EASR

Engineering and Applied Science Research

https://www.tci-thaijo.org/index.php/easr/index

Published by the Faculty of Engineering, Khon Kaen University, Thailand

Banana quality classification using lightweight CNN model with microservice integration system

Vasutorn Chaowalittawin[1], Woranidtha Krungseanmuang[1], Posathip Sathaporn[1], Fuka Morita[2], Tuanjai Archevapanich[3] and Boonchana Purahong*[4]

[1]Department of Robotics and Computational Intelligent Systems, School of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand
[2]Department of Mechanical and Intelligent Systems Engineering, Faculty of Engineering, The University of Electro-Communications, Tokyo, Japan
[3]Department of Electronics and Communication Engineering, Faculty of Engineering and Architecture, Rajamagala University of Technology Suvarnabhumi, Nonthaburi, Thailand
[4]Department of Computer Engineering, School of Engineering, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand

Received 13 January 2025
Revised 8 April 2025
Accepted 7 May 2025

Abstract

Banana sorting has been performed manually, which often leads to human error due to the high volume and diverse characteristics involved. This paper presents a banana quality classification system using ConsolutechMobileNetV2 (CST-MobileNetV2) to classify banana ripeness into four categories unripe, ripe, overripe, and rotten. A lightweight deep learning model is proposed and integrated with a uniquely designed microservice system to optimize performance while minimizing computational demands. A publicly available dataset containing 13,478 images was used, and the data split into 56% for training, 14% for validation, and 30% for testing. Image normalization and augmentation techniques were applied to enhance the model's robustness. The model's performance was evaluated using a confusion matrix, achieving 98% precision, recall, and F1-score. The proposed model was compared with other deep learning models to benchmark its performance and deployed in different operating systems to evaluate its flexibility and capabilities. The LINE platform was employed as the user interface, enabling practical interaction with users. The system also demonstrated an average response time of 9.25 seconds per image, ensuring efficient processing, delivers high accuracy and scalability making it a practical and efficient solution for automated banana quality classification.

Keywords: Banana, CNN, CST-MobileNetV2, Lightweight deep learning model, Microservice architecture

1. Introduction

Bananas are widely recognized for their nutritional benefits, which include high-energy carbohydrates, digestive fiber, and a low-calorie content. They can be cultivated in a variety of soil types, such as flat, loamy, and sandy loam soils. Bananas thrive particularly well in tropical climates, especially in Asia [1] serving as the primary hub for global production and export. A critical component of banana cultivation [2], from plant nurturing to harvest, is the quality sorting process, which is currently performed manually. This manual approach is lead to human errors and significant effort. In line with modern agricultural practices, extensive research has focused on integrating technology to improve this process. Shuprajhaa et al. [3] introduced a deep learning-based non-destructive classification method for banana fruits, categorizing them into four groups, achieving an accuracy of 91.25%. Nikhilesh et al. [4] developed a deep learning model using the EfficientNet-B7 architecture, which enhanced image augmentation and regularization techniques. Saragih and Emanuel [5] compared deep learning models for classifying banana ripeness, finding that MobileNetV2 outperformed NASNetMobile in both accuracy and speed. Zheng et al. [6] created an efficient, low-complexity detection network called Slim-Banana, building on improvements to YOLOv8. Kakati and Das [7] proposed a hybrid classification method using a self-constructed CNN model (SCCNN) combined with SVM. Rangkuti et al. [8] compared several CNN models across nine classes of banana images, totaling 7,936 images, with the EfficientNet model achieving the highest accuracy at 89%, followed by the VGG16 model at 83.8%. Upadhyay et al. [9] utilized a CNN to classify bananas as raw or ripe, achieving a remarkable accuracy of 98.34%. Arunima et al. [10] developed a CNN model using a dataset of 4,320 images, achieving 95% accuracy. Sangeetha et al. [11] advanced a fruit classification model, marking a significant milestone in food quality assessment through the use of convolutional neural networks. Christian et al. [12] presented MobileNet as a resource for deep learning by comparing four optimizers: Gradient Descent, Adagrad, RMSProp, and Adam. Chompookham and Surinta [13] proposed ensemble methods using deep convolutional neural networks (CNNs) for plant leaf recognition. They compared five CNN models to select the best base model. The results showed that the 3-Ensemble CNNs (3-EnsCNNs) performed better on plant leaf disease datasets, while the 5-Ensemble CNNs (5-EnsCNNs)

*Corresponding author.
Email address: boonchana.pu@kmitl.ac.th
doi: 10.14456/easr.2025.38

outperformed others on the mulberry leaf dataset. Puangsuwan and Surinta [14] enhanced plant leaf disease classification using a snapshot ensemble convolutional neural network. This study proposed a deep learning approach to address real-world challenges present in the PlantDoc dataset. The experimental results showed that DenseNet201, when trained with the snapshot ensemble method (4-cycle), achieved an accuracy of 69.51%. Arampongsanuwat and Chaowalit [15] implemented deep convolutional neural networks for mangosteen ripeness classification. The experimental results showed that ResNet50 achieved the highest validation accuracy of 79%. Gatchalee et al. [16] conducted a Thai text classification experiment using CNN and transformer models for timely and timeless content marketing. The research demonstrated good results with a small dataset consisting of 600 articles, each containing at least 250 words. Recent trends indicate a growing integration of computer vision and deep learning to optimize agricultural productivity [17, 18]. Many researchers have applied these technologies to address various business and manufacturing needs [19, 20], extending to other applications [21]. Yang et al. [22] developed an automated, image-based fire detection and alarm system utilizing edge computing and a cloud platform, effectively minimizing latency for enhanced accuracy. Sithiyopasakul et al. [23] presented an inventory management system based on IoT and microservices architecture, facilitating synchronization between IoT devices and web applications. Roh et al. [24] introduced the AI and IoT-enabled MRIoT/AI Convergence Platform, applicable across fields such as disaster response and military surveillance. Mehmood et al. [25] proposed a multilevel fusion method for fruit disease identification and classification, incorporating intensive pre-processing and customized image kernels for feature extraction using state-of-the-art deep learning methods. Lee and Shin [26] utilized the Faster R-CNN for object detection, implementing it in automatic detection and monitoring systems for unexpected events in tunnels. Liu et al. [27] proposed an improved Faster R-CNN model for vehicle detection and human action recognition at night using infrared thermal imaging and transfer learning. The performance evaluation showed that the proposed method achieved a mean average precision (mAP) of 0.97. Kanjanasurat et al. [28] presented a personal identification method using Delaunay triangles and optic disc retinal vascular patterns. The vascular extraction algorithm, applied to the DRIVE database, achieved an average accuracy of approximately 94.1%. Anisuzzaman et al. [29] developed an automated wound localization system using a deep neural network, integrating a comprehensive wound diagnostic mobile application. Estonilo and Festijo [30] created a deep learning-based mobile application for predicting diabetes mellitus using TensorFlow. Antony et al. [31] proposed the Dipper Throat Optimization Algorithm with Deep Learning for food crop classification, leveraging remote sensing imaging for agricultural resource management.

In order to enhance efficiency and reduce production costs, the Consolutech-MobileNetV2 (CST-MobileNetV2), a Lightweight model designed to apply during a banana cultivation workflow, especially for quality classification of sorting/grading process. This model aims to enhance accuracy and efficiency while reducing model size and computational cost, ensuring compatibility with mobile, cloud, and edge computing environments.

The methodology began with the development of CST-MobileNetV2 by customized head of MobileNetV2, the classification head was modified with a fully connected (Dense) layer of 128 neurons with ReLU activation, followed by a dropout layer with a rate of 0.2, and a final Dense layer with 4 neurons using Softmax activation for multi-class classification. Followed by the creation of a microservice system to enable seamless integration and deployment in modern applications. This system offers flexibility and ease of integrity with mobile, web applications or IoT devices, empowering farmers to leverage modern and edge computing technologies. By this approach, the solution not only enhances operational efficiency but also significantly improves the entire banana supply chain process, ensuring better quality management and reduced wastage.

2. Materials and methods

2.1 Data sets

The data preparation process began with the utilization of images from a public dataset [32], comprising a total of 13,478 images. These included 2,179 images of unripe bananas, 4,015 ripe bananas, 2,691 overripe bananas, and 4,593 rotten bananas, as shown in Figure 1. The dataset was divided into three subsets: 56% for training (8,114 images), 14% for validation (1,320 images), and 30% for testing (4,044 images), with each image having dimensions of 416 × 416 × 3. Subsequently, the images were augmented using the ImageDataGenerator with the following parameters: rotation, zoom, width shift range (0.5), height shift range (0.5), shear range (0.12), horizontal flipping, and a fill mode set to nearest. Finally, the augmented images were resized to 224 × 224 × 3 to prepare them for model training.

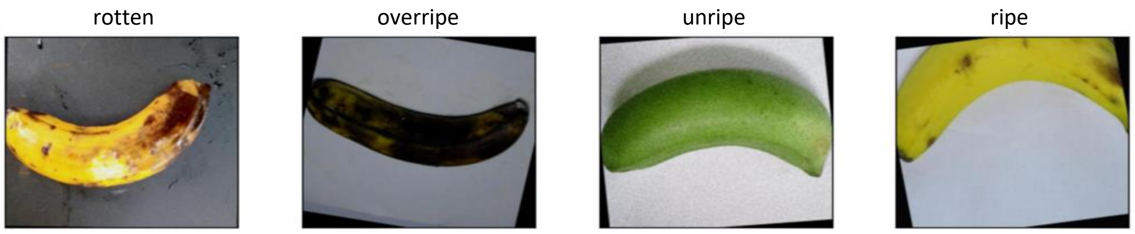

Figure 1 Type of banana

2.2 Normalization

Normalization in CST-MobileNetV2 is a crucial preprocessing step that involves adjusting pixel values from a scale of 0 to 255 to a range between -1 and 1. The transformation uses the formula (1):

$$X_{normalized} = \frac{X}{127.5} - 1 \qquad (1)$$

where X is the original pixel intensity. This approach is crucial for stabilizing the training process since centering the data around zero enhances learning efficiency. It ensures gradients behave consistently, making the model's training faster and more stable. By converting pixel values to the -1 to 1 range, CST-MobileNetV2 gains better numerical stability and avoids biases linked to features with larger value scales, ultimately improving the model's performance.

2.3 Consolutech-MobileNetV2 (CST-MobileNetV2)

The CST-MobileNetV2 was developed by customizing the MobileNetV2 [33] base model at the classification head. MobileNetV2 has a layer of convolution containing 32 filters, 19 residual congestion layers, and it is based on an inverted residual structure. However, CST-MobileNetV2, A Dense layer with 128 units and the ReLU activation function were added. It is calculated as (2). Then, a dropout layer with a 0.2 dropout rate was introduced to reduce overfitting. Subsequently, a Dense layer with 4 units was added in the output layer, utilizing the Softmax function to support multi-class classification, as illustrated in Figure 2. It is calculated as (3). With these customizations, the model delivered high performance and proved to be well-suited for integration with mobile applications.

$$\mathrm{ReLU(x)} = \max(0, \mathrm{x}) = \begin{cases} 0\ for\ x \le 0 \\ x\ for\ x > 0 \end{cases} \tag{2}$$

$$\sigma(z) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}\ for\ j = 1, \dots, K \tag{3}$$

Customizing the classification head of MobileNetV2 for a small custom dataset offers several advantages, particularly for mobile applications. By leveraging the pre-trained feature extractor of MobileNetV2, which is optimized for speed and efficiency, the model requires less computational power while maintaining high performance. This approach minimizes the risk of overfitting, especially with limited data, as only the final layers are retrained to adapt to the specific classification task. Furthermore, the Lightweight architecture of MobileNetV2 ensures the model remains suitable for deployment on resource-constrained devices, enabling integration into mobile applications while preserving accuracy and responsiveness.

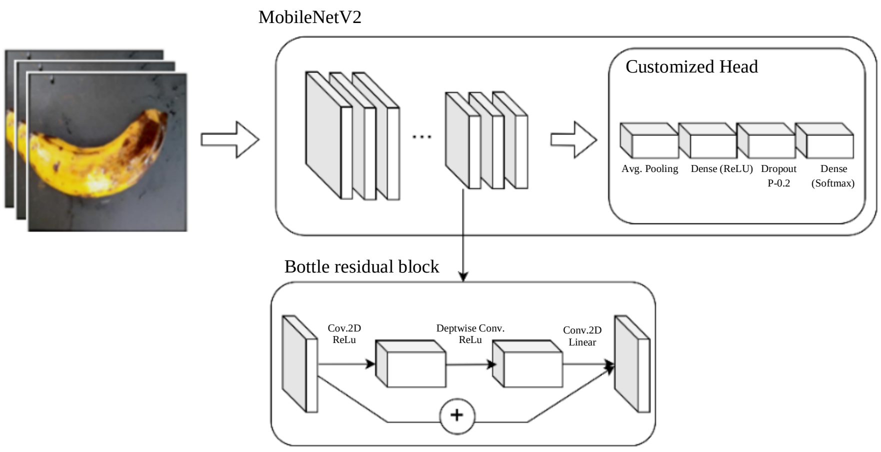

Figure 2 CST-MobileNetV2 Customization

2.4 Microservice architecture

We've designed a unique system that capable to integrate a light weight model with mobile applications which decomposes the functionality into independent container services. The infrastructure is divided into three components: the Application Service (Line Platform), the Web Server Service, and the Deep Learning AI Model Service. The core infrastructure is hosted on AWS cloud computing, with integration to the Line application to provide an intuitive user interface, as illustrated in Figure 3.

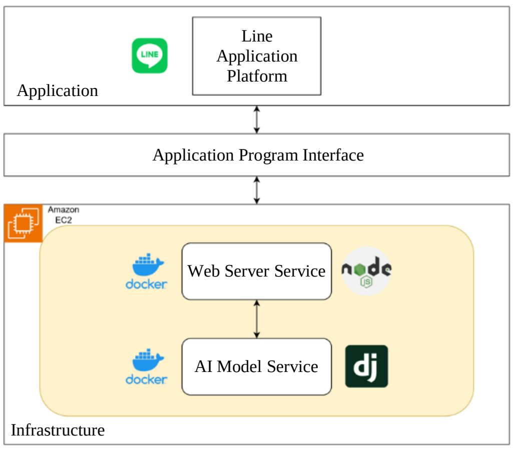

Figure 3 Proposed Microservice Architecture

2.4.1 Line platform

LINE Platform has been selected for the application layer due to its widespread use as a mobile messaging platform in Asia, particularly in Thailand and Japan. However, LINE provides the LINE Messaging API (Application Programming Interface), which enables uninteruptted integration with external applications. In this development, LINE is utilized to acquire images from mobile devices. These images are automatically distorted by the platform, reducing the original image quality by 86.34%, before being transferred to the CST-MobileNetV2 model via an API.

2.4.2 Web server service

A Docker container has been set up as the backend system on an Amazon EC2 server, with Node.js installed to serve as the web server for this solution. Node.js is chosen due to its open-source nature and its efficient interface for interacting with LINE APIs, such as sending and receiving messages and managing events like message exchanges. Acting as middleware, Node.js processes images received through the API via the line/bot-sdk library, ensuring that the images are properly formatted before being sent to the Deep Learning Model Service for further analysis and processing. This makes Node.js an ideal choice for building scalable, event-driven applications that require real-time data processing with minimal latency.

2.4.3 AI classification service

The CST-MobileNetV2 model will be deployed within a Docker container built on the Django framework, utilizing the Python programming language. First, a virtual environment will be set up to install all necessary dependencies, including package version control, the Python interpreter, and libraries such as TensorFlow and Keras. Next, the system will receive an image via URL from a web service middleware, after which the image will be preprocessed to a size of 224x224x3. Finally, the processed image will be passed through the pre-trained model, and the result will be returned to the user via the Line application interface. Figure 4 shows the system flow.

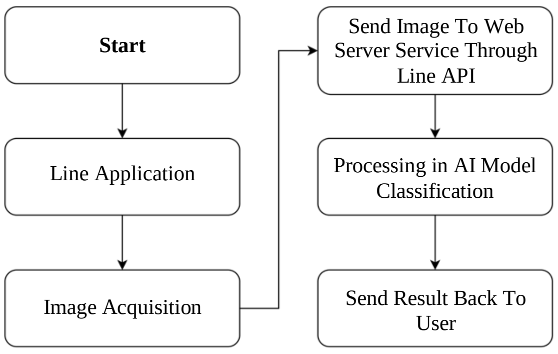
Start
Line Application
Image Acquisition
Send Image To Web Server Service Through Line API
Processing in AI Model Classification
Send Result Back To User

Figure 4 System flow

This ensures that the system operates swiftly and meets user demands in various scenarios. Furthermore, the design prioritizes flexibility for future development and enhancements, enabling the system to scale and accommodate new technologies seamlessly over time.

2.5 Experimental environment settings and model evaluation indicator

The proposed model was implemented using Python 3.10 on Ubuntu 22.04 LTS OS, Intel(R) Xeon(R) CPU@ 2.20GHz, RAM 53 GB, GPU RAM 22.5 GB. The performance of the model is measured by using confusion matrix, f1-score, accuracy, precision and recall.

A confusion matrix is a visual tool that helps assess the effectiveness of a classification model. It presents a summary of the actual versus predicted classifications, making it easier to identify how well the model is performing for each class. The matrix includes four key components as a below.

- True Positive (TP): The count of instances correctly identified as positive.
- True Negative (TN): The count of instances correctly identified as negative.
- False Positive (FP): The count of instances incorrectly identified as positive (also referred to as Type I error).
- False Negative (FN): The count of instances incorrectly identified as negative (known as Type II error).

Accuracy is a straightforward metric that reflects the proportion of correct predictions (both true positives and true negatives) relative to the total number of instances evaluated. It can be calculated using the formula as (4).

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \tag{4}$$

Precision, often called positive predictive value, measures how many of the instances predicted as positive are actually positive. It is calculated as (5).

$$Precision = \frac{TP}{(TP+FP)} \tag{5}$$

Recall, or sensitivity, evaluates how many actual positive instances were correctly predicted by the model. It is calculated as (6).

$$Recall = \frac{TP}{(TP+FN)} \tag{6}$$

The F1-score is a composite metric that combines precision and recall into a single value, reflecting the balance between the two. It is calculated as (7).

$$F1\ score = 2\frac{(Precision * Recall)}{(Precision + Recall)} \tag{7}$$

3. Results

In this section of the research, we present a comprehensive analysis of all outcomes, including model training results, encountered challenges, and implemented solutions. Additionally, we discuss advancements in system development and the integration of the model into the overall system framework.

3.1 Model training results

A comparison of the models is required to identify the most suitable option to serve as the base model. Table 1 shows the performance of different models after 5 epochs of training. The decision to limit training to 5 epochs per model was to ensure a fair and efficient comparison among the five models. Since the goal was to identify the most suitable model rather than fully optimize each one, a small number of epochs was sufficient to observe initial performance trends without excessive training time. Based on the results, considering validation accuracy and training time, MobileNetV2 performs the best overall. Therefore, MobileNetV2 has been chosen as the base model for this transfer learning task.

Table 1 Performance of different models after 5 epochs of training.

No.	Model	Train accuracy (%)	Validation accuracy (%)	Training Time (s)
1	ResNet152V2	**96.74**	96.55	166.06
2	Xception	95.56	95.23	115.45
3	InceptionV3	95.01	94.64	127.08
4	VGG19	92.55	91.57	170.94
5	EfficientNetB7	34.18	33.51	340.61
6	MobileNetV2	96.69	**96.71**	**112.61**

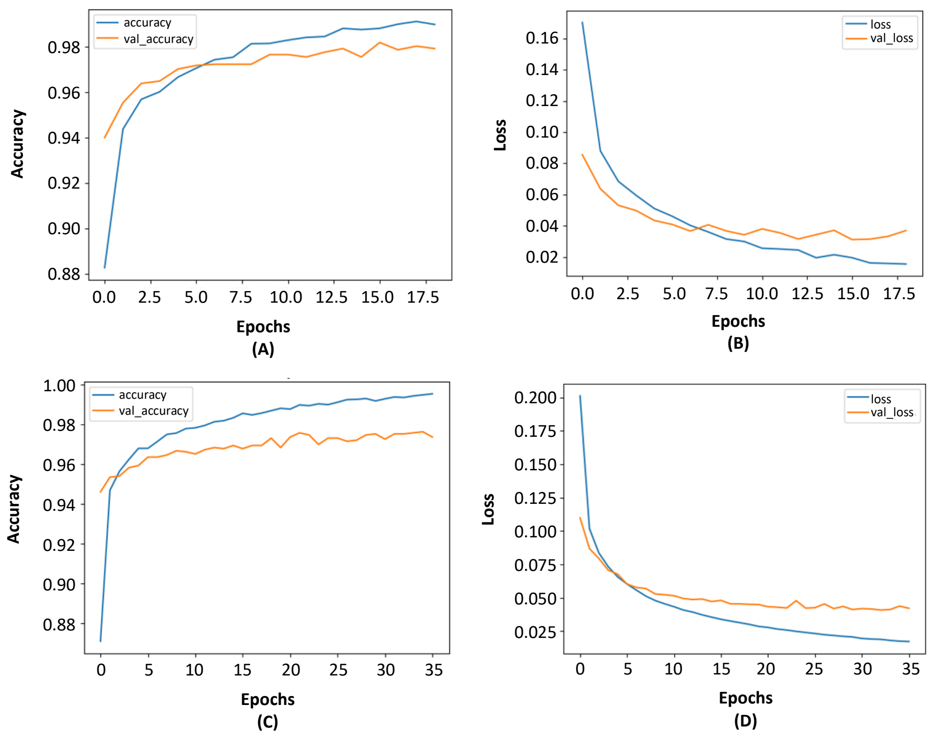

Figure 5 (A)Training accuracy and validation accuracy of CST-MobilNetV2, (B) Training loss and validation loss of CST-MobilNetV2, (C)Training accuracy and validation accuracy of MobilNetV2, (D) Training loss and validation loss of MobilNetV2

The graph in Figure 5 (a) shows the accuracy of a model over 18 training epochs. The blue line represents the training accuracy, while the orange line represents the validation accuracy. At the beginning, both lines start lower, with the training accuracy starting around 88% and validation accuracy a bit higher. Over the first few epochs, both lines increase sharply as the model learns and quickly improves. The training accuracy continues to climb steadily, reaching around 98% by the last epoch, while validation accuracy shows a similar trend but with slight fluctuations towards the end. The gap between the training and validation accuracy is relatively small, indicating that the model is generalizing well on the validation data. The leveling off in both curves towards the end suggests that the model has reached a stable level of accuracy, with only minor improvements after epoch 10.

The graph in Figure 5 (b) shows the loss of a model over 18 training epochs, with the blue line representing the training loss and the orange line representing the validation loss. At the start, both the training and validation loss are quite high, with the training loss beginning around 0.16. However, as the epochs progress, both lines show a steep downward trend, indicating that the model is learning effectively and minimizing error. By around the 5th epoch, the loss values have decreased significantly, with both lines nearing a stable range. From approximately the 10th epoch onward, the training loss continues to decline steadily, reaching a very low level near 0.02 by the end. The validation loss, while also low, shows minor fluctuations after epoch 10, hinting at slight variations in performance on unseen data. Overall, the close alignment of training and validation loss throughout suggests that the model is not overfitting and is likely generalizing well.

The upward trend and the halt at epoch 18 result from the implementation of a callback function known as early stopping, which focuses on the validation loss. This function is configured to terminate training immediately if the validation loss does not decrease within 3 epochs. Consequently, the use of early stopping helps maintain the model's accuracy and prevents overfitting.

Early stopping is a technique that prevents overfitting by stopping the training process when the model's performance on validation data stops improving. In this setup, training will monitor the validation loss, and if it increases for more than three consecutive epochs, training halts immediately. This approach saves resources and helps ensure the model stops at its best generalization point, avoiding the risk of fitting to noise in the data.

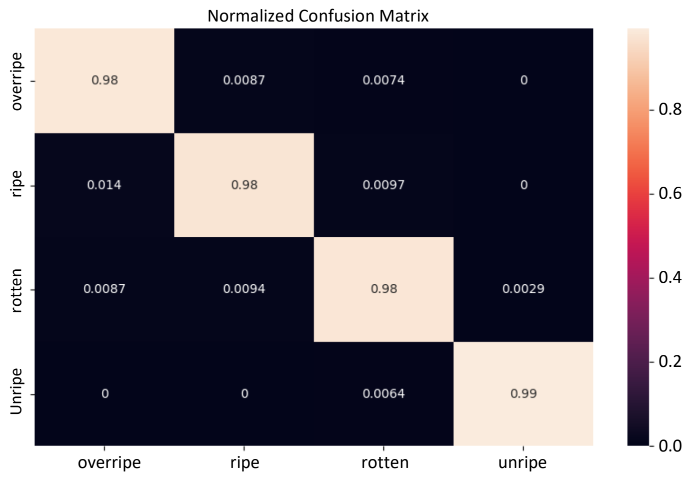

Figure 6 Normalized Confusion Matrix of CST-MobileNetV2

Figure 6 shows the result and can be described as below:

- Overripe: TP = 0.98. The FN results for overripe are 53% in ripe and 47% in rotten, but it does not predict unripe at all.
- Ripe: TP = 0.98. The FN results for ripe are 13% in ripe and 87% in rotten, but it does not fall into unripe at all.
- Rotten: TP = 0.98. The FN results are distributed across all classes, with 41% in overripe, 45% in ripe, and 14% in unripe.
- Unripe: TP = 0.99. All FN results fall into rotten, which is 100%.

From the confusion matrix result, it can be seen that the class most similar to other classes is rotten, while the class that is distinctly different from others is unripe.

Table 2 shows the performance of the proposed model in terms of precision, recall, F1-score and support. The "Unripe" class exhibits the highest values for precision, recall, and F1-score, all at 0.99. This exceptional performance is attributed to the distinct color characteristics of the Unripe class, which clearly differentiate it from the other classes, resulting in higher accuracy, sensitivity, and balance compared to the others. In contrast, both the "Ripe" and "Rotten" classes have equal precision, recall, and F1-scores of 0.98, indicating that predictions for these two classes are accurate and well-balanced. For the "Overripe" class, the precision is at 0.96, suggesting that there are more positive predictions than warranted, leading to a higher incidence of false positives compared to other classes. The recall for this class stands at 0.98, comparable to the other classes, while the F1-score is 0.97, indicating that this class has a lower balance between precision and recall than the others. Figure 7 shows examples of misclassified images. Table 3 presents a comparison of CST-MobileNetV2 with alternative deep learning methods.

Table 2 Performance Evaluation of CST- MobileNetV2

Class	Precision	Recall	F1-Score	Support
Overripe	0.96	0.98	0.97	809
Ripe	0.98	0.98	0.98	1232
Rotten	0.98	0.98	0.98	1378
Unripe	0.99	0.99	0.99	625
Accuracy	-	-	0.98	4044
Macro avg	0.98	0.98	0.98	4044
Weighted avg	0.98	0.98	0.98	4044

Table 3 Comparison of CST-MobileNetV2 with alternative deep learning methods.

Paper	Dataset	Classes	Method/Model	Classification Type	Accuracy (%)
Shuprajhaa et al. [3]	Banana Customized dataset	4	CNN-XgBoost	Quality	91.25
Nikhilesh et al. [4]	Nendran banana	4	EfficientNet-B7	Quality	95
Saragih et al. [5]	[34]	4	MobileNetV2	Quality	96.18
CST-MobileNetV2	Banana Classification [32]	4	Learning based on MobileNetV2	Quality	**98.15**

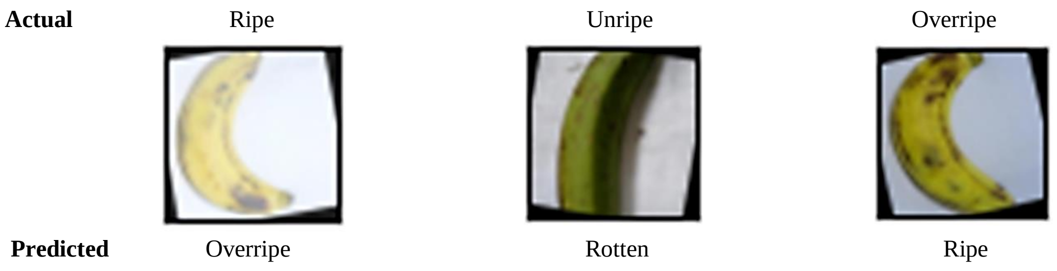

Figure 7 Examples of the misclassified images.

3.2 Application

We aim to demonstrate the integration of our proposed model with a custom-developed microservice infrastructure based on containerization. We deployed CST-MobileNetV2 into a containerized environment. This container was then deployed across three distinct target environments.

These environments were chosen as part of an initiative focused on modern application development. They represent potential directions for future implementation, including web applications leveraging cloud infrastructure, IoT applications utilizing edge computing, and mobile applications acting as backend infrastructure for real-world use cases.

The original image is first sent to the system. It is then processed and split into two versions: distorted and undistorted, before entering the prediction phase. In the undistorted case, the original 3000×4000 image is simply resized. In contrast, the distorted version undergoes a change in proportions before being resized to 224×224 pixels. This process degrades image quality and results in a loss of detail, making it more difficult for the model to accurately analyze the original features.

The system was tested using 270 images, with 265 correctly classified. The calculated mean accuracy was 0.9815, with a variance of 0.0182 and a standard deviation of 0.1349.

Table 4 Performance evaluation of the proposed model and microservice system in various environments.

Environment	Original Image (Pixels)	Process Image (Pixels)	Distortion (%)	Accuracy (%)	Response Time (s)
Cloud Computing	3000 × 4000	3000 × 4000	0	98.15	4.3
Edge Computing	3000 × 4000	3000 × 4000	0	98.15	3.75
Mobile Computing	3000 × 4000	3000 × 4000	0	98.15	0.3

Table 4 presents the results of an additional experiment in which the model was integrated with the proposed system and deployed in different environments (Cloud Computing, Edge Computing, and Mobile Computing). The objective was to evaluate the model's performance across alternative solutions. The results reveal that Mobile Computing is the most efficient environment for the proposed solution, delivering fast response times. Meanwhile, the other environments also maintained impressive processing speeds under 60 seconds.

The proposed deep learning model demonstrated high consistent accuracy across all environments and tools. Mobile Computing was chosen as the primary platform for development, as it proved to be the most effective and well-suited to the proposed solution. Accordingly, the Line application was adopted as the Mobile Computing platform and operational tool for user interactions.

Table 5 Performance evaluation of the proposed model and microservice system in Line Application

Platform/ Device	Original Image (Pixels)	Process Image (Pixels)	Distortion (%)	Accuracy (%)	Response Time (s)
Line application	3000 × 4000	960 × 1706	86.34	98.15	9.25

Table 5 presents the performance evaluation of the CST-MobileNetV2 model integrated with the Line application. The assessment was conducted using key HTTPS protocol metrics, including status codes, error rates, and response times, measured via the Postman tool.

The results show that the response time is 9.25 seconds, although the image received from LINE was automatically distorted to 86.34%, the model remains applicable and performs with a high accuracy rate of 98.15%. The response is subsequently sent to the client via the LINE Messaging API with a status code of 200, confirming that both the model and the application deployment are reliable and consistently available.

4. Discussion

This development is divided into two main components. The first focuses on model development for banana classification, while the second involves designing the system architecture for deployment. In the model development phase, six models were trained and evaluated to determine the most suitable base model for this dataset. Among these models, MobileNetV2 demonstrated the best performance. To enhance its capability, the head of MobileNetV2 was customized for transfer learning, resulting in CST-MobileNetV2, which achieved an accuracy of 98.15%. In the system architecture phase, the proposed model was integrated into a microservice-based system and deployed across three different computing environments. Mobile computing yielded the lowest response time at 0.30 seconds, compared to 3.75 seconds on edge computing and 4.30 seconds on cloud computing. Additionally, the model was deployed through the Line Application to assess its real-world applicability. A key challenge in this setup was image distortion caused by Line's compression process. Despite this issue, the model maintained high classification accuracy, demonstrating its robustness in practical applications.

5. Conclusion

In this development, we implemented a banana quality classification system using CST-MobileNetV2 to categorize banana ripeness into four stages: unripe, ripe, overripe, and rotten. The proposed Lightweight model achieved an accuracy of 98.15%, outperforming other evaluated models, as detailed in Table 3. The architecture is designed to operate as an individual service. The proposed solution was benchmarked against other deep learning models and tested on various operating systems, demonstrating consistent accuracy. It achieved the best performance on Mobile Computing with a response time of 0.30 seconds, compared to 3.75 seconds and 4.30 seconds on Edge Computing and Cloud Computing, respectively. Additionally, the model demonstrates an average response time of 9.25 seconds per image when integrated with the LINE application. This architecture is also adaptable and ready for integration with other mobile applications, web applications, or IoT systems, ensuring practical usability.

Future development will focus on adapting the lightweight model within a microservices framework for broader industrial applications, including medical, steel, and manufacturing sectors. This expansion enhances the system's versatility beyond agriculture.

6. References

[1] Mordor Intelligence. Banana market size - industry report on share, growth trends & forecasts analysis (2024 – 2029) [Internet]. 2024 [cited 2024 Oct 3]. Available from: https://www.mordorintelligence.com/industry-reports/banana-market.

[2] Fiallos-Cárdenas M, Pérez-Martínez S, Ramirez AD. Prospectives for the development of a circular bioeconomy around the banana value chain. Sustain Prod Consum. 2022;30:541-55.

[3] Shuprajhaa T, Mathav Raj J, Paramasivam SK, Sheeba KN, Uma S. Deep learning based intelligent identification system for ripening stages of banana. Postharvest Biol Technol. 2023;203:112410.

[4] Nikhilesh N, Rajesh SG, Praveen TS, Raghuram AS, Prerena N. Banana grading using deep learning model. Int Res J Mod Eng Technol Sci. 2023;5(5):5144-8.

[5] Saragih RE, Emanuel AWR. Banana ripeness classification based on deep learning using convolutional neural network. The 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT); 2021 Apr 9-11; Surabaya, Indonesia. USA: IEEE; 2021. p. 85-9.

[6] Zheng Z, Chen L, Wei L, Huang W, Du D, Qin G, et al. An efficient and lightweight banana detection and localization system based on deep CNNs for agricultural robots. Smart Agric Technol. 2024;9:100500.

[7] Kakati JB, Das TK. Classification of healthy and unhealthy banana leaves using deep learning approach: a comparative assessment. The 4th International Conference on Computing and Communication Systems (I3CS); 2023 Mar 16-18; Shillong, India. USA: IEEE; 2023. p. 1-5.

[8] Rangkuti AH, Lau SL, Hasbi VA, Indallah FH, Ranny, Aryanto R. Comparison of CNN models for optimizing banana image classification. 2023 IEEE International Conference on Computing (ICOCO); 2023 Oct 9-12; Langkawi, Malaysia. USA: IEEE; 2024. p. 456-61.

[9] Upadhyay A, Singh S, Kanojiya S. Segregation of ripe and raw bananas using convolutional neural network. Procedia Comput Sci. 2023;218:461-8.

[10] Arunima PL, Gopinath PP, Geetha Lekshmi PR, Esakkimuthu M. Digital assessment of post-harvest Nendran banana for faster grading: CNN-based ripeness classification model. Postharvest Biol Technol. 2024;214:112972.

[11] Sangeetha K, Vishnu Raja P, Siranjeevi S, Suman J, Rohith S. Classification of fruits and its quality prediction using deep learning. The 5th International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV); 2024 Mar 11-12; Tirunelveli, India. USA: IEEE; 2024. p. 342-6.

[12] Christian S, Murwantara IM, Lazarusli I. A mobile application for food and its ingredients detection using deep learning. The 1st International Conference on Technology Innovation and Its Applications (ICTIIA); 2022 Sep 23-23; Tangerang, Indonesia. USA: IEEE; 2022. p. 1-6.

[13] Chompookham T, Surinta O. Ensemble methods with deep convolutional neural networks for plant leaf recognition. ICIC Express Lett. 2021;15(6):553-65.

[14] Puangsuwan T, Surinta O. Enhancement of plant leaf disease classification based on snapshot ensemble convolutional neural network. ICIC Express Lett. 2021;15(6):669-80.

[15] Arampongsanuwat S, Chaowalit O. Application of deep convolutional neural networks for mangosteen ripeness classification. ICIC Express Lett. 2021;15(6):649-57.

[16] Gatchalee P, Waijanya S, Promrit N. Thai text classification experiment using CNN and transformer models for timely-timeless content marketing. ICIC Express Lett. 2023;17(1):91-101.

[17] Pandey DK, Mishra R. Towards sustainable agriculture: harnessing AI for global food security. Artif Intell Agric. 2024;12:72-84.

[18] Dhanya VG, Subeesh A, Kushwaha NL, Vishwakarma DK, Nagesh Kumar T, Ritika G, et al. Deep learning based computer vision approaches for smart agricultural applications. Artif Intell Agric. 2022;6:211-29.

[19] Bhatti MA, Syam MS, Chen H, Hu Y, Keung LW, Zeeshan Z, et al. Utilizing convolutional neural networks (CNN) and U-Net architecture for precise crop and weed segmentation in agricultural imagery: a deep learning approach. Big Data Res. 2024;36:100465.
[20] Yin Y, Zhao M. Application of AI, big data and cloud computing technology in smart factories. The 6th International Conference on Artificial Intelligence and Big Data (ICAIBD); 2023 May 26-29; Chengdu, China. USA: IEEE; 2023. p. 192-6.
[21] Ahmad S, Shakeel I, Mehfuz S, Ahmad J. Deep learning models for cloud, edge, fog, and IoT computing paradigms: survey, recent advances, and future directions. Comput Sci Rev. 2023;49:100568.
[22] Yang X, Li Y, Chen Q. Automated image-based fire detection and alarm system using edge computing and cloud-based platform. Internet of Things. 2024;28:101402.
[23] Sithiyopasakul P, Piyatananugoon C, Chaowalittawin V, Krungseanmuang W, Sathaporn P, Kanjanasurat I. Inventory management system based on IoT and microservices architecture design. 2023 International Electrical Engineering Congress (iEECON); 2023 Mar 8-10; Krabi, Thailand. USA: IEEE; 2023. p. 395-9.
[24] Roh BH, Choi G, Lee S, Kim SJ, Kang J. Mixed reality-enabled multilateral collaboration application platform with AI and IoT convergence. 2023 IEEE International Conference on Consumer Electronics (ICCE); 2023 Jan 6-8; Las Vegas, USA. USA: IEEE; 2023. p. 1-4.
[25] Mehmood A, Ahmad M, Ilyas QM. On precision agriculture: enhanced automated fruit disease identification and classification using a new ensemble classification method. Agriculture. 2023;13(2):500.
[26] Lee KB, Shin HS. An application of a deep learning algorithm for automatic detection of unexpected accidents under bad CCTV monitoring conditions in tunnels. 2019 International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML); 2019 Aug 26-28; Istanbul, Turkey. USA: IEEE; 2019. p. 7-11.
[27] Liu Y, Matsui K, Kageyama Y, Shirai H, Ishizawa C. Improving a faster R-CNN model for vehicle detection and human action recognition at night via infrared thermal imaging using transfer learning. Int J Innov Comput Inf Control. 2024;20(6):1573-85.
[28] Kanjanasurat I, Purahong B, Aoyama H, Benjangkaprasert C, Pintavirooj C. Personal identification using a delaunay triangle and optic disc retinal vascular pattern. Int J Innov Comput Inf Control. 2020;16(3):879-97.
[29] Anisuzzaman DM, Patel Y, Niezgoda JA, Gopalakrishnan S, Yu Z. A mobile app for wound localization using deep learning. IEEE Access. 2022;10:61398-409.
[30] Estonilo CG, Festijo ED. Development of deep learning-based mobile application for predicting diabetes mellitus. The 4th International Conference of Computer and Informatics Engineering (IC2IE); 2021 Sep 14-15; Depok, Indonesia. USA: IEEE; 2021. p. 13-8.
[31] Antony A, Ganesh Kumar R. Enhancing food crop classification in agriculture through dipper throat optimization and deep learning with remote sensing. e-Prime - Adv Electr Eng Electron Energy. 2024;9:100732.
[32] Thakar A. Banana Classification [Internet]. 2024 [cited 2024 Oct 3]. Available from: https://www.kaggle.com/datasets/atrithakar/banana-classification.
[33] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. Mobilenetv2: inverted residuals and linear bottlenecks. Proceedings of the IEEE conference on computer vision and pattern recognition; 2018 Jun 18-22; Salt Lake City, USA. USA: IEEE; 2018. p. 4510-20.
[34] Mazen FMA, Nashat AA. Ripeness classification of bananas using an artificial neural network. Arab J Sci Eng. 2019;44(8):6901-10.