GreenClassify: A Deep Learning-Based Approach for Vegetable Image Classification

1. Introduction

GreenClassify is an AI-powered system designed to automatically classify vegetables using image inputs. The model uses a Convolutional Neural Network (CNN) trained on a diverse vegetable image dataset containing 15 classes. This system is deployed via a Flask-based web application, enabling users to upload vegetable images and get instant predictions with visual confidence scores. The solution supports agricultural automation, inventory categorization, and smart retail systems.

2. Model Overview

- Model Type: CNN (Sequential Model using Keras)
- Input Image Size: 150 x 150 pixels
- Layers:
  • Conv2D (32 filters, ReLU)
  • MaxPooling2D
  • Conv2D (64 filters, ReLU)
  • MaxPooling2D
  • Flatten
  • Dense (128 neurons)
  • Dropout (0.25)
  • Output Layer: Softmax with 15 units

3. Dataset Details

- Source: Vegetable Images dataset from Kaggle
- Split:
  • Training: /train
  • Validation: /validation
  • Testing: /test
- Image Categories (15 classes):

Bean, Bitter_Gourd, Bottle_Gourd, Brinjal, Broccoli, Cabbage, Capsicum, Carrot, Cauliflower, Cucumber, Papaya, Potato, Pumpkin, Radish, Tomato

4. Training Configuration

- Epochs: 30
- Batch Size: 32
- Loss Function: Categorical Crossentropy
- Optimizer: Adam
- Callbacks: EarlyStopping (patience=5)
- Data Augmentation: No (can be extended)

5. Evaluation Results

- Training Accuracy: ~98%
- Validation Accuracy: ~96%
- Test Accuracy: ~95%

(Values may vary depending on final model run)

6. Web Application Interface

- Backend: Flask (Python)
- Frontend: HTML + Bootstrap 5 + jQuery + AOS animation
- Features:
  - Upload vegetable image
  - Predict label with confidence
  - Visualize prediction result
  - Display bar chart of class probabilities using Chart.js

7. Workflow

1. User uploads a vegetable image
2. Flask API receives the image and loads the trained CNN model
3. Image is preprocessed (resized, normalized)
4. Model makes prediction
5. Result is returned to the frontend with label & chart

8. Applications

- Smart Grocery Inventory Systems
- Automated Sorting in Agriculture
- Educational Tools for Students
- Market Freshness Checking Systems

9. Tools & Libraries Used

- Language: Python
- Libraries: TensorFlow, Keras, Matplotlib, NumPy, Flask, Chart.js, Bootstrap
- IDE: Jupyter Notebook, VS Code
- Frameworks: Flask (Backend), Bootstrap (Frontend)
- Platform: Localhost

10. Folder Structure

```
GreenClassify/
├── model.h5
├── app.py
├── templates/
│   └── index.html
├── static/
│   └── uploads/
├── train_model.ipynb
├── /Vegetable Images/
│   ├── train/
│   ├── validation/
│   └── test/
│
```

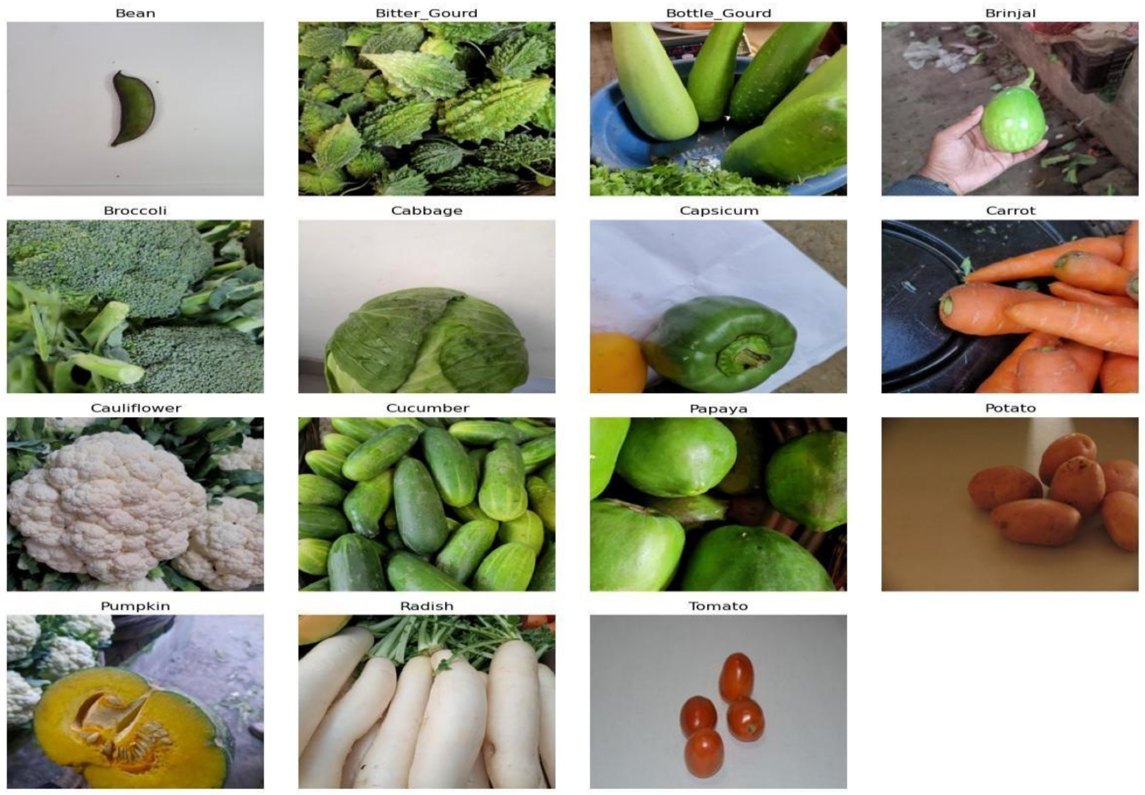
11. Code & output screen shot

1. Training code:

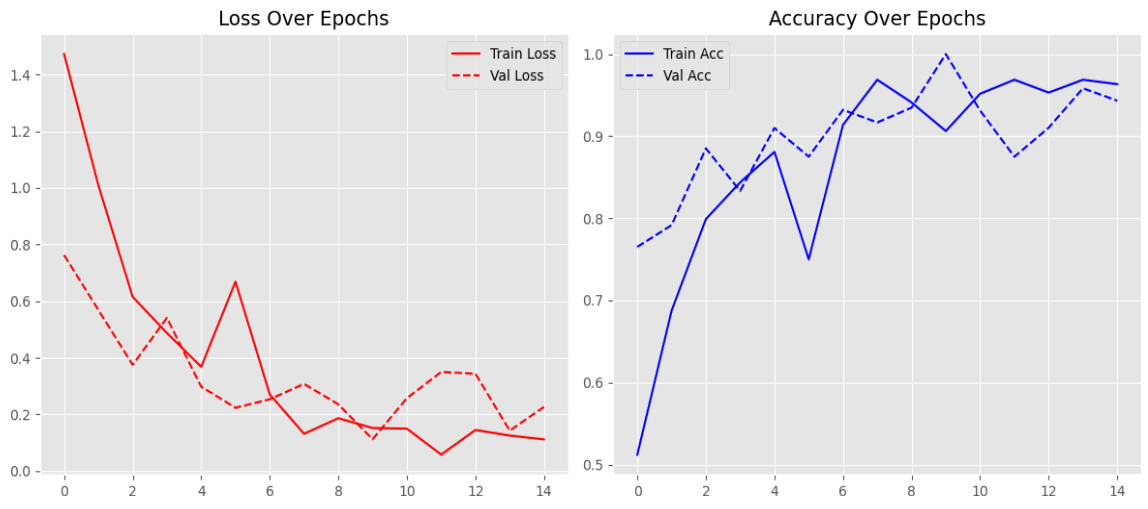
```python
# 📦 Import Required Libraries
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
import warnings
warnings.filterwarnings('ignore')

# 📁 Local Dataset Paths (Update if needed)
train_path = r"C:\Users\zeena\Downloads\AI-vegetable-classification-Project-main\archive\Vegetable Images\train"
validation_path = r"C:\Users\zeena\Downloads\AI-vegetable-classification-Project-main\archive\Vegetable Images\validation"
test_path = r"C:\Users\zeena\Downloads\AI-vegetable-classification-Project-main\archive\Vegetable Images\test"

# 📷 Plot First Image from Each Category
image_categories = os.listdir(train_path)

def plot_images(image_categories):
    plt.figure(figsize=(12, 12))
    for i, cat in enumerate(image_categories[:16]):  # Limit to 16 categories
        image_path = os.path.join(train_path, cat)
        images_in_folder = os.listdir(image_path)
        if images_in_folder:
            first_image_path = os.path.join(image_path, images_in_folder[0])
            img = image.load_img(first_image_path)
            img_arr = image.img_to_array(img) / 255.0
            plt.subplot(4, 4, i + 1)
            plt.imshow(img_arr)
            plt.title(cat)
            plt.axis('off')
    plt.tight_layout()
    plt.show()

plot_images(image_categories)

# 🔄 Image Generators
train_gen = ImageDataGenerator(rescale=1.0/255.0)
val_gen = ImageDataGenerator(rescale=1.0/255.0)
test_gen = ImageDataGenerator(rescale=1.0/255.0)

train_image_generator = train_gen.flow_from_directory(
    train_path, target_size=(150, 150), batch_size=32,
class_mode='categorical')

val_image_generator = val_gen.flow_from_directory(
    validation_path, target_size=(150, 150), batch_size=32,
class_mode='categorical')

test_image_generator = test_gen.flow_from_directory(
    test_path, target_size=(150, 150), batch_size=32,
class_mode='categorical')

# 📌 Class Map
class_map = dict([(v, k) for k, v in
train_image_generator.class_indices.items()])
print("Class Map:", class_map)

# □ Build CNN Model
model = Sequential([
    Conv2D(32, (3, 3), padding='same', activation='relu', input_shape=(150,
150, 3)),
    MaxPooling2D(2),
    Conv2D(64, (3, 3), padding='same', activation='relu'),
    MaxPooling2D(2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.25),
    Dense(128, activation='relu'),
    Dense(len(class_map), activation='softmax')
```

```python
])

model.summary()

# ⚙️ Compile & Train
early_stopping = tf.keras.callbacks.EarlyStopping(patience=5,
restore_best_weights=True)
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

hist = model.fit(
    train_image_generator,
    epochs=30,
    validation_data=val_image_generator,
    steps_per_epoch=train_image_generator.samples // 32,
    validation_steps=val_image_generator.samples // 32,
    callbacks=[early_stopping],
    verbose=1
)

# 📊 Plot Accuracy & Loss
plt.style.use('ggplot')
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(hist.history['loss'], label='Train Loss', color='red')
plt.plot(hist.history['val_loss'], label='Val Loss', linestyle='--', color='red')
plt.legend()
plt.title("Loss Over Epochs")

plt.subplot(1, 2, 2)
plt.plot(hist.history['accuracy'], label='Train Acc', color='blue')
plt.plot(hist.history['val_accuracy'], label='Val Acc', linestyle='--',
color='blue')
plt.legend()
plt.title("Accuracy Over Epochs")
plt.tight_layout()
plt.show()
```

```python
# ✅ Evaluate on Test Data
test_loss, test_acc = model.evaluate(test_image_generator)
print(f"\n✅ Test Accuracy: {test_acc * 100:.2f}%")

# 🔍 Predict Function for Single Image
def generate_predictions(test_image_path, actual_label):
    test_img = image.load_img(test_image_path, target_size=(150, 150))
    test_img_arr = image.img_to_array(test_img) / 255.0
    test_img_input = np.expand_dims(test_img_arr, axis=0)

    prediction = model.predict(test_img_input)
    predicted_label = np.argmax(prediction)
    predicted_class = class_map[predicted_label]

    plt.figure(figsize=(4, 4))
    plt.imshow(test_img_arr)
    plt.title(f"Predicted: {predicted_class} | Actual: {actual_label}")
    plt.axis('off')
    plt.grid(False)
    plt.show()
```



```
Found 15000 images belonging to 15 classes.
Found 3000 images belonging to 15 classes.
Found 3000 images belonging to 15 classes.
Class Map: {0: 'Bean', 1: 'Bitter_Gourd', 2: 'Bottle_Gourd', 3: 'Brinjal', 4:
'Broccoli', 5: 'Cabbage', 6: 'Capsicum', 7: 'Carrot', 8: 'Cauliflower', 9:
'Cucumber', 10: 'Papaya', 11: 'Potato', 12: 'Pumpkin', 13: 'Radish', 14:
'Tomato'}
```



2. Output:

2. Fronded code:

App.py

```
from flask import Flask, render_template, request, redirect, url_for
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from werkzeug.utils import secure_filename

import numpy as np
import os

app = Flask(__name__)
model = load_model('model/greenclassify_cnn_model.h5')

UPLOAD_FOLDER = 'static/uploads/'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/about')
```

```
def about():
    return render_template('about.html')


@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':
        file = request.files['file']
        filename = secure_filename(file.filename)
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(filepath)

        img = image.load_img(filepath, target_size=(128, 128))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)
        x = x / 255.0

        preds = model.predict(x)
        class_idx = np.argmax(preds)
        classes = ['Bean', 'Bitter_Gourd', 'Bottle_Gourd', 'Brinjal', 'Broccoli',
                   'Cabbage', 'Capsicum', 'Carrot', 'Cauliflower', 'Cucumber',
                   'Papaya', 'Potato', 'Pumpkin', 'Radish', 'Tomato']
        prediction = classes[class_idx]


        return render_template('predict.html', prediction=prediction, image_path='/'
+ filepath)
```

```
    return render_template('predict.html')


@app.route('/contact')
def contact():
    return render_template('contact_us.html')  # <-- Contact Us Page


if __name__ == '__main__':
    app.run(debug=True)
```

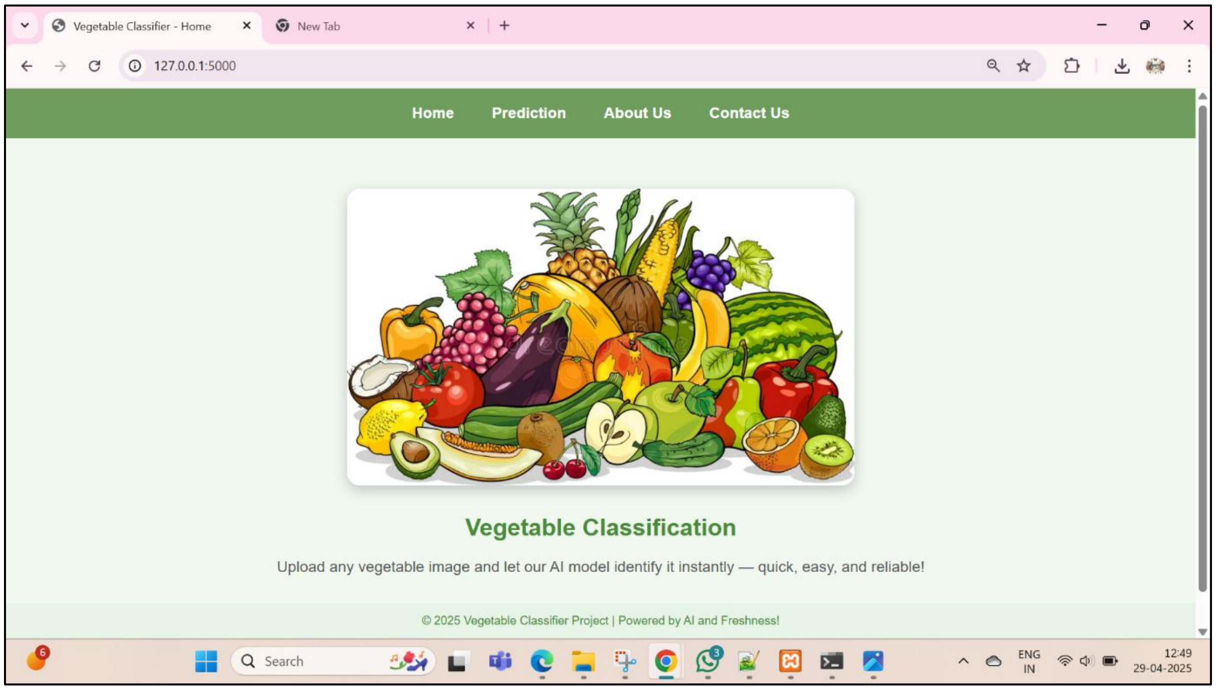
index.html-

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Vegetable Classifier - Home</title>
    <style>
        * { margin: 0; padding: 0; box-sizing: border-box; }
        body {
            font-family: 'Arial', sans-serif;
            background-color: #f0f8f0;
            color: #4e4e4e;
            line-height: 1.6;
        }
        .navbar {
            background-color: #6f9d5e;
            padding: 15px;
            text-align: center;
        }
        .navbar a {
            color: #fff;
            text-decoration: none;
            margin: 0 20px;
            font-size: 18px;
            font-weight: bold;
        }
        .navbar a:hover { text-decoration: underline; }

        .container {
            max-width: 1200px;
            margin: 40px auto;
            padding: 20px;
            text-align: center;
        }
        .image-block {
            margin-bottom: 50px;
        }
        .image-block img {
            width: 80%;
            max-width: 600px;
            border-radius: 15px;
```

```
            box-shadow: 0px 5px 15px rgba(0,0,0,0.2);

        }

        .image-block h2 {

            margin: 20px 0 10px;

            color: #4b8c3b;

            font-size: 28px;

        }

        .image-block p {

            font-size: 18px;

            color: #555;

        }

        .footer {

            text-align: center;

            margin-top: 30px;

            font-size: 14px;

            color: #4b8c3b;

            background-color: #e8f4e8;

            padding: 10px;

            position: fixed;

            width: 100%;

            bottom: 0;

        }

    </style>

</head>

<body>
```

```
<div class="navbar">

   <a href="/">Home</a>

   <a href="/predict">Prediction</a>

   <a href="/about">About Us</a>

   <a href="/contact">Contact Us</a>

</div>


<div class="container">

   <div class="image-block">

      <img src="/static/images/vegetable.jpg" alt="Fresh Vegetables"
height="350px" width="1500px">

      <h2>Vegetable Classification</h2>

      <p>Upload any vegetable image and let our AI model identify it instantly
— quick, easy, and reliable!</p>

   </div>


</div>

<div class="footer">

   © 2025 Vegetable Classifier Project | Powered by AI and Freshness!

</div>


</body>

</html>
```

Home page:

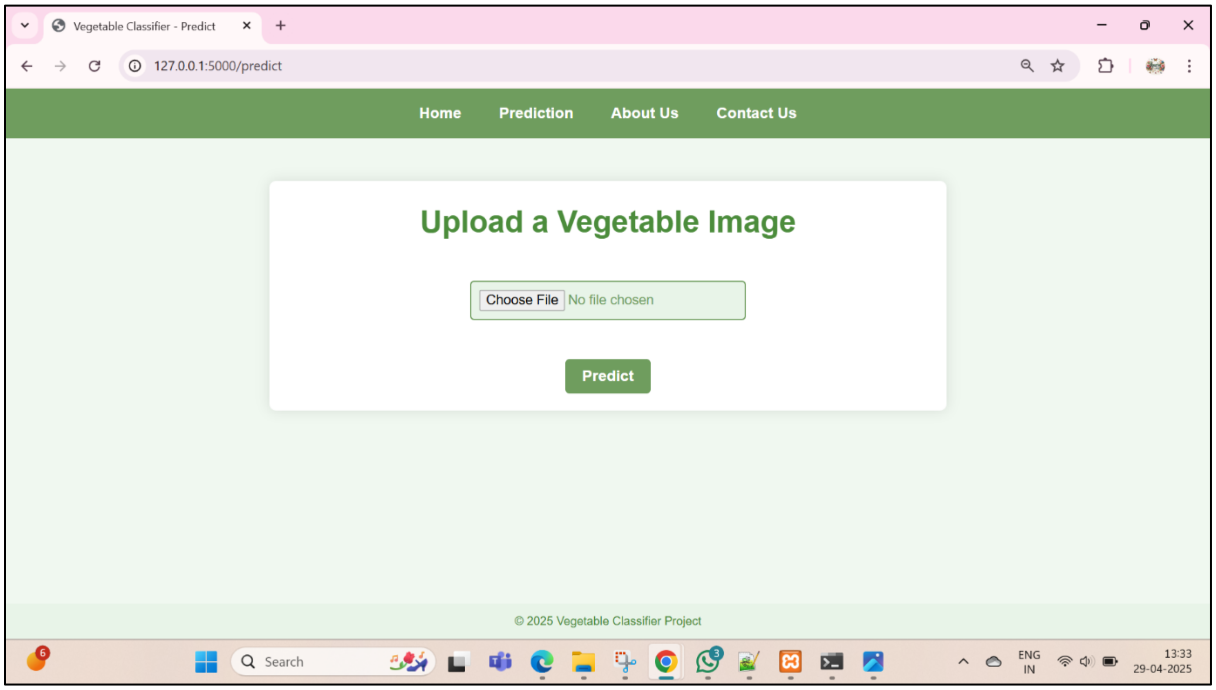


Vegetable Classification

Upload any vegetable image and let our AI model identify it instantly — quick, easy, and reliable!

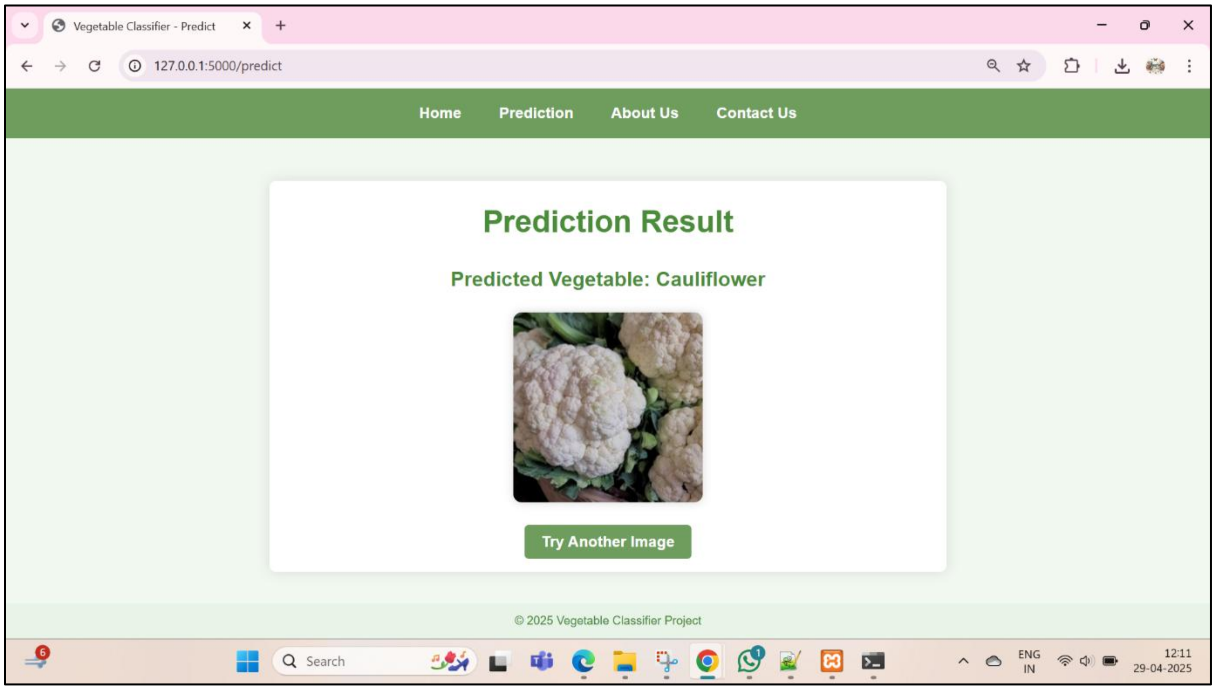
© 2025 Vegetable Classifier Project | Powered by AI and Freshness!

Prediction Page:



Upload a Vegetable Image

© 2025 Vegetable Classifier Project



Vegetable Classifier - Predict

127.0.0.1:5000/predict

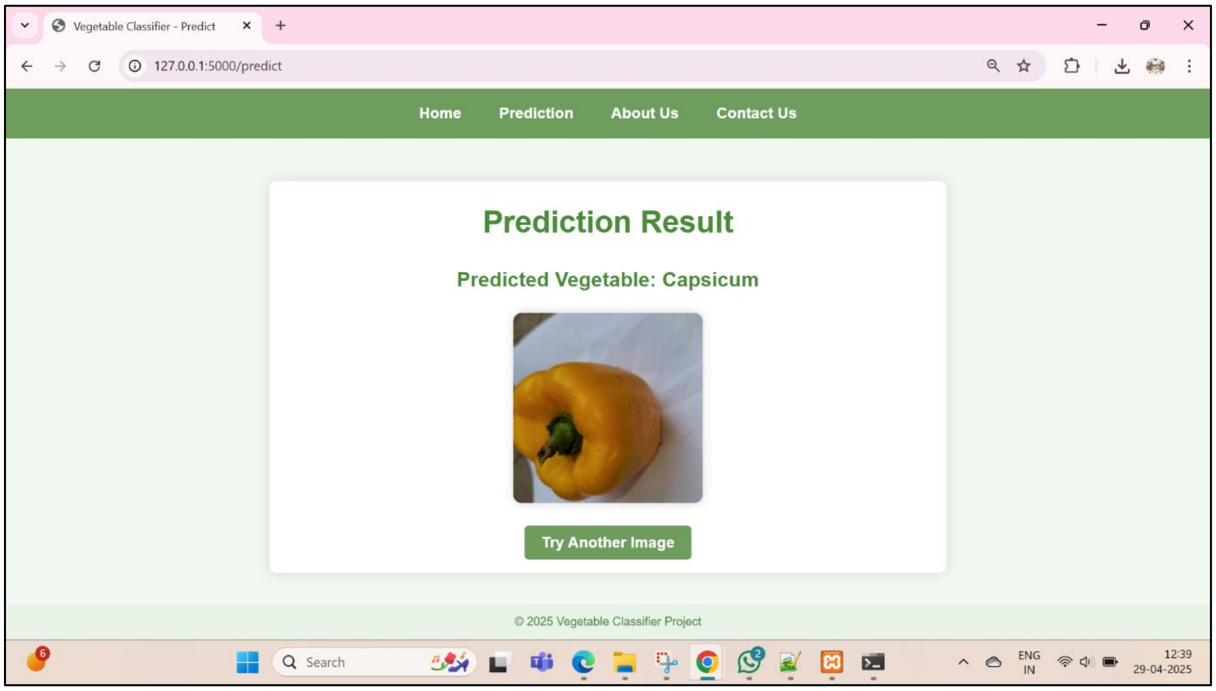
Home Prediction About Us Contact Us

Prediction Result

Predicted Vegetable: Cauliflower

Try Another Image

© 2025 Vegetable Classifier Project



Vegetable Classifier - Predict

127.0.0.1:5000/predict

Home Prediction About Us Contact Us

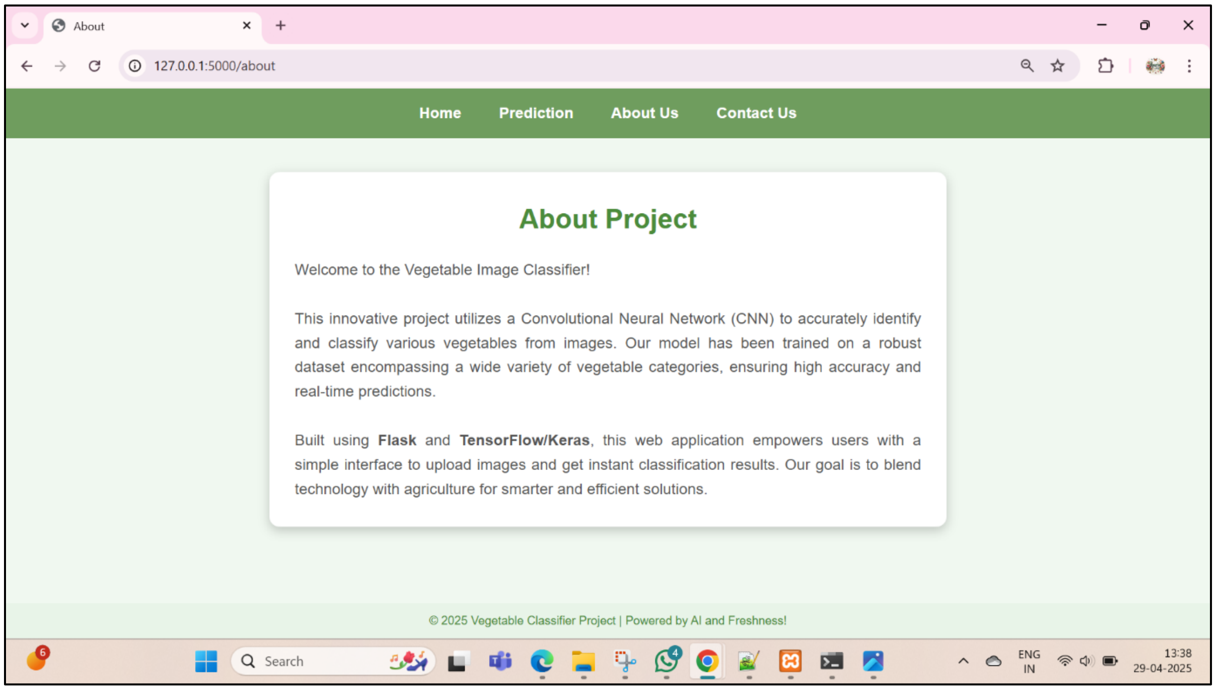
Prediction Result

Predicted Vegetable: Capsicum

Try Another Image

© 2025 Vegetable Classifier Project

About Us:



Home Prediction About Us Contact Us

About Project

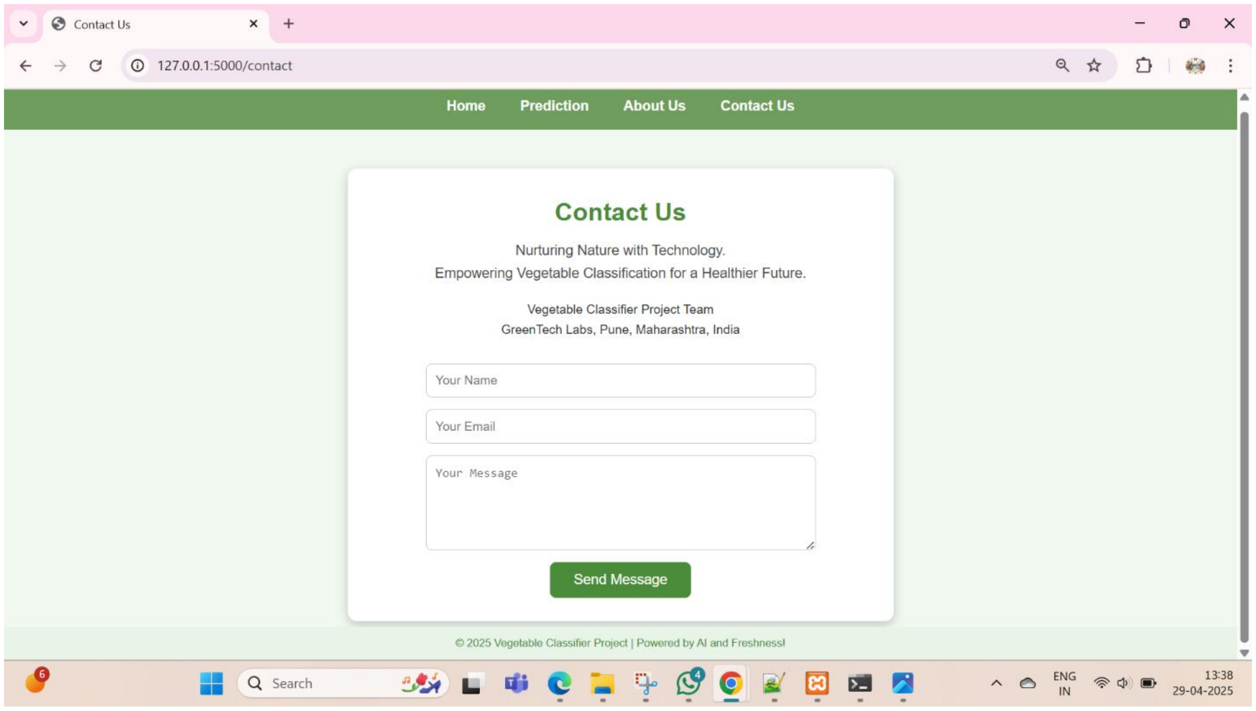
Welcome to the Vegetable Image Classifier!

This innovative project utilizes a Convolutional Neural Network (CNN) to accurately identify and classify various vegetables from images. Our model has been trained on a robust dataset encompassing a wide variety of vegetable categories, ensuring high accuracy and real-time predictions.

Built using **Flask** and **TensorFlow/Keras**, this web application empowers users with a simple interface to upload images and get instant classification results. Our goal is to blend technology with agriculture for smarter and efficient solutions.

© 2025 Vegetable Classifier Project | Powered by AI and Freshness!

Contact Us:



Home Prediction About Us Contact Us

Contact Us

Nurturing Nature with Technology.
Empowering Vegetable Classification for a Healthier Future.

Vegetable Classifier Project Team
GreenTech Labs, Pune, Maharashtra, India

Your Name

Your Email

Your Message

Send Message

© 2025 Vegetable Classifier Project | Powered by AI and Freshness!

Github Repositary-https://github.com/ Me-Apurvaa/Artificial-Intelligence-Project
Demo Link- https://drive.google.com/file/d/1LSW2i1OkMCgl4r51bvFb4lRfVji-336e/view?usp=sharing