Sweet Spot: A Web Application to Classify Levels of Banana Ripeness using Convolutional Neural Networks

Dave Shanna Marie Gigawin
dsmegigawin01325@usep.edu.ph

Waken Cean C. Maclang
wccmaclang00570@usep.edu.ph

Allan C. Tagle
actagle00562@usep.edu.ph

I. Introduction

a. Background of the Study

Bananas are among the most important fruit crops worldwide, serving as a staple food and a major source of income in many tropical and subtropical regions. Its commercial value and consumer acceptance are strongly influenced by its ripeness stage, which determines sweetness, texture, shelf life, and overall quality [1]. During ripening, bananas undergo complex biochemical changes, including chlorophyll degradation, sugar accumulation, and softening of the pulp, which are traditionally monitored through visual inspection of peel color and texture [2].

On the other hand, manual assessment of ripeness is subjective, inefficient, and inconsistent, especially when applied to large-scale production and distribution. Such limitations often result in post-harvest losses and reduce the efficiency of supply chain management [3]. With food waste being a global concern, the need for reliable, objective, and automated ripeness classification systems has become increasingly important.

Advances in computer vision and machine learning provide solutions for this problem. Studies show that image-based approaches using deep learning models, particularly convolutional neural networks (CNNs), can achieve high accuracy in classifying ripeness stages [4].

b. Statement of the Problem

The main objective is to develop an application for classifying banana ripeness using image-based classification methods. Specifically, the study aims to answer the following research questions:

1. How does the classification performance (accuracy, F1, per-class recall) of a transfer-learned ResNet-101 + classifier compare with a vanilla CNN trained from scratch on the same banana ripeness dataset?

2. How does the classification performance of a transfer-learned model(s) + classifier compare with a vanilla CNN trained from scratch on the same banana ripeness dataset?

   Comparison of models will be made through the following metrics:
   a. Classification Metrics (This is compared across the different models and their classes)
      i. Accuracy
      ii. Precision
      iii. Recall
      iv. F1 Score
   b. Training Metrics
      i. Number of Parameters
      ii. Time needed to train the model
      iii. Training Accuracy
      iv. Validation Accuracy

3. How well do both approaches generalize to images from new farms / new cameras (domain shift)? Can fine-tuning significantly close the domain gap?

4. Considering model size and inference latency, which approach is more suitable for on-device mobile inference and for server-side web inference?

c. Scope and Delimitations

This research study will have the following scopes and delimitations:

Scope:

1. Target Fruit - the model will classify the ripeness category of Bananas. These bananas may vary in size and angle as long as the banana skin can be seen in its image.
2. Ripeness Classifications - The models will only classify the following classes: Unripe, Ripe, Overripe, and Rotten.
3. Models To Be Used - We will utilize the following models for classification:
    a. Simple CNN (Designed by the researchers)
    b. Transfer Learning using ResNet101 and VGG 19
4. Programming Tools - Use Python, sci-kit learn, PyTorch/TensorFlow
5. Front-End Application - Use ___ for website application
6. Visualize and simulate the results in a website application while using actual bananas.
7. Input Image Details - Pre-process, annotate, and augment a banana image dataset categorized by different phases of ripeness.

Delimitation:

1. Limited to Bananas - This study will not create a model that can also detect the ripeness categories of fruits other than bananas.
2. Limited Ripeness Factors - Classification of ripeness is based only on the banana's external features (i.e., the skin or outer features of the banana), no internal features will be used.
3. Simulation Purposes Only - The model will be deployed into a mobile and web application alone, and will not be implemented into real-life business-like hardware.
4. Specific Input Format - The model will only accept images.
5. Limited Image Quality - The quality of banana images within the dataset is specific. It cannot generalize different lighting conditions or camera quality.

II. Methodology

a. Research Framework

Creating a banana ripeness detection model can greatly assist in the following SDGs:

- Zero hunger (SDG 2)
    - Detecting ripeness can help reduce post-harvest loss as farmers and distributors can get them to the market at the right stage, thus reducing waste
- Responsible Consumption (SDG 12)

        - Food waste is one of the biggest sustainability problems. Creating an early ripeness detection model can help prevent bananas from being unsold and wasted
    - Industry, Innovation, and Infrastructure (SDG 9)
        - The model can be integrated into different AI and IoT infrastructures, making farming more data-driven and modern
    - Decent Work and Economic Growth (SDG 8)
        - A ripeness model can introduce fairer pricing, a reduction in spoilage, and a more stable income

b. Data Understanding – Describe the dataset to be used (source, type, size, characteristics)

The datasets to be used to train the models will comprise of images, those of which are either taken from kaggle or from research papers. These will then be integrated into a singular dataset used for training, testing, and validating the model. The datasets used are as follows:

The first dataset comprises 13,478 images of bananas at various stages of ripeness (Unripe, Ripe, Overripe, and Rotten). The images in the dataset will be combined and re-split for better model training and evaluation.

- Dataset link: https://www.kaggle.com/datasets/shahriar26s/banana-ripeness-classification-dataset
- Number of bananas - they only contain 1 banana per image.
- Bananas of different ripeness - the dataset contains 4 classes: Unripe, Ripe, Overripe, and Rotten.
- Type of banana - *Musa Acuminata* bananas are used in this dataset
- Preprocessing:
    - Auto-Orient: Applied
    - Resize: Stretch to 416x416
    - Modify Classes: 2 remapped, 0 dropped
        - Fresh ripe and ripe are classified as ripe
        - Mold are classified as overripe
        - Fresh unripe and unripe are classified as unripe
- Augmentation:
    - Angle and position of bananas - Banana/s in these images are placed on a table with varying degrees of angles and positions from the camera.
    - Image Angle - 90° Rotate: Clockwise, Counter-Clockwise, Upside Down
    - Image Flip - Horizontal and Vertical

    - Image Crop - 0% Minimum Zoom, 20% Maximum Zoom
    - Image Rotation - Between -15° and +15°
    - Image Blur - Up to 1px
    - Image Lighting:
        - Hue: Between -10° and +10°
        - Saturation: Between -10% and +10%
        - Brightness: Between -10% and +10%
        - Exposure: Between -10% and +10%

The second dataset comprises 300 banana images of different ripening levels: unripe (green banana), yellowish green, mid-ripe, and overripe. The dataset has 104 green bananas, 57 yellowish bananas, 97 mid-ripen bananas, and 42 overripen bananas.

- Dataset Link: https://drive.google.com/drive/folders/1nRWBYAHNRqmL4R0SLrs6dbGQFSWGVY8V
- It was used in the studies [5, 6].
- Number of bananas - they only contain 1 banana per image.
- Bananas of different ripeness - the dataset contains 4 classes: Green (Unripe), Midripen, Overripen, and Yellowish-Green
- Type of banana - *Plantain* bananas are used in this dataset
- Image - They contain one top-down image of a banana
- Image size - 960 x 540 pixels
- Preprocessing - None
- Augmentation - None

c. Data Preparation – Explain how the data will be cleaned, preprocessed, or transformed for training.

The following data preparation methods will be used in the order listed below:

1. Before the integration of the different datasets, the second dataset may be augmented on the following levels:
    - Brightness
    - Noise - Blur
    - 90-degree rotations - Clockwise, Counter-Clockwise, and Upside Down
    - Image Flip - Horizontal and Vertical
2. The images will be resized to a dimension of 224 x 224 pixels
3. Data integration, where images from different datasets, as well as the training, testing, and validation datasets, will be combined but grouped by their ripeness

    - Classes that the model will not classify will be remodified
        - Midripen class will be placed under the ripe classification
        - Overripen class will be placed under the overripe classification
4. RGB images will be normalized to obtain good results and to speed up the computational calculations. Likewise to how [7] has implemented it to their CNN model.

b. Neural Network Architecture

The proposed Convolutional Neural Network (CNN) will include:

1. A VGGNet-like architecture, but with a smaller number of filters, starting at 16 and doubling.
2. Number of Convolutional Layers - 5
3. Convolutional Layers have a fixed kernel size of (3, 3)
4. Convolutional Layers will use a Rectified Linear Unit (ReLU) as their activation function
5. The number of filters in a convolutional layer is either 16, 32, or 64 (i.e., doubles after every pooling layer)
6. Batch Normalization will be used after every convolutional layer and before its activation
7. Pooling layers will use max pooling with a pool_size of (2,2) and a stride of 2
8. 4 output neurons in the fully connected (fc) network, each to predict one of the four classes: Unripe, Ripe, Overripe, and Rotten
9. In the final layer of our fully connected layer, a softmax function will be used to output the categorical probability distribution of the given data/image.

Additionally, the proposed CNN will have the following architecture:

Legend:

- Block 1 = Yellow
- Block 2 = Green
- Block 3 = Light Blue
- Fully Connected Layer = Deep Blue

Index	Layer (type)	Output Shape (None, X, Y, Filters)	Param #
0	conv2d_1 (Conv2D)	(None, 224, 224, 16)	448

1	batch_normalization_1 (Batch Normalization)	(None, 224, 224, 16)	64
2	re_lu_1 (ReLU)	(None, 224, 224, 16)	0
3	conv2d_2 (Conv2D)	(None, 224, 224, 16)	2,320
4	batch_normalization_2 (Batch Normalization)	(None, 224, 224, 16)	64
5	re_lu_2 (ReLU)	(None, 224, 224, 16)	0
6	max_pooling2d_1 (MaxPooling2D)	(None, 112, 112, 16)	0
7	dropout_1 (Dropout)	(None, 112, 112, 16)	0
8	conv2d_3 (Conv2D)	(None, 112, 112, 32)	4,640
9	batch_normalization_3 (Batch Normalization)	(None, 112, 112, 32)	128
10	re_lu_3 (ReLU)	(None, 112, 112, 32)	0
11	conv2d_4 (Conv2D)	(None, 112, 112, 32)	9,248
12	batch_normalization_4 (Batch Normalization)	(None, 112, 112, 32)	128
13	re_lu_4 (ReLU)	(None, 112, 112, 32)	0
14	max_pooling2d_2 (MaxPooling2D)	(None, 112, 112, 32)	0
15	dropout_2 (Dropout)	(None, 112, 112, 32)	0
16	conv2d_5 (Conv2D)	(None, 56, 56, 64)	18,496
17	batch_normalization_5 (Batch Normalization)	(None, 56, 56, 64)	256
18	re_lu_5 (ReLU)	(None, 56, 56, 64)	0
19	max_pooling2d_3 (MaxPooling2D)	(None, 56, 56, 64)	0
20	dropout_3 (Dropout)	(None, 56, 56, 64)	0
21	flatten_1 (Flatten)	(None, 50176)	0
22	dense_1 (Dense)	(None, 64)	3,211,328

23	dropout_4 (Dropout)	(None, 64)	0
24	dense_2 (Dense)	(None, 4)	260
	Total Params: 3,247,380 (12.39 MB) Trainable params: 3,247,060 (12.39 MB) Non-trainable params: 320 (1.25 KB)		

Note: The proposed CNN architecture may change if training time takes too long.

Whereas, the transfer learning models will have the following architecture:

Index	Layer (type)	Output Shape (None, X, Y, Filters)	Param #
0	base_model (Transfer Learning Model)	(Depends on the transfer learning model)	(Depends on the transfer learning model)
1	global_average_pooling2d_1 (GlobalAveragePooling2D)		0
2	dense_1 (Dense)		(Depends on the transfer learning model)
3	dropout_4 (Dropout)		0
4	dense_2 (Dense)	(None, 4)	(Depends on the transfer learning model)

Note: The architecture of the Fully Connected Layer is not final and may change during the development of the project

c. Training and Evaluation

The characteristics listed here will be used for training all of the models used in this study, which are both the proposed CNN and the transfer learning models

- Learning rate - 1e-3
- Optimizer type - Adam
- Batch size
    - For the proposed model: 32 or 64 (Depends on GPU capabilities)
    - For the transfer learning models: 16 or 32
- Number of epochs - 50
    - This may be reduced as training will also utilize early stopping
- Dropout rate
    - 0.25 for Blocks 1 and 2, and 0.4 for Block 3 under the proposed CNN model
    - 0.5 in the fully connected layer (Applied to all models)
- Weight initialization method - He Normal
- Number of images for training, testing, and validation
    - Though the number of images are not finalized, the train, val, and test split will be 70-20-10 respectively.

On the other hand, these are the evaluation metrics used to compare the different models:

- Classification Metrics:
    - Accuracy - The percentage of correct predictions from the actual classification values.
    - Precision - Ratio of true positives to all positive predictions of the model.
    - Recall - Ratio of true positives to actual positive instances.
    - F1-Score - Harmonic mean of both Precision and Recall.

- Training Metrics:
    - Training Time - How long it took to train the model(s) under the same dataset
    - Training Accuracy - How well the model performed on the data it was trained on
    - Training Loss - How far off the model's predictions are from the correct answers numerically (i.e., computed through a loss function)
    - Validation Accuracy - How well the model performs on unseen data (i.e., generalizability of the model)
    - Validation Loss - How confident the model is in predicting unseen data

d. Deployment

Possible Tech Stack:

Layer	Tool	Purpose
Front-end	HTML/CSS/JS or React Frontend	For uploading banana images and displaying classification
Backend (Python)	Django + Django REST Framework	Handles requests, stores data, and calls your model
Model	Tensorflow / Keras	Trained CNN or MobileNet model (.h5)

Here's the workflow:

- User uploads an image (banana photo) via the Django web form or REST API.
- Django backend loads the trained model as an API (e.g., banana_model.h5).
- The uploaded image is preprocessed and fed to the model for prediction.
- Django returns the ripeness label (e.g., "Unripe", "Ripe", "Overripe").

III. References

Cite and list all sources using the Association for Computing and Machinery (ACM) Citation Style and Reference Formats.

[1] Beatriz Rosana Cordenunsi-Lysenko, João Roberto Oliveira Nascimento, Victor Costa Castro-Alves, Eduardo Purgatto, João Paulo Fabi, and Fernanda Helena Gonçalves Peroni-Okyta. 2019. The starch is (Not) just another brick in the wall: the primary metabolism of sugars during banana ripening. Frontiers in Plant Science 10.

[2] Fernando Mendoza and J.M. Aguilera. 2004. Application of image analysis for classification of ripening bananas. Journal of Food Science 69, 9.

[3] Hongkun Tian, Tianhai Wang, Yadong Liu, Xi Qiao, and Yanzhou Li. 2019. Computer vision technology in agricultural automation —A review. Information Processing in Agriculture 7, 1: 1–19.

[4] Raymond Erz Saragih and Andi W. R. Emanuel. 2021. Banana Ripeness Classification Based on Deep Learning using Convolutional Neural Network. 2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT): 85–89.

[5] N. Saranya, K. Srinivasan, and S. K. Pravin Kumar. 2021. Banana ripeness stage identification: a deep learning approach. Journal of Ambient Intelligence and Humanized Computing 13, 8: 4033–4039.

[6] Fatma M. A. Mazen and Ahmed A. Nashat. 2019. Ripeness classification of bananas using an artificial neural network. Arabian Journal for Science and Engineering 44, 8: 6901–6910.

[7] Luis Chuquimarca, Boris Vintimilla, and Sergio Velastin. 2023. Banana Ripeness Level Classification Using a Simple CNN Model Trained with Real and Synthetic Datasets. Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications: 536–543.