Applied Computing and Informatics

Comparative analysis of convolutional neural network and DenseNet121 transfer learning in agriculture focusing on crop leaf disease identification

Received 15 March 2024
Revised 4 May 2024
Accepted 19 May 2024

Heru Agus Santoso, Brylian Fandhi Safsalta, Nanang Febrianto and Galuh Wilujeng Saraswati
Department of Informatics, Dian Nuswantoro University, Semarang, Indonesia, and
Su-Cheng Haw
Multimedia University – Cyberjaya Campus, Cyberjaya, Malaysia

Abstract

Purpose – Plant cultivation holds a pivotal role in agriculture, necessitating precise disease identification for the overall health of plants. This research conducts a comprehensive comparative analysis between two prominent deep learning algorithms, convolutional neural network (CNN) and DenseNet121, with the goal of enhancing disease identification in tomato plant leaves.

Design/methodology/approach – The dataset employed in this investigation is a fusion of primary data and publicly available data, covering 13 distinct disease labels and a total of 18,815 images for model training. The data pre-processing workflow prioritized activities such as normalizing pixel dimensions, implementing data augmentation and achieving dataset balance, which were subsequently followed by the modeling and testing phases.

Findings – Experimental findings elucidated the superior performance of the DenseNet121 model over the CNN model in disease classification on tomato leaves. The DenseNet121 model attained a training accuracy of 98.27%, a validation accuracy of 87.47% and average recall, precision and F1-score metrics of 87, 88 and 87%, respectively. The ultimate aim was to implement the optimal classifier for a mobile application, namely Tanamin.id, and, therefore, DenseNet121 was the preferred choice.

Originality/value – The integration of private and public data significantly contributes to determining the optimal method. The CNN method achieves a training accuracy of 90.41% and a validation accuracy of 83.33%, whereas the DenseNet121 method excels with a training accuracy of 98.27% and a validation accuracy of 87.47%. The DenseNet121 architecture, comprising 121 layers, a global average pooling (GAP) layer and a dropout layer, showcases its effectiveness. Leveraging categorical_crossentropy as the loss function and utilizing the stochastic gradien descent (SGD) Optimizer with a learning rate of 0.001 guides the course of the training process. The experimental results unequivocally demonstrate the superior performance of DenseNet121 over CNN.

Keywords Mobile application, Deep learning algorithm, CNN, DenseNet121,
Tomatoes leaf diseases identification

Paper type Full length article

1. Introduction

In agricultural research, recent studies have shown the effectiveness of deep learning in detecting plant diseases. Examples include successful disease identification in potato leaves

Applied Computing and Informatics
Emerald Publishing Limited
e-ISSN: 2210-8327
p-ISSN: 2634-1964
DOI 10.1108/ACI-03-2024-0132

[1], apple leaves [2] and various other crops [3]. These findings highlight the promising potential of deep learning for addressing agricultural challenges. In 2023, Asia's crucial tomato production reached 119,204,602 tons [4], facing challenges from pests and diseases. Given the changing agricultural landscape and climate impacts, effective pest and disease management is crucial for sustaining yields in tomato cultivation. Recognizing and identifying specific diseases, such as bacterial spot, early blight and powdery mildew, are pivotal steps in optimizing tomato plant health. This identification process involves observing distinctive characteristics manifesting on the leaves [5].

The integration of artificial intelligence, specifically machine learning for image recognition, known as computer vision, harnesses deep learning algorithms to identify objects within images. This technological advancement enables machines to replicate human-like recognition and perception of objects. A notable deep learning algorithm used for this purpose is the convolutional neural network (CNN), characterized by layered filters designed for object recognition in images [6]. In this study, the CNN algorithm served as a learning tool to identify diseases in tomato plants, utilizing leaf images affected by these agricultural concerns. However, CNN presents certain limitations, including high computational demands, a requirement for substantial labeled data and a considerable memory footprint. Additionally, interpretability poses challenges, and its effectiveness diminishes when applied to sequential data. CNNs tend to exhibit slower processing speeds, and the training phase is characterized by extended time requirements [7, 8]. We aim to demonstrate the efficacy of CNN in the identification of diseases affecting tomato leaves.

Transfer learning method, including DenseNet121, aims to transfer knowledge that has been formed in one domain to another but still is related. This method enables the maximization of previously acquired knowledge, particularly when confronted with limited datasets, thereby constituting a notable advantage of transfer learning. The application of transfer learning using DenseNet121 has demonstrated notable successes within the realm of crop disease detection research [9]. However, the literature review conducted thus far has provided limited information regarding the comparative performance analysis between CNN and DenseNet121 for the detection of diseases on tomato leaves in mobile application development. We conducted a comparative analysis, evaluating the performance of CNN against DenseNet121, with attention to hyperparameter configurations for both algorithms. Our primary objective is to determine the most effective algorithm for tomato disease identification, tailored for seamless integration into the mobile application Tanamin.id [10]. Therefore, the principal contribution of our study lies in the comprehensive comparison, accompanied by a nuanced analysis of why DenseNet121 outperformed CNN. Given the inherent complexity of deep learning models, we also delve into the aspect of interpretability, emphasizing its pivotal role within the domain of agricultural decision-making. The dataset utilized comprises a combination of publicly accessible image data obtained from Kaggle and private data collected from fields in Central Java, Indonesia. While Tanamin.id offers detection for eight types of plant diseases, our focus in this paper is specifically on tomatoes.

2. Related works

This section offers a comprehensive overview of various state-of-the-art methods employed for tomato disease identification, including hybrid CNN-support vector machine (SVM) approaches, pure CNN models and the utilization of the DenseNet121 algorithm. The research for tomato disease identification via leaves using hybrid CNN and SVM achieved accuracy rates of 92.6 %, utilizing 8,000 image data points for each disease type. This study involved data processing, wherein the red, green and blue (RGB) image was transformed into hue, saturation, value (HSV) and subsequently converted to grayscale [11]. In a subsequent investigation, the research for tomato crop disease identification was also conducted using

CNN, where they employed the Plant Village dataset, encompassing 39 classes of various plants. From this dataset, 10 disease labels related to tomato plants were extracted. By employing three convolution layers and two hidden dense layers, with a hyperparameter set to 0.001, the model achieved maximum accuracy after running 5,000 epochs [7]. Another study focused on the application of CNN to tomato plants, utilizing data from Kaggle. The dataset comprised a total of 32.535 instances. They adjusted the pixel size of the images to 224x224 and divided the data into 80% training data and 20% testing data. The data underwent processing with a CNN model architecture involving four layers of convolution and three dropouts, along with four max-pooling. The results obtained after training the model demonstrated an accuracy value of 95% [12]. CNN augmented with residual learning technique and bolstered by an attention mechanism exhibited promising performance, as reported by Ref. [13]. Their proposed methodology, validated on the Plant Village dataset, yielded an overall accuracy of 98%.

On the other hand, the study focused on implementing the DenseNet121 algorithm for the classification of multiple plant leaf diseases and achieved promising result [14]. The researchers leveraged data from the open dataset "PlantVillage" available on Kaggle. This dataset encompassed 35,779 images, categorized into 29 disease labels associated with seven types of plants. Utilizing DenseNet121, the average accuracy achieved following model training stood at an impressive 98.23%. Transfer learning has demonstrated its efficacy in disease identification for plants based on leaf conditions, predominantly relying on publicly available datasets [15]. However, in our investigation, given the intended implementation of the Tanamin.id application in tomato fields, we augmented our analysis with a dataset meticulously gathered by our team. This customization ensures greater alignment with the unique plant conditions prevalent in Indonesia. To attain our objective, we conducted a comparative analysis of the performance between CNN and DenseNet121 Transfer Learning, utilizing a composite training set comprising both publicly sourced datasets and those acquired first-hand from the tomato fields under our research purview. Another variant of DenseNet, namely DenseNet_Xception, was utilized for the identification of tomato diseases in Chinese cultivation. Following a series of experiments, the optimal performance was achieved, resulting in an accuracy of 97.10% [16].

The hybrid CNN-SVM and CNN with public datasets demonstrate notable strengths, including high accuracy achieved with relatively large datasets and ease of replication with comparison of results, respectively. However, the hybrid CNN-SVM approach requires pre-processing steps such as transformation to HSV and grayscale, adding computational overhead. Conversely, CNN models trained on public datasets may lack customization for specific regional plant varieties, potentially limiting their applicability. There are also several noteworthy gaps and considerations, such as interpretability and customization approaches for specific regional conditions. Hence, our work provided a comparative analysis between CNN and DenseNet121, utilizing a custom dataset tailored to regional tomato fields. By addressing these gaps and leveraging both publicly sourced and first-hand datasets, we aim to develop a robust and adaptable model for practical implementation in agricultural decision-making tools like Tanamin.id.

3. Research methodology

As previously discussed, given the proven effectiveness of CNN and DenseNet121 in identifying tomato diseases through leaf analysis, this study endeavors to ascertain the optimal performance of these two deep learning algorithms for integration into Tanmin.id. To achieve the objectives of this research, a series of methodological steps have been implemented. The specific order of these methods is illustrated in Figure 1 below.

Figure 1 illustrates the employed method, commencing with the collection of data through two distinct channels, i.e. public data from the Kaggle open dataset and private data sourced

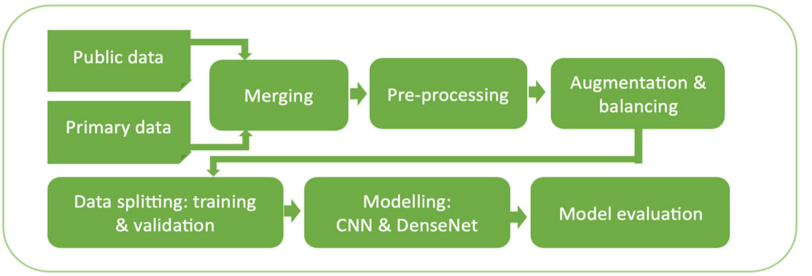

Figure 1.
Research method

Source(s): Created by authors

directly from the agricultural area. The gathered data undergo initial processing through pixel normalization of the images. Subsequently, the image dataset undergoes augmentation and balancing for each label pertaining to pests and diseases. The primary objective of this study is to identify the optimal model by comparing CNN and transfer learning using DenseNet121.

3.1 Data merging

During the data acquisition phase, our research involved a comprehensive exploration of existing online repositories or public dataset and on-site visits to local tomato farms for data collection. It is imperative that the data used for model training align closely with the intended learning objectives. Therefore, this study utilized two distinct data sources as elucidated below.

(1) *Public data.* The Kaggle repository, namely the PlantVillage Dataset [17] was used. The dataset encompasses ten distinct labels, as detailed in Table 1.

(2) *Primary data.* On-site visits were conducted at tomato farms situated in Central Java, Indonesia. Image data were acquired using two distinct devices: the Sony G7x Mark II pocket camera and a Huawei P30 Pro smartphone equipped with a 20MP wide lens.

No	Disease	Public data	Collected data
1	Bacterial_spot	2127 (86.9%)	320 (13.1%)
2	Early_blight	1000 (77.0%)	298 (23%)
3	Late_blight	1909 (94.8%)	105 (5.2%)
4	Leaf_Mold	952 (77.3%)	279 (22.7)
5	Septoria_leaf_spot	1771 (100%)	0%
6	Spider_mites_Two_spotted_spider_mite	1676 (100%)	0%
7	Target_Spot	1404 (100%)	0%
8	YellowLeaf__Curl_Virus	3209 (91.9%)	282 (8.1%)
9	mosaic_virus	373 (54.9%)	307 (45.1%)
10	Healthy	1591 (82.9%)	329 (17.1%)
11	Powdery Mildew	0%	244 (100%)
12	Magnesium Deficiency	0%	155 (100%)
13	Leaf Miner	0%	205 (100%)
	Total	16012 (85%)	2083 (15%)
		18,815 (100%)	

Table 1.
The acquired dataset

Source(s): Created by authors

Both devices operated in auto-setting mode, with a focus adjustment tailored to capture a singular leaf afflicted by the disease. The ensuing information was gathered through a comprehensive amalgamation of PlantVillage dataset as mentioned above and rigorous on-site data collection procedures.

3.2 Data pre-processing

Pre-processing of images is a pivotal stage conducted prior to model training, designed to optimize the data for subsequent analysis. The aim of image pre-processing is to elevate the quality of raw images within a given dataset [18]. In this study, we employed pixel normalization. We prioritized normalizing pixel dimensions for data without a 1:1 ratio. For images with uneven sizes, padding was applied using a pixel intensity of 0, representing a dark color. This approach aimed to preserve the shape of the leaf object within each image. The image normalization process can be elucidated through several steps. For instance, considering a leaf image with dimensions of 1,600 pixels in width and 2,112 pixels in height, we then determined the longest pixel dimension to establish a consistent 1:1 ratio. During this process, the remaining pixels underwent examination and were partitioned into two sections to maintain balance during pixel addition. Pixel addition was performed on the sides with the shortest pixel length. The final step involved resizing the image to dimensions of 400 x 400, a strategic measure designed to optimize computational efficiency.

3.3 Augmentation and dataset balancing

This involved the manipulation of image, facilitating the machine's understanding of diverse image variations within the dataset. Data augmentation encompassed the manipulation of the original labeled leaf dataset through the transformation process. The data augmentation in this study unfolds in two phases.

(1) Balancing the distribution of data across labels, achieved by subjecting the original dataset to multiple treatments for augmentation. The subsequent phase finalized the augmented data, rendering it ready for integration. We employed the ImageDataGenerator function, a Keras library for data augmentation. Five key input parameters, i.e. rotation_range, width_shift_range, height_shift_range, horizontal_flip and vertical_flip, were applied with parameter values 20, 0.15, 0.15, true and true, respectively. The augmented images were generated with 1,500 images per label.

(2) Validating and partitioning the dataset into training and validation sets. Parameters were configured to rescale the data and allocate 75% for training and 25% for validation. Data access was facilitated using the flow_from_directory method, referencing the folder in the storage results from the initial augmentation. Additionally, this process standardizes the image size to 224x224 pixels, and class_mode was specified as categorical.

3.4 Modeling with CNN

The CNN algorithm processes image inputs by conducting convolutional calculations through filters or kernels to extract distinctive features. These convolution kernels perform calculations on the original image, resulting in a new pixel with weighted values that unveil essential characteristics of the image [19]. The CNN method, as depicted in Figure 2, consists of two integral parts: feature extraction and feature classification. Feature extraction is executed through convolution and pooling layers, while feature classification employs fully connected layer computations, ultimately producing the CNN's final output. In this particular study, global average pooling (GAP) was employed as an alternative to Flatten layers.

3.5 Modeling with DenseNet

The Dense Convolutional Network, or DenseNet, represents a significant advancement in deep learning algorithms. This method employs a convolutional process that allows for more profound image analysis and a more efficient calculation process for image recognition. DenseNet121 incorporates pre-trained models trained with extensive datasets such as ImageNet and C, positioning itself as a transfer learning model. DenseNet121 operates by connecting each convolutional layer to the subsequent layer, transmitting the output feature map from the previous layer to the next input layer, a process referred to as a dense block [9]. The block diagram illustrating the transfer learning process of DenseNet121 for tomato disease identification is depicted in Figure 3. For DenseNet121, it comprises four dense blocks, with transition layers incorporating convolution and pooling situated between the dense blocks.

DenseNet offers diverse architectural versions distinguished by variations in the number of parameters they encompass. In this study, DenseNet121 was specifically chosen due to its feature of having the fewest parameters.

3.6 Experimental setup

We outline the experimental setup for developing and evaluating the classification model, which covers hardware, software and hyperparameters.

Hardware: MacBook Pro A2992/EMC 8407 M3 Pro Chip, featuring an 11-core CPU, 14-core GPU and 16-core Neural Engine, with 18 GB LPDDR5 memory and a 512 GB SSD.

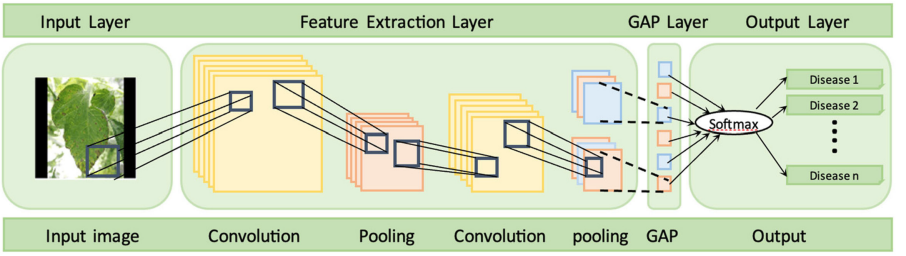

Figure 2.
Phase of CNN method for tomato disease identification

Source(s): Created by authors

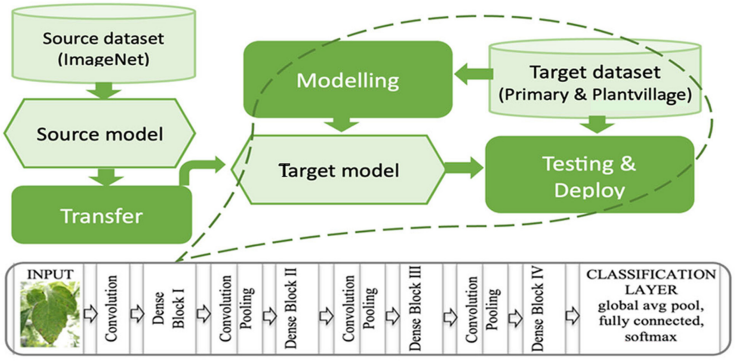

Figure 3.
DenseNet121 for tomato disease classification in detail

Source(s): Created by authors

Software: The system operates on MacOS Sonoma 14.4.1, utilizing the Anaconda3 Distribution Package with Python 3.11.7.

Hyperparameters: (1) CNN: number of layers:3 layers, Kernel Size: (3.3), Activation Function:ReLU, Input Shape: 224,224,3, Pooling Layers:Maxpool (2.2), Global Average Pooling (GAP), Dropout:0.3, Number of Fully Connected Layers:2, Output Layer: Softmax, (2) DenseNet12: Model Compilation: include_top:false, weights:imagenet, input_shape: (224,224,3), GAP, Dropout Layers: 0.2 & 0.4, Dense Layers: 2 with ReLU and softmax activations, Optimizer: SGD with learning rate of 0.001.

3.7 Performance metrics

Assessing the results of model training is essential to understand the model's learning capabilities. The confusion matrix proves instrumental in gauging the performance of a classification model, especially when dealing with more than two classes or multi-class classification [20]. This matrix contains true positive (TP), true negative (TN), false positive (FP) and false negative (FN) to represent the outcomes of the classification process. To evaluate the model's performance comprehensively, the confusion matrix values are utilized to calculate average accuracy, precision, recall and F1-score, as presented below.

Confusion Matrix =	TP : Positive data were correctly predicted	TN : Negative data were correctly predicted	FP : Positive data were incorrectly predicted as negative	FN : Negative data were incorrectly predicted as positive
	Accuracy :	Precision :	Recall :	F1 Score :
Performance Metrics =	$\frac{TP + TN}{TP + TN + FP + FN}$	$\frac{TP}{TP + FP}$	$\frac{TP}{TP + FN}$	$2\frac{Precision * Recall}{Precision + Recall}$

4. Result

4.1 Dataset

The presented dataset is detailed in Table 1. The table exhibits 13 labels, encompassing a total of 18,815 leaves, with 12 labels corresponding to various diseases and 1 label representing a healthy leaf – 13 labels in total.

Notably, three new disease types, i.e. powdery mildew, magnesium deficiency and leaf miner, have been introduced, which were not previously documented in PlantVillage datasets. In terms of data pre-processing perspective, the collected data for the labels of bacterial spot, early blight, late blight, leaf mold, yellow leaf curl virus (YLCV), mosaic virus and tomato plant health were enriched, whereas powdery mildew, magnesium deficiency and leaf miner did not receive additional data. The public dataset constitutes 85%, while the primary dataset comprises 15% of the total dataset. The model training involved 50 epochs. ModelCheckpoint function was employed to obtain the best model, utilizing the highest val_ accuracy monitor and saving the temporary model [21]. The implementation of CNN and DenseNet121 is detailed below.

4.2 CNN implementation

The processing of the input images has been explained during the data pre-processing phase, with dimensions of 224x224x3. In the feature extraction layer, the image underwent convolution and pooling processes. The results from this layer underwent processing using GAP before entering the classification stage as the final output of the method. The convolution layer carried out the computation of an image using a kernel or filter – an array-

valued matrix quantity, often referred to as a tensor. The convolution process involved matrix multiplication via dot product across all sections of the 3x3 kernel image pixel, resulting in a new image that has undergone filtering.

The pooling layer was essential for downsampling to reduce computational load during model training. The functioning of pooling closely resembled that of the convolution layer, employing a kernel with 2x2 size and a stride as a step for the calculation process within the kernel. Two approaches namely, max-pooling and average pooling were used to pool images. Max-pooling selects the maximum value from a given kernel, while average pooling calculates the average number in a kernel. Furthermore, the dropout feature is a parameter employed to prevent overfitting by reducing the computational load on the fully connected layer[s]. This feature prevents the model from overly relying on specific inputs, enhancing overall generalizability.

The GAP layer is a method used in the CNN algorithm to replace the flatten function in. While its operation is similar to max-pooling and average pooling, GAP gives an input image a 1x1 pixel size by averaging all pixels in the image. Consequently, the GAP layer yields the same number of outputs as the size of the output map. This process can be applied before fully connected calculations or classification layers [22]. The Softmax function determined the predictive probability that the input image belongs to each category. It was employed in the last classification layer, or output layer, to normalize the output into a basic probability distribution within the range of 0 to 1.

The training results of the CNN method, observed after running 50 epochs revealed a consistent increase throughout the epochs. However, starting from the 38th epoch, the validation accuracy exhibited instability. This behavior was also mirrored in the model loss. The highest training accuracy achieved was 90.41%, and the peak validation accuracy occurred at the 48th epoch, reaching 84.77%. The calculations for training and validation loss in the 50th epoch yielded a training loss of 0.0432 and a validation loss of 0.0801. The tested architecture underwent several prior refinements, involving alterations to both the architecture itself and some of its hyperparameters. In this context, incorporating the GAP layer exhibited superior performance compared to the conventional CNN architecture utilizing a flatten layer. This performance disparity was notably evident in the validation accuracy outcomes. The CNN with flatten function achieved a validation accuracy of 72% with validation loss result of 2.4. Hence, the GAP layer resulted in an 11% increase of validation accuracy, along with a substantially reduced validation loss result of 0.0801. Table 2 presents the outcomes of the training model.

4.3 DenseNet121 implementation

In configuring the DenseNet121 layer, parameters were specified including input_shape with 224x224x3 and weights using ImageNet. Following the processing outcomes from the DenseNet121 layer, the results then underwent the GAP layer, followed by the application of the dropout feature. The subsequent step involved the fully connected Dense layer with a 512-kernel, culminating in the last layer utilizing Dense with Softmax activation. As epoch progresses, the results indicated a consistent increase in the training accuracy, accompanied

Table 2. Performance of CNN

CNN	Training Best accuracy (at 50th epoch)	Validation Best accuracy (at 48th epoch)
Accuracy	90.41%	84.77%
Loss	0.0432	0.0739

Source(s): Created by authors

by a continual decrease in the training loss. Conversely, the validation accuracy and loss exhibited relatively minor changes post the 15th epoch. Notably, the best training accuracy reached 98.27% at the 50th epoch, while the training loss recorded a value of 0.0135. Additionally, the validation accuracy yielded the highest result at the 34th epoch, registering at 87.8%, with a corresponding validation loss of 0.0558, presented in Table 3. Hence, when compared to the findings of related studies, where hybrid CNN and SVM [11], CNN with the Plant Village dataset [7] and DenseNet121 with the Plant Village dataset [14] achieved accuracies of 92.6%, 98.4% and 98.23%, respectively, our proposed approach yields a promising and improved outcome of 98.27%.

Prior to finalizing the architecture for the DenseNet121, a thorough tuning process was conducted. One noteworthy outcome of this tuning effort was the adoption of the Adam optimizer. The results obtained with the Adam optimizer revealed an accuracy rate of 99.14% during the model training phase. However, a significant divergence in validation accuracy was observed, standing at 81.78%. Subsequently, the architecture ultimately chosen for this study exhibited a notable improvement in validation accuracy, surpassing the earlier result. In addition, the confusion matrix, an important statistical tool [23] was employed to assess the performance of DenseNet121. Positive predicted values or precision and sensitivity, indicating correctly identified relevant instances, demonstrated 88% and 87% presented in Table 4.

4.4 Discussion

In the final phase of this research, we evaluated the trained models using prepared data to identify the most effective method for accurately classifying tomato plant diseases based on leaf images. To gauge the performance of both methods, we analyzed accuracy from the training and validation sets of DenseNet121 and CNN models, presented in Figure 4(a) and (b). It reveals a visual comparison between DenseNet121 and CNN models in terms of accuracy and loss. The yellow line representing DenseNet121 consistently outperforms the CNN method in both training and validation results. DenseNet121 also consistently achieves lower loss values than the CNN model. A detailed comparison of the percentage accuracy over 50 epochs is provided in Table 4.

DenseNet121 surpassed CNN model in both training and validation accuracy. Specifically, DenseNet121 achieves 7.84% higher training accuracy and 4.14% higher accuracy validation

DenseNet121	Training Best accuracy (at 50th epoch)	Validation Best accuracy (at 34th epoch)
Accuracy	98.27%	87.8%
Loss	0.0135	0.0558

Source(s): Created by authors

Table 3. Performance of DenseNet121

Model	Training accuracy (%)	Validation accuracy	Training loss	Validation loss	Precision (%)	Recall (%)	F-measure (%)
CNN	90.41	83.33	0.0432	0.0801	84	83	83
DenseNet121	98.27	87.47	0.0135	0.0662	88	87	87

Source(s): Created by authors

Table 4. Performance recapitulation of CNN and DenseNet121

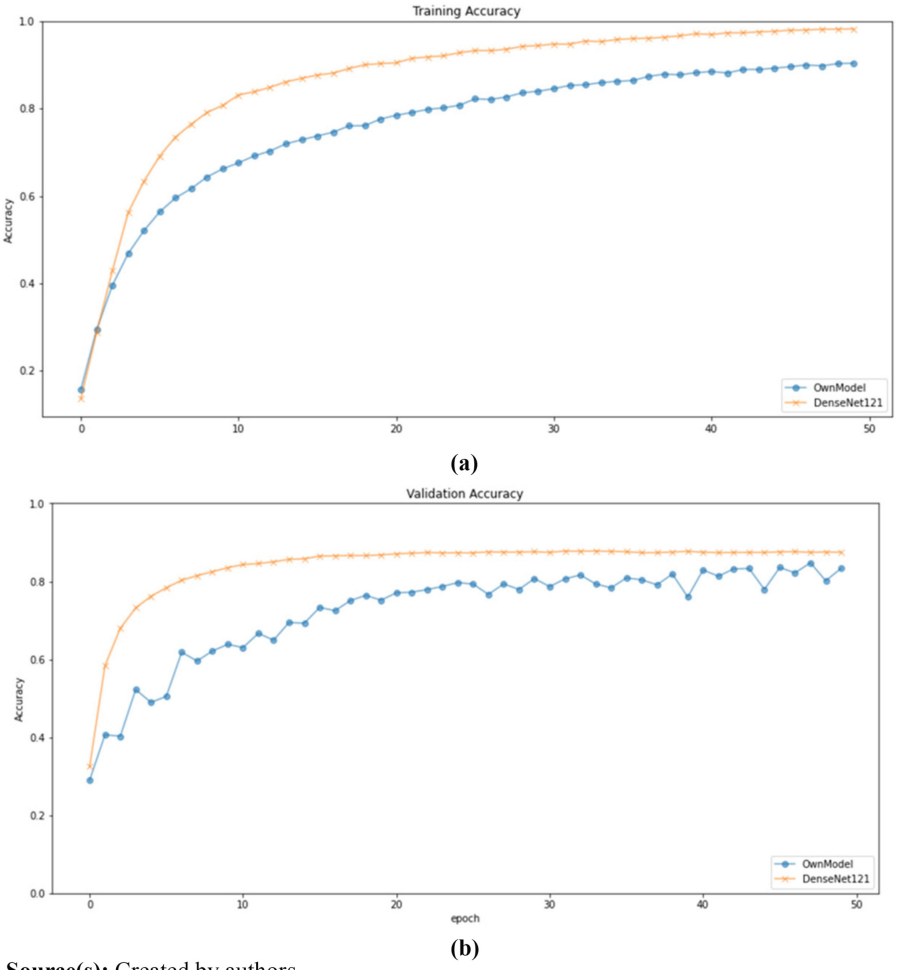

Figure 4.
(a) Training accuracy and (b) validation accuracy of CNN and DenseNet121

Source(s): Created by authors

compared to CNN. In addition of the accuracy comparison, precision, recall and F1-score calculations were employed to discern the optimal method for this study.

The findings presented in Table 4 highlight that, in the comparative analysis between the two methods, DenseNet-121 demonstrated better performance, exhibiting precision, recall and F1-score values that are 4% higher than those attained by the CNN in the identification of tomato plant diseases, incorporating both private and public datasets.

Figure 4 depicts that DenseNet121 outperforms traditional CNN architectures due to several key factors elucidated in this study. First, its dense connectivity pattern facilitates direct input from preceding layers, enhancing the optimization of hyperparameters. Second, the incorporation of a GAP layer allows for a more comprehensive representation of input data. Third, leveraging pre-trained weights from ImageNet aids in initializing the model's feature weights effectively. Fourth, the integration of dropout layers mitigates overfitting risks. Fifth, DenseNet121 exhibits remarkable robustness when trained on a complex dataset

comprising 13 classes of tomato plant diseases. Its dense connectivity and feature reuse mechanisms empower the model to learn discriminative features adeptly amidst diverse and intricate data, thereby contributing to its superior performance compared to conventional CNN.

DenseNet121's interpretability stems from its architectural design, regularization techniques, utilization of pre-trained weights and optimization strategy and ability to handle complex datasets. By analyzing how these factors interact and influence the model's predictions, one can gain valuable insights into how the model processes and interprets input data. In the context of agricultural decision-making, the interpretability of DenseNet121 plays a crucial role in understanding and trusting the model's predictions, which is essential for informed decision-making in farming practices. For instance, the connectivity pattern and GAP layer enable it to capture intricate patterns and spatial information within plant images. By interpreting the model's predictions, agricultural experts can gain insights into the specific disease patterns detected by the model. Understanding these patterns can inform decisions about disease management strategies, such as targeted pesticide application or crop rotation practices. Moreover, to bolster the validity of the research findings and offer a more comprehensive evaluation of the proposed model's effectiveness, an example of visual identification results is illustrated in Plate 1.

5. Conclusion

Based on the research findings, DenseNet121 demonstrates superior performance compared to CNN in identifying tomato plant diseases. The training dataset, which encompasses 13 classes, underscores the robustness of DenseNet121. The integration of private and public data significantly contributes to determining the optimal method for model training. Specifically, the CNN method achieves a training accuracy of 90.41% and a validation accuracy of 83.33%, whereas the DenseNet121 method excels with a training accuracy of 98.27% and a validation accuracy of 87.47%. The DenseNet121 architecture, comprising 121 layers, a GAP layer and a dropout layer, showcases its effectiveness. The hidden layers consist of 512 nodes, and the final layer comprises 12 nodes with a softmax function for image classification. Leveraging categorical_crossentropy as the loss function and utilizing the SGD Optimizer with a learning rate of 0.001 guide the course of the model training process. The experimental results unequivocally demonstrate the superior performance of

(a) (b) (c)

Source(s): Created by authors

Plate 1.
(a) Late blight disease, (b) Septoria leaf spot disease and (c) healthy leaf and are identified correctly using the proposed approach

DenseNet121 over CNN. Consequently, the findings from this research have been practically applied in the Tanamin.id application, accessible on Google Play.

An inherent limitation of our approach is its applicability to other crops, given the study's exclusive focus on tomato plant diseases. Generalizing our findings necessitates further validation and fine-tuning for different plant species and disease types. Collaborative research and data sharing present promising prospects for developing more adaptable models and fostering robust solutions to diverse agricultural challenges. Additionally, exploring hyperparameter optimization provides opportunities to enhance model performance. These advancements aim to facilitate the development of more accurate and reliable models for detecting tomato plant diseases, positively impacting agricultural production.

References

1. Mahum R, Munir H, Mughal ZUN, Awais M, Sher Khan F, Saqlain M, Mahamad S, Tlili I. A novel framework for potato leaf disease detection using an efficient deep learning model. Hum Ecol Risk Assess Int J. 2023; 29(2): 303-26. doi: 10.1080/10807039.2022.2064814.
2. Zhong Y, Zhao M. Research on deep learning in apple leaf disease recognition. Comput Electron Agric. 2020; 168: 105146. doi: 10.1016/j.compag.2019.105146.
3. Li L, Zhang S, Wang B. Plant disease detection and classification by deep learning—a review. IEEE Access. 2021; 9: 56683-98. doi: 10.1109/ACCESS.2021.3069646.
4. MFK Tomato production by country. [cited 2024 Apr 21]. Available from: https://www.worldostats.com/post/tomato-production-by-country-2023
5. Singh BK, Delgado-Baquerizo M, Egidi E, Guirado E, Leach JE, Liu H, Trivedi P. Climate change impacts on plant pathogens, food security and paths forward. Nat Rev Microbiol. 2023; 21(10): 640-56. doi: 10.1038/s41579-023-00900-7.
6. Hassaballah M, Awad AI. (Eds) Deep learning in computer vision: principles and applications. Boca Raton: CRC Press; 2020. doi: 10.1201/9781351003827.
7. Agarwal M, Gupta SK, Biswas KK. Development of Efficient CNN model for Tomato crop disease identification. Sustain Comput Inform Syst. 2020; 28: 100407. doi: 10.1016/j.suscom.2020.100407.
8. Li Y, Nie J, Chao X. Do we really need deep CNN for plant diseases identification?. Comput Electron Agric. 2020; 178: 105803. doi: 10.1016/j.compag.2020.105803.
9. Chen J, Chen J, Zhang D, Sun Y, Nanehkaran YA. Using deep transfer learning for image-based plant disease identification. Comput Electron Agric. 2020; 173: 105393. doi: 10.1016/j.compag.2020.105393.
10. Author 2. Tanamin – apps on Google play In: Tanamin – apps on Google play. [cited 2024 Mar 5]. Available from: https://play.google.com/store/apps/details?id=com.app.tanamin&hl=en_GB
11. Garg N, Gupta R, Kaur M, Kukreja V, Jain A, Tiwari RG. Classification of tomato diseases using hybrid model (CNN-SVM). In: 2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO); 2022. p. 1-5. doi: 10.1109/ICRITO56286.2022.9964708.
12. Paul SG, Biswas AA, Saha A, Zulfiker MS, Ritu NA, Zahan I, Rahman M, Islam MA. A real-time application-based convolutional neural network approach for tomato leaf disease classification. Array. 2023; 19: 100313. doi: 10.1016/j.array.2023.100313.
13. Karthik R, Hariharan M, Anand S, Mathikshara P, Johnson A, Menaka R. Attention embedded residual CNN for disease detection in tomato leaves. Appl Soft Comput. 2020; 86: 105933. doi: 10.1016/j.asoc.2019.105933.
14. Vellaichamy AS, Swaminathan A, Varun C, Kalaivani S. Multiple plant leaf disease classification using DENSENET-121 architecture. Int J Electr Eng Technol. 2021; 12(5). doi: 10.34218/IJEET.12.5.2021.005.

15. Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, Xiong H, He Q. A comprehensive survey on transfer learning. Proc IEEE. 2021; 109(1): 43-76. doi: 10.1109/JPROC.2020.3004555.

16. Hong H, Lin J, Huang F. Tomato disease detection and classification by deep learning. In: 2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE); 2020. p. 25-9. doi: 10.1109/ICBAIE49996.2020.00012.

17. Author 3. PlantVillage dataset. Kaggle. Available from: https://www.kaggle.com/datasets/emmarex/plantdisease

18. Author 4. Deep learning for vision systems. 2020. [cited 2024 Apr 21]. Available from: https://www.simonandschuster.com/books/Deep-Learning-for-Vision-Systems/Mohamed-Elgendy/9781617296192

19. Author 5. Deep learning, MIT Press. [cited 2024 Apr 21]. Available from: https://mitpress.mit.edu/9780262035613/deep-learning/

20. Grandini M, Bagli E, Visani G. Metrics for multi-class classification: an overview. arXiv.org. [cited 2024 Apr 21]. Available from: https://arxiv.org/abs/2008.05756v1

21. Zhuang S, Zuccon G. Asyncval: a toolkit for asynchronously validating dense retriever checkpoints during training. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. Madrid Spain: ACM; 2022. p. 3235-9. doi: 10.1145/3477495.3531658.

22. Author 6. Electronics | free full-text | CNN variants for computer vision: history, architecture, application, challenges and future scope. [cited 2024 Apr 21]. Available from: https://www.mdpi.com/2079-9292/10/20/2470

23. Zeng G. On the confusion matrix in credit scoring and its analytical properties. Commun Stat - Theor Methods. 2020; 49(9): 2080-93. doi: 10.1080/03610926.2019.1568485.

Corresponding author
Heru Agus Santoso can be contacted at: heru.agus.santoso@dsn.dinus.ac.id