Implementation of Convolutional Neural Network Algorithm to Pest Detection in Caisim

2023 International Conference on Computer Science, Information Technology and Engineering (ICCoSITE) | 979-8-3503-2095-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCoSITE57641.2023.10127792

1st Cendekia Luthfieta Nazalia
Informatics Engineering
Institut Teknologi PLN
Jakarta, Indonesia
lulu1831036@itpln.ac.id

2nd Pritasari Palupiningsih
Informatics Engineering
Institut Teknologi PLN
Jakarta, Indonesia
pritasari@itpln.ac.id

3rd Budi Prayitno
Informatics Engineering
Institut Teknologi PLN
Jakarta, Indonesia
budiprayitno@itpln.ac.id

4th Yudhi Setyo Purwanto
Informatics Engineering
Institut Teknologi PLN
Jakarta, Indonesia
y.purwanto@itpln.ac.id

***Abstract*—High demand for caisim in Indonesia's main export commodity must be accompanied by a good planting process. The obstacle faced is that farmers are currently able to apply pesticides when the caisim plants have holes due to being eaten by pests. This control can be a good step to maximize the yield of caisim farming. However, many farmers have not implemented proper control of pests, one of which is farmers in Kebon Raya Dempo, South Sumatera, Indonesia. The obstacles faced such as not being able to detect pests correctly and provide pesticides with precision. Motivated by CNN's success in image classification, a learning-based approach has been carried out in this study to detect the presence of pests in caisim. The experimental results show differences in accuracy in each experiment with a dataset of 1000, consisting of 500 image data with pests and 500 without pests. The accuracy of the experiment A – CNN from Scratch is 48.33%, precision 1, recall 0.48, F1-score 0.65, experiment B – CNN from Scratch is 73.00% precision 1, recall 0.64, F1- score 0.78, experiment C–CNN from Scratch experiment is 92.00% precision 0.88, recall 0.96, F1-score 0.92. Of the 3 trials, experiment A – CNN from Scratch experienced underfitting, experiment B – CNN from Scratch overfitting, and the C – CNN experiment from Scratch can be used for pest detection in ciasim.**

Keywords— Deep Learning, CNN, Classification, Caisim Pest.

I. INTRODUCTION

The agricultural sector grew positively amid the weakening national economy due to the Covid-19 pandemic. The Central Statistics Agency (BPS) stated that the Gross Domestic Product (GDP) of agriculture grew by 2.59 percent year on year (yoy) in the fourth quarter of 2020 [1]. One of these agricultural products is Green mustard. Green mustard is one of the popular leaf vegetables in Indonesia. Another name is caisim (from the Cantonese name 菜心, choy sum, which literally means "vegetable heart") [2]. Caisim plants are known to have high economic value, considering that this vegetable is one of Indonesia's main export commodities [3]. Caisim has a relatively short harvest period of 21 days of planting and the open market is attracted to see this plant [4].

However, vegetables are foodstuffs that have perishable properties. A good handling from the planting to the distribution processes are hadled properly. One of the factors that can damage the caisim is the presence of pests in the planting process. In identifying plants affected by pests, there are special characteristics shown by plants. A feature that is very easy to recognize when a plant is affected by pests is if the plant body has a different symptom or sign caused by the those causative organisms, especially on the leaves [5].

One solution that can be applied is to create a model that can detect the presence of pests in caisim. Image processing is a system in which the process is carried out with input in the form of an image and the result (output) is also an image or information from an image [6]. Machine learning can process image data on the farm to get patterns. One of the tasks in machine learning is supervised learning where this type has a labeling stage in the training phase. Supervised learning itself consists of regression and classification. Classification has one model, i.e. the neural network model with deep learning method [7].

One method that can be applied to deep learning is the classification method with the Convolutional Neural Network (CNN) algorithm. CNN has several layers that extract information from the image and determine the classification of the image in the form of a classification score. Therefore, deep learning is widely used in machine learning because of its significant capabilities in modeling complex data [8].

The model that has been formed is then evaluated using the confusion matrix. A confusion matrix is the determination of the performance of an overfitting, underfitting, or usable classification model. This can be seen from the performance measurement parameters based on the level of accuracy, recall, precision, and F1 score [9].

The purpose of this study is to apply the CNN algorithm in classifying caisim pest image data, find out which CNN architecture has the highest performance, and implement it into a caisim pest detection system.

II. RELATED WORK

As a basis for conducting research that will be carried out, related theories are discussed. In the following we describe the theoretical basis of this work. discusses the detection of healthy leaves and leaves that are attacked by pests on cotton leaf. This study aims to assist farmers in overcoming leaves affected by pests using the CNN algorithm [10]. Discusses the pest detection model in maize plantations for more efficient inputs to the Integrated Pest Management (IPM) system, both to reduce the use of chemical pesticides and a more sustainable agricultural ecosystem [11]. Discusses a

comparison of several models to find the best model in the case of pests in rice plants [12]. Discusses some of these studies have also been deployed to applications such as mobile and websites. This system is designed for farmers so that it helps to reduce the complexity, time, and cost of diagnosing leaves of any disease [13]. Discusses the CNN algorithm is proven to have better accuracy compared to other methods, namely Support Vector Machine (SVM). The SVM has an accuracy of 87.65% while CNN has 89.27% [14].

The update from previous studies is that this research was carried out on caisim. Three architectures were created without adding available models in Keras consisting of Experiment A - CNN from Scratch, Experiment B - CNN from Scratch, and Experiment C - CNN from Scratch. The purpose of this experiment is to find the best architecture that can be applied to the website system to detect caisim pests.

III. Research Method

The workflow of the research carried out in this study can be seen in Figure 1 below.

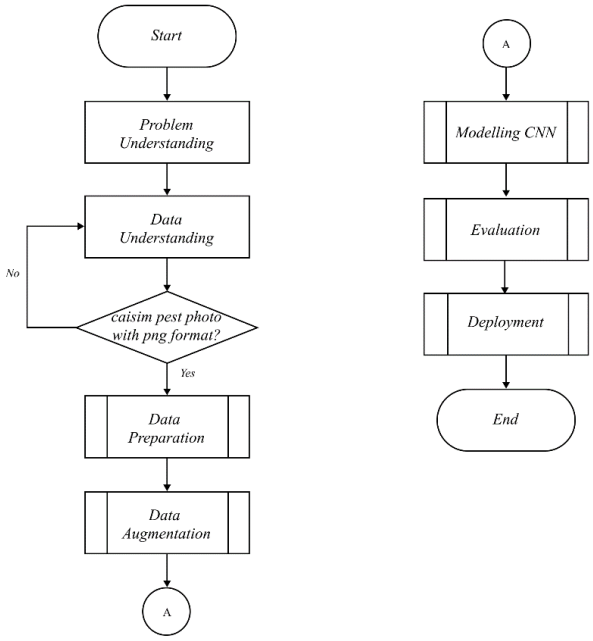

Fig. 1. Research Workflow

A. Problem Understanding

At this stage it is necessary to understanding the urgency of the problem, the substance of the class division of images, and their needs. Determine the goal for class division and make strategic planning. The understanding of the problem in this study refers to the substance of making this classification, in which the resulting classification results can help farmers in applying fertilizer to be more effective in the future as well as early warning if plants are attacked by pests.

B. Data Understanding

The phase of collecting datasets and exploring the data so you can identify which data to use. This phase tries to identify issues related to data quality. The data for this thesis uses extracted primary data. Dataset or images in .png format were successfully collected with a total of 1000. The image data model is in the form of images of caisim leaves and caisim leaves with pests. Data from the Dempo Botanical Gardens on May 14, 2022. The dataset can be accessed on kaggle [15].

C. Data Preparation

This stage is the pre-processing of the data. This stage includes resizing the image, normalizing the image, and dividing the data train, data validation, and data testing. In the pre-processing stage of this data, there are several steps that must be taken to present the data so that it is ready to be processed in the next stage, namely the data augmentation stage, the stages are as follows:

1. Pre-processed Data: The data is divided into 3 types, namely Training, Validation, and Test. The training data will be given a larger composition so that there are more learning models. The function of the training data will be used to train the model and data validation for the training process which will produce a loss function which can be considered whether the data is overfitting or underfitting, and finally data testing is used when testing the model as a simulation of using the model in the real world. Test data should never be seen by a previously trained model
2. Resizing Image: resizing the image to 128x128. This is done because the image data from the Pest dataset has various orientations, angles of view, and sizes that are too big or small, so resizing is needed on the existing dataset with the aim that the data have the same size so as to speed up the training process.

D. Data Augmentation

The more data and variety, the better the machine will learn. So that the more training data, the higher the accuracy of the model, but conversely, the less data, the more problems will occur, one of which is overfitting. Augmentation is a way to overcome this by modifying data such as rotation, zoom, horizontal flip, width shift, height shift, and so on.

E. Modelling CNN

The modeling process of pest detection systems that exist in caisim was done by utilizing the image of caisim. In General, CNN consists of several layers including the convolutional layer, pooling layer, dropout layer, flatten layer, and fully connected layer. The general description of modeling using the CNN algorithm is shown in Figure 2 below.

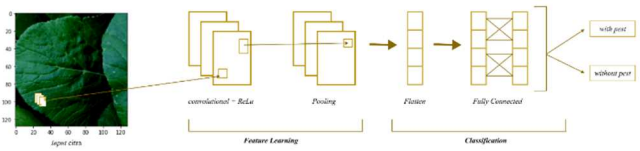

Fig. 2. CNN System Modeling

1. Convolutional Layer : These layers will calculate the output of the neurons connected to the local area in the input, the point between their weights, and the small area connected to the respective input volume. In each layer, some hyperparameters i.e. filter size, stride, and padding can be changed. Stride is a parameter that determines how much the filter shifts. If the stride is getting smaller, more

information will likely be obtained from input, but it requires more computation when compared to a large stride. Padding is an additional layer that can be added to the image boundary by increasing the pixel size by a certain value around the input data so that the receptive field is not too small hence not much information is lost. This value is usually zero so it is called zero padding. This convolution process produces an output that will be used as input for the next convolution layer. ReLU is an operation to introduce non-linearity and improve representations and models. The ReLU activation function will eliminate the vanishing gradient by applying the function f(x) = max (0,x) or the element activation will be carried out when it is at the threshold 0.

2. Pooling Layer : The pooling layer is a layer that reduces the dimensions of the feature map which can speed up computations because fewer parameters need to be updated and can also overcome overfitting. The value taken in pooling is max pooling, which is the highest value in the filter area
3. Dropout Layer : Dropout serves to reduce the complexity of the model that has been built by building unused neural networks. The model size will be the same, nothing has changed.
4. Flatten Layer : Reshape the feature map into a vector so that we can use it as input from the fully-connected layer.
5. Fully Connected Layer : This layer is fully connected. The resulting feature map is still in the form of a filter matrix, so it must be flattened so that it can be used as input from the fully connected layer. This layer has every neuron that will connect to all the numbers and volumes.

F. Evaluation

From the results of the modeling with the best optimization, then a test is carried out to classify the test data which is also an image of caisim both with pests and without pests. The confusion in the classification can be seen through the confusion matrix and the accuracy obtained from the application of the classification method with the Convolutional Neural Network algorithm.

Confusion matrix is a matrix that represents the classification results in a dataset. The Confusion Matrix is made in the form of a Table. This Table is often used to describe the performance of a classification model. This Table consists of rows and columns as many as the number of classes that show the value of false positives (FP) which is negative data but is detected as positive data, false negatives (FN) is positive data that is detected as negative data, true positives (TP) is positive data detected as true, and true negatives (TN) is the number of negative data detected as true. The equation of the calculation of accuracy of confusion matrix is at equation (1).

$$Accuracy = \frac{TP+TNP}{TP+TN+FP+FN} \; x\; 100\% \qquad (1)$$

Then measure the precision value using the following equation. Intuitive precision is the classifier's ability not to label a positive as a negative sample. The equation of the calculation of precision of confusion matrix is at equation 2.

$$Precision = \frac{TP}{TP+FP} \qquad (2)$$

Furthermore, to measure the recall value using the following equation. The intuitive recall is the classifier's ability to find all positive samples. The equation of the calculation of recall of confusion matrix is at equation 3.

$$Recall = \frac{TP}{TP+FN} \qquad (3)$$

Then, to measure the value of the F1-Score using the following equation 4.

$$F1\text{-}Score = 2\; x\; \frac{precision\; x\; recall}{precision\; +\; recall} \qquad (4)$$

G. Deployment

In this research, the deployment stages are carried out on the website using the Flask framework which is then displayed in an interface is shown in Figure 3 below.

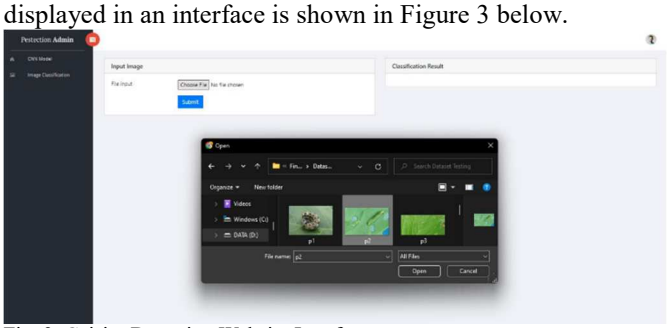

Fig. 3. Caisim Detection Website Interface

IV. Results and Discussion

The results and discussion of this study is a test system for pest detection on caisim using the classification method with the CNN algorithm through several stages, as follows:

A. Preprocessing Data

The image data that has been collected will be pre-processed. The amount of data processed is 1000 images of caisim both with pests and without pests. Those 1000 images are divided into training, validation, and test data. Where training and validation will be carried out during the training and testing processes for the trial phase. This is where the data sharing stage is carried out differently at each experiment as shown in table I below.

TABLE I. Data Partition

No	Sample	Training	Validation	Test
1	Experiment A	40%	30%	30%
2	Experiment B	80%	10%	10%
3	Experiment C	90%	5%	5%

Based on table I the result of Experiment A has data sharing with the composition of training data almost the same

as validation and test. However, for experiments B and C, data partitioning was carried out by providing more data on the training data.

B. Convolutional Neural Network Modeling

Convolutional Neural Network model was developed from several stages of iteration between the convolution layer and the pooling layer which ends with a fully connected layer. The architectural model development was carried out 3 times with different numbers of layers and hyperparameter as summarized in Figures 4 to 9 below.

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_16 (Conv2D)           (None, 126, 126, 64)      1792
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 63, 63, 64)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 61, 61, 8)         4616
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 30, 30, 8)         0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 28, 28, 4)         292
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 14, 14, 4)         0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 12, 12, 96)        3552
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 6, 6, 96)          0
_________________________________________________________________
flatten_4 (Flatten)          (None, 3456)              0
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 6914
=================================================================
Total params: 17,166
Trainable params: 17,166
Non-trainable params: 0
_________________________________________________________________
```

Fig. 4. Experiment A Model Summary – CNN from Scratch

Figure 4 has 4 convolution stages (shown in red marks) and 4 polling stages (shown in green marks) and ends with a fully connected layer (shown in yellow marks).

```
model_name='Cendekia'
# print("Building model with", base_model)

model = tf.keras.Sequential([
        # Note the input shape is the desired size of the image 128x128 with 3 bytes color
        # This is the first convolution
        tf.keras.layers.Conv2D(filters=64, input_shape=img_shape, kernel_size=(3,3), activation='relu'),
        tf.keras.layers.MaxPooling2D(pool_size=(2,2)),

        # This is the second convolution
        #base_model,
        tf.keras.layers.Conv2D(filters=4, kernel_size=(3,3),activation='relu'),
        tf.keras.layers.MaxPooling2D(pool_size=(2,2)),

        # The third convolution
        tf.keras.layers.Conv2D(filters=8, kernel_size=(3,3), activation='relu'),
        tf.keras.layers.MaxPooling2D(pool_size=(2,2)),

        # The fourth convolution
        tf.keras.layers.Conv2D(filters=81, kernel_size=(3,3), activation='relu'),
        tf.keras.layers.MaxPooling2D(pool_size=(2,2)),

        # This is fullyconnected layer
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(class_count, activation='softmax')
])

model.compile(optimizer=RMSprop(lr=0.1), loss='categorical_crossentropy', metrics='accuracy')
```

Fig. 5. Hyperparameter A Model Summary – CNN from Scratch

Figure 5 shows the layers consisting of 4 convolution layers and 1 fully connected layer. The hyperparameter of this experiment is to make 4 different types of filters from each convolution layer, namely 64, 4, 9, 81 (shown in green marks) and when compiling this model a hyperparameter experiment was carried out using the RMSprop (shown in blue marks) optimizer with a learning rate of 0.1 (shown in red marks).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 16384)             0
_________________________________________________________________
dense (Dense)                (None, 2)                 32770
=================================================================
Total params: 52,162
Trainable params: 52,162
Non-trainable params: 0
_________________________________________________________________
```

Fig. 6. Experiment B Model Summary – CNN from Scratch

Figure 6 has 2 convolution stages (shown in red marks) and 2 polling stages (shown in green marks) and ends with a fully connected layer(shown in yellow marks).

```
model_name='Cendekia'
# print("Building model with", base_model)

model = tf.keras.Sequential([
        # Note the input shape is the desired size of the image 128x128 with 3 bytes color
        # This is the first convolution
        tf.keras.layers.Conv2D(filters=32, padding='same',input_shape=img_shape, kernel_size=(3,3), activation='relu',
                               strides=1),
        tf.keras.layers.MaxPooling2D(pool_size=(2,2)),

        # This is the second convolution
        tf.keras.layers.Conv2D(filters=64, padding='same', kernel_size=(3,3), activation='relu', strides=2),
        tf.keras.layers.MaxPooling2D(pool_size=(2,2), strides=(2,2)),

        # This is fullyconnected layer
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(class_count, activation='softmax')
])

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=.001), loss='categorical_crossentropy', metrics='accuracy')
```

Fig. 7. Hyperparameter B Model Summary – CNN from Scratch

Figure 7 shows the layers consisting of 2 convolution layers and 1 fully connected layer. The hyperparameter of this experiment is to make 2 different types of filters from each convolution layer, namely 32, and 64 (shown in green marks). Besides that, padding same is also added so as not to change the pixel size during the training process (shown in orange marks). Strides are also used here to make the learning process more detailed in every pixel of the image (shown in purple marks). and when compiling this model a hyperparameter experiment was carried out using the Adam (shown in blue marks) optimizer with a learning rate of 0.001 (shown in red marks).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 128, 128, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 64, 64, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 131072)            0
_________________________________________________________________
dense (Dense)                (None, 2)                 262146
=================================================================
Total params: 263,042
Trainable params: 263,042
Non-trainable params: 0
_________________________________________________________________
```

Fig. 8. Experiment C Model Summary – CNN from Scratch

Figure 8 has 1 convolution step (shown in red marks) and 1 polling step (shown in green marks) plus a dropout layer (shown in purple marks) to reduce unused layers and ends with a fully connected layer (shown in yellow marks).In this model also added Early Stopping and Model Checkpoint. Where Early Stopping is used to stop model training and Model Checkpoint is used to save the model with the best results. The use of these two callbacks is to prevent overfitting from occurring in the model we created.

```
model_name='Cendekia'
# print("Building model with", base_model)

model = tf.keras.Sequential([
        # Note the input shape is the desired size of the image 128x128 with 3 bytes color
        # This is the first convolution
        tf.keras.layers.Conv2D(filters=32, padding='same',input_shape=img_shape, kernel_size=(3,3), activation='relu',
                               strides=1),
        tf.keras.layers.MaxPooling2D(pool_size=(2,2)),

        # This is dropout Layer
        tf.keras.layers.Dropout(rate=0.4),

        # This is fullyconnected layer
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(class_count, activation='softmax')
])

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=.001), loss='categorical_crossentropy', metrics='accuracy')
```

Fig. 9. Hyperparameter B Model Summary – CNN from Scratch

Figure 9 shows the layers consisting of 1 convolution layer, 1 dropout layer and 1 fully connected layer. The hyperparameter of this experiment is to make 1 types of filters namely 32 (shown in green marks). Besides that, padding same is also added so as not to change the pixel size during the training process (shown in orange marks). Strides are also used here to make the learning process more detailed in every pixel of the image (shown in purple marks). The dropout layer is added to reduce unused features in order to maximize the resulting model (shown in yellow marks) and when compiling this model a hyperparameter experiment was carried out using the Adam (shown in blue marks) optimizer with a learning rate of 0.001 (shown in red marks).

C. *CNN Modeling Graph*

The next stage is the application of the model formed into the training data and validation data with several selections of the number of epochs at the time of starting to converge. Each epoch goes through different epochs. The graphic image shows the level of accuracy and loss function of the training and validation data from the first epochs' journey to the last one. A good loss function is a graph that slowly decreasing but with increasing accuracy. The results of the training process for each model are shown in Figures 7 to 9 below.

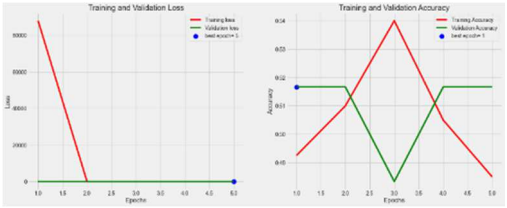

Fig. 10. Results of the Experiment A Process – CNN From Scratch

Figure 10 shows the results of the training process with 5 epochs (as shown in x axis) with not too good results where the loss function (as shown in graph "training and validation loss") decreases too sharply in both the training(as shown in red line) and validation data (as shown in green line) and accuracy did not show an increase (as shown in graph "training and validation accuracy"), even the accuracy of the training data (as shown in red line) decreased and the accuracy of the validation data stagnated (as shown in green line). So overall this model shows that underfitting is proven with an accuracy of 48.33%.

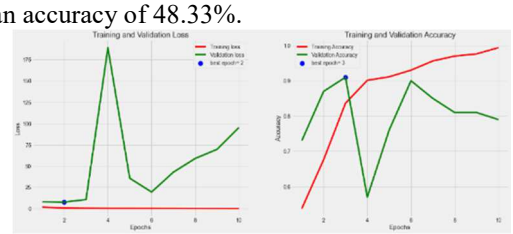

Fig. 11. Results of the Experiment B Process – CNN From Scratch

Figure 11 In the second model to improve on the shortcomings of the A-CNN From Scratch model, by adding 10 epochs (as shown in x axis), and simplifying the model layers according to the predicted output. Results of the training process shows better progress where accuracy (as shown in graph "training and validation accuracy") begins to increase, both increasing training data (as shown in red line) and data validation which is fluctuating but slowly increasing (as shown in green line). so that the accuracy obtained is 78%. However, the loss function has not decreased (as shown in graph "training and validation loss") which can be seen from the increasing validation data (as shown in green line) and stagnant training data (as shown in red line), so this model still cannot be used because it can only predict the data in the dataset, but if the data are given from outside, it becomes unable to predict the data (overfitting).

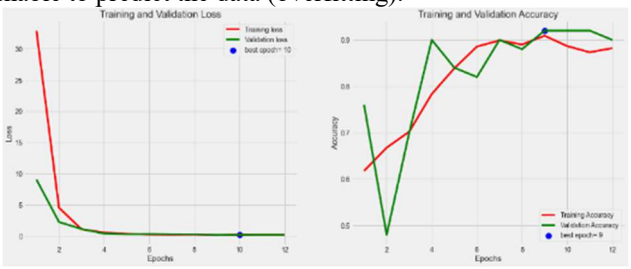

Fig. 12. Results of the Experiment C Process – CNN From Scratch

Figure 12 to overcome overfitting, several preventive measures were taken by increasing the number of epochs by 50, and also using callbacks, namely Early Stopping and Model Checkpoints. because it uses a callback, the epoch only stops at 12(as shown in x axis), which indicates the training process has reached the best point. At the end the result shows a good loss function graph (as shown in graph "training and validation loss") with line derivation from both training data (as shown in red line) and validation data (as shown in green line). Likewise, along with the graph, the accuracy is slowly increasing (as shown in graph "training and validation accuracy") both for training data (as shown in red line) and validation data (as shown in green line) both showed an increase. So that the results are 92%. This model is the most apropriate for this study.

D. *Classification*

The model that has been obtained is then tested into the test data. The testing process in this study uses performance evaluation called the Confusion Matrix method. The summary of this experiment can be found in table II below.

TABLE II. CONFUSION MATRIX'S EVALUATION SUMMARY

No	Percobaan	*Accuracy*	*Precision*	*Recall*	*F1-score*	Result
1	Experiment A – CNN *from Scratch*	48,33%	1	0.48	0.65	*underfitting*
2	Experiment *B* – CNN *from Scratch*	73,00%	1	0.64	0.78	*overfitting*
3	Experiment *C* – CNN *from Scratch*	92,00%	0.88	0.96	0.92	*good*

Calculations in the table II for accuracy calculations refer to equation 1, precision calculations refer to equation 2, recall calculations refer to equation 3, and F1-score calculations refer to equation 4. The results of the three experiments. The highest accuracy was in the third

experiment and the lowest accuracy was in the first experiment. Thus, only the third experiment had an architecture that could be used properly to detect pests in caisim

V. CONCLUSION

Based on the research that has been done, the parameters that influence the results of a model include : Data partition for training data, validation data and test data. To get maximum results, the number of training data must be greater than the validation and test data. The complexity of the layers in the CNN algorithm is also adjusted to how many output classifications it has. If you classify a few classes, the layers used don't need to be too complex. But if you classify a lot of classes, the layers can adjust. These methods can be used to avoid underfitting models.

If you have done some of the methods above and then the accuracy is good but cannot be used to predict data that is outside the dataset then this condition is called overfitting. The solution for this condition is to add a callback. There are 2 callbacks that can be used, namely the first is Early stopping to stop the training process when the validation loss drops drastically. Besides that, there is also a Model checkpoints where this callback will save the model with the best accuracy during the training process.

Apart from that, we can also change the hyperparameters contained in each layer in the CNN, including the smaller the filter value used, the more detailed the training process will be in each pixel. Padding same is also added so as not to drastically change the pixel size during the training process. Adam is the best optimizer that can be used in the case of mustard greens. the smaller the learning rate, the better the accuracy. similarly with strides filter 1 will make the training better.

So that the best results are obtained for the classification model with Convolutional Neural Network algorithm, starting from data collection and class division, image processing, data augmentation, convolutional, max pooling, flatten, relu, dropout, and dense models. With the model doing 50 epochs of training but already finding the best model in 12 epochs. From the training process, a model with an accuracy of 92% is obtained. Experiment C – CNN from Scratch with this accuracy value is the best result among experiments A to C. The results obtained from validation using the confusion matrix get an accuracy value of 92%. The process is done by implementing the C - CNN from Scratch experiment on the caisim pest detection system website using the Flask framework.

It is recommended for further development, the next research can apply CNN architecture with different hyperparameters or to implement different classification models such as Decision Tree and SVM. In addition, this study uses a dataset with a total of 1000 images. For further research, it is possible to take a larger number of datasets to get better accuracy results and avoid overfitting and underfitting.

REFERENCES

[1] C. Riyanto, I. Pusparani, M. Hardianti, N. Sari, D. Dahmhudi, and U. K. Indonesia, "Pengaruh Pembiayaan UMKM Terhadap Peningkatan PDB Sektor Pertanian Pada Masa Pandemi Covid-19," vol. 19, pp. 1–14, 2021.

[2] I. Nugraha, S. Isnaeni, and A. Rosmala, "GROWTH RESPONSE AND PRODUCTION OF CAISIM (Brassica juncea L.) ON DIFFERENT POC TYPES AND CONCENTRATIONS," vol. 5, pp. 12–22, 2021.

[3] M. Zailani, R. A. Kuswardani, and E. L. Panggabean, "Growth Response and Crop Production (Brassica Juncea L.) Against Watering Time Interval at Various Hydroponics Media," *Budapest Int. Res. Exact Sci. J.*, vol. 1, no. 1, pp. 9–22, 2019, doi: 10.33258/birex.v1i1.131.

[4] J. Akhatar *et al.*, "Genome wide association analyses to understand genetic basis of flowering and plant height under three levels of nitrogen application in Brassica juncea (L.) Czern & Coss," *Sci. Rep.*, vol. 11, no. 1, pp. 1–14, 2021, doi: 10.1038/s41598-021-83689-w.

[5] T. Yudiarti, *Ilmu Penyakit Tumbuhan Edisi 2*. Plantaxia/Graha Ilmu, 2018.

[6] D. H. Xia *et al.*, "Review-material degradation assessed by digital image processing: Fundamentals, progresses, and challenges," *J. Mater. Sci. Technol.*, vol. 53, pp. 146–162, 2020, doi: 10.1016/j.jmst.2020.04.033.

[7] T. Nakaura, T. Higaki, K. Awai, O. Ikeda, and Y. Yamashita, "A primer for understanding radiology articles about machine learning and deep learning," *Diagn. Interv. Imaging*, vol. 101, no. 12, pp. 765–770, 2020, doi: 10.1016/j.diii.2020.10.001.

[8] L. Alzubaidi *et al.*, *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*, vol. 8, no. 1. Springer International Publishing, 2021.

[9] M. Z. Alom *et al.*, "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches," 2018.

[10] M. Azath, M. Zekiwos, and A. Bruck, "Deep Learning-Based Image Processing for Cotton Leaf Disease and Pest Diagnosis," *J. Electr. Comput. Eng.*, vol. 2021, 2021, doi: 10.1155/2021/9981437.

[11] W. S. R. Souza, A. N. Alves, and Di. L. Borges, "A deep learning model for recognition of pest insects in maize plantations," *Conf. Proc. - IEEE Int. Conf. Syst. Man Cybern.*, vol. 2019-Octob, pp. 2285–2290, 2019, doi: 10.1109/SMC.2019.8914428.

[12] C. R. Rahman *et al.*, "Identification and recognition of rice diseases and pests using convolutional neural networks," *Biosyst. Eng.*, vol. 194, pp. 112–120, 2020, doi: 10.1016/j.biosystemseng.2020.03.020.

[13] M. E. Karar, F. Alsunaydi, S. Albusaymi, and S. Alotaibi, "A new mobile application of agricultural pests recognition using deep learning in cloud computing system," *Alexandria Eng. J.*, vol. 60, no. 5, pp. 4423–4432, 2021, doi: 10.1016/j.aej.2021.03.009.

[14] L. G. Tian, C. Liu, Y. Liu, M. Li, J. Y. Zhang, and H. L. Duan, "Research on plant diseases and insect pests identification based on CNN," *IOP Conf. Ser. Earth Environ. Sci.*, vol. 594, no. 1, pp. 0–6, 2020, doi: 10.1088/1755-1315/594/1/012009.

[15] C. L. Nazalia, "Dataset Caisim," *kaggle*, 2023. https://www.kaggle.com/datasets/cendekialuthfietanz/caisim (accessed Jan. 05, 2023).