4 — Training and Evaluation

After constructing the model, train the final classification layers, then save the model and check validation accuracy.

Add Custom Classifier + Compile

```
x = Flatten()(base_model.output)
output = Dense(20, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=output)

model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

model.summary()
```

Train Model

```
history = model.fit(
    train_data,
    validation_data=test_data,
    epochs=12,    # increased from 6
    steps_per_epoch=len(train_data),
    validation_steps=len(test_data)
)
```

Save the model and evaluate the model.

Milestone 5 — Deployment with Streamlit

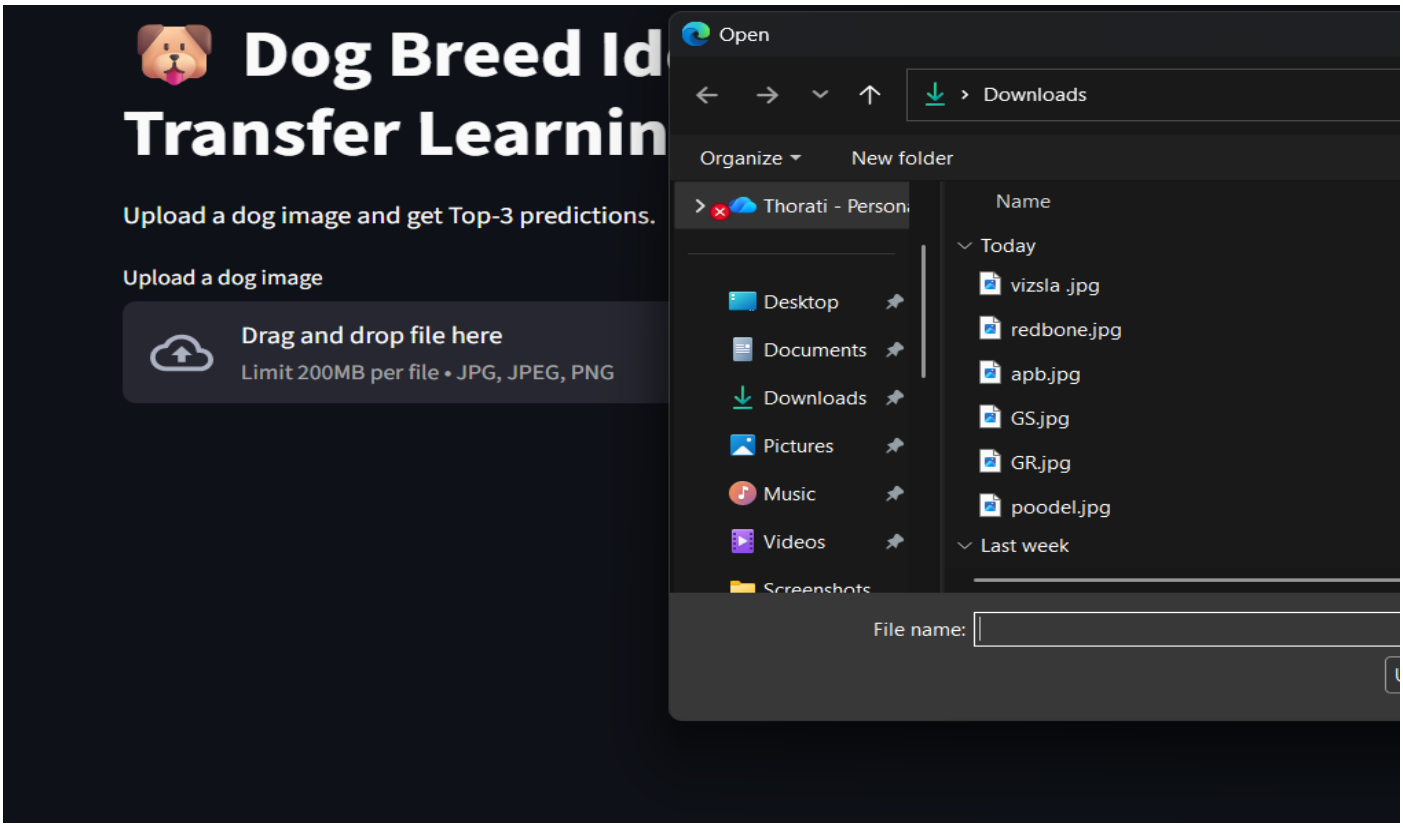
Convert the trained model into an interactive web app interface. Streamlit is used to build a frontend that accepts uploads, runs the model, and shows predictions.

```
from tensorflow.keras.applications.vgg19 import preprocess_input
from PIL import Image

st.set_page_config(page_title="Dog Breed Identification", page_icon="🐶")

st.title("🐶 Dog Breed Identification using Transfer Learning")
st.write("Upload a dog image and get Top-3 predictions.")

@st.cache_resource
def load_my_model():
    return load_model("dogbreed.h5")

model = load_my_model()

# IMPORTANT: Keep this class order SAME as training
class_names = [
    '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'
]

uploaded_file = st.file_uploader("Upload a dog image", type=["jpg","jpeg","png"])

if uploaded_file:
    img = Image.open(uploaded_file).convert("RGB")
    st.image(img, caption="Uploaded Image", use_container_width=True)

    img = img.resize((128, 128))
```

Output Screenshots

Home Page

Image Upload Interface



Prediction Output Interface

Uploaded Image

Top-3 Predictions

golden_retriever : 100.00%

whippet : 0.00%