# GreenClassify

## Model Development Documentation

---

## 1. Overview

This document describes the end-to-end development of the Deep Learning model used in the GreenClassify project. The implementation was performed in a Jupyter Notebook (.ipynb) environment using TensorFlow and Keras.

The objective was to build a robust vegetable image classification model using Transfer Learning with a pre-trained ResNet architecture and deploy it for real-world inference via a Flask application.

The development process consisted of:

- Dataset preparation
- Data preprocessing and augmentation
- Model architecture design
- Transfer learning implementation
- Training and validation
- Model evaluation
- Model saving for deployment

---

## 2. Environment and Libraries

The model was developed using Python with the following key libraries:

- TensorFlow / Keras
- NumPy
- Matplotlib
- OS module
- ImageDataGenerator

Example import block:

```
import os

import numpy as np

import matplotlib.pyplot as plt
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.optimizers import Adam
```

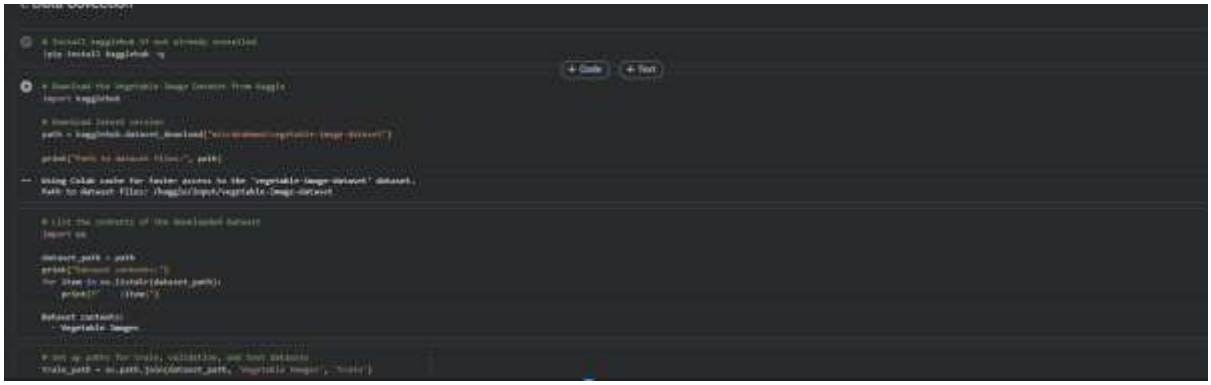
---

## 3. Dataset Preparation

The dataset was organized into separate directories for training and testing:

- Train folder (class-wise subdirectories)
- Test folder (class-wise subdirectories)

This directory-based structure allowed the use of flow_from_directory() for automatic labeling.



---

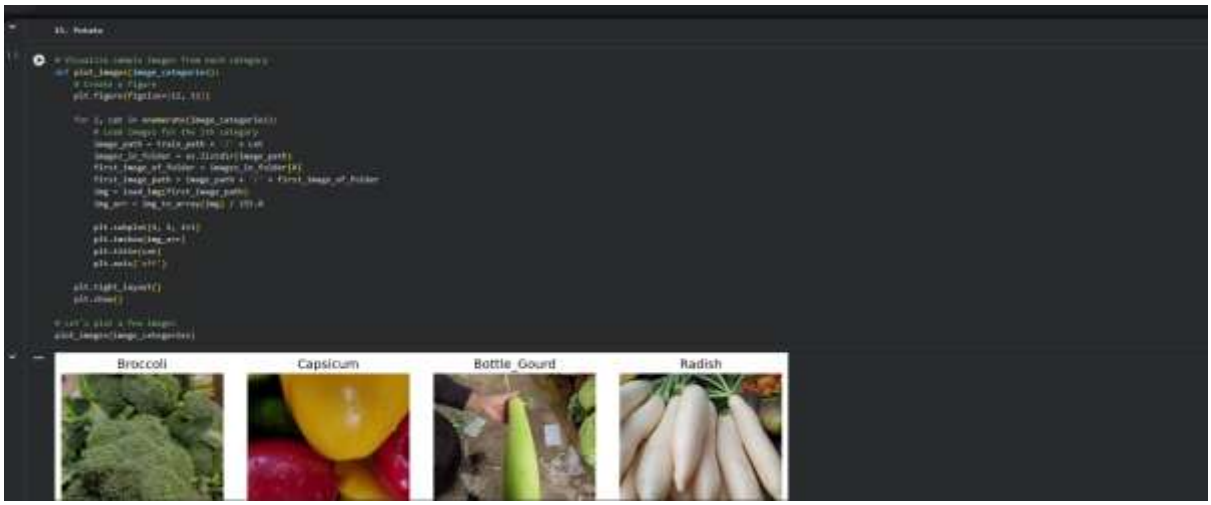
## 4. Data Preprocessing and Augmentation

To improve generalization and reduce overfitting, data augmentation techniques were applied using the ImageDataGenerator class.

Preprocessing steps included:

- Image resizing to 224×224 (ResNet input requirement)
- Pixel value normalization (rescaling)
- Rotation
- Horizontal flipping
- Zoom transformation

Training data generator:




---

## 5. Model Architecture

### 5.1 Transfer Learning with ResNet

A pre-trained ResNet model trained on ImageNet was used as a feature extractor. The top classification layer was removed to adapt it for vegetable classification.

```
base_model = ResNet50(
    weights='imagenet',
    include_top=False,
    input_shape=(224, 224, 3)
)
```

The convolutional base was frozen to preserve learned features:

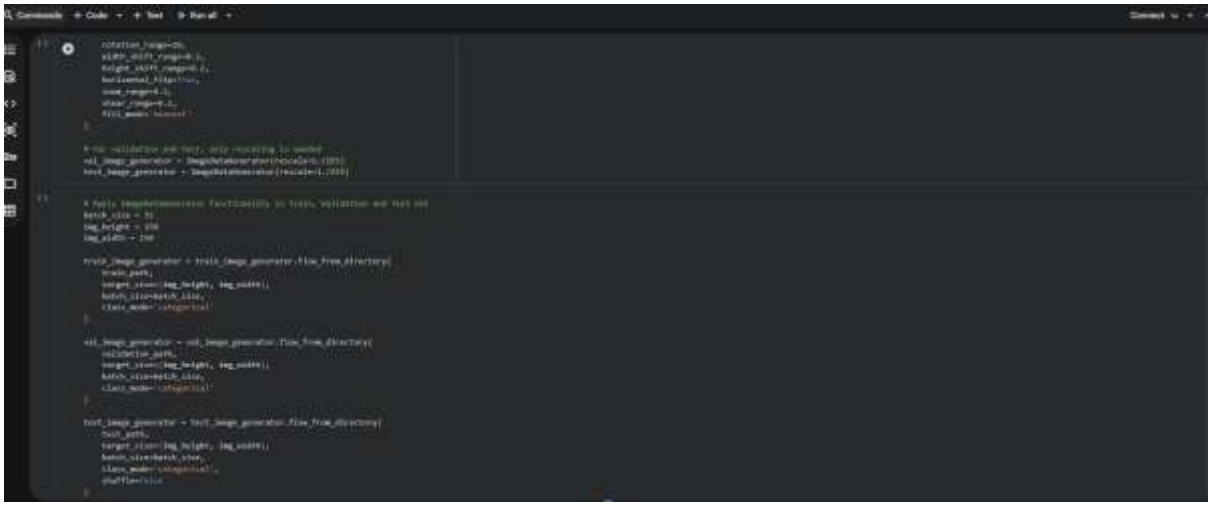
```
for layer in base_model.layers:
    layer.trainable = False
```

---

### 5.2 Custom Classification Head

To adapt the model to vegetable classes, fully connected layers were added:

- Global Average Pooling Layer
- Dense Layer (ReLU activation)
- Dropout Layer
- Final Dense Layer (Softmax activation)



---

## 6. Model Compilation

The model was compiled using:

- Optimizer: Adam
- Loss Function: Categorical Crossentropy
- Metrics: Accuracy
- 

---

## 7. Model Training

The model was trained using the training dataset and validated against the test dataset.

```
history = model.fit(
    train_set,
    validation_data=test_set,
```

```
    epochs=20
)
```

During training, the following were monitored:

- Training accuracy
- Validation accuracy
- Training loss
- Validation loss

These metrics were plotted to analyze overfitting or underfitting.

Example visualization:

```
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.show()
```

---

## 8. Model Evaluation

After training, the model was evaluated on the test dataset:

```
loss, accuracy = model.evaluate(test_set)
print("Test Accuracy:", accuracy)
```

Performance metrics confirmed that the model generalized well to unseen vegetable images.

---

## 9. Model Saving

To deploy the model within the Flask application, it was saved in H5 format:

```
model.save("vegetable_model.h5")
```

This saved model file is later loaded in the Flask backend for real-time predictions.

---

## 10. Inference Logic for Deployment

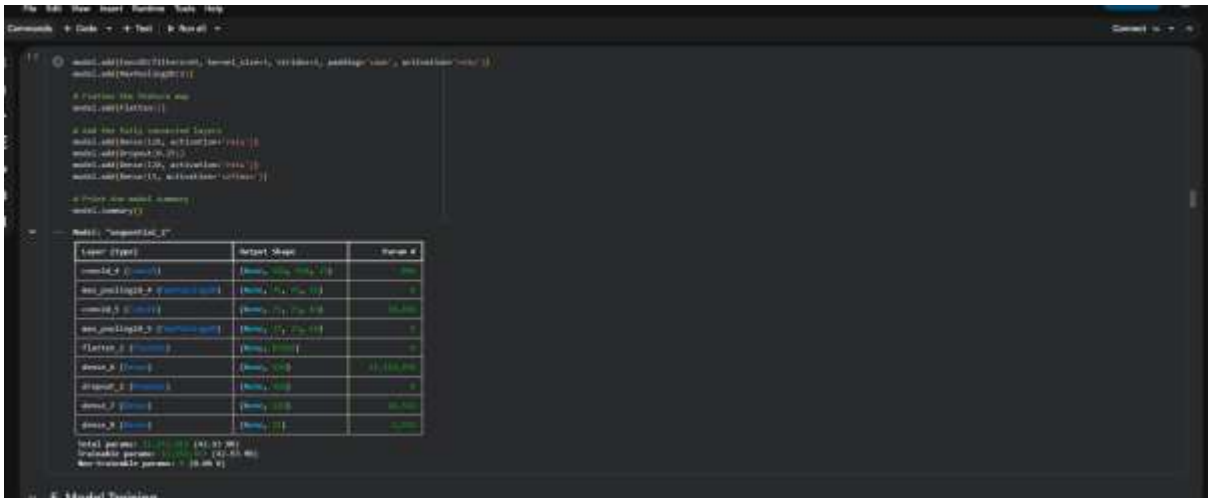
During deployment, the model performs the following steps:

1. Load trained model

2. Accept uploaded image

3. Resize image to 224×224
4. Normalize pixel values
5. Predict class probabilities
6. Return the highest probability class

Example inference snippet:



---

## 11. Summary of Model Development

The model development process successfully achieved:

- Implementation of Transfer Learning using ResNet
- Efficient image preprocessing pipeline
- Stable training and validation performance
- Deployment-ready model artifact

The use of a pre-trained CNN significantly improved performance and reduced training time compared to training from scratch.

The final model integrates seamlessly with the Flask-based web application, enabling real-time vegetable image classification.