CNN-Based Approach to Rice Plant Disease Classification: Overfitting Prevention Strategies

Sutthirak Sookkhasem[1], Petcharat Rungwachira[1] and Sakawkarn Piyawitwanich[1]*

[1]Department of Information System, Faculty of Business Administration and Information Technology, Rajamangala University of Technology Tawan-ok, Chakrabongse Bhuvanarth Campus, Thailand

***Corresponding Author E-mail**: sakawkarn_pi@rmutto.ac.th
Received: 27/07/24, **Revised**: 27/08/24, **Accepted**: 28/08/24

Abstract

Rice plant diseases can be an interference to the production of cropping rice plants. The timely and precise detection and classification of the disease is the key to reduce the potential of deprived output. Convolutional Neural Networks (CNNs) have shown a great potential in image recognition and classification, including rice plant disease classification. Nevertheless, CNNs are likely to be overfitting. This research paper proposes a use of combination of overfitting prevention techniques for CNN-based approach to rice plant disease classification. The techniques used are data augmentation, max pooling with stride, dropout and early stopping. The CNN model with these overfitting avoidance strategies is trained to classify the disease of the rice plant leaves and resulted in a prediction accuracy of 0.93. We conclude that this CNN-based architecture is considered to be effective and reliable for rice plant disease classification.

Keywords: Rice Diseases, Machine Learning, Convolutional Neural Network

1. Introduction

Rice is one of the most consumed crops in Thailand. We harvest millions of tons of rice every year. The problem comes that rice plant disease can impede the significant growth of the plant. It may yield in the loss of blood, sweat and tears of our farmers. In spite of the fact that we can indeed detect the anomalies of the rice plant disease by manual inspection, this process is time-consuming, subjective and might lead to human-error mistakes.

Convolutional Neural Network or CNN is an artificial neural network which is efficient in image recognition and classification. CNN has an ability to find the image patterns so it can identify the objects and labels. And this could be the answer for rice plant disease automated detection without prejudices. However, CNN is inclined to overfitting. Overfitting is when the model memorizes all the training data and achieves terrible performance on the unseen data. This research is to explore the CNN-based approach to rice plant disease classification that can help relieve the overfitting problem. We propose a CNN model with overfitting prevention strategies to rice plant disease classification. The noteworthy strategies are data augmentation, stride hyperparameter, dropout and early stopping. The details of this paper include the proposed CNN model, the methodology for training and testing, the results on rice plant disease classification, and the discussion of the overfitting prevention strategies and strengths and weaknesses of the model.

2. Literature Review

As a matter of fact, there are many other approaches to image classification specifically on rice plant disease. Kawcher Ahmed et al. [1] presented the techniques including Logistic Regression, K-Nearest Neighbor, Decision Tree, and Naive Bayes to detect leaf smut, bacterial leaf blight and brown spot diseases on rice plants. Daniya and Vigneshwari [2] extracted more features from the data by data pre-processing method and inputted them into K-Nearest Neighbors, Neural Network, Multiclass SVM and Naive Bayesian to classify the rice plant whether or not it carries a disease. However, Convolutional Neural Network is a neural network that is designed especially for image recognition and classification. Many researchers have successfully used CNN to analyze rice plant disease. Eusebio L. Mique et al. [3] implemented a CNN model for rice pest and disease detection and achieved an accuracy of 90.9%. Runling Wang [4] constructed convolutional neural networks to develop an agriculture pest and disease recognition system. Kosamkar et al. [5] proposed a CNN model with five layers for disease detection of the plant leaf images.

Despite their success, CNNs are prone to overfitting problems. A research paper on an introduction of convolutional neural networks by Keiron O'Shea and Ryan Nash [6] stated that overfitting is when a network is unable to learn effectively and impaired the ability to pinpoint generalized features for both training and testing datasets. Santos et al. [7] presented methods used to avoid overfitting in CNN for example data augmentation, dropout and label regularization. Another research paper by Thanapol et al. [8] suggested using data augmentation, batch normalization and dropout techniques to mitigate overfitting problems in CNN. This study found that using 30 epochs with width and height shift data augmentation and dropout techniques yields good outcomes for avoiding overfitting problems for CIFAR-10 dataset. Ferro et al. [9] also offered methods for preventing overfitting such as early stopping, training

model over a predefined number of epochs, stop when the loss function update becomes small, dropout etc. Building upon these foundations, our research examined a CNN-based architecture for rice plant disease classification that has a combination of data augmentation, max pooling with stride technique, dropout and early stopping.

3. Methodology

3.1 Dataset

The dataset used in this research is Rice Plant Diseases dataset [10]. The dataset is a collection of 4684 images and its labels. There are three diseases related to the leaves of rice plants which are Bacterial Blight, Brown Spot, and Leaf Smut. The contribution of each disease can be seen in Table 1 below.

Table 1 Distribution of diseases found in rice plant leaves

Class Label (Disease)	Count
Bacterial Blight	1604
Brown Spot	1620
Leaf Smut	1460
Total	4684

The dataset is divided into two groups of 80% training data and 20% validation data. The data is preprocessed using Keras preprocessing class. The image size used in this study is 224 x 224. And the batch size is 32. The example images from the first batch look like the pictures shown below in Fig. 1.

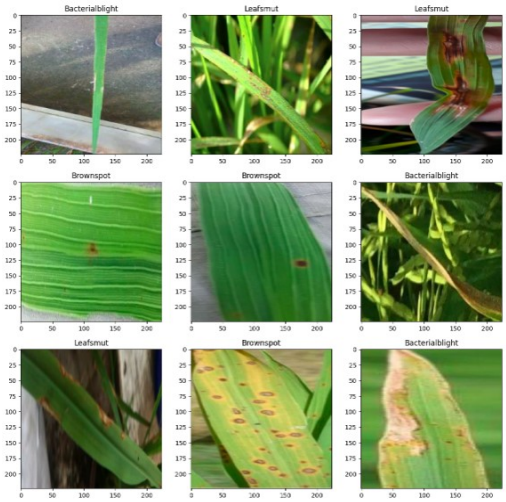

Fig. 1. The first 9 images from the first batch of training dataset

3.2 Convolutional Neural Networks

To avoid the overfitting problem that might occur when training the model, we applied the augmentation techniques on the training dataset using Keras preprocessing layers. We used RandomFlip, RandomRotation and RandomZoom layers. The first layer used in data augmentation is to randomly horizontal and vertical flip the images during training. Random rotation layer is to randomly rotate the images with the number of a chosen factor. We tried to use the float number 0.2 for the factor. To rotate the image by factor of 0.2 means that each image rotates randomly in the range from -20% * 2pi to 20% * 2pi. And the last layer, RandomZoom is to randomly zoom the images with the number of factors. We chose height_factor to be 0.2 which is zooming out by up to 20% (when width_factor is default to be None). We also added a Rescaling layer to the model to rescale the image data into numbers between 0 and 1 and change the input shape.

The Convolutional Neural Network is created using the Sequential model which is a linear stack of layers. We then added the Rescaling and data augmentation layers to the model. The first layer after preprocessing the input is a convolutional layer which applies the kernel or filter to each input image to create a matrix or a map of pixels for each one. The activation function used during convolutional operation is ReLU or Rectified Linear Unit. This activation function helps the model to learn complex patterns and relationships in the dataset. The second layer is a pooling layer which extracts features from the matrix and that results in dimensional reduction of the matrix. The hyperparameter stride is also used in the pooling layer to make it more efficient [11]. With stride, it will take the amount of given steps each time to help speed up the process. The stride value of 2, which is not too big or too small of a step, is then used in the pooling layer. The third layer is another convolutional layer and the fourth layer is another pooling layer with stride. Before the model starts to classify the output, the dropout layer is added to improve generalization of a performance and prevent the model from overfitting [12]. Dropout prevents the network from being too dependent on specific neurons by setting some of input units to zero. It helps the model to learn more about features that are better at predicting unseen data. If we have a large dataset, using a dropout rate of 0.5 or higher is effective [13]. However, the dataset we have is rather small so it is more appropriate to use a dropout rate lower than 0.5. It is also important to note that models with complex structure will largely benefit from using higher dropout rates, e.g. 0.5. Meanwhile, basic models can have some benefits as well but it is better with lower dropout rates, e.g. 0.2, to avoid model underfitting. The model we implemented is not too complex with a lot of layers and parameters or too simple so we decided to use the rates between 0.2 and 0.5 and settled with 0.3 after experimenting. The last layers are fully connected layers or dense layers which receive a flattened matrix from the last flatten layer to predict the label of the output. The ReLU activation function is then used in a dense layer. And lastly, Softmax activation function is used for an output label prediction in a last dense layer.

The architecture of convolutional neural network consists of:

- Rescaling the input to be 1./255 and defining image shape to be (224, 224, 3)
- Data augmentation layer with random flip for both horizontal and vertical, random rotation with factor of 0.2 and random zoom with height factor of 0.2
- Convolutional layer with filter = 32, kernel size = 3x3 and ReLU activation function
- Pooling layer with pool size = 2x2 and stride = 2
- Convolutional layer with filter = 16, kernel size = 3x3 and ReLU activation function
- Pooling layer with pool size = 2x2 and stride = 2
- Flatten layer
- Dropout layer with rate of 0.3
- Dense layer with output = 16 and ReLU activation function
- Dense layer with output = num_class which is 3 and Softmax activation function

3.3 Experimental Setup

The study is being done on a device with GPU T4 x 2 accelerator since it is the GPU recommended and designed for image processing and neural networks [14]. The model uses Keras SparseCategoricalCrossentropy loss function to compute cross entropy loss and Adam [15] algorithm for model optimization. The metrics used to evaluate by the model during training and testing is accuracy.

The model is set to run for 30 epochs. And to prevent the model from overfitting, the model is also required to monitor the validation loss. If the loss is not decreasing or increasing, the model will stop training if the loss increases after running for 2 epochs. This callback technique is called early stopping.

When the model completed training, we can see the evaluation metrics used to assess the performance of the model as seen below in fig. 2 and 3. And as we can see from fig. 2 and 3, the model training process stops at epoch 12.

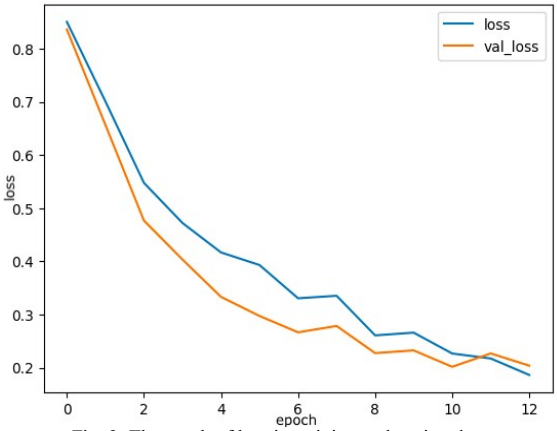

Fig. 2 The graph of loss in training and testing dataset

Fig. 2 shows that loss from training data set in blue line is still decreasing while loss from testing dataset in orange line is increasing during epoch 10 and 11. The model then stops training at epoch 12 to prevent overfitting.

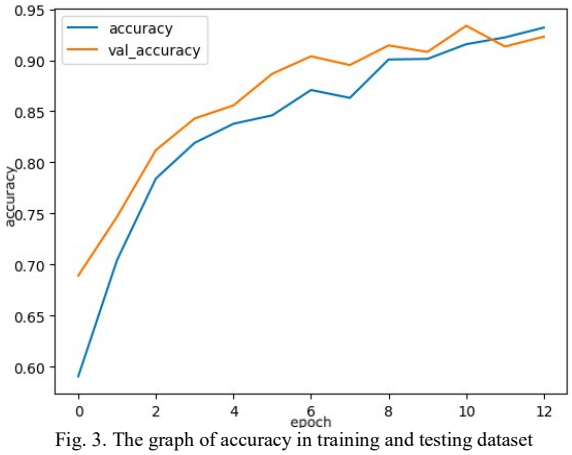

Fig. 3. The graph of accuracy in training and testing dataset

Fig. 3 shows a similar trend to fig. 2. The accuracy of training dataset in blue line is increasing while the accuracy of testing dataset in orange line is decreasing during epoch 10 and 11. The increasing trend of accuracy for training dataset is considered to be good. However, it is better to stop training the model before it memorizes all patterns and perfectly predicts nearly all images because that will result in the model to be overfitting.

Summarily, the criteria to make the training process stop at epoch 12, as seen in fig. 2 and fig. 3, is that the model is about to memorize all the data if the training does not stop. When the model knows all the data and labels, it is defined as overfitting. To prevent overfitting, the model should not continue training beyond this point. And with early stopping, it can stop at any epoch before the assigned number (which is 30 in our case) when the model decides it is becoming overtraining.

4. Results

The accuracy of the baseline method can achieve an accuracy of 0.98. Except the accuracy number shows that the baseline model seems to be remembering the patterns of data instead of learning. With the use of data augmentation, stride technique, dropout and early stopping in this study, we can prevent the model from overfitting and obtain an accuracy of 0.93.

After we train the model, we then give new data for the model to predict the label of each picture. The results can be seen in fig. 4. Fig. 4 shows the label from CNN model prediction and the correct label of the rice leaves. These 9 pictures show 8 out of 9 corrected predictions which is considered to be a proper model to predict new data.

5. Discussions

During the model training, the challenge we encountered is that the baseline model is too good at learning and memorizing the patterns. And that leads to the model being overfitting. The dataset is quite small and has insufficient variation. Hence, it results in the use

of data augmentation that provides an adequate variety of data to be used to train the model. The importance of using data augmentation is that it helps increase the dataset size and diversity. When we use RandomFlip, RandomRotation and RandomZoom layers on the dataset, we improve the generalization of the model. The model will now have more variations of the data. And to reach the main objective of this study, data augmentation helps to prevent the model from overfitting.

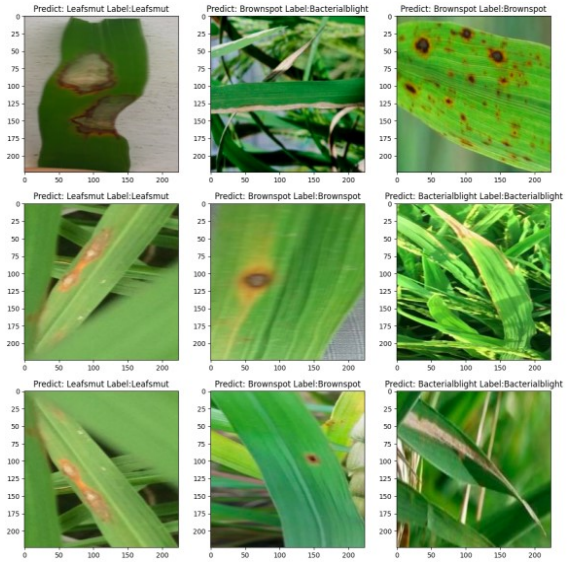

Fig. 4. The pictures of rice leaf with predicted label and actual label

When the traditional approach for CNN has been using max pooling to reduce the dimension of feature maps, we decided to use max pooling with stride to make the reduction process smoother without loss in accuracy [16]. It helps to prevent the model from overfitting by using stride since it can preserve important information and can also detect abnormalities. The model can solely focus on the most important features. The reduction of features also results in less computations and can lower the overfitting risk we might encounter when training.

Moreover, before the model reaches dense or fully connected layers, we added a dropout layer to prevent neurons from depending too much on some specific training data points and encourage the model to learn more of other features. It helps the model to discover features that have significant meanings. The model will adjust better and lead to generalized outcomes. That results in the model being less prone to overfitting.

And with early stopping when fitting the model to the dataset, we did not only monitor the accuracy of the model but also kept the validation in check. We stopped the training if the validation plateaus or starts to decline. Since we knew that the it is the right time before the performance starts to go into the wrong direction. As stated before, the model should stop training to prevent overfitting.

With the combined use of these techniques, we have improved the performance of an image classification model by minimizing the risk of model overfitting. Each technique relatively offers beneficial approach on how to handle overfitting when training CNN-based architecture.

Also consider that the results we have seen in the above section are acceptable. Although it does not always get the prediction right 100%, it can predict the label right most of the time about 93% which is considered a good prediction model. The strengths of this model is that it does not have complex or a lot of layers so the time used for training is small. It also provides a mixture of techniques to help with model overfitting. The weaknesses of this model is that it is very sensitive to data diversity. During the training of the model, it often memorized the training data and wrongly predicted the unseen data although data augmentation is applied. Therefore, we also combined other techniques to help prevent it from overfitting. We believe that the model can be trained more appropriately if we have more variations and amount of data. And since we only have a simple model and perform well, the model can be implemented in a more complicated way with additional layers and parameters to better extract the features.

One possible technique that could be used is regularization. L2 regularization or ridge regularization is one example of regularization techniques. L2 regularization works by calculating the penalty function when training and makes the weights to be smaller or is called weight decay. This technique causes the model to be less complex and lower the chance of overfitting. Another alternative technique would be using ensemble learning. Ensemble learning method uses predictions from multiple models. Therefore, it can also help reduce the dependency of one model for prediction and makes it less likely to model overfitting.

References

[1] K. Ahmed, T. R. Shahidi, S. M. Irfanul Alam and S. Momen, “Rice Leaf Disease Detection Using Machine Learning Techniques,” In *2019 Proc. International Conference on Sustainable Technologies for Industry 4.0 (STI),* Dhaka, Bangladesh, 2019, pp. 1-5, doi: 10.1109/STI47673.2019.9068096.

[2] T. Gayathri Devi and P. Neelamegam, “Image processing based rice plant leaves diseases in Thanjavur, Tamilnadu”, *Journal of Networks, Software Tools and Applications*, vol. 22, Nov., pp. 13415–13428, 2019.

[3] EL Mique Jr, TD Palaoag, “ Rice Pest and Disease Detection Using Convolutional Neural Network,” In *2018 Proc. International Conference on Information Science and System’04*, 2018, pp. 147–151. doi.org/10.1145/3209914.3209945.

[4] R. Wang, "Mechanism and Design of Agriculture Pest and Disease Recognition System Based on Convolutional Neural Network," In *2024 Proc. IEEE Eurasian Conference on Educational Innovation'07*, 2024, pp. 291-295, doi: 10.1109/ECEI60433.2024.10510856.

[5] P. K. Kosamkar, V. Y. Kulkarni, K. Mantri, S. Rudrawar, S. Salmpuria and N. Gadekar, "Leaf disease detection and recommendation of pesticides using convolution neural network," In *Proc. International Conference on Computing Communication Control and Automation*, 2019, pp. 1-4, doi: 10.1109/ECEI60433.2024.10510856.

[6] K. O'shea and R. Nash, "An introduction to convolutional neural networks," Current Issues in Neural and Evolutionary Computing, Dec, 2015. [Online serial].doi.org/10.48550/arXiv.1511.08458.

[7] C. F. G. D. Santos and J. P. Papa, " Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks," *Journal of Association for Computing Machinery*, vol. 54, no. 10s, Sep, 2022. [Online serial]. doi.org/10.48550/arXiv.2201.03299.

P. Thanapol, K. Lavangnananda, P. Bouvry, F. Pinel and F. Leprévost, "Reducing overfitting and improving generalization in training convolutional neural network (CNN) under limited sample sizes in image recognition," In *Proc. International Conference on Information Technology'05*, 2020, pp. 300-305, doi: 10.1109/InCIT50588.2020.9310787.

[8] M. Vilares Ferro, Y. Doval Mosquera, F.J. Ribadas Pena et al., " Early stopping by correlating online indicators in neural networks," *Neural Networks*, Mar, 2022. [Online serial]. doi.org/10.1016/j.neunet.2022.11.035.

[9] J. Dev, "Rice Plant diseases dataset," kaggle.com, May, 2024. [Online]. Available: https://www.kaggle.com/code/jay7080dev/rice-plant-disease-detection. [Accessed May. 21, 2024].

[10] R Riad, O Teboul, D Grangier, N Zeghidour, "Learning strides in convolutional neural networks," Current Issues in Machine Learning, pp. 1-17, Feb, 2022. [Online serial]. doi.org/10.48550/arXiv.2202.01653.

[11] S. Cai, Y. Shu, G. Chen, B. C. Ooi, W. Wang and M. Zhang, "Effective and efficient dropout for deep convolutional neural networks," https://arxiv.org/, Jul. 28, 2020. [Online]. doi.org/10.48550/arXiv.1904.03392.

[12] A. Pauls and J. A. Yoder, "Determining optimum drop-out rate for neural networks," In *2018 Proc. International Conference on the Midwest Instruction and Computing Symposium*, 2018, pp. 1-11.

[13] A. Ravikumar, H. Sriraman, PM. Sai Saketh, S. Lokesh, and A. Karanam, " Effect of neural network structure in accelerating performance and accuracy of a convolutional neural network with GPU/TPU for image analytics," *PeerJ Computer Science* 8:e909, Mar, 2022. [Online serial]. doi.org/10.7717/peerj-cs.909.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," https://arxiv.org/, Jan. 30, 2017. [Online]. doi.org/10.48550/arXiv.1412.6980/.

[15] J. T. S pringenberg, A. Dosovitskiy, T. Brox and M. Riedmiller, "Striving for simplicity: The all convolutional net," https://arxiv.org/, Apr. 13, 2015. [Online]. doi.org/10.48550/arXiv.1412.6806/.