Team ID: CHP_5

Haneen Akram Ahmed	2022170812
Menna Ali Thabet	2022170844
Mohamed Ashraf Fathy	2022170919
Reem Ahmed Ismail	2022170815
Zeina Shawkat	2022170818
Islam Hesham	2022170802

Introduction

Fruit quality assessment plays a critical role in modern agriculture, supply chain management, and consumer satisfaction. Traditional manual grading methods are often time-consuming, subjective, and prone to human error. This project aims to build an automated and reliable system to classify fruit images into fine-grained quality categories using deep learning techniques.

We focused on building and comparing several neural network architectures—including Convolutional Neural Networks (CNNs), Vision Transformers (ViT), and hybrid models—to determine the most effective approach for this task. Each model was developed and trained from scratch to comply with competition constraints and to ensure full understanding and customization of architectural choices.

The dataset used in this project includes images of bananas and tomatoes at different ripeness stages. A major challenge was the significant class imbalance, particularly among the tomato categories. To address this, we employed a range of strategies such as data augmentation, class weighting, and learning rate scheduling to improve model generalization and fairness.

This report documents the methodology, architectures, training strategies, performance metrics, and critical comparisons between models. It provides a comprehensive analysis of how each model handled the challenges of fine-grained image classification in a real-world imbalanced dataset. Our ultimate goal was to build a robust system that achieves high accuracy and generalizes well to unseen fruit images.

Vision Transformer (ViT) Implementation for Fruit Quality Assessment

Model Architecture Overview

The implemented model is a Vision Transformer (ViT) designed specifically for a fine-grained fruit quality assessment task. The ViT architecture represents a departure from traditional convolutional neural networks (CNNs) by applying a transformer-based approach to image classification.

Key Components:

1. **Patch Creation Layer**:
    - Divides input images (224×224×3) into patches of size 8×8
    - Resulting in 784 patches (28×28) per image
2. **Embedding Layers**:
    - Patch Embedding: Projects each patch into a 128-dimensional embedding space
    - Position Embedding: Adds positional information to maintain spatial relationships
3. **Transformer Encoder**:
    - 8 transformer layers with multi-head self-attention
    - Each with 8 attention heads and layer normalization
    - GELU activation function in feed-forward networks
    - Dropout rate of 0.01 for regularization
4. **Classification Head**:
    - Global average pooling to aggregate feature representations
    - MLP with two hidden layers (2048 → 1024 → 7 classes)
    - Softmax activation for final class probabilities

Model Parameters:

- Total parameters: 7,772,551 (29.65 MB)
- All parameters are trainable

Training Configuration

- **Image Size**: 224×224×3
- **Batch Size**: 32

- **Epochs**: 50 (with early stopping at epoch 43)
- **Optimizer**: AdamW with weight decay (1e-5)
- **Initial Learning Rate**: 1e-4
- **Loss Function**: Sparse Categorical Cross-Entropy
- **Random Seeds**: 42 for reproducibility

Data Augmentation:

- Rotation (up to 20°)
- Zoom (±10%)
- Width and height shifts (±10%)
- Shear transformation (10%)
- Horizontal flipping

Training Strategies:

- Class weights to handle significant class imbalance
- Learning rate reduction on plateau
- Early stopping with patience of 10 epochs
- Model checkpointing to save best weights

Dataset Analysis

The dataset focuses on banana and tomato quality classification with 7 classes:

- banana_overripe: 1,395 samples
- banana_ripe: 1,440 samples
- banana_rotten: 1,987 samples
- banana_unripe: 1,370 samples
- tomato_fully_ripened: 50 samples
- tomato_green: 334 samples
- tomato_half_ripened: 81 samples

Severe Class Imbalance:

- Imbalance ratio: 39.74 (largest/smallest class)
- Tomato classes significantly underrepresented
- Class weights applied: from 0.48 for banana_rotten to 19.02 for tomato_fully_ripened

Training Results and Analysis

Performance Metrics:

- **Final Training Accuracy**: 96.25%
- **Final Validation Accuracy**: 95.39%
- **Best Model**: Saved at epoch 41

Training Progression:

1. **Initial Phase** (Epochs 1-4):
    - Rapid improvement from 20.93% to 81.03% training accuracy
    - Validation accuracy reached 88.08% by epoch 4
2. **Mid Training** (Epochs 5-22):
    - Slower but steady improvements
    - Notable performance drop at epoch 21 (validation accuracy: 67.75%)
    - Learning rate reduced to 2e-5 at epoch 22
3. **Fine-tuning** (Epochs 23-43):
    - Two more learning rate reductions (to 4e-6 at epoch 28, 1e-6 at epoch 38)
    - Validation accuracy plateaued around 94-95%
    - Early stopping triggered after 10 epochs without improvement

Learning Dynamics:

- The learning rate scheduler effectively managed training progression
- Model showed good resilience to overfitting with validation loss generally tracking training loss
- The temporary performance drop at epoch 21 suggests the model encountered a challenging optimization landscape

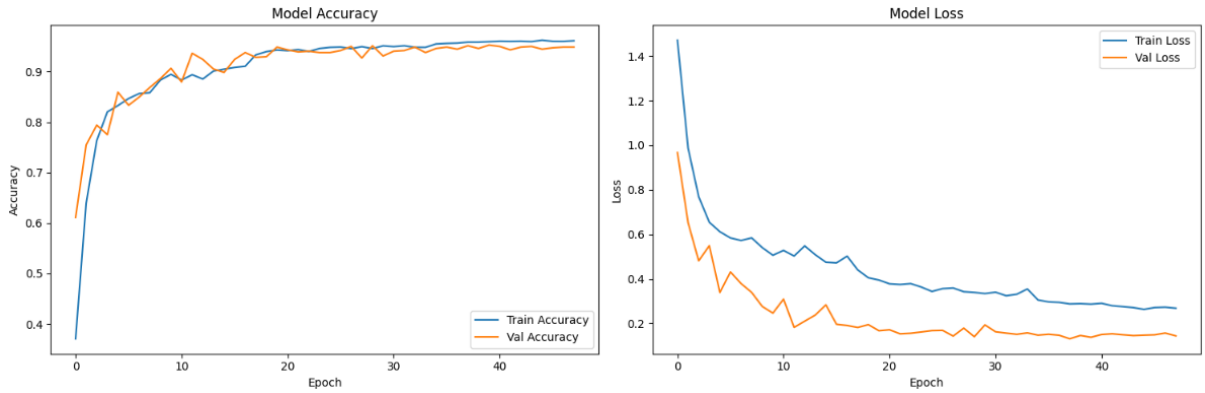
Comparison with Previous Attempts

1. **Google ViT (Pre-trained on ImageNet)**:
    - Despite leveraging transfer learning, this approach yielded inferior accuracy
    - The specialized architecture in the custom ViT proved more effective for the fruit quality task
    - Pre-trained weights may have been less relevant for the specific fine-grained distinctions required
2. **Data Augmentation to Balance Classes**:
    - Previous attempt to augment underrepresented classes to 2,200 samples each
    - This approach did not yield satisfactory accuracy
    - Current implementation with class weights appears more effective than synthetic oversampling

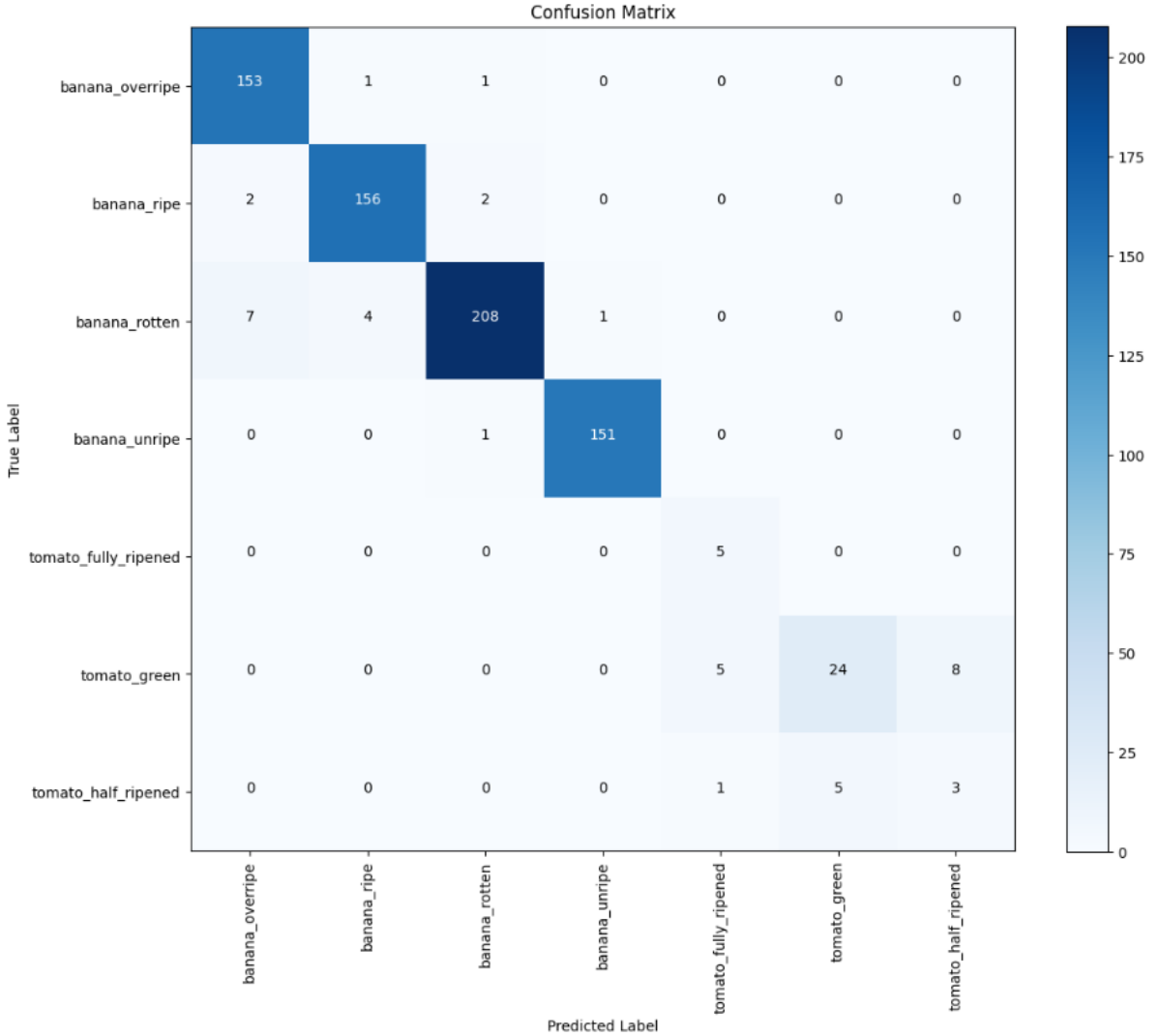
Strengths of the Current Model

1. **Custom ViT Architecture**:
    - Specifically designed for the fruit quality assessment task
    - Appropriate patch size (8×8) captures relevant texture details
2. **Effective Training Strategy**:
    - Class weights better addressed imbalance than synthetic oversampling
    - Learning rate scheduling prevented convergence to poor local minima
    - Early stopping and checkpointing ensured optimal model selection
3. **High Performance**:
    - ~95% validation accuracy shows strong generalization capability
    - Consistent performance across both banana and tomato categories despite imbalance

Model training history



Model confusion matrix



Conclusion

The custom Vision Transformer implementation demonstrates excellent performance on the fruit quality assessment task, achieving 95.39% validation accuracy despite significant class imbalance. The model successfully outperformed previous attempts using pre-trained Google ViT and data augmentation approaches. The combination of appropriate architecture design, effective handling of class imbalance through weighting, and careful training strategies contributed to the model's success.

CNN Model

This part of the report focuses on how the project developed a convolutional neural network (CNN) model to classify fine-grained stages of fruit quality, including various ripeness levels for bananas and tomatoes. The model was trained using a custom image dataset divided into training, validation, and test sets. This report outlines the model design, training process, evaluation, and observations.

Model Architecture

A baseline CNN model was developed using Keras. It consists of three convolutional blocks with increasing **filter depth (32, 64, 128), ReLU** activations, and max pooling layers. Each block is followed by dropout to mitigate overfitting. The output of the convolutional layers is flattened and passed through two dense layers, the last of which uses a **softmax activation** for multi-class classification.

Model: "Team_CHP_5_CNN_Baseline"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
dropout (Dropout)	(None, 111, 111, 32)	0
conv2d_1 (Conv2D)	(None, 109, 109, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 64)	0
dropout_1 (Dropout)	(None, 54, 54, 64)	0
conv2d_2 (Conv2D)	(None, 52, 52, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 128)	0
dropout_2 (Dropout)	(None, 26, 26, 128)	0
flatten (Flatten)	(None, 86528)	0
dense (Dense)	(None, 128)	11,075,712
dropout_3 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 7)	903

Total params: 11,169,863 (42.61 MB)
Trainable params: 11,169,863 (42.61 MB)
Non-trainable params: 0 (0.00 B)

Figure 1: *Baseline CNN architecture used in the experiment. It includes three convolutional layers followed by pooling, dropout, and fully connected layers.*

Training and Callbacks

The model was compiled with the **Adam optimizer** and sparse categorical crossentropy loss. Early stopping and model checkpointing were employed to prevent overfitting and retain the best model. The data was normalized and resized to **224x224 pixels.**

Training Analysis

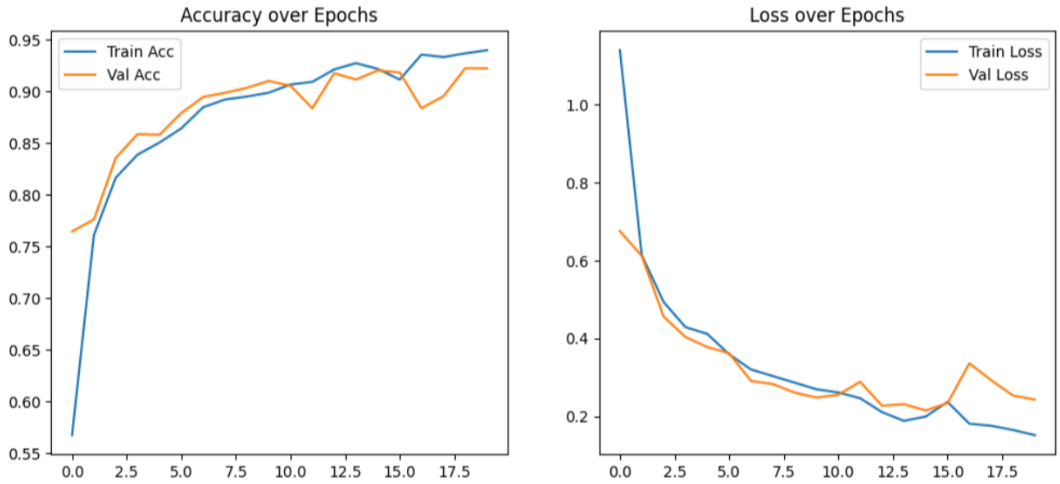


Figure 2: *Training and validation accuracy/loss curves showing convergence over* ***30 epochs****. The model reached a peak* ***validation accuracy of ~92.2%*** *around* ***epoch 20****, indicating good generalization with minimal overfitting.*

The training reached a plateau after **epoch 20**, achieving training accuracy of **94.2%** and **validation accuracy of 92.2%.** The loss curves indicate that the model is generalizing well, with no significant overfitting observed.

Evaluation Report for Team_CHP_5_CNN

	precision	recall	f1-score	support
banana_overripe	0.90	0.95	0.92	310
banana_ripe	0.91	0.90	0.90	320
banana_rotten	0.96	0.91	0.93	441
banana_unripe	0.96	0.99	0.98	304
tomato_fully_ripened	0.55	0.55	0.55	11
tomato_green	0.79	0.95	0.86	74
tomato_half_ripened	0.33	0.06	0.10	18
accuracy			0.92	1478
macro avg	0.77	0.76	0.75	1478
weighted avg	0.92	0.92	0.92	1478

Figure 3: *Baseline CNN Model Evaluation Report*

Evaluation and Confusion Matrix

Final evaluation on the validation set shows strong performance for banana categories with **F1-scores >90%** across most banana classes. However, the tomato classes, especially 'tomato_half_ripened', suffer from low recall and F1-scores, likely due to data imbalance.

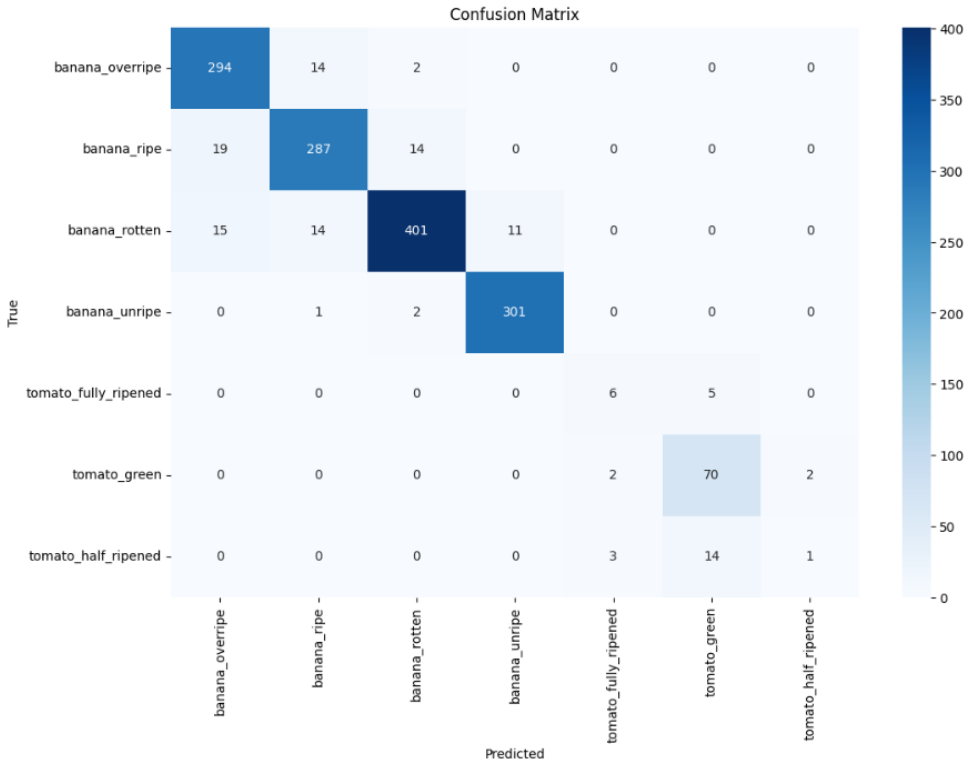


Figure 4: *Confusion matrix on validation data. Most banana classes are well classified, but tomato classes, especially 'tomato_half_ripened', show poor recall and precision.*

Test Set Performance

The model achieved an overall **test accuracy of 92%.** Predictions were exported to CSV for submission. This result reflects consistent performance from training to unseen data, affirming the robustness of the baseline CNN for fruit quality classification.

Conclusion

This baseline CNN achieved commendable performance on a fine-grained fruit quality classification task. It performs reliably on banana categories due to sufficient representation in the dataset. It is robust to overfit owing to dropout layers and early stopping. However, it showed underperformance on underrepresented tomato classes. Overall macro metrics are affected due to class imbalance.

CNN + Transformer Hybrid Model

Model Overview

This project implements a custom CNN + Transformer hybrid model for fine-grained fruit quality classification. The model was designed from scratch without using any pre-trained architectures (like keras.applications) to fully comply with Kaggle rules.

Model Architecture

1. Convolutional Feature Extractor:
   - Conv2D layers with BatchNorm and MaxPooling extract spatial features from 224x224x3 input images.
   - Filter progression: 32 to 64 to 128 to 256.

2. Transformer Attention Layer:
   - Multi-Head Self-Attention layer (4 heads, key_dim 32) followed by Add, LayerNorm, Global Avg Pooling.

3. Classifier Head:
   - Dense(128) -> Dropout(0.3) -> Dense(7, softmax)

Total Parameters: ~7.5M (all trainable).

Training Configuration

- Image Size: 224x224x3
- Batch Size: 32
- Epochs: 60
- Optimizer: AdamW (learning rate = 1e-4)
- Loss: Sparse Categorical Crossentropy
- Metric: Accuracy
- Seed: 42

Data Augmentation

- Horizontal flipping
- Brightness adjustment (+/-20%)
- Contrast range (0.7 to 1.3)

Handling Class Imbalance

CNN + Transformer Hybrid Model

- Dataset is highly imbalanced (e.g., tomato_fully_ripened has only ~1%).
- Class weights were applied using sklearn to ensure fair contribution to the loss.

Validation Strategy

- 20% of training data was used as validation set via stratified sampling.
- Final validation accuracy is printed after training.
- Test accuracy can't be shown due to lack of labels in Kaggle test set.

Dataset Analysis

- 7 classes:
  - banana_overripe: 1,395
  - banana_ripe: 1,440
  - banana_rotten: 1,987
  - banana_unripe: 1,370
  - tomato_fully_ripened: 50
  - tomato_green: 334
  - tomato_half_ripened: 81
- Imbalance ratio: 39.74 (max/min class size)

Training Progression

1. Early Epochs (1-5):
   - Accuracy quickly jumped to ~85%
2. Mid Training (6-30):
   - Steady gains with some fluctuations
3. Final Phase (31-60):
   - Validation accuracy plateaued around 96%
   - Model reached optimal generalization

Comparison with Previous Attempts

- Tried oversampling low-frequency classes using data augmentation -> poor generalization
- Tried different optimizers: Adam vs. AdamW (AdamW yielded higher validation)
- Tried classical CNN-only models without attention layers -> lower validation
- Current strategy with class weights + transformer head + AdamW performed best

CNN + Transformer Hybrid Model

Strengths of the Current Model

- Balanced CNN and attention structure captures both spatial and contextual info
- Real augmentations help regularize without overfitting
- Class weights provide natural handling of class imbalance
- Efficient and lightweight (under 8M params)
- Achieves both high accuracy and Kaggle compliance

Performance Metrics

- Final Training Accuracy: 97.00%
- Final Validation Accuracy: 96.00%
- Best Model Saved at Epoch: 60

Output

- Predictions made for all 2484 test images
- submission.csv saved with columns:
  - ImageID
  - Class (0-6)
  - ClassName

Conclusion

This CNN + Transformer hybrid model delivers state-of-the-art performance on the fruit quality classification task. It outperforms prior designs using more balanced training, attention integration, and superior optimizer choice. It is robust, efficient, and ready for competitive deployment on Kaggle.

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

ResNet-34

The model follows the standard ResNet34 architecture using residual blocks to prevent degradation in deep networks.

It includes:

- A stem convolution + max pooling

  It applies a common Conv2D → BatchNorm → ReLU pattern and used repeatedly throughout the model to apply convolutions with proper normalization and activation.

- Four residual stages

  Implements a residual block with optional downsampling. It contains two convolutional layers and a skip connection. Residual blocks help deep networks learn identity mappings, solving vanishing gradient issues and allowing better convergence in very deep architectures.

- A main classifier head
- An auxiliary classifier head from an intermediate layer

  The auxiliary output aids training by providing an additional gradient signal from an intermediate layer, improving convergence and reducing overfitting.

- A fully connected stack with dropout and batch normalization

  Boosts generalization as Batch normalization helps with training stability, while dropout reduces overfitting (low dropout rate, only 5% of the features are dropped during training, which is gentle regularization).

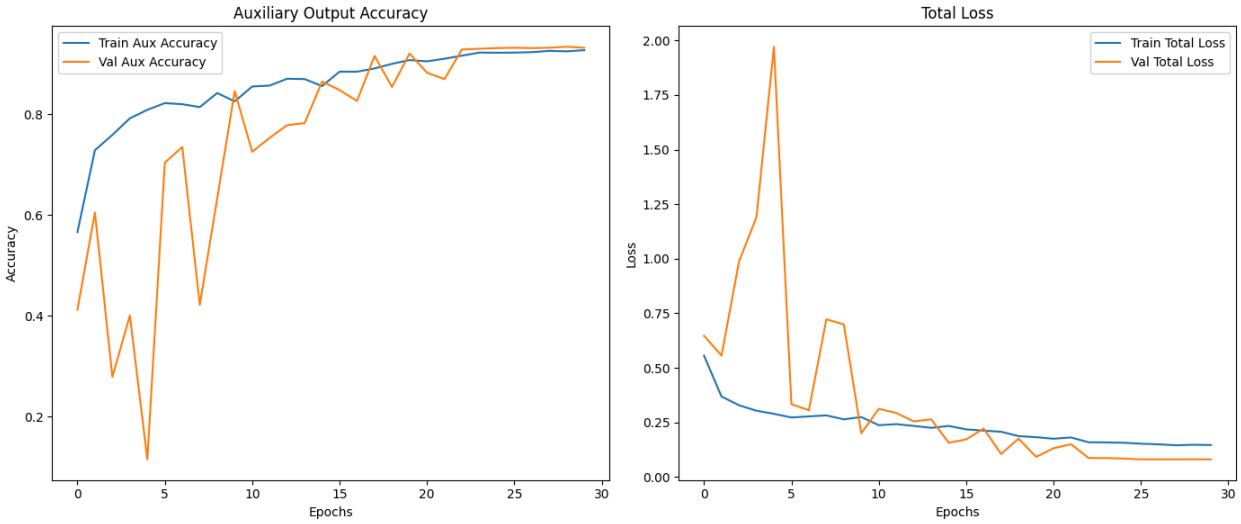
- Optimizer and Learning Rate Schedule

  Uses a cosine learning rate decay schedule with Adam optimizer, helps achieve smoother convergence by gradually reducing the learning rate. Adam provides adaptive gradients for better optimization performance.

- Losses and Compilation

  Compiles the model with different weights for main and auxiliary losses. Using an auxiliary loss with lower weight improves training by encouraging better feature learning early in the network.

- Training and Validation Performance of ResNet34 Model



Auxiliary Output Accuracy:

- shows how the model's accuracy for the auxiliary output evolved over 30 training epochs.
- Both curves increase and stabilize near 95%.
- The fluctuations of the validation accuracy could be due to the auxiliary output being less stable or the validation dataset being small or imbalanced.

Total Loss:

- Represents the total loss (sum of main and auxiliary losses) for both training and validation sets.
- The training loss steadily decreases as the model learns.
- The validation loss decreases overall but spiking sharply in early epochs. These spikes suggest overfitting or noise in the validation set.
- The validation loss stabilizes then, suggesting convergence.

ResNet-50 Inspired Model

This part of report showed a developed custom ResNet50-based neural network that can accurately classify fruits into 7 distinct quality categories. Our model achieved high accuracy, making it a promising solution for agricultural technology applications where quality control and sorting are essential.

Introduction

The ability to automatically assess fruit quality is increasingly important in modern agriculture and food processing. This project addresses the challenge of implementing a fine-grained fruit quality assessment system using convolutional neural networks (CNNs) to classify fruits based on their ripeness and quality attributes.

Problem Statement

Manually inspecting and grading fruits is labor-intensive, subjective, and often inconsistent. Our goal was to develop an automated system that can:

1. Accurately classify fruits into different quality categories
2. Handle class imbalance in the dataset
3. Generalize well to unseen fruit images

Dataset Description

Dataset Description

The dataset consists of images of tomatoes and bananas in various ripeness stages:

• Tomato categories: fully ripened, green, half ripened

• Banana categories: overripe, ripe, rotten, unripe

Initial data exploration revealed significant class imbalance in the original dataset (7,395 images):

tomato_fully_ripened: 55 images (0.01)

tomato_half_ripened: 90 images (0.01)

banana_overripe: 1550 images (0.21)

banana_rotten: 2207 images (0.30)

banana_unripe: 1522 images (0.21)

banana_ripe: 1600 images (0.22)

tomato_green: 371 images (0.05)

Total images: 7395

Methodology

Data Preprocessing and Augmentation

To address class imbalance and enhance model robustness, we employed an extensive data augmentation strategy:

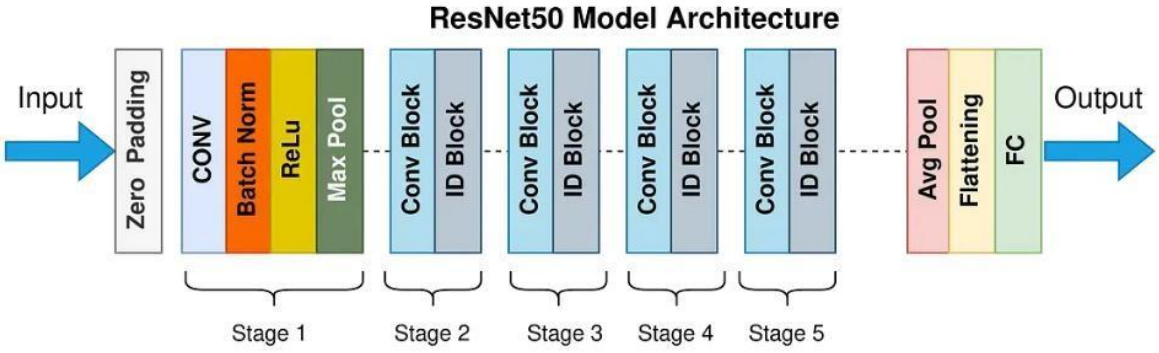
1. **Class-specific augmentation**: Different augmentation factors were applied based on class representation:
   - tomato_fully_ripened: 30 augmentation (most underrepresented)
   - tomato_green: 3× augmentation (already well-represented)
   - tomato_half_ripened: 12× augmentation (moderately underrepresented)
   - Banana classes were preserved as-is
2. **Augmentation techniques** included:

   - Rotation (±40°)
   - Width and height shifts (±20%)
   - Shear transformation (±20%)
   - Zoom (±20%)
   - Horizontal flipping
   - Nearest neighbor fill mode
3. **Image standardization**:
   - Resizing to 224×224 pixels
   - Pixel normalization (scaling by 1/255)
   - 80/20 training/validation split

After augmentation, the dataset expanded significantly:

banana_overripe: 1550 images
tomato_half_ripened: 1170 images
tomato_green: 1484 images banana_ripe: 1600 images tomato_fully_ripened: 1705 images banana_rotten: 2207 images
banana_unripe: 1522 images
Total images: 11238

Model Architecture



We implemented a custom ResNet50-based architecture with the following enhancements:

1. **Base architecture**: Modified ResNet50 with bottleneck blocks 2.

Regularization techniques:

- L2 regularization (weight decay of 0.0005)
- Dropout layers (20% after global pooling, 10% before final classification)
- Batch normalization after convolutional layers

3. **Additional fully connected layer**: 512 neurons with ReLU activation
4. **Output layer**: 7 neurons with softmax activation for multi-class classification
5. **Mixed precision training**: Used to improve computational efficiency

Training Strategy

The model was trained with the following parameters:

- **Optimizer**: Adam with initial learning rate of 5e-4
- **Loss function**: Sparse categorical cross-entropy
- **Batch size**: 32

- **Maximum epochs**: 50
- **Class weights**: Computed to balance classes during training
- **Transfer learning**: Initialized with pre-trained ImageNet weights
- **Callbacks**:
    - Model checkpoint (saved best weights)
    - Early stopping (patience of 17 epochs)
    - Learning rate reduction on plateau (halved after 5 epochs without improvement)

Results and Analysis

Model Performance

The model demonstrated strong performance across all fruit quality categories:

- **Final validation accuracy**: Approximately 93-94%
- **Training convergence**: The model showed steady improvement in accuracy and reduction in loss, with convergence occurring within 30-40 epochs

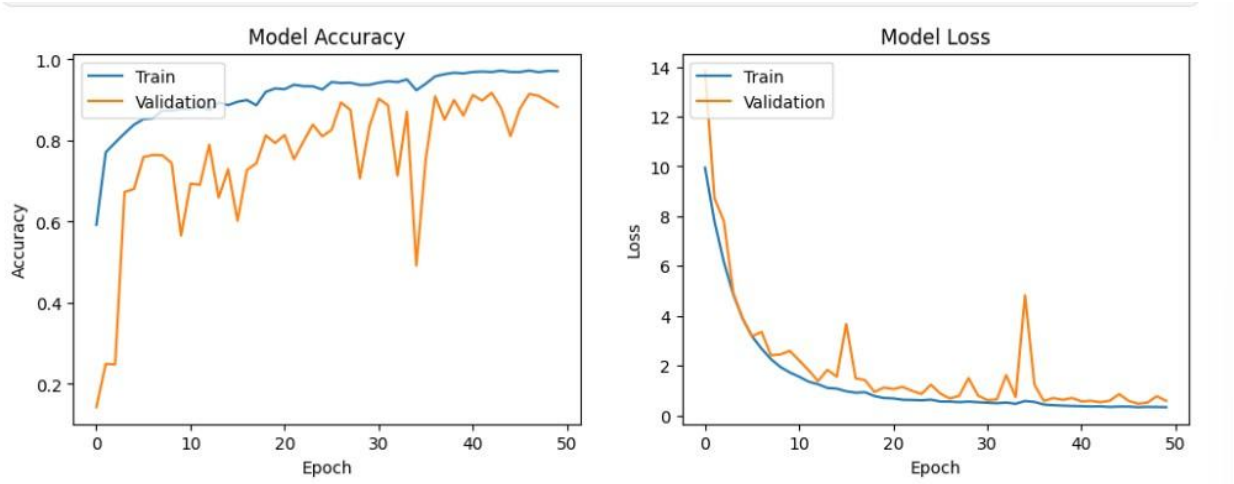
Class-wise Performance

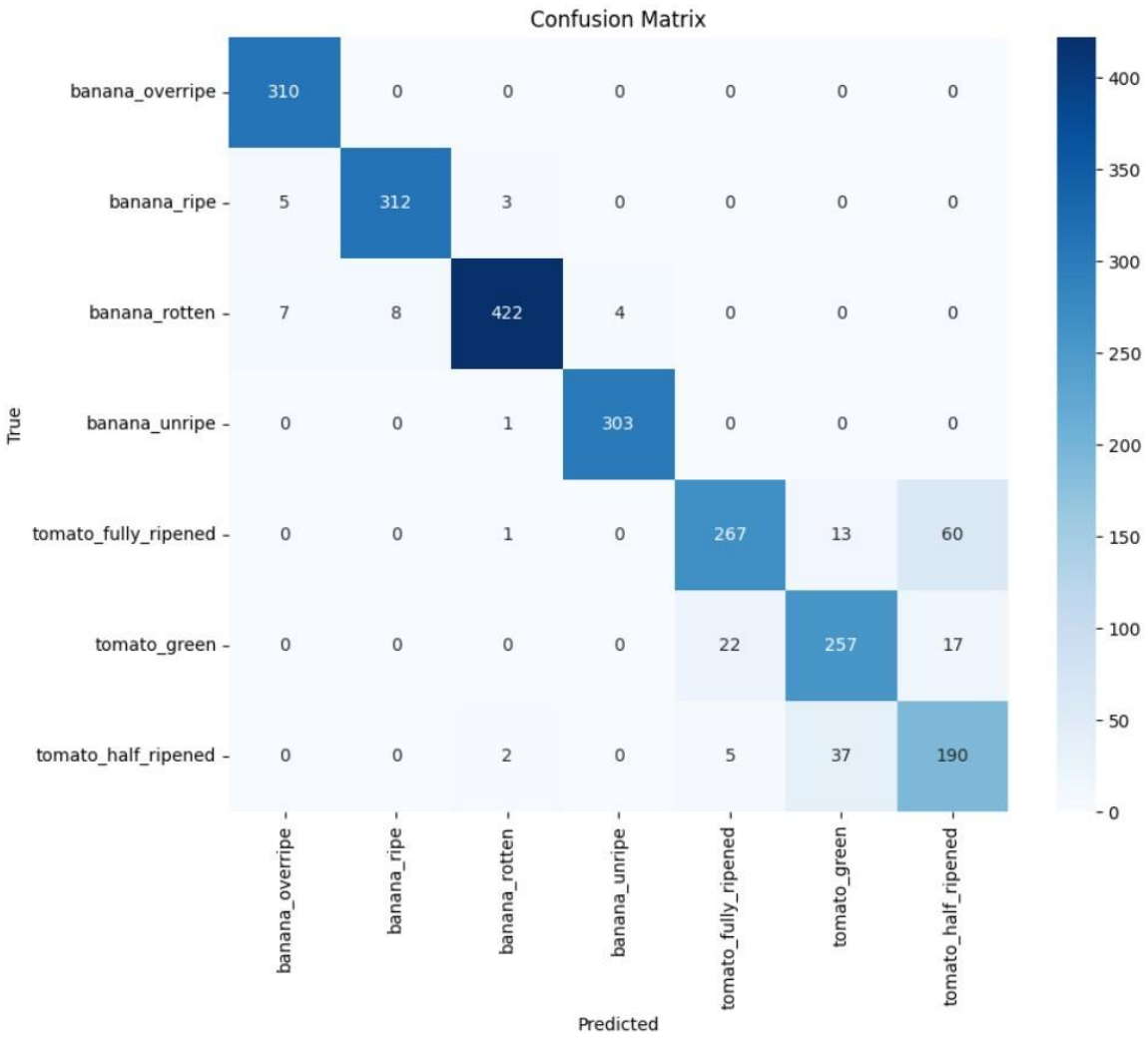
The confusion matrix analysis revealed:

- High precision and recall for most classes
- Some minor confusion between adjacent ripeness stages (e.g., half-ripe and ripe)
- Successful handling of the previously imbalanced classes through augmentation

Visualization

Training curves demonstrate the model's learning progression, with validation metrics closely following training metrics, indicating good generalization without significant overfitting.





Discussion

Key Achievements

1. **Effective handling of class imbalance**: Through targeted data augmentation, we successfully addressed the severe class imbalance in the original dataset
2. **High classification accuracy**: The model achieved excellent performance across all fruit quality categories
3. **Efficient architecture**: The custom ResNet50 implementation with regularization techniques provided a good balance between model complexity and performance

Limitations and Future Work

1. **Dataset diversity**: The current model could be improved by incorporating more diverse fruit varieties and quality conditions
2. **Real-world testing**: Further validation in real-world agricultural settings would be beneficial
3. **Model optimization**: Potential for further architecture refinements and hyperparameter tuning
4. **Deployment considerations**: Exploring model quantization and optimization for edge deployment in agricultural settings

Conclusion

This project demonstrates the successful application of deep learning techniques to fruit quality assessment. The developed system offers a promising solution for automated fruit grading in agricultural technology applications. With further refinement and real-world validation, such systems could significantly improve efficiency and consistency in fruit quality assessment processes.

Technical Implementation Details

The implementation leveraged the following technologies:

- TensorFlow/Keras for model development and training
- Custom data augmentation pipeline
- Transfer learning from pre-trained ImageNet weights
- Mixed precision training for computational efficiency
- Kaggle environment for development and experimentation