Jurnal Teknik Informatika (JUTIF) Vol. 6, No. 2, April 2025, Page. 501-520
P-ISSN: 2723-3863 https://jutif.if.unsoed.ac.id
E-ISSN: 2723-3871 DOI: https://doi.org/10.52436/1.jutif.2025.6.2.4168

Development of a Convolutional Neural Network Method for Classifying Ripeness Levels of Servo Variety Tomatoes

Rosalina*[1], Ario Yudo Husodo[2], I Gede Pasek Suta Wijaya[3]

[1,2,3]Master of Information Technology, Mataram University, Indonesia

Email: [1]rosaaaaa174@mhs.unram.ac.id

Received : Dec 4, 2024; Revised : Jan 3, 2025; Accepted : Jan 4, 2025; Published : Apr 26, 2025

Abstract

The distribution of tomatoes in Indonesia is huge, making it an important commodity in the agricultural sector. However, manual classification of tomato ripeness can lead to human error and decrease supply chain efficiency. Therefore, an automated system capable of classifying tomatoes quickly and accurately is needed, in order to reduce the potential for human error and improve supply chain efficiency. This research aims to develop the Convolutional Neural Network (CNN) method to improve the accuracy of tomato ripeness detection through modifications to the architecture, such as reducing several layers, adding batch normalization, and adding dropouts. The dataset used in this study consists of 500 images taken by the researcher himself which are divided into 5 classes, namely unriped, half-riped, riped, half-rotten, and rotten, with each class containing 100 images. There are 3 proposed CNN models, namely the standard model, as well as the addition of batch normalization and dropout in the architecture. The results showed that the proposed model 3 with the addition of dropout on several layers of its architecture is the optimal model with a parameter of 2.4 million and using a batch size of 16 resulting in an accuracy of 98%, as well as precision, recall, and F1-score values of 98%. With these results, the proposed CNN model is effective in identifying the ripeness level of tomato fruit. This research is expected to be applied in the agricultural industry to improve the efficiency of sorting and distributing tomato fruits according to the desired quality standards.

Keywords : *Batch Normalization, Classification, CNN, Dropout, Tomato.*

1. INTRODUCTION

Tomato *(Solanum Lycopersicum)* is one of the agricultural commodities in Indonesia with production levels that tend to increase every year [1]. Based on data from the Badan Pusat Statistik Indonesia (BPS Indonesia), tomato production in Indonesia in 2021 reached 1,114,399 tons, and increased to 1,168,744 tons in 2022. With this huge production volume, the need for a fast and accurate tomato sorting system is very important to support an efficient distribution process and maintain product quality. The tomato sorting process, which is currently generally done manually, often faces various challenges. As the tomato production increases, this manual process becomes less efficient, time and labor intensive, and error-prone, mainly due to factors such as fatigue, boredom, limited human concentration, and different perceptions of fruit quality [2], [3], [4]. This can lead to inaccuracies in tomato classification, leading to the distribution of products that do not match quality standards and potential economic losses. Therefore, there is a need for an automated system that is able to assist the tomato classification process quickly and accurately to reduce human error and improve efficiency in the supply chain. With a more efficient system, tomato distribution can be done in a timely manner, ensuring the product arrives at the hands of consumers in optimal conditions and reducing the risk of spoilage or quality loss during delivery. This is important because tomatoes are one of the fruits with a specific level of maturity in a relatively short time span [5], [6]. Tomatoes can be classified using image processing based on obvious characteristics such as the ripeness level of the fruit. Image processing also makes use of the physical properties of tomatoes, such as color, shape, size, and surface defects, to sort and grade the fruit accordingly [7]. The main indicator used in this automated system is the color of the tomato skin. The color feature is the easiest characteristic to identify the ripeness level of tomatoes, as it can be observed directly without the need to touch the fruit to check the texture [8], [9].

Many classification using image processing and machine learning techniques have been applied, such as Convolutional Neural Networks (CNN), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). These methods are used to classify tomatoes into different quality classes based on color to accurately determine the ripeness or quality level of tomatoes. Based on research [10], [11], [12], [13], Support Vector Machine (SVM) and K-Nearest Neighbor (KNN) methods have been used for tomato fruit classification and showed good results. However, both methods have limitations in handling complex image variations. In addition, these methods are not capable of automatic feature extraction, which requires manual feature selection. In this process, researchers must determine the most relevant features for the classification task. The process can be complicated, time-consuming and error-prone if the selected features are not representative enough to distinguish between the classes [14]. Some research [15], [16], [17], [18], have also utilized the CNN method for tomato fruit classification with good accuracy results. However, these studies generally only focus on two or three classes of tomato fruit conditions, and not many have tried to develop methods for classification with more than three classes of conditions. Many developments have been made to the CNN architecture to improve model accuracy, such as the addition or reduction of architectural layers, as well as the use of regularization techniques such as Batch Normalization and Dropout. Several studies [19], [20], [21], prove that CNN architecture modifications, such as the use of Batch Normalization and Dropout, can improve model performance in classification. This is because the addition of regularization such as Batch Normalization can speed up model training and dropout is proven effective in reducing overfitting by randomly disabling neurons during training [22].

Therefore, this study will use the CNN method for classification, because CNN is able to extract visual features such as color, texture, and shape well, making it very suitable for tomato image classification. The automatic capability of CNN in feature extraction makes this deep learning model highly accurate in object classification, allowing the system to learn from visual data without requiring a manual feature extraction process, and proves to be more effective than conventional methods [23], [24] [25]. This study aims to develop a Convolutional Neural Network (CNN) method specifically modified by researchers to classify the ripeness level of tomato fruit based on the image of its skin color. This modification is made by reducing the number of layers, adding batch normalization, and adding dropouts. It aims to improve classification accuracy by maximizing the network's ability to recognize the unique characteristics of each class. Most previous studies have only used three ripeness classes, such as unriped, riped, and rotten, and are unable to capture the visual complexity of other ripeness categories. Therefore, in this study, the CNN method was designed to recognize visual differences in five categories, namely unriped, half-riped, riped, half-rotten, and rotten tomatoes. The addition of the number of classes aims to create a more detailed model and fit the needs of complex classification in real applications. With the development of this particular CNN, the research is expected to produce an accurate and efficient classification model in identifying the maturity level of tomato fruit based on its visual image, so that it can be applied in the agricultural industry to improve the quality of the process of sorting and distributing tomato fruit according to the desired standard.

2. METHOD

The following is a flowchart of the research conducted, which visualizes the main stages from dataset collection to model evaluation. This diagram is designed to provide an understanding of the process flow applied in the development of a tomato ripeness classification system.

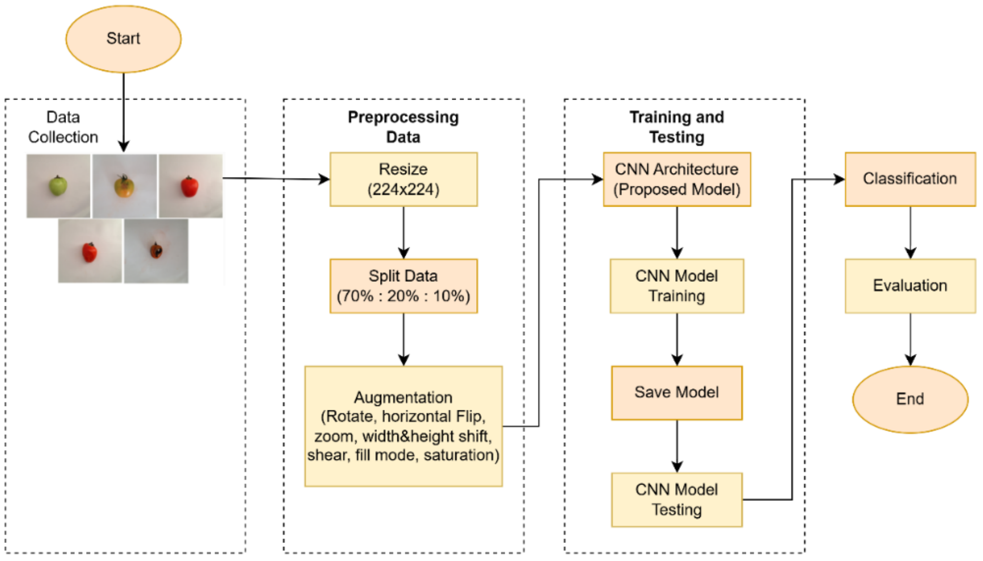

Start
Data Collection
Preprocessing Data
Resize (224x224)
Split Data (70% : 20% : 10%)
Augmentation (Rotate, horizontal Flip, zoom, width&height shift, shear, fill mode, saturation)
Training and Testing
CNN Architecture (Proposed Model)
CNN Model Training
Save Model
CNN Model Testing
Classification
Evaluation
End

Figure 1. Research Flow Diagram

Based on Figure 1, the first step in this research begins with the collection of tomato fruit image datasets, which will be used as data for classification of maturity levels. After the data is collected, pre-processing is performed which includes adjusting the image size, and splitting the data. The next stage is data augmentation, where various changes are applied to the training data to increase the variety and amount of data. Once the augmentation process is complete, the data is used to train the model using a custom CNN architecture developed by the researcher. After training, the model is tested to measure its performance against data that the model has never seen. Finally, the results from training and testing are analyzed at the evaluation stage, where the model's performance is evaluated based on certain metrics to assess its success in classifying the ripeness level of tomatoes.

2.1. Data Collection

The dataset used in this research is data in the form of images of servo variety tomatoes collected by researchers themselves. Furthermore, image capture was carried out using a Samsung Galaxy A54 cellphone camera with a resolution of 9 Megapixels and a ratio of 1:1 to ensure that more tomato objects were captured than the background in the image without going through the cropping process. In addition, magnification is done twice so that the tomato object can be seen more clearly. The data collection process was carried out by placing the tomatoes in a mini studio box with a white background. The camera was fixedly positioned through a hole on top of the box, with a distance of 30 cm from the object. This step aims to reduce the risk of vibration or movement that can cause the image to become blurry. An illustration of the dataset collection carried out by researchers can be seen in Figure 2.

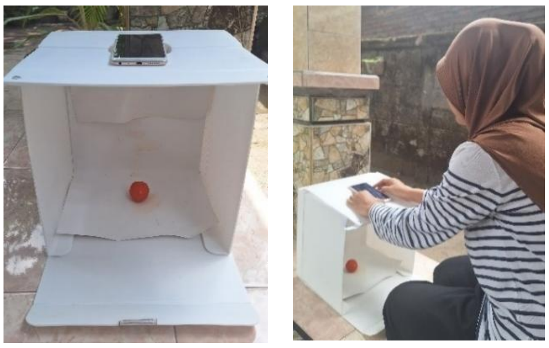


Figure 2. Dataset Collection Process

The collected dataset totals 500 images in (.jpg) format and is divided into 5 categories, namely 'Unriped', 'Half-Riped', 'Riped', 'Half-Rotten', and 'Rotten' with 100 images each. The dataset can be accessed here: [Tomato Fruit Dataset](). An example dataset of each class can be seen in Figure 3.

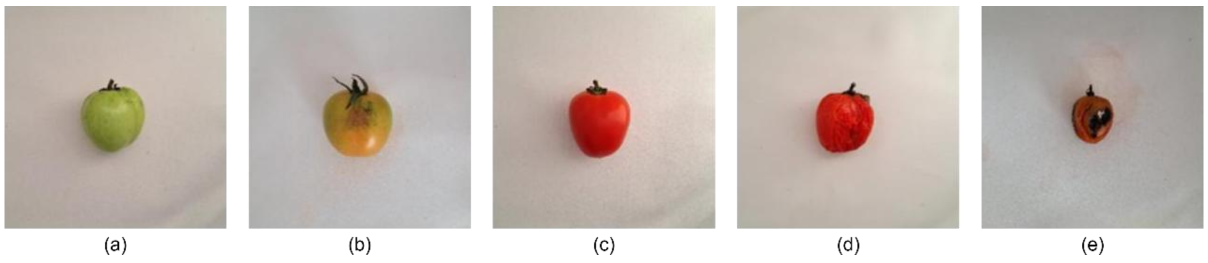


Figure 3. Sample Dataset. (a) Unriped, (b) Half-Riped, (c) Riped, (d) Half-Rotten, (e) Rotten.

2.2. Pre-processing

After collecting the data, the next step is to perform pre-processing which aims to improve the image quality so as to facilitate identification and recognition. In this research, a rescale process is carried out to change the RGB value from the range 0-255 to 0 to 1, because models generally cannot process such high values [26]. Then, resize the image from 3056×3056 to 224×224 pixels. The 224x224 size is proven to increase the effectiveness during the training process [17].

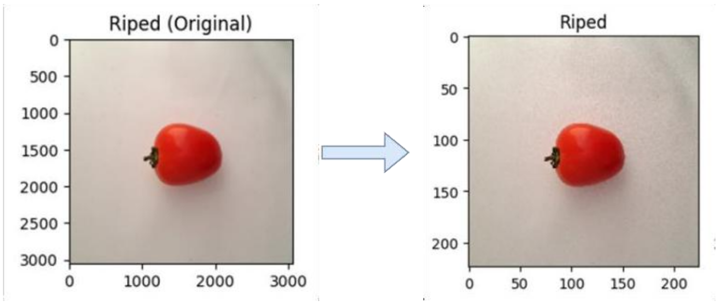


Figure 4. Resized Image of Tomato Fruit

2.3. Split Data

In this study, data was split with a ratio of 70% : 20% : 10% for each data set, namely training, validation, and testing data. This ratio was chosen to ensure the model can be trained effectively with sufficient data, monitor performance and prevent overfitting through validation data, and evaluate the accuracy and generalization of the model on new data with testing data and help optimize hyperparameter adjustments [27]. This split aims to ensure that the built model has enough data to learn and test effectively. With this ratio, the total number of images used in this study is 500 images. Therefore, after the data splitting process, 350 images of training data are obtained, which are used to train the model to recognize the patterns and characteristics of each class. Furthermore, 100 images are allocated for validation, which serves to evaluate the performance of the model during the training process and avoid overfitting. The remaining 50 images are used as testing data, which will be tested to measure the model's ability to classify new images that have never been seen before.

2.4. Augmentation

Augmentation is a technique in machine learning used to increase the number of training data samples by manipulating the existing samples [28]. In this research, an augmentation process is performed on the training data using the ImageGenerator library from Keras. The augmentation stages applied include:

a. Rotation : Setting rotation_range=40 randomly rotates the image by 40°, increasing the variety of viewing angles of objects.
b. Horizontal Flip : Horizontally flip the image to increase the variety of object orientations.
c. Zoom : zoom_range=0.2 to randomly enlarge or reduce the image by up to 20%.
d. Width Shift : width_shift_range=0.2 to shift the image horizontally by 20% of the image width.
e. Height Shift : height_shift_range=0.2 to shift the image vertically up to 20% of the image height.
f. Shear : Shear distortion with shear_range=0.2 for additional variations.
g. Fill Mode : Set fill_mode='nearest' to fill the empty areas that appear after the transformation using the nearest pixels to keep the image intact.

In addition, the saturation augmentation technique was also used to increase the color variation in the image dataset by changing the color intensity. With these parameters, augmentation produces new image variations from the pre-existing data, bringing the total data after augmentation to 1739 images. Here is an example of an image that has been augmented can be seen in Figure 5.

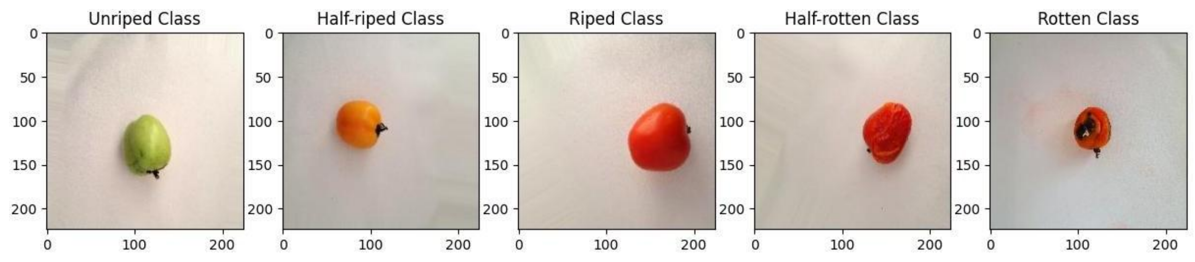


Figure 5. Augmented Dataset

2.5. Proposed Convolutional Neural Network Models

In this research, the model used is Convolutional Neural Network (CNN) with custom architecture. In general, CNN consists of 3 types of layers, namely Convolutional for feature extraction, pooling for dimensionality reduction, and a fully connected layer collecting features to determine classification results [29]. Researchers proposed three CNN models as an extension of the models in previous research, with self-designed hidden layer configurations. CNN models from previous research can be seen in Figure 6.

Vol. 6, No. 2, April 2025, Page. 501-520
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.2.4168

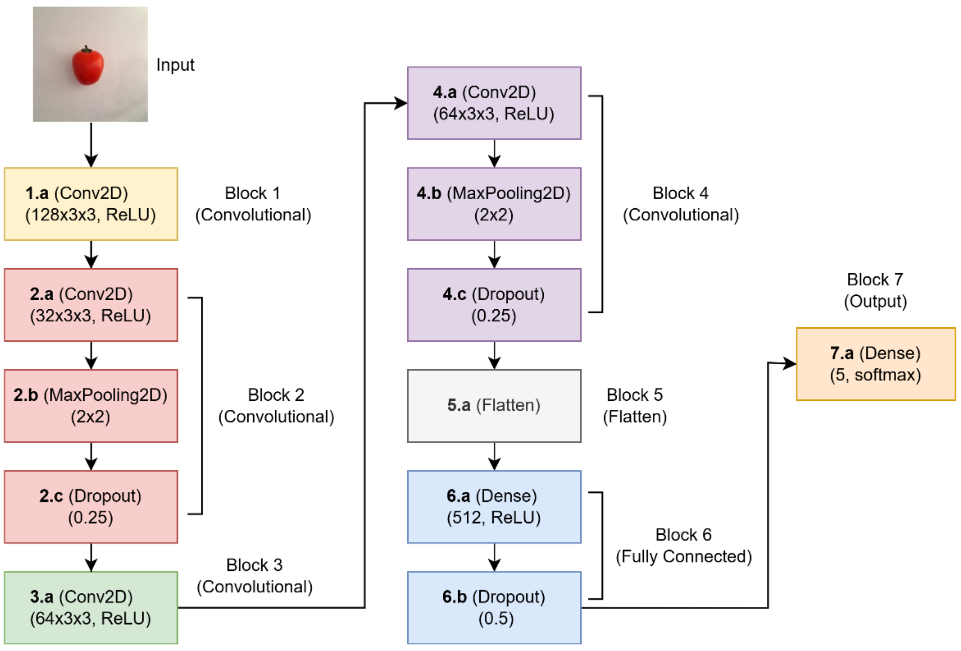


Figure 6. Previous Research Model

Figure 6 shows the architecture of the CNN model used in the previous study, which serves as a reference for researchers in developing and modifying the model. In the proposed model, we reduce the number of layers and add batch normalization and dropout in some layers to improve the performance of tomato ripeness classification.

2.5.1. Proposed Model 1

Proposed model 1 is a standard CNN architecture designed based on the rule of thumbs [30]. This architecture consists of multiple layers, consisting of convolutional and pooling layers, ending with a fully connected layer and an output layer. The researcher used 4 main blocks for the convolutional layer just like the model in the previous study. In this study, a Conv2D layer was used with an increasing number of filters and ReLU activation. Block 1 consists of the Conv2D layer (1.a) and the addition of MaxPooling which serves to reduce the dimensionality of the feature map and extract important information [31], as well as block 2 consists of the Conv2D layer (2.a) and Maxpooling (2.b), but without the addition of dropout (2.c). Then the same as block 1, in block 3 there is a Conv2D layer (3.a) with the addition of Maxpooling. Furthermore, block 4 consists of Conv2D (4.a) and MaxPooling (4.b) layers without any dropouts (4.c). Block 5 consists of flatten (5.a) which is used to flatten the output into a vector before entering the fully connected layer. Furthermore, in block 6 for fully connected, the researcher only uses the dense layer (6.a) without dropout (6.b). Then finally in the output layer, the researcher uses dense (7.a) with softmax activation because the processed data has more than two classes or multiclass classification [32]. For more details, the details of the architecture of the proposed model 1 can be seen in Table 1.

Table 1. Proposed Model 1

Block Layer	Layer	Layer in the Block
Block 1	Convolutional	Conv2D (32 filter, kernel 3x3, ReLU) MaxPooling2D (pool size 2x2)
Block 2	Convolutional	Conv2D (64 filter, kernel 3x3, ReLU) MaxPooling2D (pool size 2x2)
Block 3	Convolutional	Conv2D (128 filter, kernel 3x3, ReLU) MaxPooling2D (pool size 2x2)
Block 4	Convolutional	Conv2D (256 filter, kernel 3x3, ReLU) MaxPooling2D (pool size 2x2)
Block 5	Flatten	Flatten
Block 6	Fully Connected	Dense (256, ReLU)
Block 7	Output	Dense (5 units, softmax)

The below is a summary of Proposed Model 1, which is designed to improve performance in tomato ripeness classification. This summary presents complete information about the architecture structure, parameters, and configurations in this model. For more details, it can be seen in Table 2.

Table 2. Summary of Proposed Model 1

No	Name	Output Size	Param #
1	conv2d_4 (Conv2D)	(None, 222, 222, 32)	896
2	max_pooling2d_4 (MaxPooling2D)	(None, 111, 111, 32)	0
3	conv2d_5 (Conv2D)	(None, 109, 109, 64)	18496
4	max_pooling2d_5 (MaxPooling2D)	(None, 54, 54, 64)	0
5	conv2d_6 (Conv2D)	(None, 52, 52, 64)	73856
6	max_pooling2d_6 (MaxPooling2D)	(None, 26, 26, 128)	0
7	conv2d_7 (Conv2D)	(None, 24, 24, 256)	295168
8	max_pooling2d_7 (MaxPooling2D)	(None, 12, 12, 256)	0
9	flatten_1 (Flatten)	(None, 36864)	0
10	dense_2 (Dense)	(None, 256)	9437440
No	Name	Output Size	Param #
11	dense_3 (Dense)	(None, 5)	1285

Total params: 9,827,141
Trainable params: 9,827,141
Non-trainable params: 0

Table 2 shows the number of parameters generated by proposed model 1, which is 9,827,141 parameters. This number indicates that the model is quite complex in learning detailed patterns. With a large number of parameters, the model is expected to provide accurate results, but requires efficient training to avoid overfitting.

2.5.2. Proposed Model 2

The architecture in proposed model 2 consists of 6 block layers, with 3 block layers Conv2D, flatten, fully connected, and output. Block 1 consists of the Conv2D layer (1.a) with additional Batch Normalization, and MaxPooling. Likewise, block 2 contains Conv2D (2.a) and Maxpooling (2.b) layers with additional Batch Normalization and no dropout (2.c). Furthermore, similar to block 1, block 3 consists of a Conv2D layer (3.a) with additional Batch Normalization, and MaxPooling. Each convolution block consists of Conv2D with an increasing number of filters and there is “same” padding to clarify the image boundaries, so that detection becomes more focused [33]. In addition, the use of batch normalization helps the model to reach the optimal value faster during training [34]. After that, block 4 has a flatten layer and block 5 is fully connected with 2 dense layers, and block 6 has an output layer. For more details, the architecture of model 2 can be seen in Table 3.

Table 3. Proposed Model 2

Block Layer	Layer	Layer in the Block
Block 1	Convolutional	Conv2D (32 filter, kernel 3x3, padding='same')
		BatchNormalization
		Activation (ReLU)
		MaxPooling2D (pool size 2x2)
Block 2	Convolutional	Conv2D (64 filter, kernel 3x3, padding='same')
		BatchNormalization
		Activation (ReLU)
		MaxPooling2D (pool size 2x2)
Block 3	Convolutional	Conv2D (128 filter, kernel 3x3, padding='same')
		BatchNormalization
		Activation (ReLU)
		MaxPooling2D (pool size 2x2)
Block 4	Flatten	Flatten
Block 5	Fully Connected	Dense (64, ReLU)
		Dense (32, ReLU)
Block 6	Output	Dense (5 units, softmax)

The following is a summary of Proposed Model 2, this summary contains information about the structure, parameters, and model settings. For details can be seen in Table 4.

Table 4. Summary of Proposed Model 2

No	Name	Output Size	Param #
1	conv2d (Conv2D)	(None, 224, 224, 32)	896
2	batch_normalization (BatchNormalization)	(None, 224, 224, 32)	128
3	activation (Activation)	(None, 224, 224, 32)	0
4	max_pooling2d (MaxPooling2D)	(None, 112, 112, 32)	0
No	Name	Output Size	Param #
5	conv2d_1 (Conv2D)	(None, 112, 112, 64)	18496
6	batch_normalization_1 (BatchNormalization)	(None, 112, 112, 64)	256
7	activation_1 (Activation)	(None, 112, 112, 64)	0
8	max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 64)	0
9	conv2d_2 (Conv2D)	(None, 56, 56, 128)	73856
10	batch_normalization_2 (BatchNormalization)	(None, 56, 56, 128)	512
11	activation_2 (Activation)	(None, 56, 56, 128)	0
12	max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 128)	0
13	flatten (Flatten)	(None, 100352)	0
14	dense (Dense)	(None, 64)	6422592
15	dense_1 (Dense)	(None, 32)	2080
16	dense_2 (Dense)	(None, 5)	165

Total params: 6,518,981
Trainable params: 6,518,533
Non-trainable params: 448

Table 4 shows that the number of parameters generated by Proposed Model 2 is 6,518,981. This number is smaller than Proposed Model 1 which has 9,827,141 parameters. Thus, the parameters in Proposed Model 2 are about 33.67% smaller than those in Proposed Model 1. This reduction reflects a more efficient model design without sacrificing the ability to capture important patterns in the data.

2.5.3. Proposed Model 3

In the proposed model 3, 7 block layers are used with 4 block layers of convolution, flatten, fully connected, and output. In block 1 there are Conv2D (1.a) and MaxPooling layers. Then in block 2 there are Conv2D (2.a) and MaxPooling (2.b) layers without any dropout (2.c). Furthermore, in block 3 there is a Conv2D layer (3.a) with MaxPooling. In block 4 there is Conv2D (4.a), MaxPooling (4.b), and dropout (4.c). Furthermore, block 5 consists of a flatten layer (5.a), block 6 with a dense layer (6.a) and 2 dropout layers. Finally, block 7 consists of an output layer with a dense layer of 5 units and softmax activation. The use of dropouts in some layers aims to reduce overfitting by randomly removing neurons during the training [35]. The architecture of the proposed model 3 can be seen in Table 5.

Table 5. Proposed Model 3

Block Layer	Layer	Layer in the Block
Block 1	Convolutional	Conv2D (32 filter, kernel 3x3, ReLU)
		MaxPooling2D (pool size 2x2)
Block 2	Convolutional	Conv2D (64 filter, kernel 3x3, ReLU)
		MaxPooling2D (pool size 2x2)
Block 3	Convolutional	Conv2D (64 filter, kernel 3x3, ReLU)
		MaxPooling2D (pool size 2x2)
Block 4	Convolutional	Conv2D (128 filter, kernel 3x3, ReLU)
		MaxPooling2D (pool size 2x2)
		Dropout (rate 0.5)
Block 5	Flatten	Flatten
Block 6	Fully Connected	Dropout (rate 0.5)
		Dense (128, ReLU)
		Dropout (rate 0.5)
Block 7	Output	Dense (5 units, softmax)

Here is a summary of the Proposed Model 3, which is designed for tomato ripeness classification. This summary contains information about the structure, parameters, and settings of the model. For more details, please refer to Table 6.

Table 6. Summary of Proposed Model 3

No	Name	Output Size	Param #
1	conv2d (Conv2D)	(None, 222, 222, 32)	896
2	max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
3	conv2d_1 (Conv2D)	(None, 109, 109, 64)	18496
4	max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 64)	0
5	conv2d_2 (Conv2D)	(None, 52, 52, 64)	36928
6	max_pooling2d_2 (MaxPooling2D)	(None, 26, 26, 64)	0
7	conv2d_3 (Conv2D)	(None, 24, 24, 128)	73856
8	max_pooling2d_3 (MaxPooling2D)	(None, 12, 12, 128)	0
9	dropout (Dropout)	(None, 12, 12, 128)	0
10	flatten (Flatten)	(None, 18432)	0
11	dropout_1 (Dropout)	(None, 18432)	0
12	dense (Dense)	(None, 128)	2359424
13	dropout_2 (Dropout)	(None, 128)	0
14	dense_1 (Dense)	(None, 5)	645

Total params: 2,490,245
Trainable params: 2,490,245
Non-trainable params: 0

Table 6 shows that the number of parameters generated by Proposed Model 3 is 2,490,245. This number is smaller than Proposed Model 1 which has 9,827,141 parameters, so the parameters in Proposed Model 3 are about 74.73% smaller. Compared to Proposed Model 2 which also has 6,518,981 parameters, Proposed Model 3 is about 61.83% smaller. This number of parameters not only indicates a more efficient model in memory capacity, but can also reduce computational load, speed up training time, and reduce the risk of overfitting.

2.6. Hyperparameter

In this research, several hyperparameters such as optimizer, learning rate, batch size, and epoch are used. The selection of values for each hyperparameter aims to achieve an optimal balance between training speed and model accuracy. The value of each hyperparameter can be seen in Table 7.

Table 7. Hyperparameter of Three Proposed Model

Hyperparameter	Value
Optimizer	Adam
Learning Rate	0.001
Batch Size	16, 32, 64
Epoch	50

Based on Table 7. The optimizer used is Adam (Adaptive Moment Estimation). This optimizer is proven effective because it combines momentum and adaptive learning rate, accelerates convergence and improves model stability [36]. The selected learning rate value is 0.001. Then several batch size values (16, 32, 64) were used to determine the effect of batch size on model performance because it is a good default value and provides a balance between parameter update stability and training efficiency [37]. The last one, 50 epochs were used to give the model enough time to explore and learn patterns from the training data, with the aim of reducing the risk of overfitting that can arise if the model is trained for too long [38].

2.7. Evaluation

In this research, an evaluation was conducted to measure the performance of the trained model in classifying the ripeness level of tomato fruit. The evaluation is done using confusion matrix, which provides information about accuracy, precision, recall, and F1-score for each class [39].

$$Accuracy = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

$$F1 - Score = 2 * \frac{\text{Presition} \times \text{Recall}}{\text{Presition} + \text{Recall}} \tag{4}$$

3. RESULT

The following are the performance results of the research that has been conducted, along with a graphical display of loss and accuracy.

3.1. Model Performance

In this research, training with 3 proposed models was carried out using training, validation, and test data using several batch size values. The following are the results of the training that has been conducted.

3.1.1. Training Result with Train Data

The following are the results of model training using train data by showing precision, recall, F1-score, and accuracy, and displaying the parameter values of the model can be seen in Table 8.

Table 8. Training Result with Train Data

Models	Batch size	Precision	Recall	F1-Score	Accuracy	Parameters (m)
Proposed Medel 1	16	1.00	1.00	1.00	1.00	9.8
	32	**1.00**	**1.00**	**1.00**	**1.00**	**9.8**
	64	1.00	1.00	1.00	1.00	9.8
Proposed Medel 2	**16**	**0.91**	**0.85**	**0.85**	**0.85**	**6.5**
	32	0.90	0.82	0.79	0.82	6.5
	64	0.87	0.84	0.84	0.84	6.5
Proposed Medel 3	**16**	**1.00**	**1.00**	**1.00**	**1.00**	**2.4**
	32	0.98	0.97	0.97	0.97	2.4
	64	0.99	0.99	0.99	0.99	2.4

Table 8 shows the training results using the train data. In all batch sizes, proposed model 1 achieved perfect performance with 100% accuracy, precision, recall, and F1-Score. While in proposed model 2, the highest accuracy value is found in the use of batch size 16 which is 85%. Then in proposed model 3, the best performance was achieved at batch size 16 with accuracy, precision, recall, and F1-Score of 100% respectively, while at batch sizes 64 and 32 the accuracy decreased slightly to 99% and 97%. Proposed Model 3 has the smallest number of parameters of proposed models 1 and 2. Based on these results, proposed model 3 with batch size 16 is the best choice because it is able to achieve perfect performance with the smallest number of parameters.

3.1.2. Training Result with Validation Data

The training results on the validation data are presented in Table 9 showing the precision, recall, F1-Score, and accuracy values of the model.

Table 9. Training Result with Validation Data

Models	Batch size	Precision	Recall	F1-Score	Accuracy	Parameters (m)
Proposed Medel 1	16	0.95	0.95	0.95	0.95	9.8
Models	Batch size	Precision	Recall	F1-Score	Accuracy	Parameters (m)
	32	**0.98**	**0.98**	**0.98**	**0.98**	**9.8**
	64	0.96	0.95	0.95	0.95	9.8
Proposed Medel 2	**16**	**0.91**	**0.85**	**0.85**	**0.85**	**6.5**
	32	0.89	0.84	0.83	0.84	6.5
	64	0.92	0.86	0.84	0.86	6.5
Proposed Medel 3	**16**	**0.96**	**0.96**	**0.96**	**0.96**	**2.4**
	32	0.95	0.95	0.95	0.95	2.4
	64	0.95	0.94	0.94	0.94	2.4

The results in Table 9 show the results of training using validation data. In proposed model 1, the highest accuracy value is 98% at batch size 32. While in proposed model 2, the highest accuracy value of 86% is obtained at batch size 64. However, it is less effective because the accuracy results on the validation data are greater than the train data, indicating the potential for overfitting. Likewise, the results using batch size 32, so the use of batch size 16 is considered more optimal even with a lower accuracy of 85%. Finally, proposed model 3 produced the highest accuracy value of 96% at batch size 16. Using batch sizes 32 and 64 also produced accuracies of 95% and 94% respectively, slightly lower

Vol. 6, No. 2, April 2025, Page. 501-520
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.2.4168

but still stable. Overall, training with validation data shows that Proposed Model 3 with fewer parameters and using batch size 16 provides the best balance between performance and efficiency.

3.1.3. Training Result with Test Data

The following are the training results using the test data presented in Table 10. This table displays the performance of the model on the data with precision, recall, F1-Score, and accuracy metrics.

Table 10. Training Result with Test Data

Models	Batch size	Precision	Recall	F1-Score	Accuracy	Parameters (m)
Proposed Medel 1	16	0.98	0.98	0.98	0.98	9.8
	32	**0.98**	**0.98**	**0.98**	**0.98**	**9.8**
	64	0.98	0.98	0.98	0.98	9.8
Proposed Model 2	**16**	**0.91**	**0.82**	**0.82**	**0.82**	**6.5**
	32	0.91	0.84	0.82	0.84	6.5
	64	0.89	0.82	0.79	0.82	6.5
Proposed Model 3	**16**	**0.98**	**0.98**	**0.98**	**0.98**	**2.4**
	32	0.98	0.98	0.98	0.98	2.4
	64	0.96	0.96	0.96	0.96	2.4

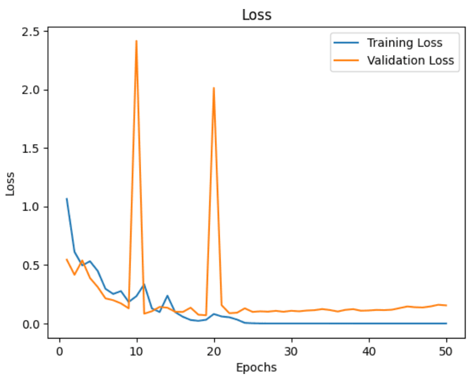
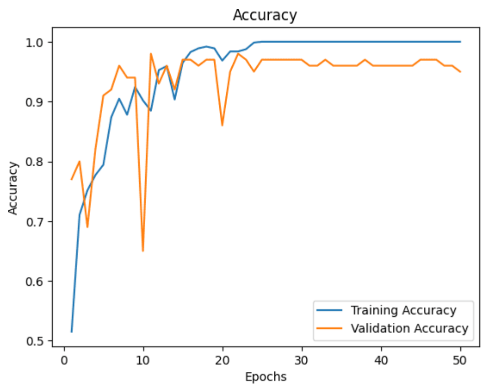
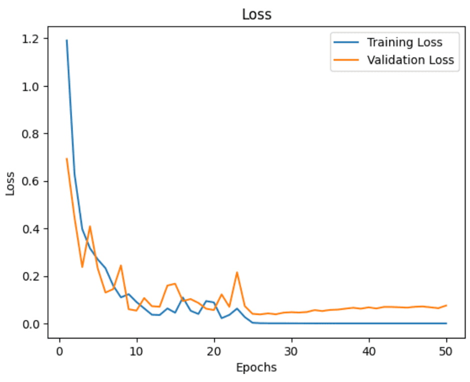
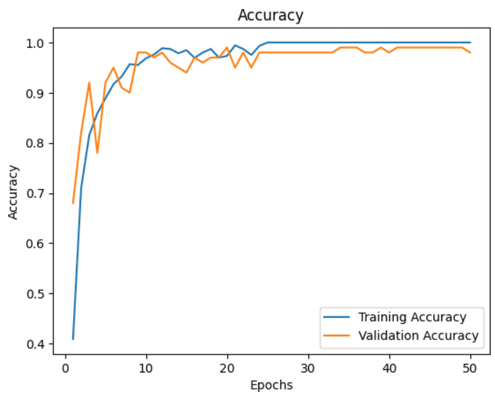
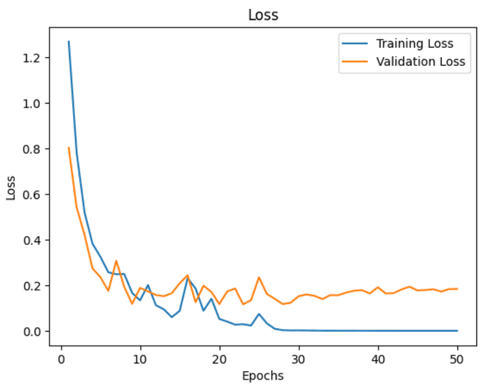
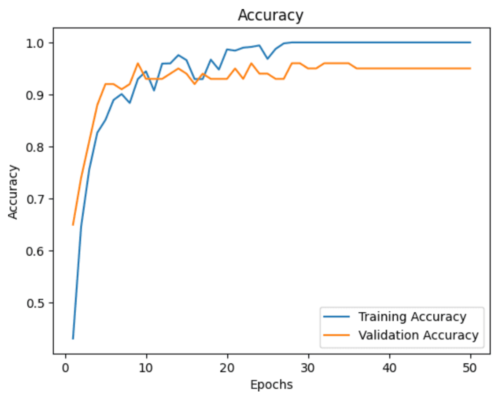
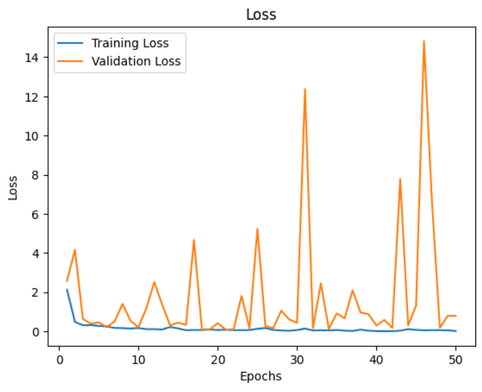
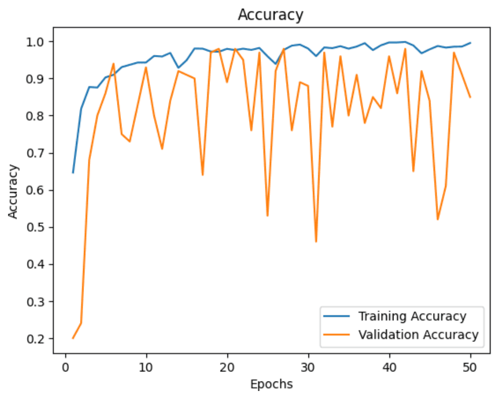
Based on Table 10, proposed model 1 shows the best performance in each batch size, achieving an accuracy of 98%, as well as precision, recall, and F1-score values which are also 98%. Furthermore, proposed model 2, achieved the highest accuracy of 84% at batch size 32. However, these results are less than optimal because the accuracy is greater than the accuracy of the model using train data. Then using a batch size of 64 produces an accuracy of 82%, but training with validation data produces greater accuracy than training using train data. So that in proposed model 2, the use of batch size 16 is considered more optimal. Finally, proposed model 3 also showed good performance with 98% accuracy at batch sizes 16 and 32, and 96% accuracy at batch size 64. Based on these results, proposed model 3 with batch size 16 is the optimal choice because it has the least number of parameters so it is more efficient and the use of a small batch size means that the model is better at recognizing new patterns, because it produces more diverse gradient updates.

3.2. Loss and Accuracy Graph of Proposed Models

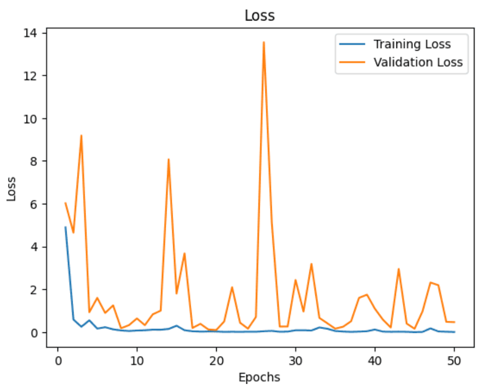
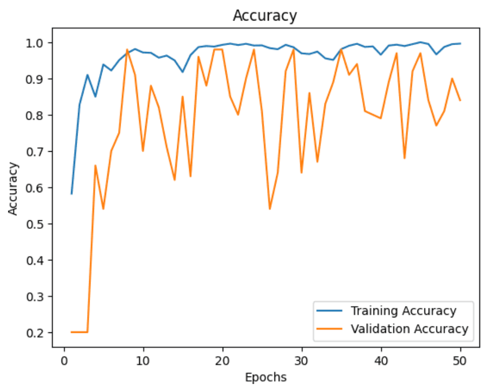
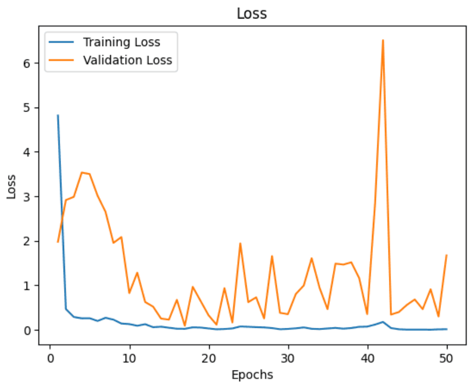
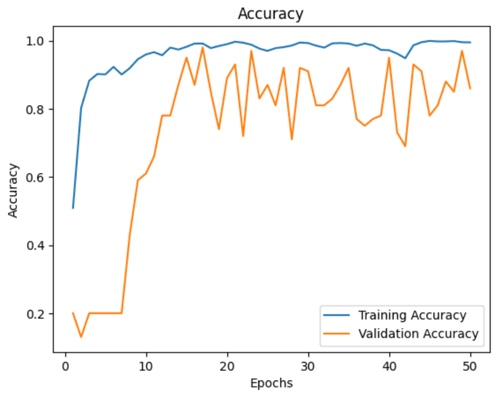
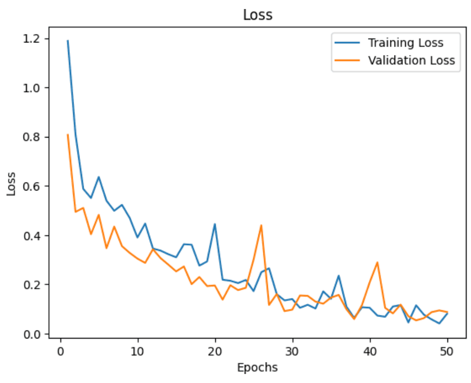
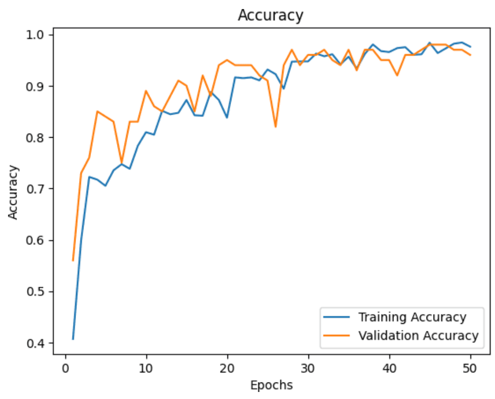
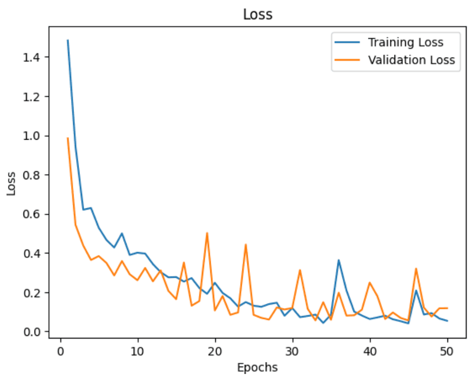
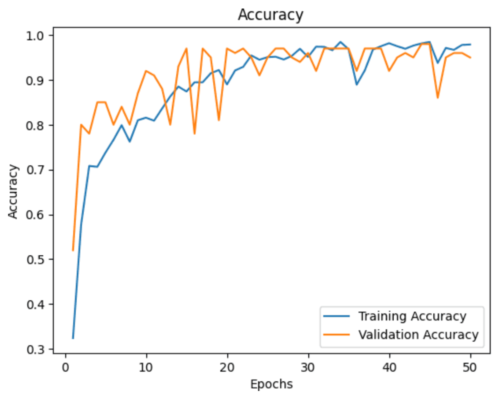
Although all three models show good results, it is important to look at the loss and accuracy graps graphs to ensure stability and consistency of performance during training. The loss and accuracy graphs

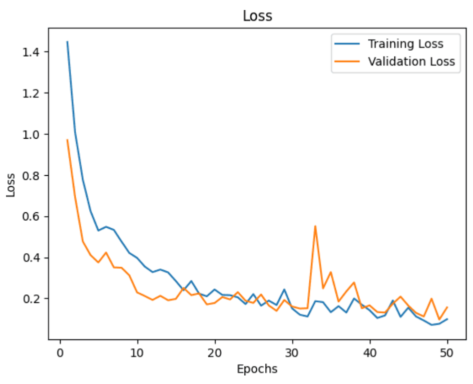
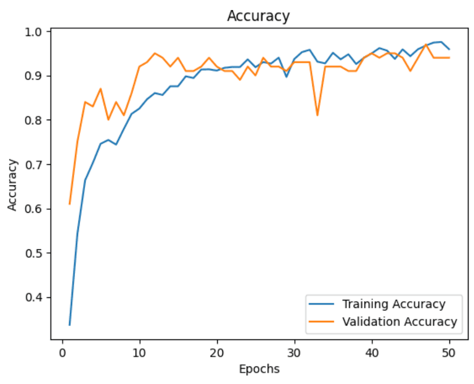
of each model can be seen in Table 11.

Table 11. Loss and Accuracy Graph of the Three Proposed Models

Models	Batch Size	Loss Graph	Accuracy Graph
Proposed Model 1	16		
	32		
	64		
Proposed Model 2	16		

Vol. 6, No. 2, April 2025, Page. 501-520
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2025.6.2.4168

	32		
	64		
Proposed Model 3	16		
	32		

64		

Based on Table 11, it can be seen that proposed model 1 with batch size 16 testing shows significant fluctuations in its validation loss. This means that the model is not yet stable in learning the data pattern at the validation stage. While testing batch sizes 32 and 64 shows more stable results. Then the loss graph on proposed model 2 with batch size testing (16, 32, 64) shows instability between training loss and validation loss, where validation loss experiences significant fluctuations. The same can be seen in the accuracy graph, where training accuracy and validation accuracy are also unstable. Validation accuracy often fluctuates drastically, indicating that the model's performance is inconsistent. Whereas the loss and accuracy graphs on proposed model 3 with batch size testing (16, 32, 64) look more consistent and stable during the training and testing process compared to Proposed models 1 and 2. The difference between training loss and validation loss is not too large. Likewise, the training accuracy and validation accuracy also look more consistent and stable compared to the two proposed models. For more details, the performance comparison of the three models with the best results can be seen in the graph visualization in Figure 7.

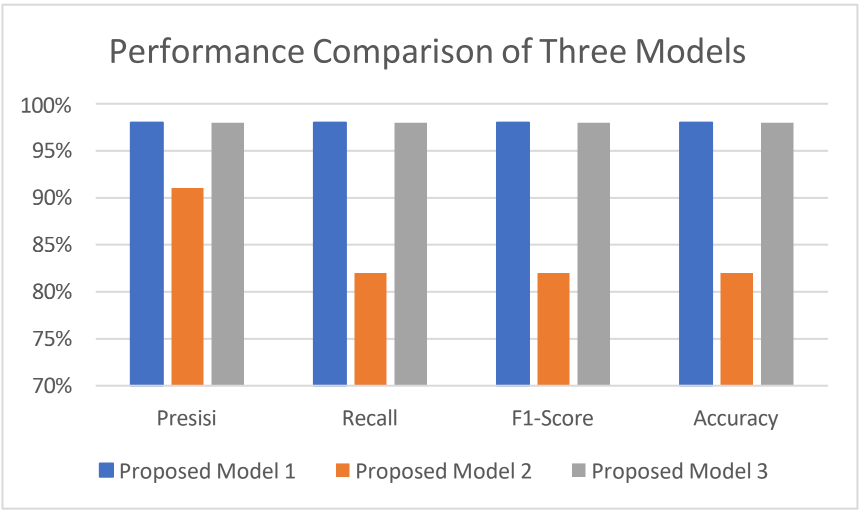


Figure 7. Graph Visualization of the Three Proposed Models

4. DISCUSSIONS

Based on the research that has been conducted, the best results are obtained from each of the proposed models. In similar research that has been done [15], for the classification of tomatoes using 3 categories namely, 'raw', 'half-ripe', and 'ripe'. by using the CNN method, the research results get a good accuracy value by using a batch size of 16 with a total dataset of 1148 images measuring

150x150. The performance of the three proposed models with models in previous studies can be seen in Table 12.

Table 12. Comparison of the Three Proposed Models with Previous Research Model

Models	Batch size	Precision	Recall	F1-Score	Accuracy	Parameters (m)
Proposed Medel 1	32	0.98	0.98	0.98	0.98	9.8
Proposed Medel 2	16	0.91	0.82	0.82	0.82	6.5
Proposed Medel 3	**16**	**0.98**	**0.98**	**0.98**	**0.98**	**2.4**
Previous Research [15]	16	0.90	0.92	0.91	0.93	42.5

Table 12 is a comparison between the three proposed models with models from previous research. Based on the computational load as well as the loss and accuracy graphs of each model in Table 11, it can be concluded that proposed model 3 has the most optimal performance compared to Proposed models 1 and 2, as well as models in previous research. Proposed Model 3 has a parameter value 33.67% smaller than Proposed Model 1, and 74.73% smaller than Proposed Model 2, and 94.15% smaller than the model in the previous study. This is certainly very influential in the computational process, where a smaller number of parameters can speed up the computational process, and vice versa. Then when viewed from its performance, proposed model 3 is superior with 98% accuracy compared to the model in previous research with 93%. Although the results have provided good accuracy, it should be considered that future research is important to increase the dataset size to obtain better results with larger and more diverse datasets. The interpretation of this research is that the development in the form of adding dropouts to several layers of the model, this research has succeeded in improving the ability to classify the ripeness level of tomato fruit well. This indicates progress in the development of technology for object classification, which has a positive impact in the agricultural industry that can help the process of automatically sorting tomatoes based on their maturity level effectively and efficiently.

5. CONCLUSION

Based on the research conducted, testing CNN models with batch sizes of 16, 32, and 64 shows that batch size 32 performs best in proposed model 1, while batch size 16 is optimal for Proposed models 2 and 3. Proposed model 1 achieved 98% accuracy, Proposed model 2 reached 82%, and Proposed model 3 also achieved 98%. Proposed model 3 proved more efficient with 2.4 million parameters and a stable loss and accuracy graph, outperforming Proposed models 1 and 2. Furthermore, compared to the previous research model with 93% accuracy and larger parameters, proposed model 3 demonstrates superior performance and efficiency. This indicates that using additional dropouts with a batch size of 16 achieves optimal performance, resulting in accuracy, precision, recall, and F1-score of 98%.

This research makes an important contribution to the Informatics Engineering field, especially in developing a more efficient deep learning model. The results show that models with a small number of parameters can still achieve high accuracy without reducing performance. This is very useful for applications on devices with limited resources. With a lighter model, memory usage is reduced, making it more efficient to implement. In addition, the use of dropout techniques and proper batch size selection can also help improve the overall performance of the model. To improve future research results, it is recommended to increase the number of datasets to be more numerous and diverse so that the model can recognize patterns better. In addition, the lighting when taking datasets also needs to be considered. Make sure the light remains similar so that the image results are consistent and make it easier for the model in the analysis process.

CONFLICT OF INTEREST

The authors declare no conflict of interest in this research.

ACKNOWLEDGEMENT

The authors would like to thank the Intelligent System Laboratory, Master of Information Technology, University of Mataram for supporting this research.

REFERENCES

[1] A. Sadri Agung, A. S. Farid Dirgantara, M. Syachrul Hersyam, A. Baso Kaswar, and D. Darma Andayani, "Classification of Tomato Quality Based on Color Features And Skin Characteristics Using Image Processing Based Artificial Neural Network," vol. 4, no. 5, pp. 1021–1032, 2023, doi: 10.52436/1.jutif.2023.4.5.780.

[2] S. Sanjaya, "Aplikasi pengenalan tingkat kematangan buah tomat menggunakan fitur warna hsv berbasis android," *Jurnal Teknoinfo*, vol. 16, no. 1, pp. 26–33, 2022.

[3] H. S. Mputu, A. Abdel-Mawgood, A. Shimada, and M. S. Sayed, "Tomato Quality Classification Based on Transfer Learning Feature Extraction and Machine Learning Algorithm Classifiers," *IEEE Access*, vol. 12, pp. 8283–8295, 2024, doi: 10.1109/ACCESS.2024.3352745.

[4] P. Li, J. Zheng, P. Li, H. Long, M. Li, and L. Gao, "Tomato maturity detection and counting model based on MHSA-YOLOv8," *Sensors*, vol. 23, no. 15, p. 6701, 2023.

[5] A. M. K. Putri and A. F. Rozi, "Implementasi Convutional Neural Network Dalam Menentukan Tingkat Kematangan Mentimun Dan Tomat Berdasarkan Warna Kulit," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 8, no. 5, pp. 10388–10394, 2024.

[6] F. Su, Y. Zhao, G. Wang, P. Liu, Y. Yan, and L. Zu, "Tomato maturity classification based on SE-YOLOv3-MobileNetV1 network under nature greenhouse environment," *Agronomy*, vol. 12, no. 7, p. 1638, 2022.

[7] S. M. Nassiri, A. Tahavoor, and A. Jafari, "Fuzzy logic classification of mature tomatoes based on physical properties fusion," *Information Processing in Agriculture*, vol. 9, no. 4, pp. 547–555, 2022.

[8] I. R. M. Fatah, A. H. Ginting, and W. T. Ina, "Klasifikasi Tingkat Kematangan Buah Tomat Berdasarkan Warna," *JTekEL: Jurnal Teknik Elektro*, vol. 1, no. 1, pp. 20–25, 2024.

[9] T. Kim, D.-H. Lee, K.-C. Kim, T. Choi, and J. M. Yu, "Tomato maturity estimation using deep neural network," *Applied Sciences*, vol. 13, no. 1, p. 412, 2022.

[10] N. Astrianda, "Klasifikasi Kematangan Buah Tomat Dengan Variasi Model Warna Menggunakan Support Vector Machine," *VOCATECH: Vocational Education and Technology Journal*, vol. 1, no. 2, pp. 45–52, Apr. 2020, doi: 10.38038/vocatech.v1i2.27.

[11] B. M. Alfaruq, D. Erwanto, and I. Yanuartanti, "Klasifikasi Kematangan Buah Tomat Dengan Metode Support Vector Machine," *Generation Journal*, vol. 7, no. 3, pp. 93–101, 2023.

[12] L. Ningsih and P. Cholidhazia, "Classification Of Tomato Maturity Levels Based on RGB And HSV Colors Using KNN Algorithm," *RIGGS: Journal of Artificial Intelligence and Digital Business*, vol. 1, no. 1, pp. 25–30, Jul. 2022, doi: 10.31004/riggs.v1i1.10.

[13] A. B. M. Widat, A. Baijuri, and F. Lazim, "Klasifikasi Kematangan Citra Buah Tomat Berdasarkan Ekstraksi Fitur Warna Menggunakan Metode K-NN," *G-Tech: Jurnal Teknologi Terapan*, vol. 8, no. 3, pp. 1779–1786, 2024.

[14] B. S. Utami, "Komparatif Antara Pendekatan Tradisional Dan Metode Deep Learning Dalam Pengenalan Tulisan Tangan Pada Aplikasi Komputer Visi," *Jurnal Teknologi Terkini*, vol. 3, no. 7, 2023.

[15] N. A. Ayunda, E. Haryatmi, and T. A. Riyadi, "Classification of Tomato Ripeness Based on Convolutional Neural Network Methods," *Journal of Information Systems and Informatics*, vol. 5, no. 4, pp. 1658–1675, Dec. 2023, doi: 10.51519/journalisi.v5i4.613.
[16] K. S. Nurbidin and I. K. G. Suhartana, "Identifikasi Tingkat Kematangan Buah Tomat Menggunakan Convolution Neural Network (CNN)," *Jurnal Nasional Teknologi Informasi dan Aplikasinya*, vol. 1, no. 1, pp. 747-750, Nov. 2022.
[17] T. Hidayat, M. Fatchan, and W. Hadikristanto, "CNN Algorithm Approach for Classification of Tomato Fruit Maturity Levels," *International Journal of Sustainable Applied Sciences*, vol. 2, pp. 421–432, May 2024, doi: 10.59890/ijsas.v2i5.1862.
[18] S. R. Nagesh Appe, G. Arulselvi, and G. Balaji, "Tomato Ripeness Detection and Classification using VGG based CNN Models", *Int J Intell Syst Appl Eng*, vol. 11, no. 1, pp. 296–302, Feb. 2023.
[19] U. N. Oktaviana, R. Hendrawan, A. D. K. Annas, and G. W. Wicaksono, "Klasifikasi penyakit padi berdasarkan citra daun menggunakan model terlatih resnet101," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 5, no. 6, pp. 1216–1222, 2021.
[20] E. Zidni and M. Akbar, "Klasifikasi Citra Makanan Khas Kota Pasuruan menggunakan Convolutional Neural Network," 2024.
[21] E. S. Budi, A. N. Chan, P. P. Alda, and M. A. F. Idris, "Optimasi Model Machine Learning untuk Klasifikasi dan Prediksi Citra Menggunakan Algoritma Convolutional Neural Network," *Resolusi: Rekayasa Teknik Informatika dan Informasi*, vol. 4, no. 5, pp. 502–509, 2024.
[22] I. Salehin and D. K. Kang, "A Review on Dropout Regularization Approaches for Deep Neural Networks within the Scholarly Domain," Jul. 01, 2023, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/electronics12143106.
[23] A. S. Riyadi, I. P. Wardhani, D. S. Widayati, and K. Kunci, "Klasifikasi Citra Anjing Dan Kucing Menggunakan Metode Convolutional Neural Network (CNN)," *Universitas Gunadarma Jl. Margonda Raya No*, vol. 5, no. 1, p. 12140, 2021.
[24] A. L. Tandung, M. Abduh, M. Arafah, and A. Halid, "Klasifikasi Objek Kapal Berbasis Deep Learning Untuk Maritime Surveillance," *EDUCATIONAL: Jurnal Inovasi Pendidikan & Pengajaran*, vol. 4, no. 4, pp. 360–386, 2024.
[25] S. S. S. Palakodati, V. R. R. Chirra, Y. Dasari, and S. Bulla, "Fresh and rotten fruits classification using CNN and transfer learning," *Revue d'Intelligence Artificielle*, vol. 34, no. 5, pp. 617–622, Oct. 2020, doi: 10.18280/ria.340512.
[26] D. Bharali, K. C. Bora, and R. Sarkar, "An Improved CNN Model for Identifying Tomato Leaf Diseases," *Transdiscipl J Eng Sci*, vol. 15, pp. 17–38, Jan. 2024, doi: 10.22545/2024/00253.
[27] M. Genç and F. Akar, "Detection of Lung Cancer Cells Using Deep Learning Methods," *Bitlis Eren Üniversitesi Fen Bilimleri Dergisi*, vol. 13, no. 2, pp. 445–459, Jun. 2024, doi: 10.17798/bitlisfen.1422869.
[28] E. Sentosa, D. I. Mulyana, A. F. Cahyana, and N. G. Pramuditasari, "Implementasi Image Classification Pada Batik Motif Bali Dengan Data Augmentation dan Convolutional Neural Network," *Jurnal Pendidikan Tambusai*, vol. 6, no. 1, pp. 1451–1463, 2022.
[29] G. Arther Sandag, J. Waworundeng Universitas Klabat, J. Arnold Mononutu, and A. -Minahasa Utara, "Identifikasi Foto Fashion Dengan Menggunakan Convolutional Neural Network (CNN) Identify Fashion Images Using Convolutional Neural Network (CNN)," *Cogito Smart Journal* |, vol. 7, no. 2, p. 2021.
[30] M. Raihan, R. Allaam, and A. T. Wibowo, "Klasifikasi Genus Tanaman Anggrek Menggunakan Metode Convolutional Neural Network (CNN)," *e-Prceeding Eng*, vol. 8, no. 2, pp. 3147–3179, 2021.
[31] M. S. Gumilang and D. Avianto, "Recognition Of Real-Time Handwritten Characters Using Convolutional Neural Network Architecture," *Jurnal Teknik Informatika (Jutif)*, vol. 4, no. 5, pp. 1143–1150, Oct. 2023, doi: 10.52436/1.jutif.2023.4.5.993.

[32] N. Saefulloh, J. Indra, and A. Ratna Juwita, "Implementasi Algoritma Convolutional Neural Network (CNN) Untuk Klasifikasi Kecacatan Pada Proses Welding di Perusahaan Manufacturing," *Technology and Science (BITS)*, vol. 6, no. 1, pp. 387–394, 2024, doi: 10.47065/bits.v6i1.5321.

[33] I. Bakti and M. Firdaus, "Klasifikasi File Gambar Hasil X-Ray Paru-Paru Dengan Arsitektur Convolution Neural Network (CNN)," *JIFOTECH (Journal of Information Technology*, vol. 3, no. 1, 2023.

[34] O. Fagbohungbe and L. Qian, "The Effect of Batch Normalization on Noise Resistant Property of Deep Learning Models," *IEEE Access*, vol. 10, pp. 127728–127741, 2022, doi: 10.1109/ACCESS.2022.3206958.

[35] A. Lutfhi, "The effect of layer batch normalization and droupout of CNN model performance on facial expression classification," *JOIV: International Journal on Informatics Visualization*, vol. 6, no. 2–2, pp. 481–488, 2022.

[36] R. O. Ogundokun, R. Maskeliunas, S. Misra, and R. Damaševičius, "Improved CNN based on batch normalization and adam optimizer," in *International Conference on Computational Science and Its Applications*, Springer, 2022, pp. 593–604.

[37] I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT Express*, vol. 6, no. 4, pp. 312–315, Dec. 2020, doi: 10.1016/j.icte.2020.04.010.

[38] G. P. H. P. Gusti, E. Haerani, F. Syafria, F. Yanto, and S. K. Gusti, "Implementasi Algoritma Convolutional Neural Network (Resnet-50) untuk Klasifikasi Kanker Kulit Benign dan Malignant," *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 4, no. 3, pp. 984–992, Jun. 2024, doi: 10.57152/malcom.v4i3.1398.

[39] C. Mahaputri, Y. Kristian, and E. Setyati, "Pengenalan Makanan Tradisional Indonesia Beserta Bahan-bahannya dengan Memanfaatkan DCNN Transfer Learning," *Journal of Intelligent System and Computation*, vol. 4, no. 2, pp. 61–68, Oct. 2022, doi: 10.52985/insyst.v4i2.252.