Paradigma, Vol. 27, No. 1, March 2025, P-ISSN 1410-5063, E-ISSN: 2579-3500, Page. 48-54
Published: LPPM Universitas Bina Sarana Informatika
PARADIGMA
website: http://jurnal.bsi.ac.id/index.php/paradigma/

Classification of Vegetable Types Using the Convolutional Neural Network (CNN) Algorithm

Wilis Arum Karunia[1], Safriya Murni Puspita[2], Dwi Rolliawati[3], Ahmad Yusuf[4]

[1,2,3,4] Universitas Islam Negeri Sunan Ampel Surabaya, Surabaya, Indonesia

ARTICLE INFORMATION

Artikel History:
Received: January 6, 2025
Revised: February 18, 2025
Accepted: March 11, 2025
Available Online: March 30, 2025

Keyword:
Convolutional Neural Network
Deep Learning
Keras
TensorFlow
Vegetable Classification

ABSTRACT

This study aims to classify vegetable types using the Convolutional Neural Network (CNN) algorithm with a dataset encompassing 15 vegetable classes and a total of 31,000 images. By utilizing the TensorFlow and Keras libraries, the CNN model was designed with convolutional, pooling, and dense layers to recognize visual features such as color, texture, and shape. The results indicate a highest validation accuracy of 95.83% and a testing accuracy of 93%. This research contributes to the application of the CNN algorithm for image classification and demonstrates its potential in handling multi-class datasets effectively. However, since the vegetables used have very distinct shapes and textures, this study is more relevant in the context of the technical application of the CNN algorithm rather than practical benefits. The research would be more impactful if applied to vegetables with similar shapes and characteristics, thereby supporting farmers or individuals studying vegetable traits in greater depth. Additionally, such an approach could address challenges in differentiating visually similar vegetable types, making the technology more valuable in real-world agricultural or educational settings.

Corresponding Author:

Wilis Arum Karunia,
Departement of Information System Faculty of Science and Technology,
Universitas Islam Negeri Sunan Ampel Surabaya,
Jl. Dr. Ir. Soekarno No. 682 Gunung Anyar, Surabaya, Indonesia, 60294,
Email: wilissarrum@gmail.com

INTRODUCTION

The advancement of technology in the present era has significantly impacted various aspects of life. Technological development brings both positive and negative effects across numerous fields, including agriculture. Technology enables individuals to save time by performing tasks more easily and effectively.

The world hosts an immense variety of plants, totaling around 374,000 species, including 308,312 vascular plants and 295,383 vegetable plants. Vegetable plants, encompassing 416 families, 13,164 genera, and 295,383 species, stand as the most diverse group of terrestrial plants worldwide (Nurrani, Andi Kurniawan Nugroho, & Sri Heranurweni, 2023).

Vegetables are plant-based food materials with high water content that are consumed either fresh or minimally processed. According to Budiawan and Hartono, many young individuals are unfamiliar with the names or types of both local and imported vegetables (Budiawan & Hartono, 2023).

One application of technology is the development of methods to classify vegetable types based on visual imagery (Aditya Dwi Putro Wicaksono ; Henri Tantyoko, 2023). These methods enhance efficiency in managing and identifying harvested vegetable varieties.

In the context of visual image processing, Convolutional Neural Network (CNN) algorithms are employed to automate the classification and recognition of vegetable types rapidly. CNN has proven effective in identifying visual patterns from image data (Prinzky & Lubis, 2022). CNN can adapt to input image variations like translation, rotation, and scaling, drawing inspiration from the simple and complex cells of the mammalian visual cortex (Hasan, Riyanto, & Riana, 2021). Leading companies such as Google and Facebook also implement this algorithm

DOI: https://doi.org/10.31294/p.v27i1.7577

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License

for facial recognition processes (Rasywir, Sinaga, & Pratama, 2020).

CNN is a deep learning approach. In the application of vegetable classification, CNN plays a critical role in distinguishing different types of vegetables based on visual characteristics such as color, texture, and shape (Prinzky & Lubis, 2022). Moreover, the classification process emphasizes not only speed but also accuracy.

Previous studies have explored the classification of fresh or spoiled fruits and vegetables using CNN (Munfaati & Witanti, 2024) and the detection of vegetable types using CNN methods (Malik & Zuliarso, 2021). However, these studies utilized TensorFlow Lite, which is specifically designed for mobile devices. Another study focused on "Smart GreenGrocer," which implemented CNN algorithms for automatic vegetable classification and tested the model with a 50-epoch setting (Wijaya, Putri, & Fudholi, 2023).

According Dhamayanti and friends that Convolutional Neural Network (CNN) is effective in classifying cabbage quality based on its physical images, achieving the highest accuracy of 80% using the Adam algorithm. However, the model relies solely on color features, limiting its capability to classify non-cabbage objects and leading to potential errors under extreme lighting conditions. Adding more features and improving preprocessing are recommended to enhance accuracy and generalization (Dhamayanti, Fatchiyatur Rohma, & Zahara, 2021). Rizky Abdul Malik and Eri Zuliarso utilized the Convolutional Neural Network (CNN) with TensorFlow to classify 16 types of vegetables, achieving an accuracy of 85% on a dataset comprising 2,400 images. However, this study has limitations in terms of the small dataset size and an accuracy level that could be further improved for real-world applications (Malik & Zuliarso, 2021).

The objective of this research is to classify vegetable types using the Convolutional Neural Network (CNN) algorithm, specifically focusing on the unique visual characteristics of each vegetable type. The dataset comprises 15 distinct vegetable types, including beans, bitter gourd, bottle gourd, brinjal, broccoli, cabbage, capsicum, carrot, cauliflower, cucumber, papaya, potato, pumpkin, radish, and tomato. Each vegetable type is represented by 1,000 images, totaling 15,000 images (Ahmed & Mamun, 2021).

This study differentiates itself from previous research by not only utilizing the CNN algorithm but also by employing a comprehensive dataset that encompasses a wide variety of vegetables, allowing for a more robust classification process. The implementation of the CNN algorithm leverages the TensorFlow and Keras libraries, which are essential for training deep learning models. Keras provides a high-level API that simplifies the development of these models, enabling efficient feature extraction and classification based on the visual attributes of the vegetables (Bagas Valentino, 2023).

RESEARCH METHOD

The process of applying the CNN model for vegetable classification involves several stages, including data collection, data preprocessing, CNN architecture design, model training, and model testing. The research stages are illustrated in Figure 1.

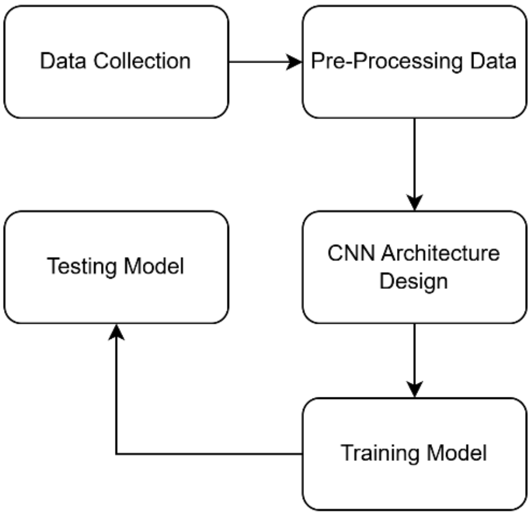

Figure 1. Research Flow

1. Data Collection

This research utilizes secondary data obtained from Kaggle. The dataset used is the "Vegetable Image Dataset" (Ahmed & Mamun, 2021). The dataset comprises training, testing, and validation data. The training data consists of 15,000 images, while the testing and validation data each contain 3,000 images. Therefore, the total number of images in the dataset amounts to 21,000.

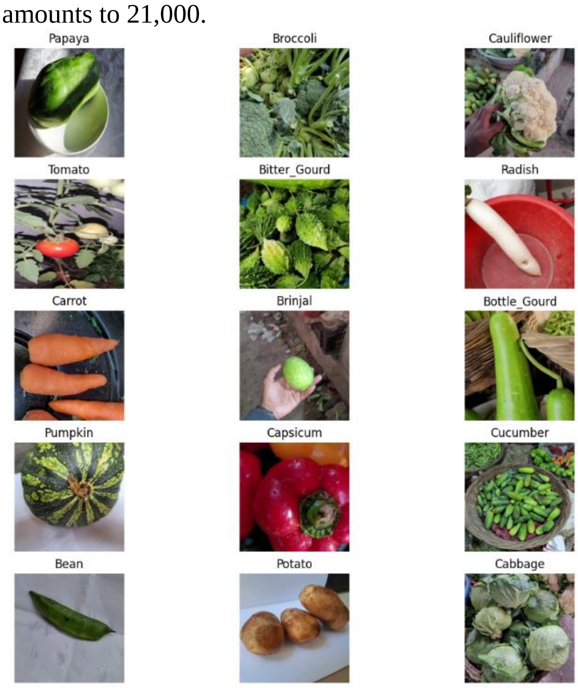

Figure 2. 15th Dataset Vegetables Class

Figure 2 illustrates a sample image representing the 15 vegetable classes in the dataset, which include bean, bitter gourd, bottle gourd, brinjal, broccoli, cabbage, capsicum, carrot, cauliflower, cucumber, papaya, potato, pumpkin, radish, and tomato.

2. Pre-Processing Data

Before the data is utilized, a preprocessing stage is conducted. This is a standard technique in image analysis to improve the model's accuracy (Maulana, Fauzan, Kholiq, A, & F, 2024). Data pre-processing was performed by scaling the image values from 0-255 to 0-1 through a normalization process. This step is used to enhance numerical stability, accelerate training convergence, ensure data consistency with the model, optimize activation functions, and adhere to standard practices in deep learning, making the model more effective and efficient (Pratitis, Kurniasari, & Fata, 2023).

3. CNN Architecture Design

The design was carried out by constructing a Sequential model consisting of multiple Conv2D, MaxPooling2D, Dropout, and Dense layers. These layers were designed to process images, extract key features, prevent overfitting, and perform classification (Mirwansyah & Arief Wibowo, 2022). The model was fine-tuned with the Adam optimizer and used the categorical_crossentropy loss function to measure prediction errors in the multi-class classification task (Prinzky & Lubis, 2022).

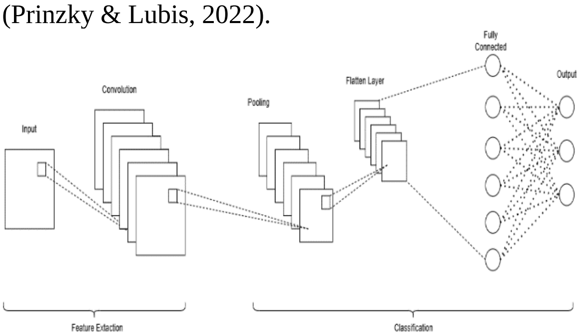

Figure 3. CNN Architecture

Figure 3 illustrates that the CNN algorithm comprises two primary components. The first is classification, which handles the categorization of processed data and organizes it into elements such as Flatten, Fully-Connected Layer, and Softmax. The second focuses on feature learning, encompassing the Convolutional Layer and Pooling Layer (Malik & Zuliarso, 2021). According to Alkhatib, among all CNN operations, convolutional layers, pooling layers, and fully connected layers are the most essential (Alkhatib et al., 2023). According to Nurcahyati and friends, the convolutional layer serves as the initial stage where convolution is applied to the image output from the preceding process (Nurcahyati, Akbar, & Zahara, 2022). Pooling layers involve matrix sampling using filters of specific sizes and stride (Minarno, 2021). In a CNN, a fully connected layer is a network layer where every neuron is linked to all neurons in the previous layer, enabling the integration of extracted features for the final output (Roy, Krishna, Dubey, & Chaudhuri, 2020)

4. Training Model

The model training process involved the optimization settings using the Adam optimizer, accuracy monitoring, and the implementation of the EarlyStopping callback mechanism. During training, the callback function was utilized to automatically halt the training process once the desired accuracy level was achieved (Winiarti, Wukir, Ahdiani, & Ismail, 2022).

5. Testing Model

The vegetable images in the training dataset were used for model testing. The testing results are presented in graphical form, illustrating accuracy and loss during the training and validation processes. These graphs provide insights into the model's performance in learning and recognizing patterns from the data (Minarno, 2021). At this stage, the success rate of the predefined method is evaluated by calculating the accuracy score (Munfaati & Witanti, 2024).

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

RESULTS AND DISCUSSION

The classification of vegetable types using a CNN was executed on Google Colab using Python and several libraries. This study utilized the TensorFlow and Keras libraries, which played a key role in constructing and training the model.

1. Normalization Image

```
# Creating Image Data Generator for train, validation and test set

# 1. Train Set
train_gen = ImageDataGenerator(rescale = 1.0/255.0) # Normalise the data
train_image_generator = train_gen.flow_from_directory(
                                            train_path,
                                            target_size=(150, 150),
                                            batch_size=32,
                                            class_mode='categorical')

# 2. Validation Set
val_gen = ImageDataGenerator(rescale = 1.0/255.0) # Normalise the data
val_image_generator = train_gen.flow_from_directory(
                                            validation_path,
                                            target_size=(150, 150),
                                            batch_size=32,
                                            class_mode='categorical')

# 3. Test Set
test_gen = ImageDataGenerator(rescale = 1.0/255.0) # Normalise the data
test_image_generator = train_gen.flow_from_directory(
                                            test_path,
                                            target_size=(150, 150),
                                            batch_size=32,
                                            class_mode='categorical')
```

Figure 4. Normalization Image Process

In Figure 4, this process focuses on normalizing image pixel values from a range of 0–255 to 0–1. The normalization is achieved by utilizing the rescale parameter in Keras ImageDataGenerator function, where each pixel value is divided by 255. This adjustment ensures the model works with data on a smaller and more consistent scale, which is essential for speeding up training and enhancing the stability of the optimization process.

Normalization is applied consistently across the entire dataset, including the training set, validation set, and test set, to ensure uniform input data throughout the training and evaluation process.

Notably, this process does not involve data augmentation, meaning the original images in the dataset are used without additional modifications. By adopting this approach, the model is trained using normalized data while retaining the original representation of the dataset, avoiding artificial variations.

Table 1. Normalization Image Results

Data	Number of Images	Percentage	Number of Class
Train	15000	71.43%	15 Kelas
Test	3000	14.29%	15 Kelas
Validation	3000	14.29%	15 Kelas

Table 1 shows that the dataset contains 15,000 training images and 3,000 images each for testing and validation, spread across 15 classes. The images have undergone preprocessing to standardize scaling and size, ensuring a consistent input format for effective model training and evaluation.

2. CNN Architecture Model

The Convolutional Neural Network (CNN) architecture employed in this study was sequentially constructed using the Keras framework.

```
# Build a custom sequential CNN model
model = Sequential() # model object

# Add Layers
model.add(Conv2D(filters=32, kernel_size=3, strides=1, padding='same',
                 activation='relu', input_shape=[150, 150, 3]))
model.add(MaxPooling2D(2, ))
model.add(Conv2D(filters=64, kernel_size=3, strides=1, padding='same',
                 activation='relu'))
model.add(MaxPooling2D(2))

# Flatten the feature map
model.add(Flatten())

# Add the fully connected layers
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(128, activation='relu'))
model.add(Dense(15, activation='softmax'))

# print the model summary
model.summary()
```

Figure 5. CNN Architecture Draft

As illustrated in Figure 5, the model comprises several key layers designed to process image data with dimensions of 150x150 pixels and three-color channels (RGB). The architecture starts with an initial Convolutional layer comprising 32 filters of size 3x3, utilizing same padding and the ReLU activation function to extract basic image features. This is followed by a MaxPooling2D layer, which halves the spatial dimensions, reducing computational complexity and helping to prevent overfitting. A second convolutional layer with 64 filters and similar configurations is added, followed by another MaxPooling layer to further enhance feature extraction.

After feature extraction, the data is flattened using a Flatten layer to transform multidimensional arrays into a one-dimensional vector. The network is connected to two Dense (fully connected) layers, each containing 128 neurons with ReLU activation functions. To minimize the risk of overfitting, a Dropout layer with a 25% dropout rate is introduced before the second Dense layer. The output layer consists of 15 neurons with a softmax activation function, producing probabilities for each of the 15 classes in the dataset.

This architecture is designed to optimize feature extraction and classification across the dataset's 15 classes. A summary generated using the summary() function shows the model comprises a total of 11,252,815 trainable parameters, representing a balanced complexity between achieving high accuracy and computational efficiency.

Table 2. CNN Architecture Draft Results

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 150, 150, 32)	896
max_pooling2d_8 (MaxPooling2D)	(None, 75, 75, 32)	0
conv2d_9 (Conv2D)	(None, 75, 75, 64)	18,496
max_pooling2d_9 (MaxPooling2D)	(None, 37, 37, 64)	0
flatten_4 (Flatten)	(None, 87616)	0
dense_12 (Dense)	(None, 128)	11,214,976
dropout_4 (Dropout)	(None, 128)	0
dense_13 (Dense)	(None, 128)	16,512
dense_14 (Dense)	(None, 15)	1,935
Total params: 11,252,815 (42.93 MB)		
Trainable params: 11,252,815 (42.93 MB)		
Non-trainable params: 0 (0.00 B)		

Table 2 outlines the architecture of the Convolutional Neural Network (CNN) model built using the Sequential framework. This framework incorporates various layers, such as convolutional, pooling, and fully connected layers, to effectively classify images (Sharma & Phonsa, 2021). The model begins with two Conv2D layers containing 32 and 64 filters, respectively, each followed by a MaxPooling2D layer to reduce feature dimensions. After feature extraction, a Flatten layer converts the output into a one-dimensional vector of 87,616 units.

This vector is fed into the first Dense layer with 128 units, where a Dropout layer is included to mitigate overfitting. Another Dense layer with 128 units follows, culminating in a final Dense layer with 15 units to generate outputs corresponding to the 15 classes. The model consists of 11,252,815 trainable parameters, reflecting its high complexity, which is designed to effectively learn patterns from input data with dimensions of 150x150x3.

3. Training Model

The model training process was conducted using a Deep Learning framework, incorporating several essential components. Figure 6 the training process began with the EarlyStopping callback, configured to stop training automatically if validation performance failed to improve after five consecutive epochs.

```python
# Compile and fit the model
early_stopping = keras.callbacks.EarlyStopping(patience=5) # Set up callbacks
model.compile(optimizer='Adam',
              loss='categorical_crossentropy', metrics=['accuracy'])
hist = model.fit(train_image_generator,
                 epochs=100,
                 verbose=1,
                 validation_data=val_image_generator,
                 steps_per_epoch = 15000//32,
                 validation_steps = 3000//32,
                 callbacks=early_stopping)
```

Figure 6. Model Training

This strategy aimed to prevent overfitting while optimizing time and resource usage. The model was compiled with the Adam optimizer, chosen for its ability to accelerate convergence, and the categorical_crossentropy loss function, which is well-suited for multi-class classification tasks. Accuracy was selected as the primary evaluation metric to track performance during training.

A data generator was employed to iteratively supply training and validation data to the model. The training ran for up to 100 epochs, with each epoch consisting of a defined number of steps for both training and validation, specified by the steps_per_epoch and validation_steps parameters. During this process, the model gradually minimized the loss function and improved its accuracy based on the training and validation data. Metrics such as loss and accuracy for each epoch were recorded in the variable hist, which can be analyzed to assess model performance, including patterns of convergence or signs of overfitting. This approach ensured an optimal and efficient model training process.

Table 3. Model Training Results

Epoch	Training Time	Accuracy	Loss	Val_Accuracy	Val_Loss
1	667s	0.3809	1.9166	0.8357	0.5560
2	1s	0.8438	0.7607	0.9167	0.2848
3	669s	0.8140	0.5641	0.9076	0.3062
4	2s	0.7812	0.6792	0.8750	0.3137
5	688s	0.9074	0.2900	0.9311	0.2385
6	1s	1.0000	0.0643	1.0000	0.0598
7	669s	0.9462	0.1711	0.9399	0.2142
8	1s	0.9375	0.1190	0.8333	0.7347
9	659s	0.9580	0.1359	0.9368	0.2333
10	1s	0.9375	0.1291	0.9583	0.1541
11	681s	0.9621	0.1160	0.9170	0.3150

Table 3 presents the results of training a CNN model over 11 epochs, including metrics such as training time per epoch, accuracy, and loss for the training data, as well as accuracy and loss for the validation data. The table indicates a significant improvement in accuracy from the first to the sixth epoch, where the training accuracy reached 100%, and the validation accuracy also achieved 100%, with very low loss values (0.0643 for training and 0.0598 for validation). However, fluctuations began to appear in subsequent epochs. For instance, in the eighth epoch, the validation loss increased to 0.7347, despite the training accuracy remaining high. This suggests the possibility of overfitting or inconsistency in the validation data during certain epochs. Overall, the model demonstrates strong performance, achieving the highest validation accuracy of 95.83% in the tenth epoch.

4. Testing Model

After training the model, vegetable images from the test dataset were used for testing and evaluating the model's performance (Munfaati & Witanti, 2024). To illustrate the model's learning process and its performance on the validation data during training, the results can be visualized in the form of a graph, as shown in Figure 7.

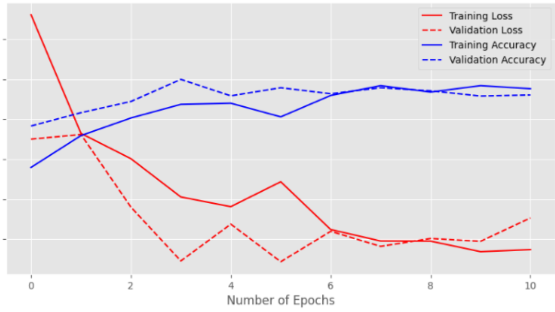

Figure 7. Accuracy and Loss Curves for Training and Test Data

Based on the displayed graph, it is evident that the model exhibited a decrease in training loss and validation loss during the initial epochs, indicating effective learning on the given data. However, in subsequent epochs, the validation loss began to increase while the training loss continued to decline, signaling the occurrence of overfitting. Additionally, the training accuracy steadily increased as the epochs progressed, whereas the validation accuracy showed a fluctuating trend and tended to stagnate after a certain number of epochs. This suggests that the model performed better on the training data than on the validation data, which could be attributed to the model's complexity or limited generalization capability.

The model's evaluation on the test data aimed to assess its generalization ability. Unlike the training and validation data, the test data was entirely separate, providing an objective measure of the model's performance on new, unseen data. The results, reflected in the accuracy and loss metrics, offer insight into how well the model recognizes patterns in the test dataset. The evaluation demonstrated an accuracy of 93.47% and a loss of 0.2499 (24%), indicating the model's strong capability to classify the test data accurately.

During the model testing phase, test images were used to evaluate the model's ability to accurately classify vegetable types. Figure 8 illustrates a test image example, where one image from the dataset (cabbage) was resized to 150x150 pixels and normalized by dividing its pixel values by 255. The image was then converted into a four-dimensional array to ensure compatibility with the model. The model generated predictions mapping the results to

Figure 8. Testing Model 1

Figure 9. Testing Model 2

Figure 10. Testing Model 3

class names based on the index of the highest prediction value mapping the results to class names based on the index of the highest prediction value. In the visualization presented in Figure 8, the original image is shown with a title displaying the model's predicted label ("Cabbage") alongside the actual label ("Cabbage"). Similarly, Figure 9 and Figure 10 also demonstrate correct classifications by the model, reinforcing its strong performance in this context. These visualizations provide additional support for the analysis by comparing the predicted outcomes with the actual labels.

CONCLUSION

This research successfully demonstrates that the Convolutional Neural Network (CNN) algorithm is effective in classifying 15 types of vegetables based on their visual images. Using a vegetable dataset sourced from Kaggle, the data is divided into training, testing, and validation sets. The training set consists of 15,000 images, while the test and validation sets each contain 3,000 images, amounting to a total of 21,000 images.

Using this dataset, processes such as normalization, training, and testing were conducted with TensorFlow and Keras libraries. The designed CNN model achieved the highest validation accuracy of 95.83% at the 10th epoch, despite showing signs of overfitting during certain epochs. The model effectively identifies vegetable types, as evidenced by a final testing accuracy of 93%.

Future researchers who wish to explore this research concept are encouraged to go beyond merely replicating the study with different objects. They should aim to introduce significant advancements, such as incorporating larger and more diverse datasets, applying novel CNN architectures, or integrating other deep learning techniques like transfer learning or hybrid models. Additionally, practical implementation in real-world environments, such as developing mobile or IoT-based solutions, could enhance the research's relevance and impact.

REFERENCES

Aditya Dwi Putro Wicaksono ; Henri Tantyoko. (2023). *JTIM : Jurnal Teknologi Informasi dan Multimedia Hybrid Algoritma Vgg16-Net dengan Support Vector Machine*. *5*(2), 56–65.

Ahmed, M. I., & Mamun, S. M. (2021). Vegetable Image Dataset. https://doi.org/10.34740/KAGGLE/DSV/2965251

Alkhatib, M. Q., Al-Saad, M., Aburaed, N., Almansoori, S., Zabalza, J., Marshall, S., & Al-Ahmad, H. (2023). Tri-CNN: A Three Branch Model for Hyperspectral Image Classification. *Remote Sensing*, *15*(2), 1–19. https://doi.org/10.3390/rs15020316

Budiawan, R. S., & Hartono, B. (2023). Pengembangan Sistem Pendeteksi Jenis Sayuran dengan Metode CNN Berbasis Android. *Jurnal Informatika Dan Rekayasa Perangkat Lunak*, *5*(1), 62. https://doi.org/10.36499/jinrpl.v5i1.7833

Dhamayanti, R., Fatchiyatur Rohma, M., & Zahara, S. (2021). Penggunaan Deep Learning Dengan Metode Convolutional Neural Network Untuk Klasifikasi Kualitas Sayur Kol Berdasarkan Citra Fisik. *SUBMIT: Jurnal Ilmiah Teknologi Informasi Dan Sains*, *1 (1)*, 08–15.

Hasan, M. A., Riyanto, Y., & Riana, D. (2021). Grape leaf image disease classification using CNN-VGG16 model. *Jurnal Teknologi Dan Sistem Komputer*, *9*(4), 218–223. https://doi.org/10.14710/jtsiskom.2021.14013

Malik, R. A., & Zuliarso, E. (2021). Metode Convolutional Neural Network Untuk Mendeteksi Jenis Sayur Menggunakan Tensorflow. *Media Bina Ilmiah*, *15*(1978), 5873–5882. Retrieved from http://ejurnal.binawakya.or.id/index.php/MBI/article/view/1147

Maulana, N., Fauzan, M., Kholiq, R., A, M. D. H., & F, G. S. (2024). *Literature Review : Klasifikasi Citra Medis Penyakit Pneumonia dengan Convolutional Neural Network*. *3*(9), 2339–2342.

Minarno, A. E. (2021). Klasifikasi Jenis Batik Menggunakan Algoritma Convolutional Neural Network. *Jurnal Repositor*, *3*(2), 199–206. https://doi.org/10.22219/repositor.v3i2.1201

Mirwansyah, D., & Arief Wibowo. (2022). Fruit Image Classification Using Deep Learning Algorithm: Systematic Literature Review (Slr). *Multica Science and Technology (Mst) Journal*, *2*(2), 120–123. https://doi.org/10.47002/mst.v2i2.356

Munfaati, E. A. N., & Witanti, A. (2024). Klasifikasi Buah dan Sayuran Segar atau Busuk Menggunakan Convolutional Neural Network. *JISKA (Jurnal Informatika Sunan Kalijaga)*, *9*(1), 27–38. https://doi.org/10.14421/jiska.2024.9.1.27-38

Nurcahyati, A. D., Akbar, R. M., & Zahara, S. (2022). Klasifikasi Citra Penyakit pada Daun Jagung Menggunakan Deep Learning dengan Metode Convolution Neural Network (CNN). *SUBMIT: Jurnal Ilmiah Teknologi Infomasi Dan Sains*, *2*(2), 43–51. https://doi.org/10.36815/submit.v2i2.1877

Nurrani, H., Andi Kurniawan Nugroho, & Sri Heranurweni. (2023). Image Classification of Vegetable Quality using Support Vector Machine based on Convolutional Neural Network. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, *7*(1), 168–178. https://doi.org/10.29207/resti.v7i1.4715

Pratitis, W. L., Kurniasari, K., & Fata, H. Al. (2023). Classification of Spotted Disease on Sugarcane Leaf Image Using Convolutional Neural Network Algorithm. *JTECS : Jurnal Sistem Telekomunikasi Elektronika Sistem Kontrol Power Sistem Dan Komputer*, *3*(2), 117. https://doi.org/10.32503/jtecs.v3i2.3433

Prinzky, & Lubis, C. (2022). Klasifikasi Buah Segar Dan Busuk Menggunakan Convolutional Neural Network Berbasis Android. *Jurnal Ilmu Komputer Dan Sistem Informasi*, *10*(2), 10823–10827. https://doi.org/10.24912/jiksi.v10i2.22551

Rasywir, E., Sinaga, R., & Pratama, Y. (2020). Analisis dan Implementasi Diagnosis Penyakit Sawit dengan Metode Convolutional Neural Network (CNN). *Paradigma - Jurnal Komputer Dan Informatika*, *22*(2), 117–123. https://doi.org/10.31294/p.v22i2.8907

Roy, S. K., Krishna, G., Dubey, S. R., & Chaudhuri, B. B. (2020). HybridSN: Exploring 3-D-2-D CNN Feature Hierarchy for Hyperspectral Image Classification. *IEEE Geoscience and Remote Sensing Letters*, *17*(2), 277–281. https://doi.org/10.1109/LGRS.2019.2918719

Sharma, A., & Phonsa, G. (2021). Image Classification Using CNN. *SSRN Electronic Journal*, (Icicc), 1–5. https://doi.org/10.2139/ssrn.3833453

Wijaya, R. B. M. A. A., Putri, D. N. A., & Fudholi, D. R. (2023). Smart GreenGrocer: Automatic Vegetable Type Classification Using the CNN Algorithm. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, *17*(3), 271. https://doi.org/10.22146/ijccs.82377

Winiarti, S., Wukir, C., Ahdiani, U., & Ismail, T. (2022). Klasifikasi Image Untuk Jenis Buku Bacaan Anak-Anak dengan Menggunakan Convolutional Neural Network. *Jurnal Media Informatika Budidarma*, *6*(2), 738. https://doi.org/10.30865/mib.v6i2.3504