Institute of Research and Publication Indonesia (IRPI)
Public Research Journal of Engineering, Data Technology and Computer Science
Journal Homepage: https://journal.irpi.or.id/index.php/predatecs
Vol. 3 Iss. 2 January 2026, pp: 126-137
ISSN(P): 3024-921X | ISSN(E): 3024-8043

Comparative Study of Convolutional Neural Network Architectures and Optimizers for Flower Image Classification

Ekatri Yulisara[1], Nayla Husna Ryanda[2*], David Martin[3], Candrawati Ariesta[4]

[1,2]Department of Information System, Faculty of Science and Technology,
Universitas Islam Negeri Sultan Syarif Kasim Riau, Indonesia
[3]Department of Nursing Care and Welfare, Fukuoka Childcare,
Children's Medical Welfare Vocational School, Japan
[4]Department and Faculty of Chinese Teaching, Jinan University, China

E-Mail: [1]12250325372@students.uin-suska.ac.id, [2]12250321428@students.uin-suska.ac.id,
[3]fukuoka_email@mail.o-hara.ac.jp, [4]osmartedu@jnu.edu.cn

Received Jun 12th 2025; Revised Jan 21th 2026; Accepted Jan 25th 2026; Available Online Jan 31th 2026
Corresponding Author: Nayla Husna Ryanda

Abstract

This study aims to comparatively evaluate the performance of different Convolutional Neural Network (CNN) architectures and optimization algorithms for flower image classification. Three widely used CNN architectures DenseNet201, InceptionV3, and MobileNetV2 are implemented using transfer learning with pre-trained ImageNet weights and tested with two optimizers, Adam and RMSProp. The experiments are conducted on the Flowers Recognition dataset consisting of five flower classes: daisy, dandelion, rose, sunflower, and tulip. Image normalization and data augmentation are applied to improve model generalization, while performance is evaluated using accuracy, precision, recall, and F1-score. The main contribution of this study lies in a systematic comparison of CNN architectures and optimizers within a unified experimental framework, which is rarely addressed in previous studies. The results show that DenseNet201 combined with the Adam optimizer achieves the highest classification accuracy of 90%, followed by MobileNetV2 with RMSProp, while InceptionV3 yields the lowest accuracy of 85%. These results confirm that the research objective is achieved, demonstrating that both CNN architecture and optimizer selection significantly influence flower image classification performance.

Keyword: Convolutional Neural Network, DenseNet201, Flower Classification, InceptionV3, MobileNetV2, RMSProp

1. INTRODUCTION

Along with the rapid advancement of computing technology, biodiversity has become an increasingly important focus in scientific research, particularly in the field of life sciences. Indonesia is recognized as one of the countries with the highest levels of plant biodiversity in the world, possessing approximately 35,000 species of flowering plants, which represent about 10% of global flowering plant diversity. Despite this richness, only around 19,232 species have been formally identified and documented [1][2].The ecological, cultural, and economic significance of these plant species makes their accurate identification essential. However, recognizing flower species in daily life remains challenging, as traditional identification relies on morphological characteristics such as color, shape, and structure, which require extensive expertise and time, and often depend on professional knowledge [3][4].

The wide variety of flower appearances and complex visual patterns makes manual classification inefficient and impractical for large-scale use. Conventional methods such as consulting experts, using reference books, or searching online, are time-consuming and unsuitable for rapid or automated identification [5]. Therefore, there is a strong need for an automated, accurate, and scalable system capable of effectively classifying flower images. This urgency is further amplified by the increasing availability of digital image data and the demand for intelligent systems in agriculture, education, and biodiversity conservation [6].

In this context, Convolutional Neural Networks (CNNs) have emerged as a powerful approach for image classification tasks. CNNs are specifically designed to process visual data and can automatically learn hierarchical features directly from images, eliminating the need for manual feature extraction [7]. Numerous studies have demonstrated that CNNs outperform traditional machine learning methods in object recognition and image classification, thanks to their ability to capture complex spatial patterns and semantic features.

These advantages make CNNs particularly suitable for flower image classification, where subtle visual differences between species must be accurately distinguished[8].

Several previous studies have applied CNNs to flower classification with promising results. Intyanto (2021) compared two CNN architectures and found that VGG16 achieved an accuracy of 80%, outperforming a custom-designed CNN model with 62%. Fitriani (2021) implemented a CNN-based flower classification system using MobileNetV2 and reported that VGG16 achieved an accuracy of 91%[9]. Furthermore, Munandar and Rozi (2024) evaluated VGG16 and NasNetMobile architectures with and without fine-tuning, showing that NasNetMobile achieved the highest accuracy of 99.15% when fine-tuned. These studies confirm the effectiveness of CNNs for flower classification; however, most of them focus on a limited number of architectures or do not systematically compare optimization strategies [10][11].

Despite these advancements, there remains a research gap in the comparative evaluation of multiple modern CNN architectures combined with different optimization algorithms using a standardized experimental setup [12][13]. In particular, there is limited research that simultaneously examines the performance of lightweight, deep, and densely connected architectures on the same dataset while also analyzing the impact of different optimizers on classification performance. Addressing this gap is crucial to identifying the most effective and efficient model configuration for practical deployment [14][15].

Therefore, this study aims to evaluate and compare the performance of three widely used CNN architectures: MobileNetV2, InceptionV3, and DenseNet201. Each architecture is optimized using two popular optimization algorithms, Adam and RMSprop. Model performance is assessed using accuracy, precision, recall, and F1-score metrics [16][17]. The dataset is divided using a hold-out validation scheme with an 80:20 ratio. Unlike previous studies that focused on a single model or lacked systematic comparison, this research provides a comprehensive evaluation framework to identify the best architecture-optimizer combination for flower image classification. The findings are expected to contribute to the development of more accurate, efficient, and practical CNN-based classification systems applicable to agriculture, education, and biodiversity preservation [18][19][20].

2. MATERIAL AND METHOD

The research process is Data Collection, Data Preprocessing, Data Splitting Process, Deep Learning Modeling Process, along with Optimizer and Evaluation of the methodology flow model for this research can be seen in Figure 1.

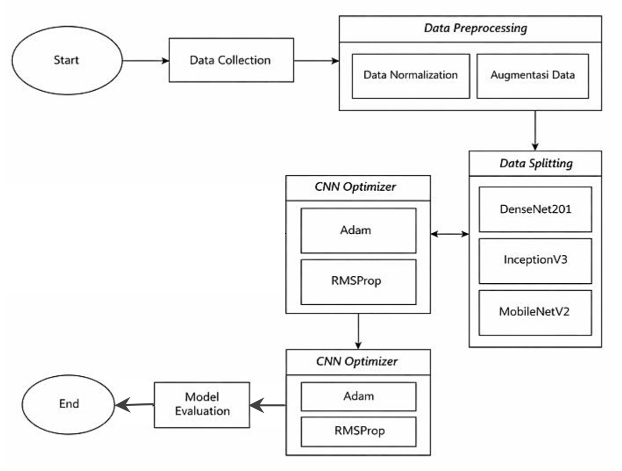

Figure 1. Research Methodology

2.1. Data Collection

At the data collection stage in this research, the dataset used comes from an open source, namely the Kaggle site. This dataset was chosen because it provides a fairly complete collection of flower images and has been classified into several types of flowers, such as daisy, dandelion, rose, sunflower, and tulip. The data obtained from this collection stage includes images with various resolutions. All data was used for training and validation of the CNN model, without involving data from case studies or other institutions. The use of the Kaggle dataset allowed the research process to be more efficient and provided a consistent standard of evaluation

2.2. Processing Data

The next process is preprocessing the X-ray image data by performing image normalization and augmentation. Image normalization is done by changing the image pixel scale from 0 to 255 to 0 to 1. This process is carried out to produce uniform image data. Thus, normalization can allow the model to process data more stably and also allow model training to be coordinated more quickly. Initially, the raw data still has different sizes or shapes, therefore, it is important to normalize the data. The data augmentation process aims to improve the data source so that the diagnostic process is more accurate. The training data carried out includes rotating random images up to 45 degrees, as well as rescaling the image pixel values to a range of 0.1 by dividing each pixel by 255, flipping the image angle n and horizontally, and filling in empty pixels by applying the nearest pixel value. As for the test data, there is no augmentation process, only rescaling the pixel values and dividing the validation data. The next stage in data preprocessing is to determine the batch size, i.e., the number of image samples generated by the generator each time it is run.

2.3. Split Data

For the next stage, the dataset is split into training, validation, and test sets. This process is an important step in preparing the dataset before testing the model. The training, validation, and test data are split using the Hold-Out technique with a 80:20 ratio. The data is split into separate paths to facilitate the process. The purpose of the following data division is to ensure that the model built can be evaluated properly and has good generalization capabilities to new data.

2.4. CNN Architecture Model

In this stage, the model testing process is carried out using a CNN architecture for flower recognition image classification. This model consists of several main layers that progressively extract and abstract important features from the image. We will apply DenseNet201, Inception V3, and MobileNetV2, and then compare the performance of each architectural model. Overall, this CNN architecture is designed to efficiently and effectively extract image features and achieve high classification accuracy, making it one of the most agile and successful approaches in image classification research.

1. DenseNet201
   DenseNet201 is a deep CNN with 201 layers, known for its dense connections that directly link each layer to all subsequent layers, enabling efficient feature sharing. Its architecture consists of Dense Blocks and Transition Layers, with Dense Blocks made up of multiple layers that maintain consistent output sizes. The network controls channel growth using Bottleneck Layers, Transition Layers, and a Growth Rate, which helps reduce parameters, prevent overfitting, and lower computational cost. This makes DenseNet201 especially effective for classifying data with small sample sizes [21].

2. InceptionV3
   Inception V3 is a Convolutional Neural Network (CNN) architecture developed by Google as an improvement over previous Inception models. It enhances computational efficiency and accuracy in image recognition by using convolution factorization (e.g., replacing 5×5 convolutions with smaller ones like 3×3 or 1×3 + 3×1). The model includes various Inception modules (Types A, B, and C) to capture multi-scale features and reduction modules to downsample spatial dimensions efficiently. It has over 40 modular layers, uses an auxiliary classifier to reduce overfitting, and applies batch normalization for stability and faster convergence. Its modular design allows for easy customization [22].

3. MobileNetV2
   MobileNetV2 is a lightweight and efficient deep learning architecture developed by Google, optimized for mobile and resource-constrained environments. It improves upon MobileNetV1 by offering better performance while maintaining efficiency. The architecture includes a feature extractor (base layer) and a classifier (upper layer). In this case, the base layer is frozen, and only the upper layer is trained for binary classification, replacing the original classification layer. MobileNetV2 features an inverted residual structure, enabling high-precision image processing with minimal latency [23].

2.5. Optimizers

Optimization algorithms play an important role in steering the training process towards optimal convergence during the development of deep learning models. Learning rate, which is controlled by the optimizer, is one of the hyperparameters that greatly affects model performance. After selecting the transfer learning approach for the CNN model architecture, this research conducted tests with two types of optimizers, Adam and RMSProp. Both are known to have advantages in speeding up the training process and improving model accuracy and stability. The Adam optimizer is effective in handling data with different gradients as it combines the advantages of momentum and adjustable learning speed. RMSProp, on the other hand, works for datasets that have noise or inhomogeneous characteristics by dynamically adjusting the learning rate for each parameter. The selection of an appropriate optimizer is critical to the design and use of this flower image classification model, as it can affect the convergence speed and final accuracy level of the built model.

1. Adam

   Adam (Adaptive Moment Estimation) is an optimization algorithm that dynamically adjusts each parameter's learning rate using a combination of RMSprop and momentum techniques. It maintains an exponential moving average of both the first-order (momentum) and second-order (variance) gradients, allowing for adaptive momentum and learning rate adjustments. This results in stable parameter updates, efficient exploration of the parameter space, and bias correction[24]. Adam is widely used due to its computational efficiency and effectiveness in handling sparse gradients and noisy data. The following is the mathematical calculation of Adam:

   $$ri + 1 = ri - \omega i\ \mu i \otimes 1/\sqrt{di} \tag{1}$$

   At each iteration, using wi = $\omega i = (\omega/\sqrt{i})$ updates the learning rate. Adam's RMSprop is reduced, and RMSProp is combined with Momentum for Adam [25].

2. RMSProp

   RMSProp (Root Mean Square Propagation) is an adaptive optimization algorithm that improves training speed and performance in deep learning. It adjusts the learning rate for each parameter individually based on the magnitude of recent gradients by maintaining a moving average of squared gradients. This normalization helps handle nonstationary objectives and sparse gradients, issues common in deep learning. RMSProp overcomes AdaGrad’s limitation of a diminishing learning rate and ensures appropriately scaled updates, making it well-suited for deep neural network training[26].

   $$v_{dw} = \beta \cdot v_{dw} + (1 - \beta) \cdot dw^2 \tag{2}$$

   $$v_{db} = \beta \cdot v_{db1} + (1 - \beta) \cdot db^2 \tag{3}$$

   $$W = W - \alpha \cdot \frac{dw}{\sqrt{v_{dw}} + \epsilon} \tag{4}$$

   $$b = b - \alpha \cdot \frac{db}{\sqrt{v_{db}} + \epsilon} \tag{5}$$

   Where β is a hyperparameter, v<sub>dw</sub> and v<sub>db</sub> are the accumulated squared gradients for weights and biases, dw and db are the gradients, α is the learning rate, and ε is a small value to prevent division by zero.

2.6. Deep Learning Model Evaluation

Model evaluation is an important stage in image classification research, which aims to measure the extent to which the trained CNN model is able to classify new images accurately. In this study, the evaluation used a confusion matrix to provide a comprehensive overview of the model's performance across each flower class. The confusion matrix provides detailed information regarding correct and incorrect predictions in each class, thus facilitating error analysis. From the confusion matrix, evaluation metrics such as accuracy, precision, recall, and F1-score are obtained, which are the main indicators in assessing model performance. Through this evaluation, certain error patterns can be identified that are useful for improving the model through architectural adjustments, hyperparameter settings, or data augmentation strategies. The entire evaluation process was conducted using the interest dataset obtained from Kaggle, without reference to a specific case study.

3. RESULTS AND DISCUSSION

This section presents the results and analysis of flower classification using three CNN architectures: DenseNet201, InceptionV3, and MobileNetV2, tested with Adam and RMSProp optimizers. The DenseNet201 with Adam achieved the highest accuracy (90%) and best overall performance, showing that architecture and optimizer choice significantly affect model effectiveness in flower image classification.

3.1. Data Collection

In this research, using the Flowers Recognition dataset which has 5 classes, namely Daisy, Dandelion, Rose, Sunflower, and Tulip. This dataset is available on Kaggle as a training dataset for CNN deep learning research. This dataset is used to be able to identify types of flowers based on the classes in the Flowers

Recognition dataset. Table 1 is classification of flower recognition and Figure 2 is a visualization of the dataset used in this study.

Table 1. Classification of Flower Recognition

No	Class	Data
1.	Daisy	764
2.	Dandelion	1052
3.	Rose	784
4.	Sunflower	733
5.	Tulip	984
	Total	4.317

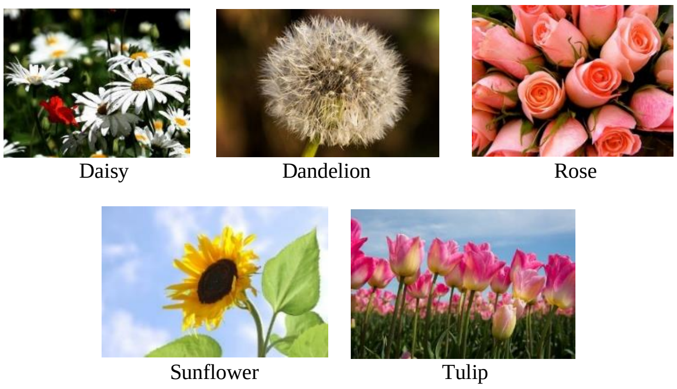

Figure 2. Visualization of Flowers Recognition Dataset

3.2. Preprocessing

The next stage is data preprocessing, which consists of 2 processes, namely normalization and augmentation of the image. In the image normalization process, rescale the image pixel values from the range 0-255 to the range 0-1, which aims to make each pixel value have the same data distribution. Then, the augmentation process in this study aims to add data variations by performing transformations such as image rotation by 45° and flipping the image horizontally. For random image deviation, it is determined by sliding angles up to 15%, enlarging random images up to 15%, flipping images vertically and horizontally, and filling in empty pixels using the closest pixel value. The composition of the augmented data can be seen in Figure 3.

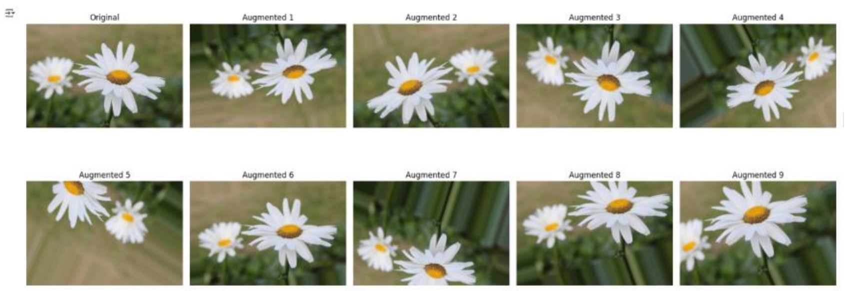

Figure 3. Augmentation Result

With this augmentation, the resulting images will have various variations in position, orientation, and scale, aiming to maintain consistency in image quality as seen in Figure 3.1. This process is very beneficial in improving the model's ability to recognize flower characteristics from different angles and image conditions.

3.3. Data Sharing Proses

The data division used in deep learning is divided into three types: training data, validation data, and testing data. Training data is used to train the deep learning model to find out the types of flowers, while validation data serves to evaluate the performance of the model during the training process carried out at the end of each epoch. Testing data is a separate data set that is not involved in the training or validation process, which aims to assess the final performance of the deep learning model. Data division is done using the holdout technique with a ratio of 80:20, where 80% is allocated for training data and 20% for testing data. Furthermore, the training data is further divided into training data and validation data with a ratio of 80:20, which is 80% for training and 20% for validation. The distribution of data division with the holdout technique can be seen in Table 2.

Tabel 2. Result of 80:20 Hold Out Data Sharing Data

Hold Out	80:20		
Class	Training Data	Validation Data	Testing Data
Daisy	611	76	77
Dandelion	841	105	106
Rose	627	78	79
Sunflower	586	73	74
Tulip	787	98	99
Total	3452	430	435

3.4. Modelling and Visualizing

In this modeling, three popular Convolutional Neural Network (CNN) architectures, namely DenseNet201, Inception V3, and MobileNetV2, are used to classify flower images into five classes: Daisy, Dandelion, Rose, Sunflower, and Tulip. All architectures used are pre-trained models without a top classification layer, so they can be readjusted with custom layers such as GlobalAverage Pooling 2D, Dense (512, activation='ReLU'), Dropout (25%), and Dense(5, activation='softmax') as the output layer. The softmax activation function is used because the classification is performed on five flower classes. The purpose of adding dropout is to reduce overfitting and improve model generalization.

In the training process, two types of optimizers are used, namely Adam and RMSProp, with a learning rate of 0.0001 and a batch size of 16. The training process is carried out for 15 epochs, where at each epoch the model will learn the pattern of the training data and update its weights based on the validation results. The loss function used is 'categorical_crossentropy' because the classification is multiclass, and the evaluation metric monitored is accuracy. The training process of each architecture and its optimizer combination can be seen in Figures 4 to 9, which shows the progress of the model performance during the training and validation process.

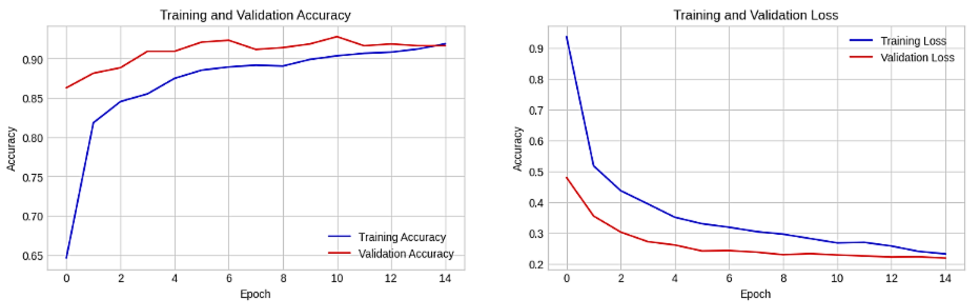

Figure 4. Deep Learning DenseNet201 Optimizer Adam Model Curve

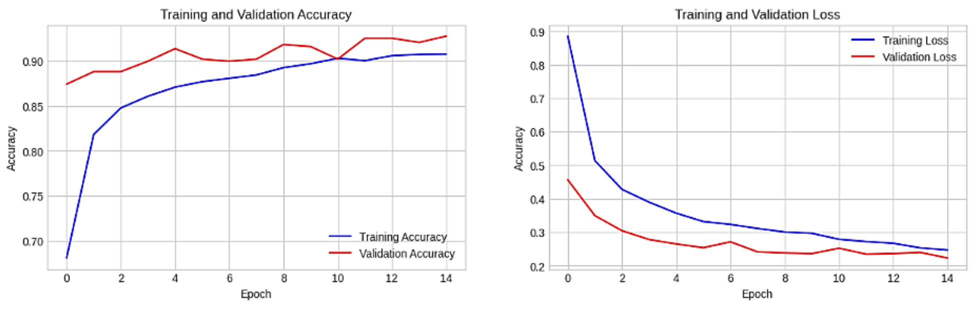

Figure 5. Deep Learning DenseNet201 Optimizer RSMProp Model Curve

Figures 4 and 5 show that DenseNet201 performs strongly and consistently with both Adam and RMSProp optimizers. With Adam, training accuracy converges at around 91% and validation accuracy at approximately 92%, accompanied by smoothly decreasing loss curves that indicate stable learning and good generalization. Using RMSProp, performance improves slightly, with training accuracy reaching about 92% and validation accuracy 93%, while closely aligned loss curves suggest better convergence. Overall, both optimizers are effective, although RMSProp provides a marginal advantage in accuracy and training stability.

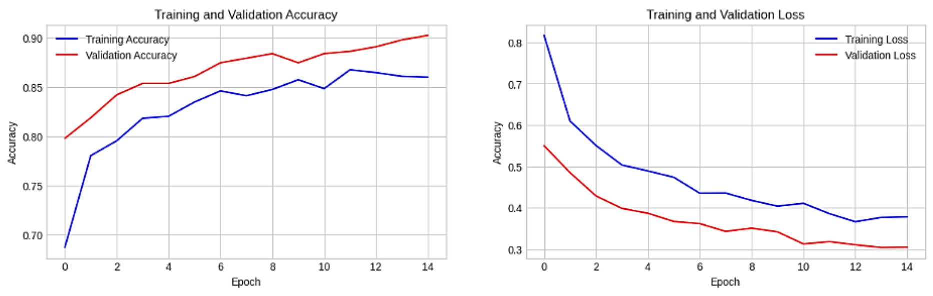

Figure 6. Deep Learning InceptionV3 Optimizer RSMProp Model Curve

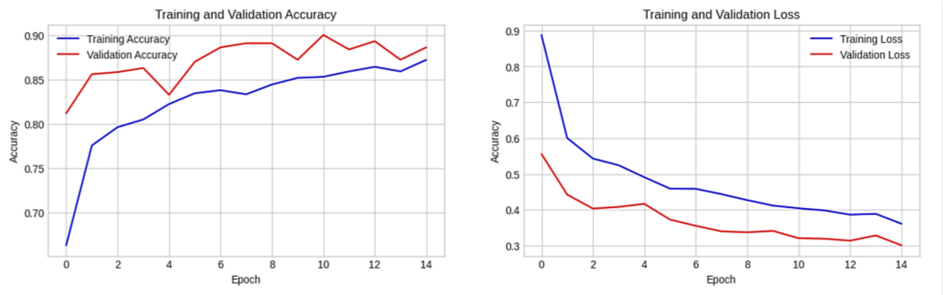

Figure 7. Deep Learning InceptionV3 Optimizer Adam Model Curve

Figures 6 and 7 show that InceptionV3 exhibits stable performance with both RMSProp and Adam optimizers. Using RMSProp, training accuracy reaches approximately 87% with validation accuracy around 90%, accompanied by steadily decreasing and closely aligned loss curves, indicating good generalization. With Adam, training accuracy slightly improves to about 88%, while validation accuracy remains around 89–90%, with smooth loss reduction. Overall, both optimizers perform reliably, with RMSProp offering slightly better validation stability and Adam providing marginally higher training accuracy.

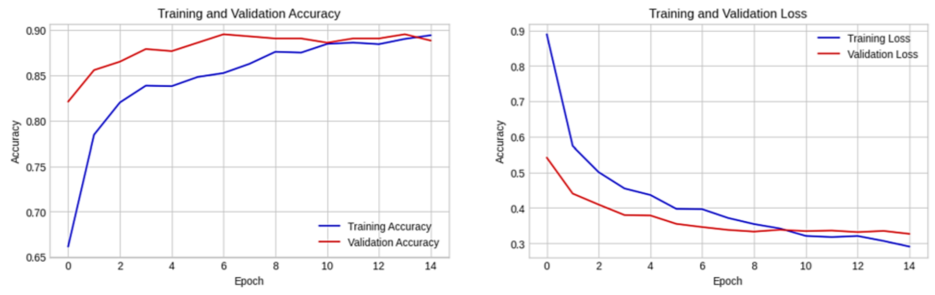

Figure 8. Deep Learning MobileNetV2 Optimizer RMSProp Model Curve

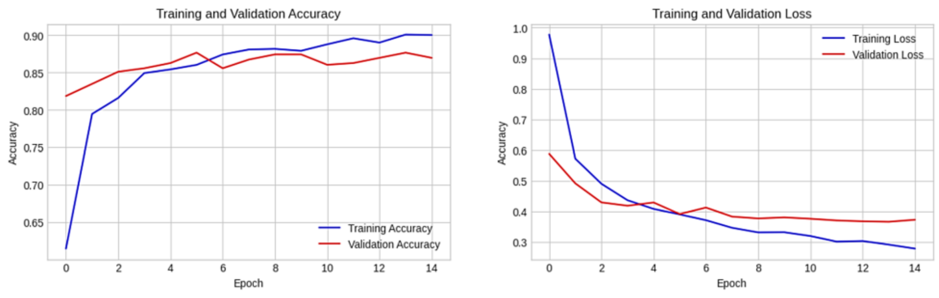

Figure 9. Deep Learning MobileNetV2 Optimizer Adam Model Curve

Figures 8 and 9 indicate that MobileNetV2 performs stably with both RMSProp and Adam optimizers. Using RMSProp, training accuracy stabilizes at around 88% with validation accuracy reaching approximately 90%, supported by closely aligned and consistently decreasing loss curves that indicate good generalization. With Adam, training accuracy slightly improves to about 89%, but validation accuracy decreases to around 87%, accompanied by minor loss fluctuations. Overall, RMSProp provides better validation performance and generalization for MobileNetV2, while Adam shows slightly stronger fitting to the training data. Table 3 is model training accuracy results.

Table 3. Model Training Accuracy Results

Arsitektur	Optimizer	Train Accuracy	Valid Accuracy	Testing Accuracy	Training Loss	Valid Loss	Testing Loss
DenseNet201	Adam	92.61%	92.79%	89.42%	0.2095	0.2229	0.2985
	RMSprop	92.67%	91.62%	89.88%	0.2168	0.2193	0.2857
InceptionV3	Adam	87.80%	89.76%	85.28%	0.3327	0.3038	0.3870
	RMSprop	87.86%	88.60%	84.82%	0.3385	0.3007	0.3933
MobileNetV2	Adam	91.19%	88.83%	90.11%	0.2651	0.3267	0.2787
	RMSprop	91.65%	87.67%	8804%	0.2485	0.3659	0.3559

3.5. Evaluation

Next, evaluate the model to classify data using Confusion Matrix. The results of the confusion matrix of the DenseNet201 architecture using the RMSProp and Adam optimizers can be seen in Figures 10, The results of the confusion matrix of the Inception V3 architecture using the RMSProp and Adam optimizers can be seen in Figures 11, The results of the confusion matrix of the MobileNetV2 architecture using the RMSProp and Adam optimizers can be seen in Figures 12.

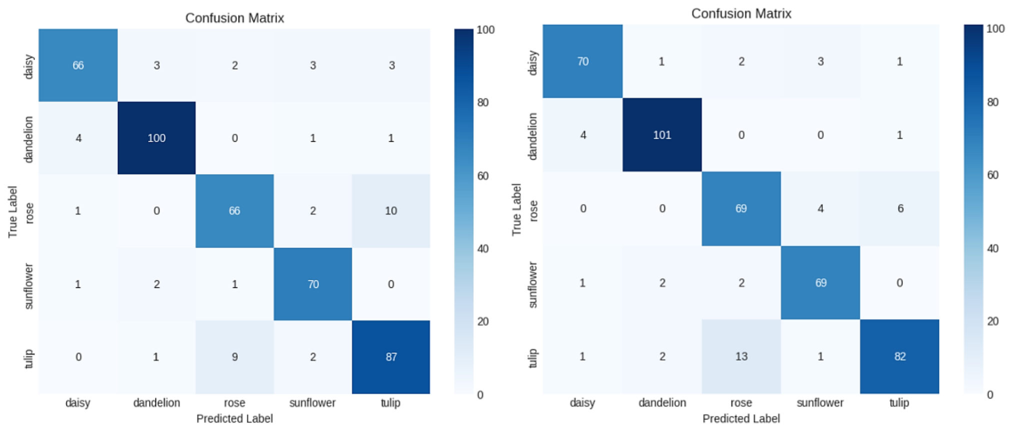

Figure 10. Confusion Matrix of Adam,s, and RMSProp DenzeNet201 Optimizer Architecture

Based on Figure 10, DenseNet201 shows good classification performance with both RMSProp and Adam optimizers. Using RMSProp, the model achieves an accuracy of 84.8%, with dandelion as the best-performing class (94.34%) and rose as the lowest (78.57%), mainly due to confusion with the tulip class. With Adam, overall accuracy improves to 86.0%, while dandelion remains the best-performing class (95.28%) and the lowest accuracy shifts to tulip (82.0%) due to misclassification as rose and sunflower. Overall, most errors occur among visually similar flower classes, whereas distinctive classes are consistently classified accurately.

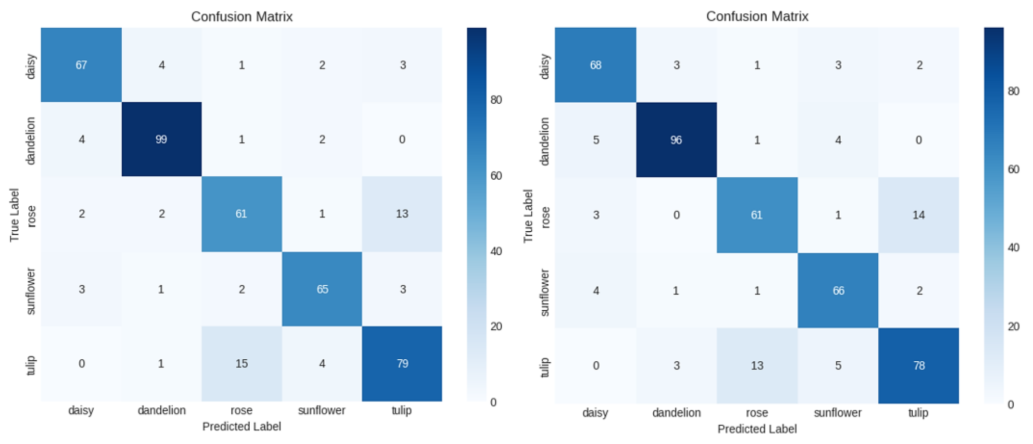

Figure 11. Confusion Matrix of Adam,s, and RMSProp InceptionV3 Optimizer Architecture

Based on Figure 11, InceptionV3 shows stable performance with both RMSProp and Adam optimizers. Using RMSProp, the model achieves 84.2% accuracy, with dandelion as the best-performing class (94.29%) and tulip as the lowest (79.0%), mainly misclassified as rose. With Adam, overall accuracy improves to 85.6%, the best class shifts to daisy (90.67%), while tulip remains the most challenging (78.0%). Overall, classification errors are concentrated among visually similar classes, with Adam providing a slight accuracy improvement.

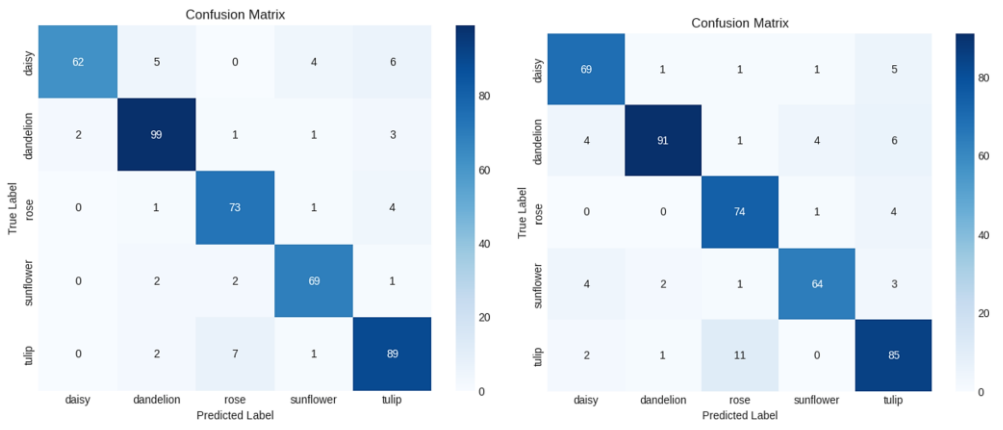

Figure 12. Confusion Matrix of Adam,s, and RMSProp MobileNetV2 Optimizer Architecture

Based on Figure 12, MobileNetV2 shows consistent performance with both RMSProp and Adam optimizers. Using RMSProp, the model achieves 84.8% accuracy, with dandelion as the best-performing class (94.29%), indicating strong feature discrimination, while daisy records the lowest accuracy (77.5%) due to misclassification as tulip and sunflower. With Adam, overall accuracy increases to 86.6%; dandelion remains the best-performing class (91.07%), and the lowest accuracy shifts to tulip (77.98%), often misclassified as rose. Overall, classification errors mainly occur among visually similar flower classes, with detailed metric results presented in Table 4.

Table 4. CNN Model Evaluation Result

Arsitektur	Optimizer	Kelas	Accuracy	Precision	Recall	F1-Score	Support
DenseNet201	RMSProp	Daisy	89%	0.92	0.86	0.89	77
		Dandelion		0.94	0.94	0.94	106
		Rose		0.85	0.84	0.84	79
		Sunflower		0.90	0.95	0.92	74
		Tulip		0.86	0.88	0.87	99
	Adam	Daisy	90%	0.92	0.91	0.92	77
		Dandelion		0.95	0.95	0.95	106
		Rose		0.80	0.87	0.84	79
		Sunflower		0.90	0.93	0.91	74
		Tulip		0.91	0.83	0.87	99
InceptionV3	RMSProp	Daisy	85%	0.88	0.87	0.88	77
		Dandelion		0.93	0.93	0.93	106
		Rose		0.76	0.77	0.77	79
		Sunflower		0.88	0.88	0.88	74
		Tulip		0.81	0.80	0.80	99
	Adam	Daisy	85%	0.85	0.88	0.87	77
		Dandelion		0.93	0.91	0.92	106
		Rose		0.78	0.77	0.78	79
		Sunflower		0.84	0.89	0.86	74
		Tulip		0.81	0.79	0.80	99
MobileNetV2	RMSProp	Daisy	90%	0.97	0.81	0.88	77
		Dandelion		0.91	0.93	0.92	106
		Rose		0.88	0.92	0.90	79
		Sunflower		0.91	0.93	0.92	74
		Tulip		0.86	0.90	0.88	99
	Adam	Daisy	88%	0.87	0.90	0.88	77
		Dandelion		0.96	0.86	0.91	106
		Rose		0.84	0.94	0.89	79
		Sunflower		0.91	0.86	0.89	74
		Tulip		0.83	0.86	0.84	99

Based on the evaluation results of three CNN architectures, the InceptionV3 architecture, with both the RMSProp and Adam optimizers, achieved 85% accuracy. However, the performance of this model tends to be lower than DenseNet201 and MobileNetV2, especially in the Rose and Tulip classes. With RMSProp, the highest F1-Score was recorded for the Dandelion class (0.93), followed by Daisy (0.88), Sunflower (0.88), Tulip (0.80), and Rose (0.77). While with Adam, the highest F1-Score is still in the Dandelion class (0.92), followed by Sunflower (0.86), Daisy (0.87), Tulip (0.80), and Rose (0.78). The MobileNetV2 architecture performed very well with 90% accuracy when using the RMSProp optimizer. In this configuration, the highest F1-score was achieved in the Dandelion (0.92) and Sunflower (0.92) classes, followed by Rose (0.90), Tulip (0.88), and Daisy (0.88). While with Adam's optimizer, the accuracy slightly decreased to 88%, with the highest F1-Score in the Dandelion class (0.91), then Sunflower (0.89), Daisy (0.88), Rose (0.89), and Tulip (0.84).

Thus, it can be concluded that the architecture that has the highest accuracy is DenseNet201 with Adam optimizer at 90%, showing the best classification performance in general. Followed by MobileNetV2 with RMSProp, which also obtained 90% accuracy, but with a stability score slightly below DenseNet201. The architecture with the lowest accuracy of the three was Inception V3, with both RMSProp and Adam, which only achieved 85% accuracy.

3.6 Comparison of Evaluation Result Accuracy

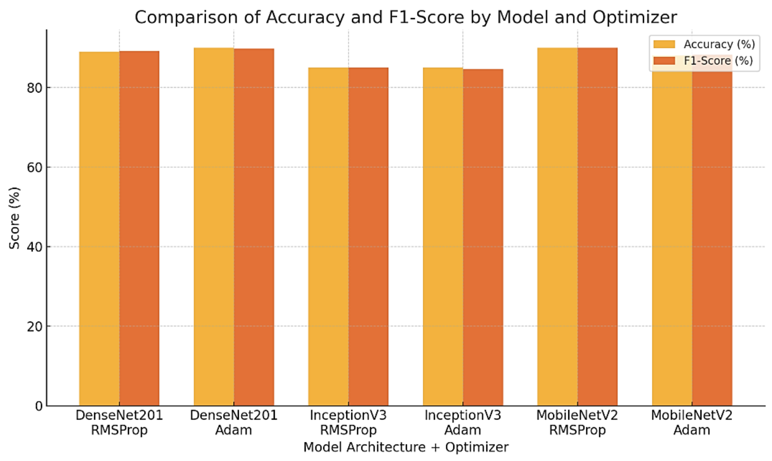

Figure 13. Comparison of CNN Evaluation Accuracy

Figure 13 is a visualization of the Accuracy and F1-score comparison for each combination of model architecture and optimizer based on the CNN evaluation table. This chart helps compare the performance of DenseNet201, InceptionV3, and MobileNetV2 models optimized using RMSProp and Adam.

3.7 Misclassification Pattern Analysis Across Flower Classes

Although the overall model performance is satisfactory, confusion matrix analysis reveals consistent misclassification patterns among several flower classes. The dandelion and sunflower classes achieve the highest accuracy across all models, indicating that these flower types possess more distinctive visual features such as petal structure, shape, and color contrast. These characteristics allow CNN models to recognize them more easily and consistently compared to other classes.

In contrast, the rose and tulip classes exhibit higher confusion rates, primarily due to similarities in petal shape, color distribution, and overall floral appearance. A recurring misclassification occurs between these two classes, suggesting that the models struggle to capture subtle visual differences between them. Additionally, some confusion is observed between the daisy and sunflower classes, which share radial symmetry and similar petal arrangements, although sunflower images generally produce more confident predictions.

Among the evaluated architectures, DenseNet201 demonstrates superior class separation, particularly for visually similar classes such as rose and tulip. This advantage is likely due to its dense connectivity structure, which enables better feature reuse and deeper representation learning. MobileNetV2 also shows competitive performance, especially considering its lightweight and efficient design, while InceptionV3 exhibits relatively lower robustness in distinguishing fine-grained visual patterns. These findings highlight the need for enhanced feature extraction strategies, such as deeper fine-tuning, attention mechanisms, and more diverse data augmentation, to further reduce misclassification errors and improve model generalization.

3.8 Discussion

This study evaluates three CNN architectures, MobileNetV2, InceptionV3, and DenseNet201, using Adam and RMSProp optimizers for flower image classification. DenseNet201 with Adam achieved the highest accuracy (90%), followed by MobileNetV2 with RMSProp, which reached the same accuracy but showed slightly less stable F1-scores. InceptionV3 consistently produced the lowest performance (85%) with both optimizers.

The results indicate that optimizer effectiveness depends on architectural characteristics. DenseNet201, with its deep and densely connected layers, benefits from Adam, which provides stable parameter updates through adaptive learning rates. In contrast, MobileNetV2, as a lightweight architecture, shows better generalization with RMSProp, which helps control overfitting in compact models. The similar performance of InceptionV3 across both optimizers suggests that additional fine-tuning is needed to fully leverage its architecture.

Compared to prior studies reporting higher accuracy using fine-tuned models such as NasNetMobile, VGG16, and InceptionResNetV2, the lower results in this study are mainly due to the absence of extensive fine-tuning. Nevertheless, this work contributes by systematically analyzing the interaction between CNN architectures and optimizers. Class-level analysis further shows that dandelion and sunflower are consistently well classified, while rose and tulip remain more challenging due to visual similarity. Overall, CNNs are effective for flower classification, but optimal performance requires appropriate alignment between architecture, optimizer, and training strategy.

4. CONCLUSION

This study addresses its primary objective by systematically evaluating and comparing the performance of three CNN architectures, DenseNet201, InceptionV3, and MobileNetV2, combined with two optimization algorithms, Adam and RMSProp, for flower image classification. The experimental results demonstrate that DenseNet201 with the Adam optimizer achieves the best overall performance, while MobileNetV2 with RMSProp provides competitive accuracy with improved computational efficiency. In contrast, InceptionV3 yields lower and more consistent performance across both optimizers.

The novel contribution of this research lies in its systematic analysis of the interaction between CNN architectures and optimizer selection within a unified experimental framework. Unlike previous studies that focused on a single architecture or relied heavily on fine-tuning, this study highlights how optimizer choice influences classification performance differently depending on architectural characteristics. The class-level evaluation further reveals that visually distinctive flower classes, such as dandelion and sunflower, are consistently classified with high accuracy, whereas rose and tulip remain challenging due to visual similarity.

The strengths of this study include a clear comparative evaluation of multiple CNN architectures, the use of multiple performance metrics, and an analysis of class-wise misclassification patterns. However, this research also has several limitations. The experiments are conducted using a single dataset, without validation on more diverse or imbalanced datasets. Additionally, advanced training strategies such as extensive fine-tuning, learning rate scheduling, and systematic hyperparameter optimization are not applied, and the training process is limited to a fixed number of epochs.

Future research should explore deeper and more complex architectures, such as NasNetMobile and InceptionResNetV2, combined with comprehensive fine-tuning, adaptive learning rate strategies, and advanced data augmentation. Evaluating the models on multiple datasets and using cross-validation is also recommended to improve robustness and generalization. These directions are expected to further enhance the performance and practical applicability of CNN-based flower classification systems.

REFERENCES

[1] Legito, R. Nuraini, L. Judijanto, dan A. I. Lubis, “The Application of Convolutional Neural Networks in Floristic Recognition,” Int. J. Softw. Eng. Comput. Sci., vol. 3, no. 3, hal. 520–528, 2023, doi: 10.35870/ijsecs.v3i3.1827.

[2] W. Fu dan Z. Chen, “A globally convergent SQP-type method with least constraint violation for nonlinear semidefinite programming,” Optimization, vol. 74, no. 10, hal. 2333–2367, Jul 2025, doi: 10.1080/02331934.2024.2345397.

[3] A. C. R. Ngo, C. Conrad, Á. G. Baraibar, A. Matura, K. H. van Pée, dan D. Tischler, “Characterization of two hydrogen peroxide resistant peroxidases from rhodococcus opacus 1cp,” Appl. Sci., vol. 11, no. 17, 2021, doi: 10.3390/app11177941.

[4] M. M. Sedky, W. E. Abdel-Azim, A. S. Abdel-Khalik, dan A. M. Massoud, “Integrated Switched Reluctance Starter/Generator for Aerospace Applications: Particle Swarm Optimization for Constant Current and Constant Voltage Control Designs,” Appl. Sci., vol. 12, no. 15, hal. 7583, Jul 2022, doi: 10.3390/app12157583.

[5] I. Patel dan S. Patel, “An optimized deep learning model for flower classification using NAS-FPN and faster RCNNa,” Int. J. Sci. Technol. Res., vol. 9, no. 3, hal. 5308–5319, 2020.

[6] M. Alom, M. Y. Ali, M. T. Islam, A. H. Uddin, dan W. Rahman, “Species classification of brassica napus based on flowers, leaves, and packets using deep neural networks,” J. Agric. Food Res., vol. 14, no. November 2022, hal. 100658, 2023, doi: 10.1016/j.jafr.2023.100658.

[7] A. Srinivasan, V. Sasirekha, S. L. Krishna, V. R. Reddy, dan T. Vengatesh, “Deep Learning for Plant Species Classification Urooj Arif , Shaik Aamena Thanveer , P . Rakshitha , N . Gowthami , Amatul Ali Sameera , Image Image Feature Classification,” vol. 14, no. 15, hal. 14–21, 2025.

[8] L. Zhou, N. Wu, H. Chen, dan Q. Wu, “applied sciences RRT * -Fuzzy Dynamic Window Approach (RRT * -FDWA) for Collision-Free Path Planning,” 2023.

[9] L. Picado-santos, “applied sciences Mechanical Performance of Cement Bound Granular Mixtures Using Recycled Aggregate and Coconut Fiber,” 2025.

[10] H. Mo dan L. Wei, “SA-ConvNeXt : A Hybrid Approach for Flower Image Classification Using Selective Attention Mechanism,” 2024.

[11] Y. Lu dan K. Yi, “applied sciences Enhancing Fine-Grained Image Recognition with Multi-Channel Self-Attention Mechanisms : A Focus on Fruit Fly Species Classification,” 2024.

[12] D. Kobiela, J. Groth, M. Hajdasz, dan M. Erezman, “Vehicle type recognition: a case study of MobileNetV2 for an image Classification task,” Procedia Comput. Sci., vol. 246, no. C, hal. 3947–3956, 2024, doi: 10.1016/j.procs.2024.09.169.

[13] V. S. Syamala dan G. S. Kumar, “IMAGE CLASSIFICATION OF THE FLOWER SPECIES Page No : 1239 ISSN NO : 0377-9254 Page No : 1240,” vol. 13, no. 06, hal. 1239–1246, 2022.

[14] A. N. R. Munandar dan A. F. Rozi, “Analisis Arsitektur Convolutional Neural Network Untuk Klasifikasi Citra Bunga,” J. Teknol. Dan Sist. Inf. Bisnis, vol. 6, no. 3, hal. 522–531, 2024, doi: 10.47233/jteksis.v6i3.1413.

[15] Y. A. Bezabh, A. M. Ayalew, B. M. Abuhayi, T. N. Demlie, E. A. Awoke, dan T. E. Mengistu, “Classification of mango disease using ensemble convolutional neural network,” Smart Agric. Technol., vol. 8, no. May, hal. 100476, 2024, doi: 10.1016/j.atech.2024.100476.

[16] J. Sumpena, “A Comparative study of Transfer Learning CNN for Flower Type Classification,” J. Appl. Intell. Syst., vol. 8, no. 3, hal. 389–399, 2023, doi: 10.33633/jais.v8i3.9380.

[17] J. Sebastian, J. Pablo, L. Fu, dan F. Auat, “Deep Learning based flower detection and counting in highly populated images : A peach grove case study Deep Learning based flower detection and counting in highly populated images : A peach grove case study,” J. Agric. Food Res., vol. 15, hal. 100930, 2024, doi: 10.1016/j.jafr.2023.100930.

[18] X. Qu, “Flower Species Classify System Based on Deep Learning,” Adv. Transdiscipl. Eng., vol. 51, hal. 97–101, 2024, doi: 10.3233/ATDE240063.

[19] S. Cao dan B. Song, “Visual attentional-driven deep learning method for flower recognition,” vol. 18, no. November 2020, hal. 1981–1991, 2021, doi: 10.3934/mbe.2021103.

[20] B. E. Bic, “Flower Recognition Using Convolutional Neural Network,” 2019.

[21] S. Desai, C. Gode, dan P. Fulzele, “Flower Image Classification Using Convolutional Neural Network,” 2022 1st Int. Conf. Electr. Electron. Inf. Commun. Technol. ICEEICT 2022, 2022, doi: 10.1109/ICEEICT53079.2022.9768635.

[22] F. BOZKURT, “A Study on CNN Based Transfer Learning for Recognition of Flower Species,” Eur. J. Sci. Technol., no. January, 2022, doi: 10.31590/ejosat.1039632.

[23] P. M. Shanthappa, M. Bayari, G. B. Abhilash, K. V. Gokul, dan P. J. Ashish, “Parkinson’s disease detection using inceptionV3: A Deep learning approach,” MethodsX, vol. 14, no. April, hal. 103333, 2025, doi: 10.1016/j.mex.2025.103333.

[24] R. Elshamy, O. Abu-Elnasr, M. Elhoseny, dan S. Elmougy, “Improving the efficiency of RMSProp optimizer by utilizing Nestrove in deep learning,” Sci. Rep., vol. 13, no. 1, hal. 1–16, 2023, doi: 10.1038/s41598-023-35663-x.

[25] J. Zhou et al., “Intelligent classification of maize straw types from UAV remote sensing images using DenseNet201 deep transfer learning algorithm,” Ecol. Indic., vol. 166, no. June, hal. 112331, 2024, doi: 10.1016/j.ecolind.2024.112331.

[26] J. Kang, X. Zhu, L. Shen, dan M. Li, “Fault diagnosis of a wave energy converter gearbox based on an Adam optimized CNN-LSTM algorithm,” Renew. Energy, vol. 231, no. November 2023, hal. 121022, 2024, doi: 10.1016/j.renene.2024.121022.