CoAtNet Model:

# 1. Model Overview

This implementation uses **CoAtNet**, a hybrid vision architecture that combines convolutional layers for **efficient local feature extraction** and transformer blocks for **global relational modeling**.

This hybrid approach is ideal for fine-grained classification problems such as fruit quality assessment, where **minor visual differences** (e.g., spots, color gradients, texture) matter and need to be captured at multiple scales.

## Key Specifications

**Input Shape**: 224×224 RGB images

**Output**: 7-class classification (banana/tomato ripeness stages)

**Architecture Depth**: 4 convolutional + 4 transformer blocks **Channel Progression**: 64 → 128 → 256

2. Model Architecture :

The architecture is composed of three major stages: **Convolutional Stem**, **MBConv Blocks**, and **Transformer Blocks**. Each is designed with specific strengths that complement each other.

## A. Convolutional Stem (Early Feature Extraction)

**Goal**: Extract low-level texture and color features such as edges, gradients, and small blemishes. **Why it works**:

Convolutions are **inductive-biased** toward spatial locality and translation invariance—perfect for detecting localized spots or textures in fruits.

Early downsampling helps reduce computational cost while increasing abstraction. **Structure**:

Conv2D (3×3, stride=2) → BatchNorm → Swish Output resolution reduces as:

224×224 → 112×112 → 56×56 → 28×28 Channels increase: 3 → 64 → 128 → 256

## MBConv Blocks (Efficient Mid-Level Representation)

**Goal**: Capture more abstract spatial-channel interactions efficiently.

**Why MBConv**:

Originally from MobileNetV2, MBConv blocks use **depthwise separable convolutions** to reduce FLOPs while keeping representational power.

Includes a **bottleneck expansion** to allow richer feature learning in a lightweight manner.

**Block Structure**:

Expansion (1x1 conv) → Depthwise conv (3x3) → Pointwise projection (1x1)

**Residual connection**: Stabilizes training and preserves input information.

**Swish activation**: Smooth and differentiable, leads to better gradient flow.

**Advantages**:

Preserves detail while increasing abstraction.

Efficiently captures feature interactions with fewer parameters.

## B. Transformer Stage (Global Reasoning)

**Goal**: Model long-range dependencies and subtle inter-region relationships, such as correlation between fruit color on one side and texture on another.

**Why Transformers**:

Unlike convolutions, self-attention **looks at the entire image** and captures **global context**.

Crucial for distinguishing subtle differences like:

Even ripening

Surface defect spread

Overall fruit shape distribution

**Architecture**:

**Tokenization**: 28×28×256 → 784 sequence tokens of 256-D each **Attention**:

Multi-head attention (4 heads, key\_dim=32)

Relative positional encoding for spatial awareness **MLP**: 256 → 1024 (Swish) → 256 **Regularization**:

LayerNorm, Dropout, Residual connections

**Advantages**

:

Enhances

**contextual reasoning**

Reduces false positives in similar-looking classes

Complements localized CNN features with global interactions

## Classification Head (Final Prediction)

**Layers**:

GlobalAveragePooling → LayerNorm → Dense → Softmax

**Purpose**:

Aggregates features from all positions

Produces probability distribution over the 7 quality categories3 Training Methodology

## Class Balancing

Fine-grained fruit datasets are often **imbalanced**. For example, overripe bananas may be overrepresented compared to rare green tomatoes.

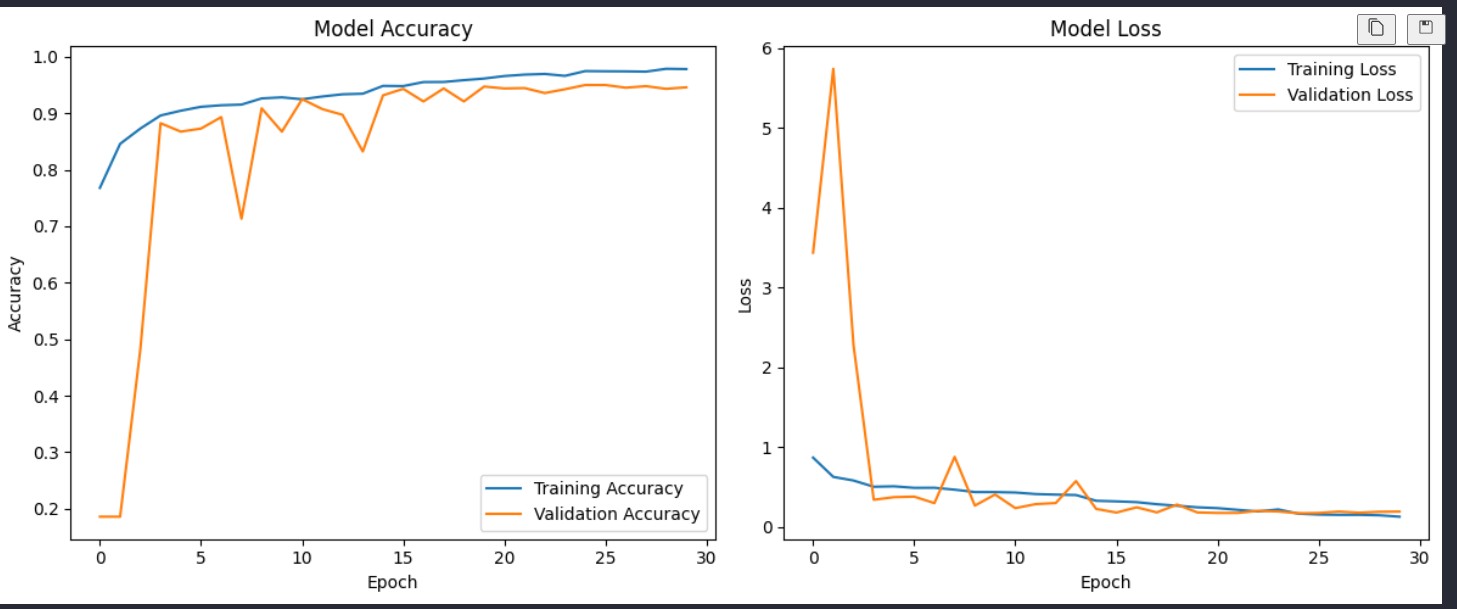
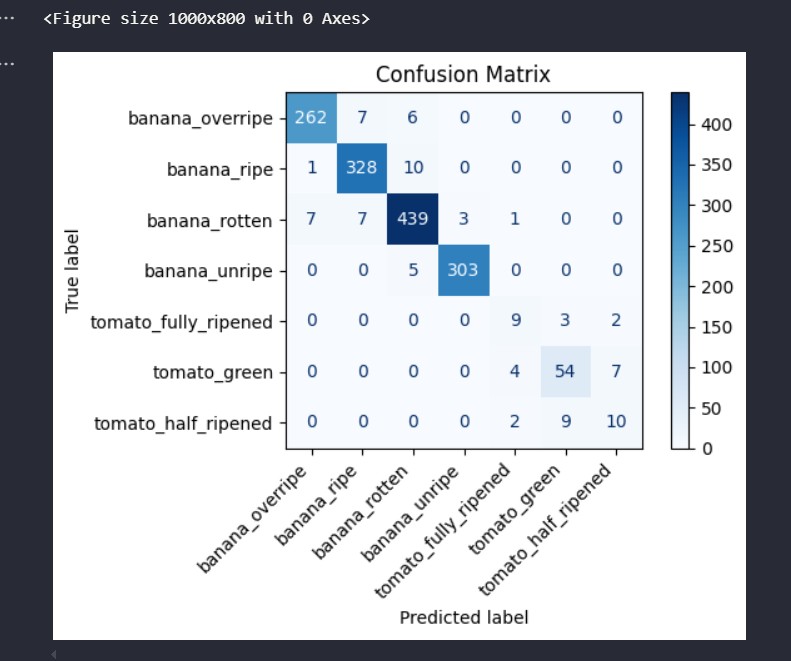
**Solution**:

Used class\_weight from sklearn.utils to dynamically compute loss weights based on label frequency.

Ensures rare classes contribute more to the loss during training.

# 4. Performance Results

## Overall Metrics:



## Class-Wise Performance

**Banana Categories**:

High precision and recall (F1 > 0.95)

Very accurate across ripeness stages **Tomato Categories**:

Harder to classify due to fewer samples

Best F1 = 0.82 on tomato\_green

## Confusion Matrix Insights

Strong diagonal dominance (high correct classification)

Logical misclassifications between **adjacent ripeness stages**

Effective class separation between banana and tomato types

# 5 Data Handling

## Class Weighting

Automatically adjusted class influence in the loss

## Augmentation Strategy

Only used **label-preserving** transformations:

Rotations, flips: spatial but not class-distorting

Color and zoom changes: preserve ripeness cues

## Batch Loading

Class-balanced batches through smart data generators

Maintained smooth and representative learning curve

# 6. Why This Architecture Works for Fine-Grained Fruit Classification

**Module Purpose Why It Helps**

|  |  |  |
| --- | --- | --- |
| **CNN Layers** | Local texture & edge detection | Captures blemishes, shape, spots |
| **MBConv Blocks** | Lightweight deep features | Efficient learning of mid-level semantics |
| **Transformer**  **Layers** | Global dependency modeling | Captures correlations across regions (e.g., gradient of ripeness) |
| **Class Weighting** | Imbalance handling | Prevents overfitting to common categories |

# 7. Conclusion

The **CoAtNet architecture** proves to be effective for fine-grained fruit quality assessment:

**95.45% validation accuracy**

Robust handling of **imbalanced classes**

Efficient training via **hybrid design**

Powerful combination of **local & global** understanding

By merging the strengths of **CNNs for fine textures** and **Transformers for global relationships**, this model is well-suited for task