Transfer Learning-Based Classification of Poultry Diseases for Enhanced Health Management

Documentation

Introduction

Project Title:

Transfer Learning -Based classification of poultry Diseases

for Enhanced Health Management

Team Members:

- Somu.Manikanta Sai
- Lakka Bhargavi
- Velpuri Pavani Yagna Priya
- Shaik Mohammad Khasim

Purpose:

To identify poultry diseases quickly and accurately. Helps farmers detect problems at an early stage.Supports timely treatment and reduces bird deaths.Limits the spread of infections on farms.Reduces the need for expert help every time.Makes poultry health monitoring easier and faster.Improves overall farm productivity and animal care

Goals:

 To identify common poultry diseases effectively using visible signs or symptoms.

 To support early detection so that treatment can begin without delay.

 To reduce the spread of infections among birds in poultry farms.

 To minimize the need for constant expert supervision by providing a user-friendly solution.

Key Features:

- **Automatic Disease Detection :**Identifies poultry diseases quickly based on visible symptoms or images.

- **Easy to Use:** Designed with a simple interface for farmers .

Prerequisites

- ✓ To complete this project, you must require the following software, concepts, and packages

- ✓ **STEP 1: Anaconda Navigator:**
- ✓ Refer to the link below to download Anaconda Navigator.
- ✓ **STEP 2:Python packages:**
- ✓ Open anaconda prompt as administrator
- ✓ Type "pip install numpy" and click enter.
- ✓ Type "pip install pandas" and click enter.
- ✓ Type "pip install scikit-learn" and click enter.
- ✓ Type "pip install matplotlib" and click enter.
- ✓ Type "pip install scipy" and click enter.
- ✓ Type "pip install seaborn" and click enter.
- ✓ Type "pip install tenserflow" and click enter.
- ✓ Type "pip install Flask" and click enter.

Prior Knowledge

You must have prior knowledge of the following topics to complete this project.

- **DL Concepts**
- **Neural Networks:: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/**
  - **Deep Learning Frameworks:: https://www.knowledgehut.com/blog/data-science/pytorch-vs-tensorflow**
  - **Transfer Learning: https://towardsdatascience.com/a-demonstration-of-transfer-learning-of-vgg-convolutional-neural-network-pre-trained-model-with-c9f5b8b1ab0a**
  - **VGG16: https://www.geeksforgeeks.org/vgg-16-cnn-model/**

- **Convolutional Neural Networks (CNNs): https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/ s://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning**

- **Overfitting regulariz: https://www.analyticsvidhya.com/blog/2021/07/prevent-overfitting-using-regularization-techniques/**

- **Optimizers: https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/**

- **Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0**

Project Objectives

By the end of this project, you will:

- Know fundamental concepts and techniques used for Deep Learning.
- Gain a broad understanding of data.
- Have knowledge of pre-processing the data/transformation techniques on outliers and some visualization concepts.

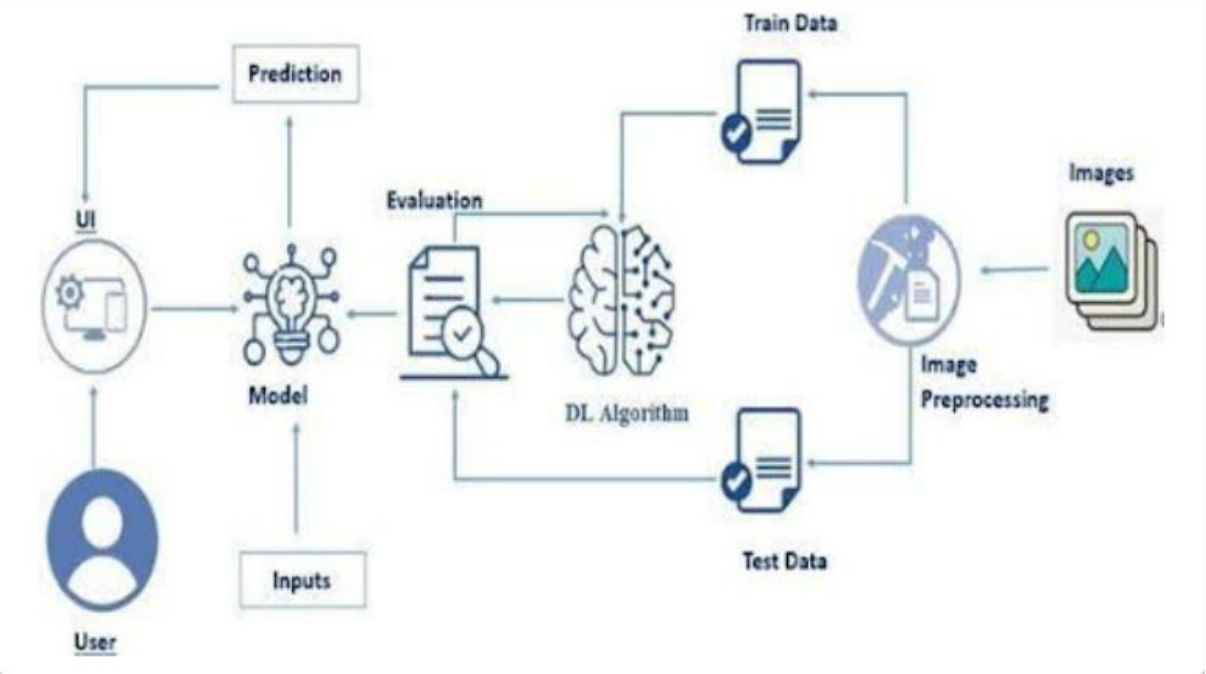
Project Flow

- The user interacts with the UI (User Interface) to choose the image.
- The chosen image is analyzed by the model which is integrated with the flask application.
- Once the model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Data Collection: Collect or download the dataset that you want to train.
- Data pre-processing
    - Data Augmentation
    - Splitting data into train and test
- Model building
    - Import the model-building libraries
    - Initializing the model
    - Training and testing the model
    - Evaluating the performance of the model
    - Save the model
- Application Building
    - Create an HTML file
    - Build python code

Architecture



Project Structure

Create the Project folder which contains files as shown below

Folder Structure

```
project_root/
│
├── static/
│   ├── assets/
│   ├── forms/
│   └── uploads/
│
├── templates/
│   ├── blog-single.html
│   ├── blog.html
│   ├── index.html
│   └── portfolio-details.html
│
├── app.py
├── healthy_vs_rotten.h5
├── ipython.html
└── Readme.txt    ├── Testing_set.csv       ← Test dataset metadata
    └── README.txt            ← Basic setup and usage
```

- We are building a Flask application with HTML pages stored in the templates folder and a Python script app.py for scripting.

- Healthy_vs_rotten.h5 is our saved model. Further, we will use this model for flask integration.

Data Collection and Preparation

- ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset

Using ImageDataGenerator

- Simple image upload panel (via tailwind css or Flask).
- Displays predicted pollengrain with confidence score.

```
from keras.preprocessing.image import ImageDataGenerator

gen = ImageDataGenerator()

train_gen=gen.flow_from_dataframe(train_df, x_col='path', y_col='label', target_size=(224,224),seed=123,
                                  class_mode='categorical', color_mode='rgb', shuffle=True, batch_size=32)

Found 2000 validated image filenames belonging to 4 classes.

test_gen=gen.flow_from_dataframe(test_df, x_col='path', y_col='label', target_size=(224,224),seed=123,
                                  class_mode='categorical', color_mode='rgb', shuffle=True, batch_size=32)

Found 2000 validated image filenames belonging to 4 classes.

val_gen=gen.flow_from_dataframe(val_df, x_col='path', y_col='label', target_size=(224, 224),seed=123,
                                  class_mode='categorical', color_mode='rgb', shuffle=True, batch_size=32)

Found 2000 validated image filenames belonging to 4 classes.
```

- This Python code utilizes Keras, a deep learning library, to preprocess image data for a machine learning model. The code imports libraries and creates three data generators: one for training ('train_gen'), one for testing ('test_gen'), and likely one for validation ('val_gen'). These generators are configured to access data from a DataFrame, which provides the location of images, labels, and other relevant information. In essence, this code snippet prepares the image data for the training process of a machine learning model.

Testing Model & Data Prediction

```
train_gen.class_indices.keys()

dict_keys(['Coccidiosis', 'Healthy', 'New Castle Disease', 'Salmonella'])

labels = ['Coccidiosis', 'Healthy', 'New Castle Disease', 'Salmonella']

from tensorflow.keras.preprocessing.image import load_img, img_to_array

def get_model_prediction(image_path):
    img = load_img(image_path, target_size=(224, 224))
    x = img_to_array(img)
    x = np.expand_dims(x, axis=0)
    predictions = model.predict(x, verbose=0)
    return labels[predictions.argmax()]
```

The code defines a function named build_model that creates a CNN model using Keras.

Application Building

This section has the following tasks

- Building HTML Pages
- Building server-side script

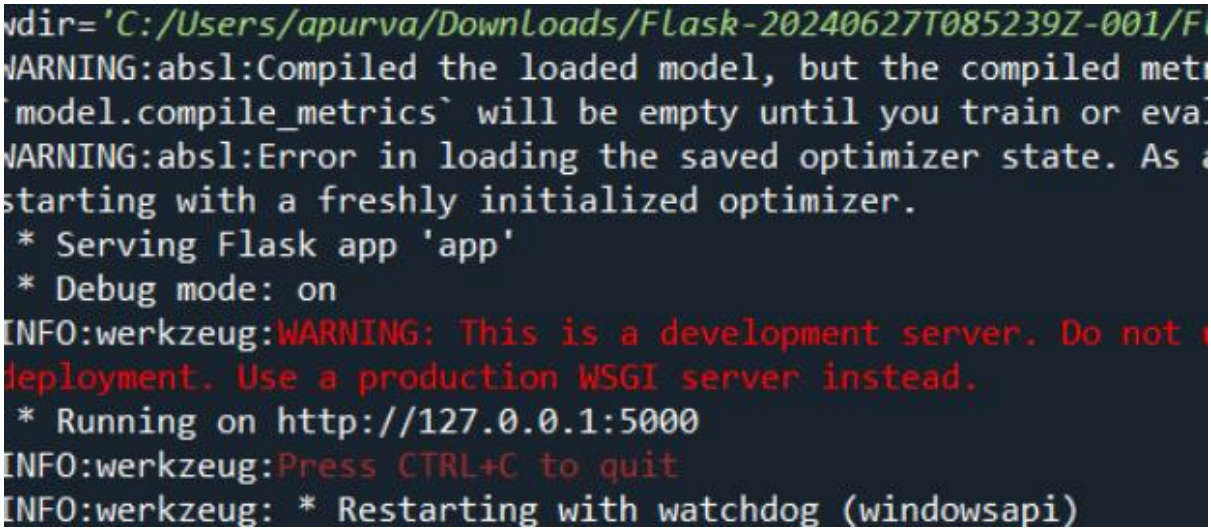
Html Pages

```
@app. route( '/predict', methods =['GET','POST'] )
def output():
    if request.method =='POST':
        f=request.files['pc_image']
        img_path = "static/uploads/" + f.filename
        f.save(img_path)
        img=load_img(img_path,target_size=(224,224))
        # Resize the image to the required size
        # Convert the image to an array and normalize it
        image_array = np.array(img)
        # Add a batch dimension
        image_array = np.expand_dims(image_array, axis=0)
        # Use the pre-trained model to make a prediction
        pred=np.argmax(model.predict(image_array),axis=1)
        index=['Coccidiosis', 'Healthy' , 'New Castle Disease', 'Salmonella']
        prediction = index[int(pred)]
        print("prediction")
        #predict = prediction
        return render_template("contact.html", predict = prediction)
    return render_template('contact.html')
```

Python Code

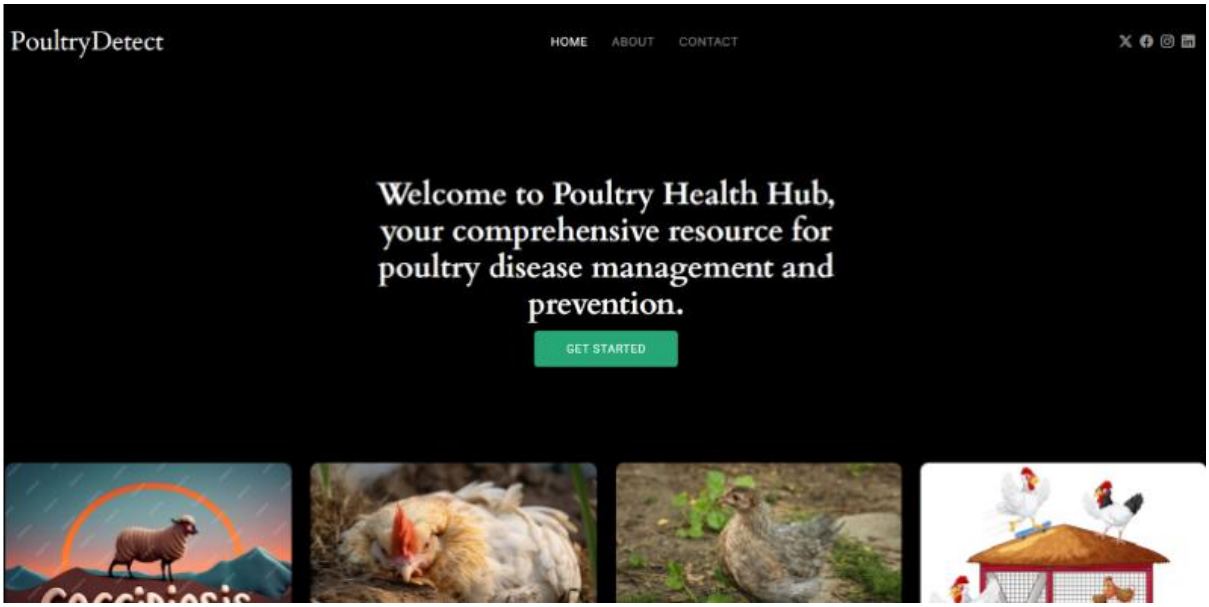
Run the web application

- Open the Anaconda prompt from the start menu.
- Navigate to the folder where your Python script is.
- Now type the “app1.py” command.
- Navigate to the localhost where you can view your web page



- Click on the Get Start button, enter the inputs, click on the submit button, and see the result/prediction on the web.

- The home page looks like this. When you click on the get started “Drop in the image you want to validate!”, you’ll be redirected to the predict section



Here are the images the model predicted