Fine-Tuning a Pre-trained CNN Model (VGG16) for Image Classification

1. Introduction

In this project, we utilize a pre-trained **VGG16** model and modify it to suit our custom image classification task. The model is fine-tuned by freezing initial layers, adding new layers for feature extraction, and gradually unfreezing some layers for better learning.

2. Steps Implemented

Step 1: Load Pre-trained VGG16 Model

- The **VGG16** model is loaded with **pre-trained weights from ImageNet**.
- The top classification layer is removed (include_top=False).
- The input size is set to (224, 224, 3).

Step 2: Modify the Model Architecture

- We **freeze** all layers initially to retain pre-trained features.
- A **Global Average Pooling (GAP)** layer is used instead of Flatten to improve performance.
- A **Dense layer (128 neurons, ReLU activation) with L2 Regularization** is added to reduce overfitting.
- A **Dropout layer (0.5)** is introduced to prevent overfitting.
- The final **Dense output layer (Sigmoid activation)** is added for binary classification.

Step 3: Data Preprocessing & Augmentation

- Images are loaded using ImageDataGenerator with **data augmentation techniques**:
    - Rescaling
    - Rotation, width/height shifts, shear, zoom
    - Horizontal flipping
    - Filling empty areas with nearest pixels

- Data is split into **training (80%) and validation (20%)** sets.

Step 4: Initial Training with Frozen Layers

- The model is **compiled** using:
  - **Adam optimizer**
  - **Binary Cross-Entropy loss** (for binary classification)
  - **Accuracy as the evaluation metric**
- Training is performed for **5 epochs** with frozen layers.
- **Early Stopping** is used to prevent overfitting.
- **Model Checkpoint** is used to save the best model.

Step 5: Fine-Tuning by Unfreezing Some Layers

- The last **10 layers of VGG16** are unfrozen for fine-tuning.
- The learning rate is reduced (1e-5) to **avoid catastrophic forgetting**.
- The model is retrained for another **5 epochs** to improve feature extraction.

Step 6: Save the Final Model

- The trained model is saved in **Google Drive** as final_cnn_model.keras for future use.

3. Conclusion

This approach leverages **transfer learning** to improve accuracy while minimizing training time. By combining **feature extraction (frozen layers)** and **fine-tuning (unfreezing layers gradually)**, the model learns meaningful patterns while adapting to the new dataset.