ResNet50 Model Documentation

1. Model Overview: -

The model is a deep convolutional neural network based on ResNet50, designed for multi-class plant disease classification using transfer learning and fine-tuning. ResNet50 employs a 50-layer deep residual architecture that utilizes shortcut connections to effectively address the vanishing gradient problem, enabling stable training of deep networks.

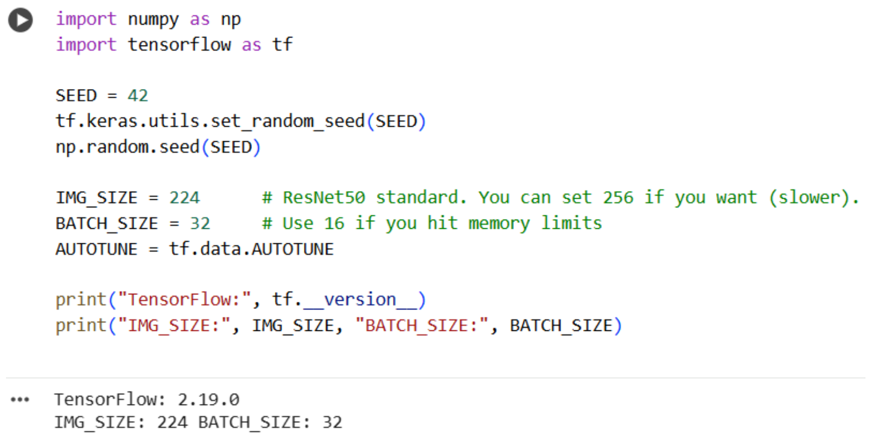
The architecture leverages pre-trained ImageNet weights to extract discriminative low-level and high-level visual features from plant leaf images. A custom classification head is added on top of the backbone network to adapt the model to the target plant disease classes. This approach improves learning efficiency, accelerates convergence, and enhances generalization performance, especially when training data is limited**.**

2. Model Architecture: -

2.1 Base Network:

Pre-training: ImageNet

Input Shape: 224 × 224× 3



```
import numpy as np
import tensorflow as tf

SEED = 42
tf.keras.utils.set_random_seed(SEED)
np.random.seed(SEED)

IMG_SIZE = 224      # ResNet50 standard. You can set 256 if you want (slower).
BATCH_SIZE = 32     # Use 16 if you hit memory limits
AUTOTUNE = tf.data.AUTOTUNE

print("TensorFlow:", tf.__version__)
print("IMG_SIZE:", IMG_SIZE, "BATCH_SIZE:", BATCH_SIZE)
```

```
TensorFlow: 2.19.0
IMG_SIZE: 224 BATCH_SIZE: 32
```

include_top=False

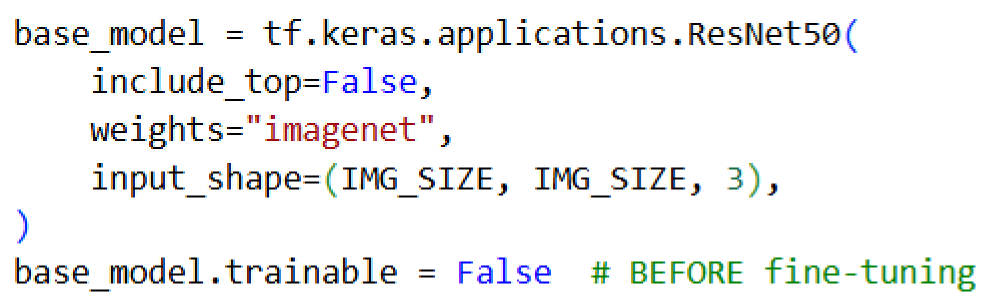
- The **top layers** of ResNet50 are **fully connected layers** trained to classify **1000 ImageNet classes**
- task has **different classes** (plant diseases)
- So **remove** the original classifier and add **custom head**

base_model.trainable = False

- Freezes **all ResNet50 weights**
- Prevents them from being updated during backpropagation
- Only **new classification head learns**

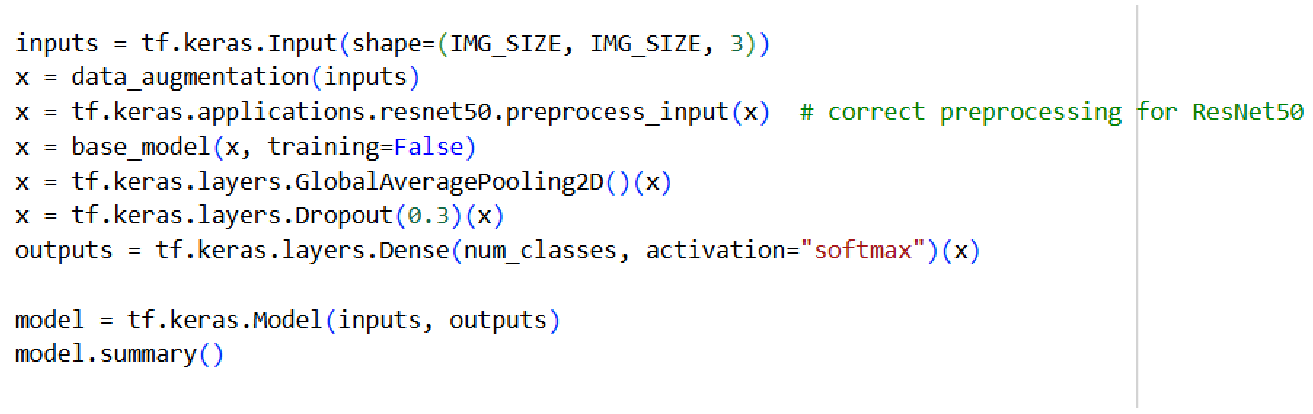
Why this is important

- ImageNet features are already very good
- Freezing prevents:
    - Overfitting
    - Destroying pre-trained knowledge
    - Unstable training on small datasets



```
base_model = tf.keras.applications.ResNet50(
    include_top=False,
    weights="imagenet",
    input_shape=(IMG_SIZE, IMG_SIZE, 3),
)
base_model.trainable = False  # BEFORE fine-tuning
```

2.2 classification head:



```
inputs = tf.keras.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
x = data_augmentation(inputs)
x = tf.keras.applications.resnet50.preprocess_input(x)  # correct preprocessing for ResNet50
x = base_model(x, training=False)
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dropout(0.3)(x)
outputs = tf.keras.layers.Dense(num_classes, activation="softmax")(x)

model = tf.keras.Model(inputs, outputs)
model.summary()
```

- The input layer receives RGB images of size **IMG_SIZE × IMG_SIZE × 3**.
- **Data augmentation** is applied to improve model robustness and generalization.

- Images are normalized using the **ResNet50 preprocessing function** to match the ImageNet training distribution.
- The pre-trained **ResNet50 backbone** is used for feature extraction with training=False, ensuring frozen weights and batch normalization statistics remain unchanged.
- **Global Average Pooling** reduces spatial feature maps into a compact feature vector, lowering model complexity and overfitting.
- **Dropout (rate = 0.3)** is applied as a regularization technique to prevent overfitting.
- A **dense output layer with softmax activation** produces class probability scores for multi-class classification.

3. Loss Function: -

Sparse Categorical Cross-Entropy is employed as the loss function.

This loss function is suitable for multi-class classification tasks where labels are provided as integer indices rather than one-hot encoded vectors. It efficiently penalizes incorrect predictions while maintaining numerical stability.

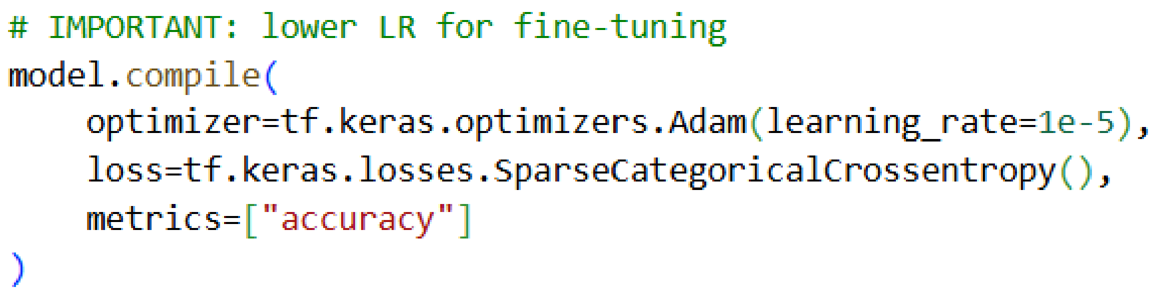
4. Optimization Strategy: -

The **Adam optimizer** is employed due to its adaptive learning rate and efficient convergence properties.

- **Initial Learning Rate (Stage 1):** Default Adam learning rate
- **Fine-Tuning Learning Rate (Stage 2):**

1×10*-5

A lower learning rate is applied during fine-tuning to allow gradual adaptation of pre-trained weights without disrupting learned representations.



```
# IMPORTANT: lower LR for fine-tuning
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(),
    metrics=["accuracy"]
)
```

<u>5. Training Strategy: -</u>

5.1 Stage 1 – Feature Extraction

- ResNet50 backbone fully frozen
- Only the classification head is trained
- Prevents overfitting on limited data
- Stabilizes early training

5.2 Stage 2 – Fine-Tuning

- Backbone unfrozen
- Only the **last 30 layers** of ResNet50 are trainable
- Earlier layers remain frozen
- **Batch Normalization layers remain frozen**
- Lower learning rate applied
- Enables task-specific feature refinement.

```
# Unfreeze only the last N layers (keeps training stable)
N = 30
for layer in base_model.layers[:-N]:
    layer.trainable = False
```

<u>6. Regularization Techniques: -</u>

```
callbacks_stage2 = [
    tf.keras.callbacks.ModelCheckpoint(
        filepath=stage2_ckpt,
        monitor="val_accuracy",
        mode="max",
        save_best_only=True,
        save_weights_only=True,
        verbose=1
    ),
    tf.keras.callbacks.EarlyStopping(
        monitor="val_accuracy",
        mode="max",
        patience=6,
        restore_best_weights=True,
        verbose=1
    ),
    tf.keras.callbacks.ReduceLROnPlateau(
        monitor="val_loss",
        factor=0.2,
        patience=3,
        min_lr=1e-6,
        verbose=1
    )
]
```

- **Model Checkpointing:** Saves the model weights corresponding to the **highest validation accuracy**, ensuring that the best-performing model is preserved during training.
- **Early Stopping:** Terminates training when validation accuracy fails to improve for a predefined number of epochs, preventing unnecessary training and overfitting while restoring the best weights.
- **Learning Rate Reduction on Plateau:** Automatically reduces the learning rate when validation loss stops improving, allowing finer weight updates and improving convergence during fine-tuning.

7. Model Accuracy: -

Before tuning

```
===== BEFORE FINE-TUNING (Stage 1) =====
Best Val Accuracy (Stage 1): 1.0
Test Accuracy (Stage 1):     0.9988558292388916
Saved Stage-1 model to: /content/drive/MyDrive/Plantvillage_Dataset/resnet50_stage1_feature_extraction.keras
```

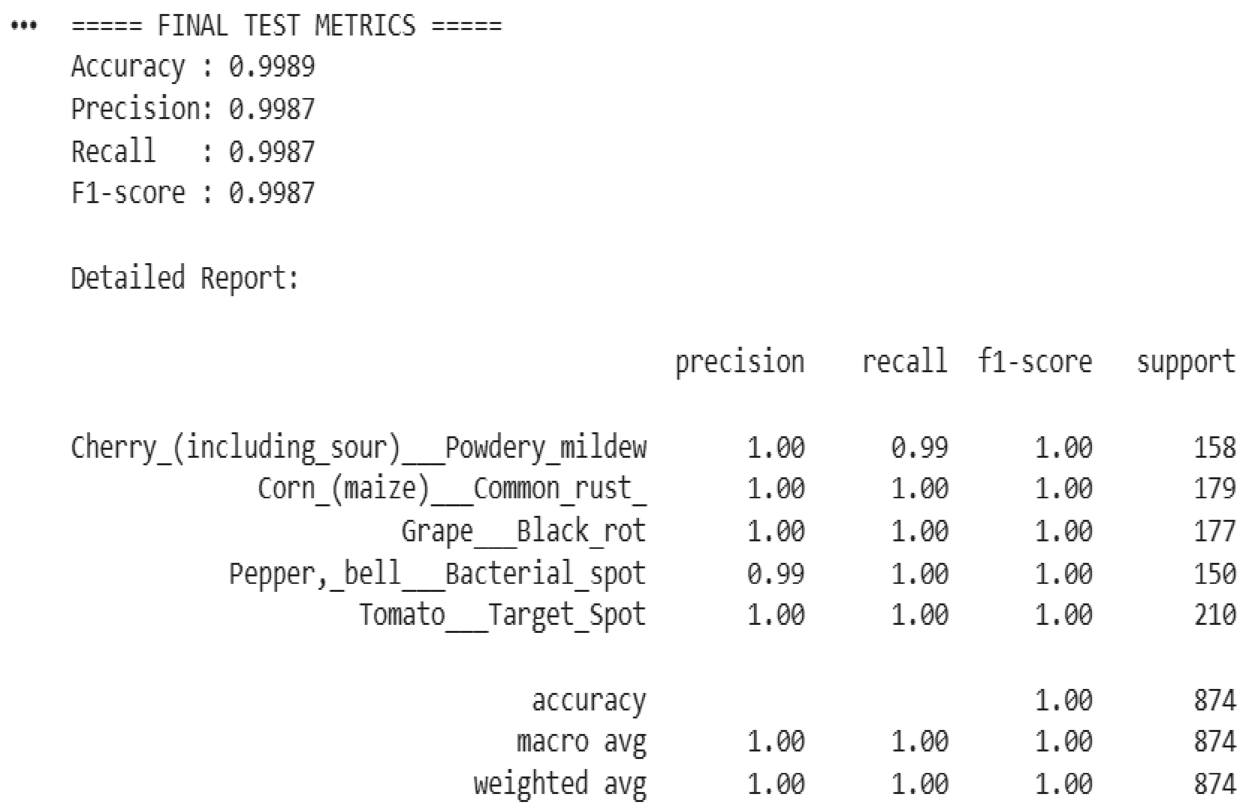
After tuning

```
===== AFTER FINE-TUNING (Stage 2) =====
Best Val Accuracy (Stage 2): 1.0
Test Accuracy (Stage 2):     0.9988558292388916
```

8. Evaluation: -

Classification Report:

The classification report shows consistently high **precision, recall, and F1-scores** across all classes, indicating robust and well-balanced classification performance.

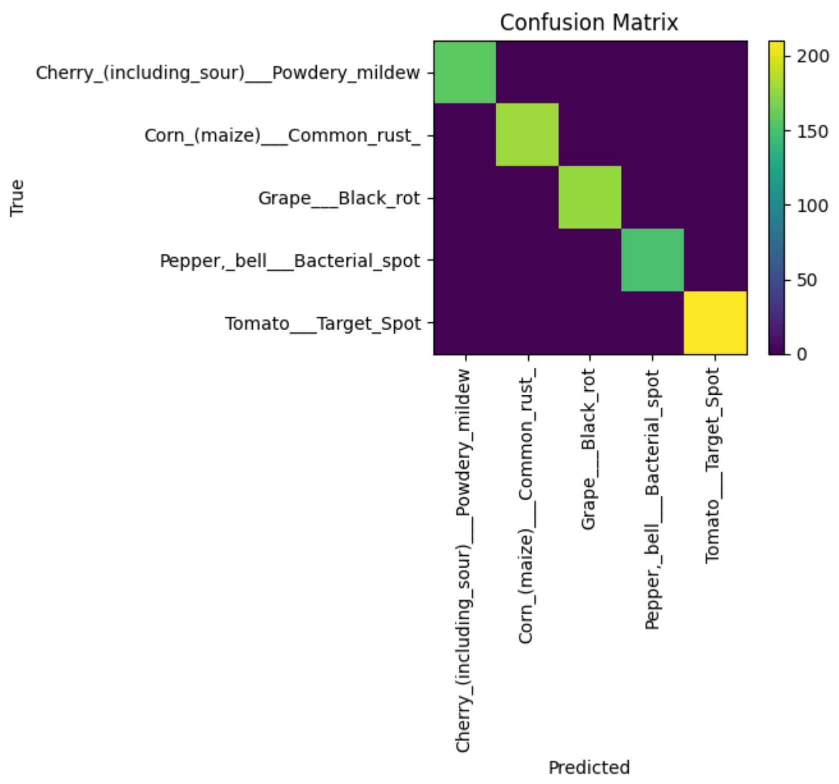


```
•••  ===== FINAL TEST METRICS =====
     Accuracy : 0.9989
     Precision: 0.9987
     Recall   : 0.9987
     F1-score : 0.9987

     Detailed Report:

                                          precision    recall  f1-score   support

     Cherry_(including_sour)___Powdery_mildew       1.00      0.99      1.00       158
                   Corn_(maize)___Common_rust_       1.00      1.00      1.00       179
                             Grape___Black_rot       1.00      1.00      1.00       177
                Pepper,_bell___Bacterial_spot       0.99      1.00      1.00       150
                           Tomato___Target_Spot       1.00      1.00      1.00       210

                                      accuracy                           1.00       874
                                     macro avg       1.00      1.00      1.00       874
                                  weighted avg       1.00      1.00      1.00       874
```

Confusion Matrix:

The confusion matrix demonstrates that most predictions lie along the diagonal, indicating high classification accuracy with minimal confusion between visually similar disease classes.

•••



9. Grad-CAM: -

The **Grad-CAM visualization** highlights disease-affected regions of plant leaves, confirming that the model focuses on relevant pathological features rather than background noise. This provides interpretability and confidence in the model's decision-making process.

•••