

Applying convolutional neural network and Nadam optimization in flower classification

Qurrotul Aini¹, Zulfiandri¹, Rezky Firmansyah², Yunifa Miftachul Arif³

¹Department of Information Systems, Faculty of Science and Technology, UIN Syarif Hidayatullah, Jakarta, Indonesia

²Information Technology Support Division, Bekasi Municipal Communication and Information Office, Bekasi, Indonesia

³Department of Informatics Engineering, Faculty of Science and Technology, UIN Maulana Malik Ibrahim, Malang, Indonesia

Article Info

Article history:

Received Mar 11, 2023

Revised Nov 16, 2023

Accepted Jan 11, 2024

Keywords:

Convolutional neural network

Transfer learning

ResNet

Nadam optimization

Flower classification

ABSTRACT

Flowers have a variety of shapes, colors and structures, the images of which need to be classified using guided learning techniques. Several studies classify flowers using machine learning, but their accuracy performance is not good. The thing is, the flowers come in a variety of colors that can sometimes look similar to the background. Therefore, this study aims to classify flowers using a convolutional neural network (CNN) and measure its performance. The method used is mixed methods by collecting existing data from previous studies and connecting it with the realities in the field. The Kozłowski and Steinbrener models were used, while the image data was obtained from the Oxford17 and Oxford102 dataset with 17 and 102 flower types, respectively. The results show 60% and 84% accuracy of CNN using the scratch and transfer learning approach for the Oxford17 dataset. The Oxford102 dataset shows 42% and 64%, respectively, with CNN from baseline and transfer learning.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Qurrotul Aini

Department of Information Systems, Faculty of Science and Technology, UIN Syarif Hidayatullah

Jakarta, Indonesia

Email: qurrotul.aini@uinjkt.ac.id

1. INTRODUCTION

Flowers are part of plants that produce seeds and undergo pollination and fertilization to develop into a fruit. Many species of flowers have various shapes, colors, and structures usually classified traditionally by botanists. However, the classification of faces problems because flowers have different colors, sometimes similar to their background, making their identification difficult.

Machine learning has been widely used for various purposes and needs, such as classifying flowers to cut time and ensure high accuracy. Several studies classified flowers using machine learning in a particular neural network [1]-[3]. Previous study proposed the classification model of flower images by applying deep convolutional neural network (CNN) to extract the features. It also applied image augmentation for better performance results. The commonly utilized method known as CNN processes incoming data through a number of hidden layers using artificial neural network (ANN) algorithms. One of the often employed deep learning algorithms for picture categorization is CNN. The three major layers of a CNN architecture, the convolutional layer, the pooling layer, and the fully connected layer, can be built into a variety of other topologies.

The flower images used Oxford17 and Oxford102 datasets divided into the training and test sets by 0.8 and 0.2, respectively. The performance results were compared with other techniques, such as support vector machine (SVM) and multi-layer perceptron (MLP) classifiers. It showed that the accuracy with CNN was less than SVM and MLP classifiers [1]. Furthermore, Almodgady *et al.* [2] introduced a flower

recognition system for the Oxford102 dataset using ANN. The images were applied segmentation to part the flower object from the background. Moreover, the chan-ve-se algorithm was used, while the classification process applied back-propagation ANN and achieved an accuracy rate of 81.19%. Another previous study was conducted on a 9500 flower images dataset using CNN. The dataset was categorized into four groups and then trained into five groups, and the testing was conducted on all the datasets. The different CNN architectures were designed and tested to obtain better accuracy. Also, various pooling schemes were implemented to improve the classification rates, resulting in a recognition rate of 97.78% [3]. Another study on the modeling process used two approaches with the CNN model architecture and the transfer learning method with ResNet. The results showed that the transfer learning approach has the best performance [4]. The Nesterov-accelerated adaptive moment estimation (Nadam) algorithm is used to achieve optimal training results. This algorithm saves computational resources and performs better than other deep learning optimization algorithms [5].

Supervised learning techniques are needed to classify the types of flowers from all inputs using machine learning. To achieve better performance with the Oxford17 and Oxford102 datasets, a new model for the classification of flowers is proposed. First, data preprocessing is done by gathering, labeling, resizing the pixels of images, and augmenting the image from the entire training image. Second, build the CNN model from scratch and transfer learning ResNet. Third, optimize the training process, which requires a loss evaluation based on data categories using Nadam. Fourth, build testing interface of the model based on Android application to obtain the metric performance of flowers. The CNN and Nadam optimization have the following advantages: i) the training data is in the form of categories; therefore, the loss evaluation is carried out based on data categories and ii) Nadam optimization is more efficient, effective, and does not use many resources.

2. METHOD

This research is quantitative, which explores a problem and solves it in a factual or characteristic manner in certain populations or fields factually and carefully. The research framework is shown in Figure 1, which started with providing data, then data preprocessing, designing CNN architecture, CNN model implementation, and testing to get the accuracy level.

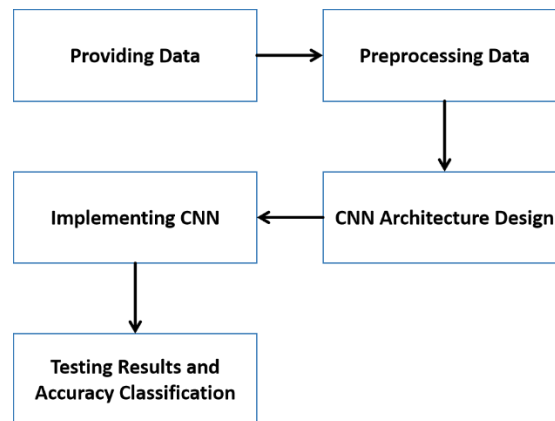


Figure 1. Research framework

2.1. Providing data

This study uses a mixed method of qualitative research and quantitative research. Qualitative research is research by collecting existing data from existing research. This research can also obtain data from interviews and direct observation of the object of research. The gathered information will be merged into one unit so that a conclusion and more details can be drawn.

2.2. Preprocessing data

We can determine which data can be used as a research sample by looking at the data that has been gathered. With respect to initial learning and transfer, the sample's accuracy for the Oxford102 dataset is 42% and 64%, respectively.

- a. Data cleaning

When the information gathered is put together, missing values get filled in, noisy data is smoothed, and any inconsistencies are resolved. Data can also be cleaned by separating it into similar-sized pieces and then being smoothed.

b. Data integration

The vast amount of data that has been gathered yields a variety of justifications and conclusions. At this point, the data are coordinated to enhance one another and produce more precise and comprehensive data. Figure 2 illustrates the data integration process which using the dynamic link library (DLL) approach.

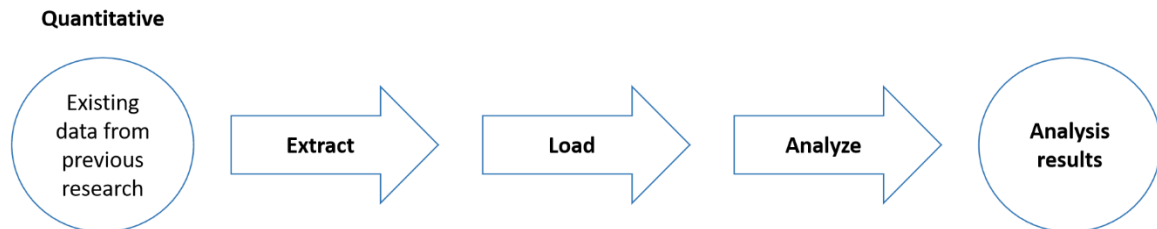


Figure 2. Data integration process using the DLL approach

2.3. Convolutional neural network architecture design

CNN is a deep learning method which analyzes incoming picture data, adds weights and biases that can be learned to various features of the image, and assists in distinguishing between different objects in the image. One common neural network design for data images is CNN, and the design explained in subsection 3.2.

3. BENCHMARK DATASET

3.1. Dataset

This study used the 17 category flower dataset known as Oxford17 [6], [7]. The image consisted of 17 flower plants, including Daffodil, Snowdrop, Lily Valley, Bluebell, Crocus, Tigerlily, Tulip, Fritillary, Sunflower, Daisy, Colts Foot, Dandelion, Cowslip, Buttercup, Windflower, and Pansy, each with 80 images. This dataset had a total of 1,360 images, shown in Figure 3. The 102 category flower dataset known as Oxford102 was also used [8]. It consisted of 102 flower plants, including Pink Primrose, Hard-leaved Pocket Orchid, Canterbury Bells, Sweet Pea, English Marigold, Tiger Lily, Moon Orchid, Bird of Paradise, Monkshood, Globe Thistle, Snapdragon, Coltsfoot, King Protea, and Spear Thistle, each with 40 to 258 images. This dataset had a total of 8,189 images. The performance of the proposed method was investigated by splitting the entire images into 80% and 20% for training and testing data, respectively. Figure 3(a) displays samples of dataset images from Oxford17, while Figure 3(b) displays images from Oxford102.

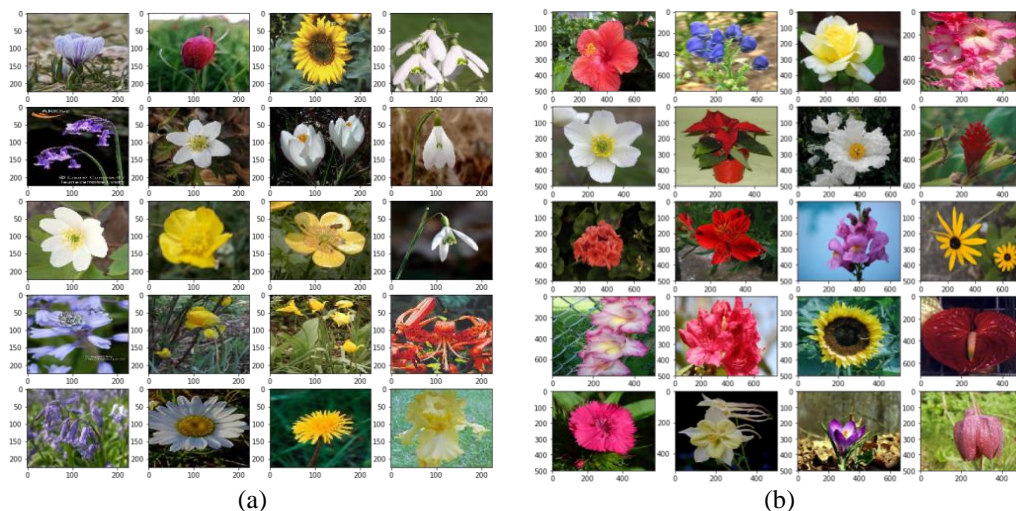


Figure 3. Sample image from dataset; (a) Oxford17 and (b) Oxford102

3.2. Convolutional neural network architecture design

This study implemented CNN from scratch and with ResNet50 transfer learning proposed by Steinbrener *et al.* [9], Kozłowski *et al.* [10]. GoogleNet machine learning model was employed as the basis for transfer learning, while the modeling process used Python with Keras library [11]-[13]. The proposed model used four convolutional and three fully connected layers Figure 4. The first layer accepts an image input with 224×224 pixels and 3 RGB channels. Furthermore, the first convolution process was conducted, as shown in Figure 4 point 1. The process involved extracting features from the image with 16 filters, 5×5 kernel, and padding valid using the rectified linear unit (ReLU) activation function. The pooling process was conducted by reducing the image to 2×2 and stride 2×2 using max-pooling to produce an image 50% smaller. This layer produced an image with a size of 64×64 pixels. The second layer received 64×64 pixels image input, while the convolution process was conducted to extract the features from the image with 32 filters, 5×5 kernel, and padding valid using the ReLU activation functions. Similarly, this layer used pooling with a size of 2×2, stride 2×2, and max pooling. Dropout layers were also used to avoid over-fitting, as shown in Figure 4 point 2[14], [15].

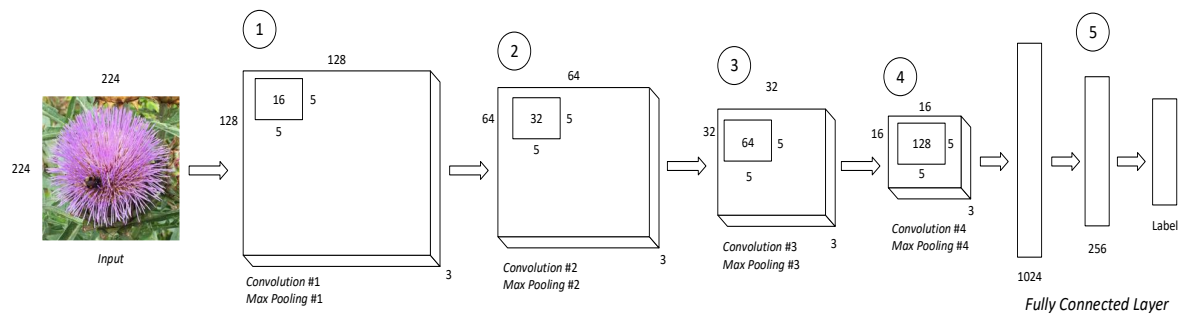


Figure 4. Proposed CNN design

The third layer in Figure 4 point 3 received 32×32 pixels image input. The convolution process extracted features from the image with 64 filters, 5×5 kernel, and valid padding using the ReLU activation function. This layer used pooling with a size of 2×2, stride 2×2, and max pooling. The fourth layer in Figure 4 point 4 received 16×16 pixels image input, while the convolution process extracted features from the image with 128 filters, 5×5 kernel, and padding using the ReLU activation function. Similarly, this layer used pooling with a size of 2×2, stride 2×2, max pooling, and a dropout layer added. All parameters were then connected as a vector on flatten to enter the fully connected layer. Entering the dense layer in Figure 4, point 5, the results were reduced to 1024 and 256 outputs. In the last layer, classification was conducted using the softmax function to produce the number of classes [16], [17].

In an approach with the transfer learning method Figure 5, feature extraction used a layer from ResNet50 and two fully connected layers. The first process was to load the weight of the previously trained model and part of the feature extraction layer without taking the fully connected layer. The fully connected layer used the dense layer, reducing the results to 256 outputs. The last layer used the output appropriate to the number of categories with softmax as a classification function [18], [19].

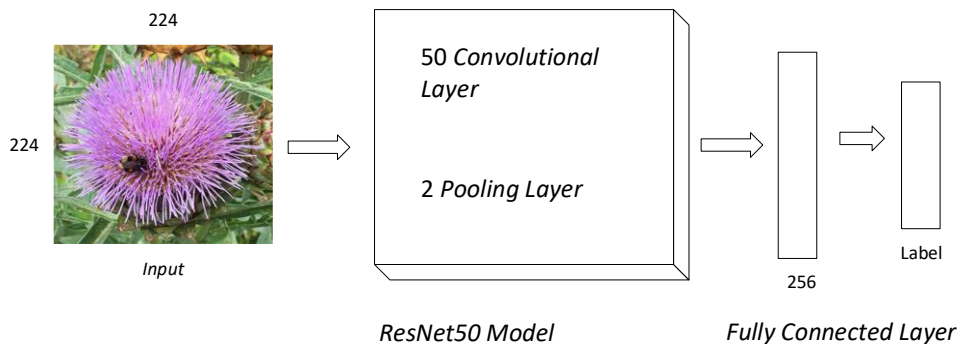


Figure 5. CNN model with ResNet50 transfer learning

3.3. Nadam optimization

Nadam is an optimization algorithm developed by Kingma and Ba [20]. It is a technique for efficient stochastic optimization requiring only a first-order gradient with little memory requirement. This method is easy to implement, computationally efficient and has few memory requirements. However, it does not fluctuate for diagonal gradient scales and is well suited for large problems with data or parameters. Nadam used Nesterov's accelerated gradient (NAG) and maintained Adam's momentum component as an advantage. It produced a substitution that increased the convergence speed and the model's quality [21]. Nadam is part of stochastic gradient descent. This algorithm functions in deep learning to update the weight and bias values to reduce the resulting loss. The Nadam algorithm in (1) was used in the back-propagation process to update the weight and bias on parameter w_{t+1} , where t is the timestep, and w_t is the current weight. L is the value of the loss function, while the parameter values of the learning rate a and decay β are adjustable as needed. Additionally, the average exponential motion of the gradient V and the squared gradient S was initiated with 0 [21].

$$w_{t+1} = w_t - \frac{a}{\sqrt{\hat{s}_t + \epsilon}} \left(\beta_1 \hat{V}_t + \frac{1 - \beta_1}{1 - \beta_1^t} \cdot \frac{\delta L}{\delta w_t} \right) \tag{1}$$

Applying Nadam optimization was chosen because it is more efficient and effective and does not use a lot of resources. The metrics used in this research are accuracy. The programming code in Python:

```
model.compile(loss='categorical_crossentropy',
optimizer=keras.optimizers.Nadam(),
metrics = ['accuracy'])
```

3.4. Convolutional neural network model training

It covers training setup to regulate the training process to obtain the best model efficiently and effectively. The training process was conducted by selecting a fit generator to carry out training in parallel and ensure efficiency. The use of a fit generator allowed data augmentation on the CPU and GPU training to be conducted in parallel in real-time. The training process used a learning rate of 0.0001 with a batch of 32. The process consisted of:

- a. Input layer

The image was converted into a three-dimensional matrix with a length×width×3 RGB channel. The RGB value of each pixel was normalized to a range of 0-1 to simplify the computation process by dividing each pixel by 255 Figure 6.

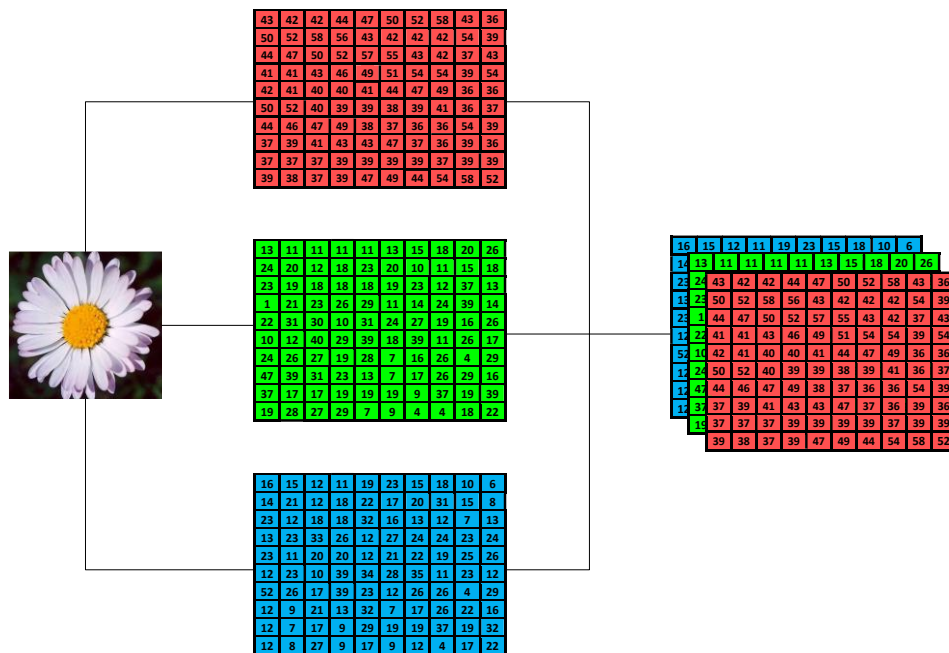


Figure 6. Input layer

b. Convolution layer

The value of the RGB channel in each pixel was processed in this layer, which extracts feature maps in the image using filters. For instance, Figure 7 is an image input on the red color channel. These numbers represent the value of the red color intensity in the image (0-255). A convolution process with filters was conducted to obtain features from the image. The filter was then convoluted with the input data to produce a feature map that could detect edges. Also, continuous convolution and pooling processes would form a more detailed pattern, as shown in Figure 8.

43	42	42	44	47	50	52	58	43	36
50	52	58	56	43	42	42	42	54	39
44	47	50	52	57	55	43	42	37	43
41	41	43	46	49	51	54	54	39	54
42	41	40	40	41	44	47	49	36	36
50	52	40	39	39	38	39	41	36	37
44	46	47	49	38	37	36	36	54	39
37	39	41	43	43	47	37	36	39	36
37	37	37	39	39	39	39	37	39	39
39	38	37	39	47	49	44	54	58	52

Figure 7. Input of red channel

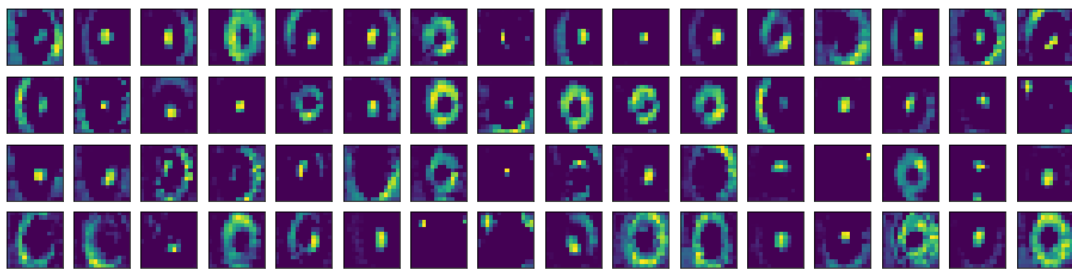


Figure 8. Pattern of feature map

c. Activation function

Figure 9(a) shows the results of the activation function in the object area selection, and Figure 9(b) shows the results in layers 1 and 2. This research uses the ReLU activation function to determine whether a neuron is active in the neural network.

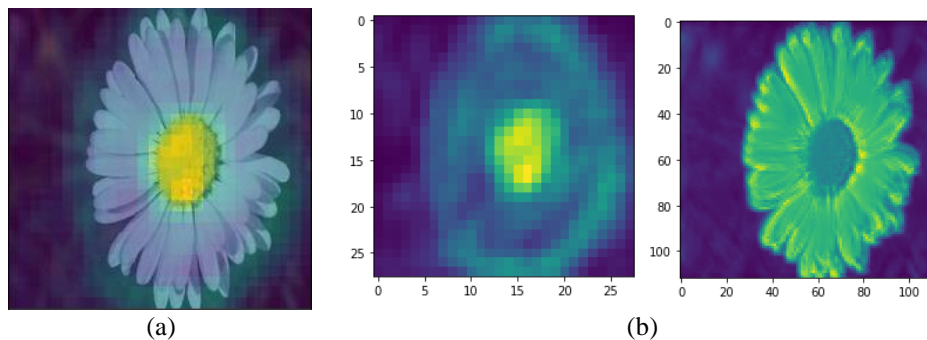


Figure 9. The results of the activation function; (a) selection of the object area and (b) at 1st and 2nd layer

d. Pooling layer

The image’s spatial size and the number of parameters and calculations in the neural network were reduced by independently operating the pooling layer on each feature. This layer received input from the feature map results from the convolution process. Max pooling with a size of 2×2 was used with a stride of 2, resulting in an image 50% smaller.

e. Fully connected layer

It changed the 3-dimensional matrix data at the convolution stage into a one-dimensional vector (shown in (2)). The last stage of the fully connected layer was softmax functioning (shown in (3)) to generate probabilities from classification predictions called the classification stage. This stage used the value of the previous neuron and applied the softmax activation function [22]. Figure 10 illustrates the one of neural network in this study.

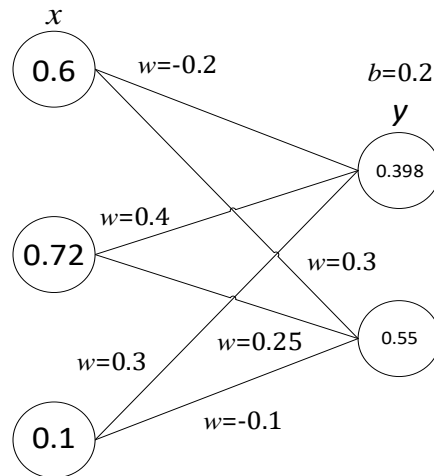


Figure 10. Example of a neural network

To get the value on the destination neuron (y) using (2): [11], [17]:

$$y = g(\sum_{i=1}^n x_i w_i + b) \tag{2}$$

with the neuron value (x) calculated by weight (w) and added with bias (b) and then activated with function (g). Hence, the value of y is:

$$y = g(0.6 \times (-0.2) + 0.72 \times 0.4 + 0.1 \times 0.3 + 0.2) = 0.398$$

$$y = g(0.6 \times 0.3 + 0.72 \times 0.25 + 0.1 \times (-0.1) + 0.2) = 0.55$$

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{3}$$

where the neuron in the final classification layer calculates the probability value (S) of the class (y) by dividing it by the total number of exponential values.

f. Loss function

The model’s performance was measured using a loss function to determine the difference between the prediction and the actual. The softmax results obtained the predictive value, while the actual value was obtained from the label index [14], [15].

Example:

Daisy is in the 7th index in Table 1, so only the 7th value is 1, and other classes/types are 0.

Actual=[0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0.]

Predictive=[0.05345823, 0.05345614, 0.05345496, 0.05345551, 0.05354749, 0.05346881, 0.1443803, 0.05345564, 0.05345527, 0.05348306, 0.05345568, 0.05354218, 0.05345708, 0.05345512, 0.05345495, 0.05345769, 0.05356189]

Loss=0.1655677436840361

Table 1. The last neuron in training

e	S	Label		
6.2711093e-05	0.05345823	0	0	Bluebell
2.3693376e-05	0.05345614	1	0	Buttercup
1.5455488e-06	0.05345496	2	0	Colts Foot
1.1844796e-05	0.05345551	3	0	Cowslip
1.7310450e-03	0.05354749	4	0	Crocus
2.6060341e-04	0.05346881	5	0	Daffodil
9.9361295e-01	0.1443803	6	1	Daisy
1.4375035e-05	0.05345564	7	0	Dandelion
7.3801125e-06	0.05345564	8	0	Fritillary
5.2712415e-04	0.05345564	9	0	Iris
1.4992844e-05	0.05345564	10	0	Lily Valley
1.6319246e-03	0.05345564	11	0	Pansy
4.1258110e-05	0.05345564	12	0	Snowdrop
4.4960607e-06	0.05345564	13	0	Sunflower
1.4411211e-06	0.05345564	14	0	Tigerlily
5.2668565e-05	0.05345564	15	0	Tulip
1.9999850e-03	0.05345564	16	0	Windflower

g. Backpropagation

This process involves updating weights and biases to reduce the overall training loss using the stochastic gradient descent method, where Nadam was used in this study. As seen by the following data, where the accuracy will have a high value if the loss is low and the loss will decrease with each epoch, the better the model and the better the accuracy.

Epoch 1/50

48/48[=====] - 15s 313ms/step - loss: 1.6959 - acc: 0.4375 - val_loss: 1.8872 - val_acc: 0.4529

Epoch 2/50

48/48[=====] - 13s 261ms/step - loss: 1.2639 - acc: 0.5426 - val_loss: 1.1795 - val_acc: 0.5924

Epoch 3/50

48/48[=====] - 11s 238ms/step - loss: 1.0954 - acc: 0.5866 - val_loss: 1.1596 - val_acc: 0.6105

Epoch 4/50

48/48[=====] - 11s 236ms/step - loss: 1.0841 - acc: 0.5834 - val_loss: 1.0522 - val_acc: 0.6377

Epoch 5/50

48/48[=====] - 11s 239ms/step - loss: 1.0078 - acc: 0.6207 - val_loss: 1.0909 - val_acc: 0.6467

Epoch 6/50

48/48[=====] - 11s 236ms/step - loss: 0.9148 - acc: 0.6518 - val_loss: 1.0561 - val_acc: 0.6051

3.5. Performance measurement

In the last stage, CNN model performance was obtained by measuring the loss and confusion matrix. This performance was compared with ANN and SVM, which used the same dataset as previous studies. Confusion matrix generated accuracy (4), precision (5), recall (6), and F1 score (7), representing the predicted and actual values [23], [24]. The confusion matrix shown in Table 2 represents the true and false negative and positive.

$$Accuracy = \frac{TP+TN}{TP+FN+TN+FP} \quad (4)$$

$$Precision = \frac{TP}{TP+FP} \quad (5)$$

$$Recall = \frac{TP}{TP+TN} \quad (6)$$

$$F1\ Score = 2 \times \frac{precision \times recall}{precision + recall} \quad (7)$$

Table 2. Confusion matrix

Actual	Predicted	
	Positive	Negative
Positive	True positives (TP)	False negatives (FN)
Negative	False positives (FP)	True negatives (TN)

4. RESULTS AND DISCUSSION

This section explains the results of the training process of the proposed CNN model from scratch and ResNet50 transfer learning. It also discusses the performance value and the comprehensive discussion.

4.1. Loss of convolutional neural network model

The training process results show the fluctuation of the loss values in the data tests from 0.7 to 2.7 from scratch and 0.05 to 3.4 for the transfer of learning into the Oxford17 dataset. They also show a loss of data train between 1.1 to 2.75 from scratch and 0.05 to 3.2 for transfer of learning, as indicated in Figures 11(a) and (b) respectively. In Figures 12(a) and (b), the results also indicate the fluctuating loss data test between 1.0 to 4.5 from scratch and 0.7 to 7.5 for transfer learning in an Oxford102 dataset, respectively. Loss of data train between 2.0 to 4.5 from scratch and 0.02 to 3.5 indicates that the loss performance of the Oxford102 dataset is less compared to Oxford17.

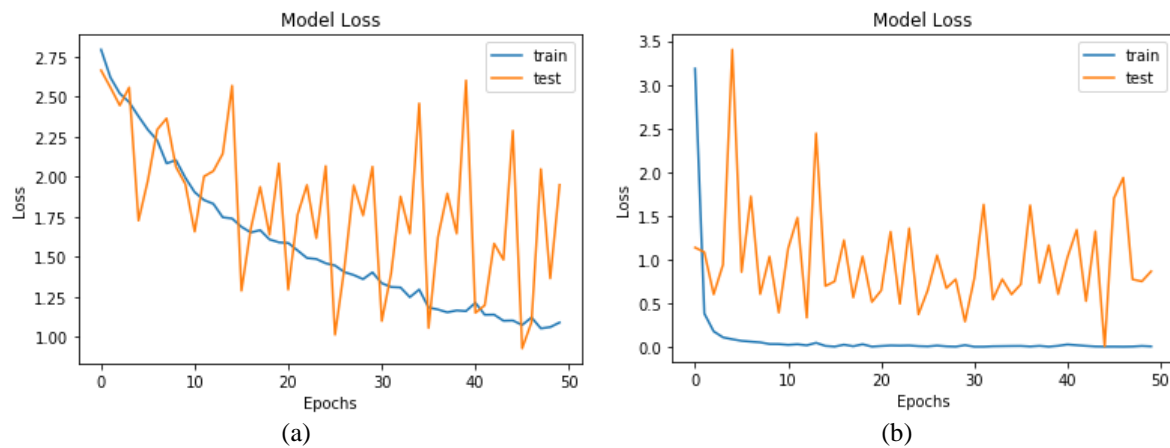


Figure 11. Loss for the Oxford17 dataset with; (a) CNN from scratch and (b) CNN transfer learning

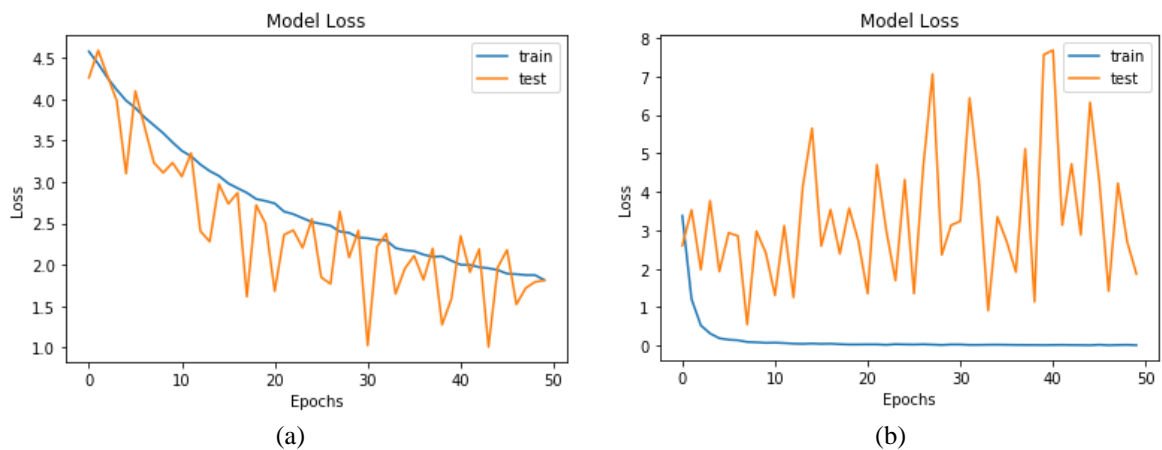


Figure 12. Loss for the Oxford102 dataset with; (a) CNN from scratch and (b) CNN transfer learning

4.2. Performance of convolutional neural network model

On the Oxford17 dataset, the average accuracy value is 0.54 for validation results and 0.65 for training outcomes. On the other hand, training with transfer learning from ResNet50 produced training and validation accuracy of 0.98 and 0.83 for the Oxford17 dataset. Figures 13(a) and (b) display the CNN model's accuracy, respectively. In the meantime, the maximum accuracy on the Oxford102 dataset was 0.58 for training results and 0.47 for validation. ResNet50 transfer learning training can yield the best accuracy results, with 0.98 in training and 0.6 in validation. The accuracy of the CNN model using the Oxford102 dataset from scratch and transfer learning is shown in Figures 14(a) and (b), respectively.

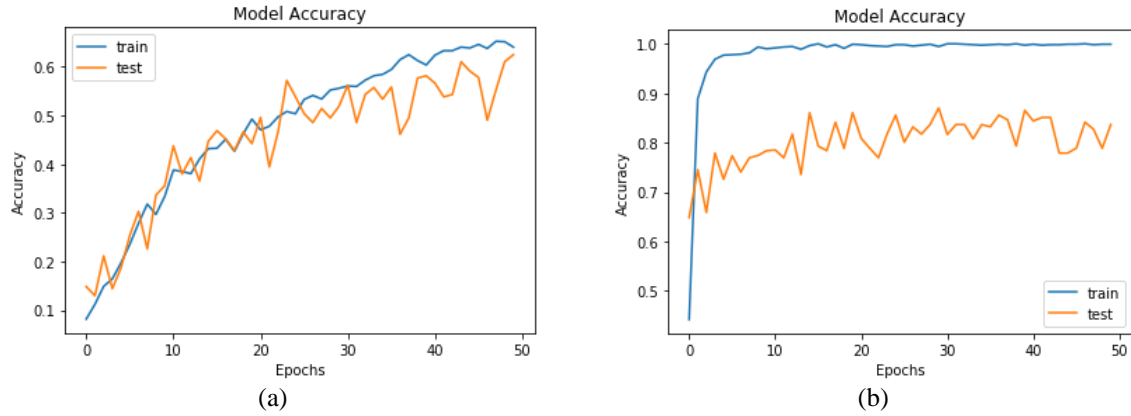


Figure 13. The accuracy of the Oxford17 with; (a) CNN from scratch and (b) CNN transfer learning

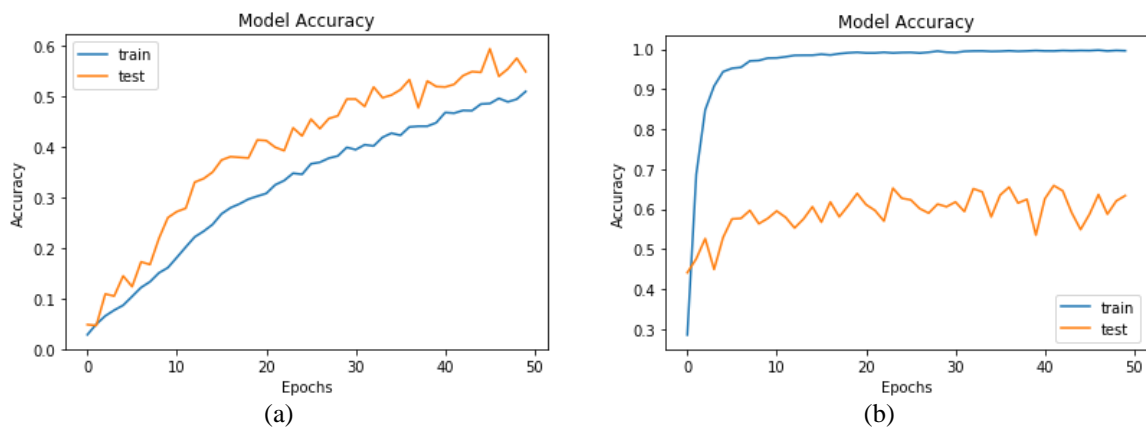


Figure 14. The accuracy of the Oxford102 with; (a) CNN from scratch and (b) CNN transfer learning

Performance measurement, besides accuracy, also uses precision, recall, and the F1-score. The measurement results show that ResNet50 transfer learning has better performance than the baseline in Table 3. The Oxford17 dataset shows a 37% higher average on precision, recall, and F1-score values, whereas the Oxford102 dataset shows nearly 26% higher values.

Table 3. CNN confusion matrix

	Oxford17 dataset			Oxford102 dataset		
	Precision	Recall	F1-score	Precision	Recall	F1-score
CNN from scratch	0.66	0.60	0.60	0.58	0.54	0.52
CNN with transfer learning	0.88	0.84	0.83	0.79	0.64	0.64

4.3. Testing result of convolutional neural network model

Testing of the CNN model is carried out on the android application. The android platform is chosen to make it easier to get images because it can directly use. In the application display, the classification process is carried out from the model, the results of the three highest classification probability are selected and displayed at the bottom of the application (Figure 15(a)). While the setting serves to select the classification model to be used, in this study, four models can be used to classify, namely: Oxford17, Oxford17 ResNet, Oxford102, and Oxford102 ResNet. In this setting, we can choose where the processing is done. There are two choices of CPU and GPU and threads to use. This setting page also has info on the classification input media (Figure 15(b)). The performance of the proposed CNN model was determined by running the program to test data of 50 images for each model. The data were classified by scanning using a smartphone camera on the images on the laptop screen. The result was recorded for each correct prediction (shown in Figure 16) to calculate the accuracy of each model, as shown in Table 4.

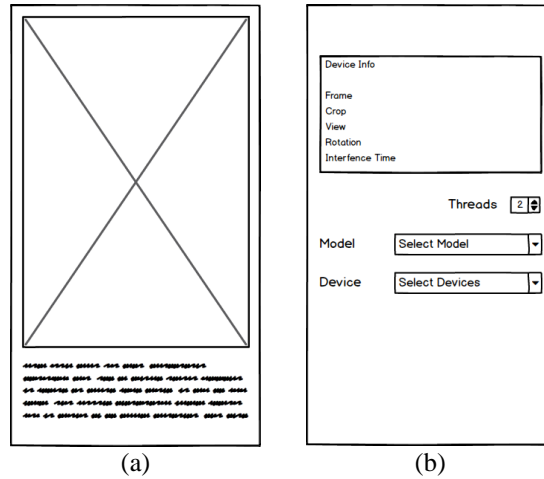


Figure 15. The interface design of flower application is android-based, which consist of; (a) classification displays and (b) settings display

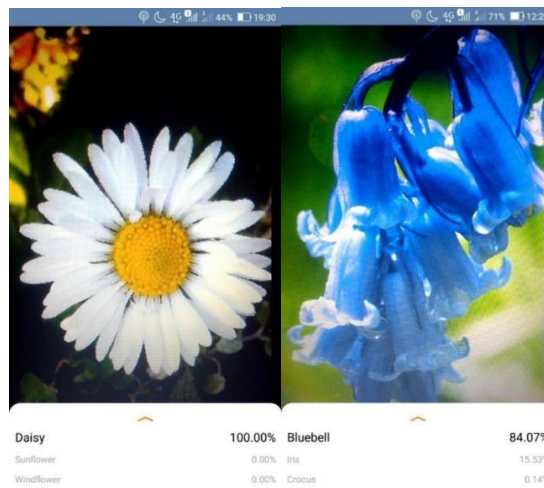


Figure 16. Testing with true prediction

Table 4. The testing results

Prediction	Oxford17		Oxford102	
	Scratch	Transfer learning	Scratch	Transfer learning
True	30	42	21	32
False	20	8	29	18
Accuracy	0.6	0.84	0.42	0.64

CNN scratch accuracy Oxford17= $\frac{30}{50} = 0.6$, transfer learning accuracy= $\frac{42}{50} = 0.84$

CNN scratch accuracy Oxford102= $\frac{21}{50} = 0.42$, CNN transfer learning accuracy = $\frac{32}{50} = 0.64$

4.5. Discussion

Based on the CNN model performance with 119 flowers, accuracy results were obtained in Table 5 and compared with SVM and ANN. CNN transfer learning method has higher accuracy of 84% than SVM [25] and ANN, which have accuracies of 83.52% and 72%, respectively. However, the CNN model from scratch cannot outperform ANN and SVM because it is only 60% accurate. The preprocessing data for the flower image in the SVM method used the region growing segmentation technique that separates the flower object from its background. The CNN model from scratch has lower accuracy than ANN [2], with an accuracy of 81.19%. However, it is higher than the SVM model, which obtained an accuracy of 32.4%. For Oxford102, the CNN model using transfer learning obtained an accuracy of 64%. ANN used the HSV color

descriptor, gray level co-occurrence matrix (GLCM), and Invariant Moments to separate the flower object from its background during data preprocessing. Therefore, the CNN model trained by the transfer learning method has an accuracy advantage. Transfer learning produces a model with high accuracy in a short time compared to a CNN from scratch with the same epochs.

Table 5. Accuracy comparison of CNN, SVM, and ANN

Dataset	CNN		SVM (%)	ANN (%)
	Scratch (%)	Transfer learning (%)		
Oxford17	60	84	83.52 [25]	72
Oxford102	42	64	32.4	81.19 [2]

5. CONCLUSION

This study proposed a flower classification model using a CNN and Nadam optimization. It used the Oxford17 and Oxford102 datasets with two approaches from scratch and transfer learning. The CNN transfer learning model has high accuracy, though the trained model is short by several epochs than the model from scratch. Furthermore, the increase in the type of the image during the classification reduced the result's accuracy value. The Oxford17 classification showed that 17 flowers have better accuracy of 60% than the Oxford102 containing 102 flowers, which is 42% accurate when using the same model. The CNN model's performance for classifying the Oxford17 dataset was 60% and 84% accurate for scratch and transfer learning, respectively. The accuracy for the Oxford102 dataset was 42% and 64% from scratch and transfer learning, respectively. Previous studies showed that CNN transfer learning and Nadam optimization perform better than SVM and ANN on the Oxford17 dataset. Their accuracy performance of 64% does not exceed ANN, which is 81.19% on the Oxford102 dataset. Therefore, future studies should investigate the accuracy of CNN with other methods, such as deep reinforcement learning, generative adversarial network (GAN), or random forest.




REFERENCES

- [1] T. Ensari and B. R. Mete, "Flower Classification with Deep CNN and Machine Learning Algorithms," in *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Ankara Turkey, Oct. 2019, pp. 1–5.
- [2] H. Almogdady, S. Manaseer, and H. Hiary, "A flower recognition system based on image processing and neural networks," *International Journal of Scientific & Technology Research*, vol. 7, no. 11, pp. 166–173, Nov. 2018.
- [3] M. V. D. Prasad *et al.*, "An efficient classification of flower images with convolutional neural networks," *International Journal of Engineering & Technology (UAE)*, vol. 7, no. 1.1, pp. 384–391, 2018, doi: 10.14419/ijet.v7i1.1.9857.
- [4] A. R. Saikia, K. Bora, L. B. Mahanta, and A. K. Das, "Comparative assessment of CNN architectures for classification of breast FNAC images," *Tissue and Cell*, vol. 57, pp. 8–14, Apr. 2019, doi: 10.1016/j.tice.2019.02.001.
- [5] E. M. Dogo, O. J. Afolabi, N. I. Nwulu, B. Twala, and C. O. Aigbavboa, "A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks," in *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, Belgaum, India, Jul. 2018, pp. 92–99, doi: 10.1109/CTEMS.2018.8769211.
- [6] M. E. Nilsback and A. Zisserman, "17 Category Flower Dataset," VGG, [Online]. Available: <https://www.robots.ox.ac.uk/~vgg/data/flowers/17/>. (accessed: 2020, 1 April).
- [7] M. E. Nilsback and A. Zisserman, "A visual vocabulary for flower classification," in *Proceeding of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, New York, NY, USA, 2006, pp. 1447–1454, doi: 10.1109/CVPR.2006.42.
- [8] M. E. Nilsback and A. Zisserman, "102 Category Flower Dataset," VGG, [Online]. Available: <https://www.robots.ox.ac.uk/~vgg/data/flowers/102/>. (accessed: 2020, 1 April).
- [9] J. Steinbrener, K. Posch, and R. Leitner, "Hyperspectral fruit and vegetable classification using convolutional neural networks," *Comput. Electron. Agric.*, vol. 162, pp. 364–372, Jul. 2019, doi: 10.1016/j.compag.2019.04.019.
- [10] M. Kozłowski, P. Górecki, and P. M. Szczypiński, "Varietal classification of barley by convolutional neural networks," *Biosyst. Eng.*, vol. 184, pp. 155–165, Aug. 2019, doi: 10.1016/j.biosystemseng.2019.06.012.
- [11] I. Vasilev, D. Slater, G. Spacagna, P. Roelants and V. Zocca, *Python Deep Learning: Exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow*. Packt Publishing Ltd, Birmingham, 2019.
- [12] M. Sewak, M. R. Karim, and P. Pujari, *Practical convolutional neural networks: implement advanced deep learning models using Python*. Packt Publishing Ltd, Birmingham, 2018.
- [13] F. Pedregosa *et al.*, "Scikit-learn: machine learning in Python," *Journal of machine Learning research*, vol. 12, pp. 2825–2830, Nov. 2011.
- [14] W. You, C. Shen, X. Guo, X. Jiang, J. Shi, and Z. Zhu, "A hybrid technique based on convolutional neural network and support vector regression for intelligent diagnosis of rotating machinery," *Advances in Mechanical Engineering*, vol. 9, no. 6, pp. 1–17, Jun. 2017, doi: 10.1177/16878140177041.
- [15] M. Z. Alom *et al.*, "The history began from alexnet: A comprehensive survey on deep learning approaches," *arXiv preprint arXiv:1803.01164*. 2018.
- [16] M. Sargül, B. M. Ozyildirim, and M. Avci, "Differential convolutional neural network," *Neural Networks*, vol. 116, pp. 279–287, Aug. 2019, doi: 10.1016/j.neunet.2019.04.025.
- [17] P. Kim, "MATLAB Deep Learning," *With Machine Learning, Neural Networks and Artificial Intelligence*, vol. 130, no. 21, p. 151, 2017, doi: 10.1007/978-1-4842-2845-6.




- [18] S. Aneja, N. Aneja, P. E. Abas, A. G. Naim, "Transfer learning for cancer diagnosis in histopathological images," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 11, no. 1, pp. 129-136, Mar. 2022, doi: 10.11591/ijai.v11.i1.pp129-136.
- [19] Z. Widyantoko, T. P. Widowati, Isnaini, P. Trapsiladi, "Expert role in image classification using CNN for hard to identify object: distinguishing batik and its imitation," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 1, pp. 93-100, Mar. 2021, doi: 10.11591/ijai.v10.i1.pp93-100.
- [20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [21] T. Dozat, "Incorporating nesterov momentum into adam," in *International Conference on Learning Representations (ICLR) Workshop*, San Juan, Puerto Rico, pp. 1-4, 2016.
- [22] Y. L. He, X. L. Zhang, W. Ao, and J. Z. Huang, "Determining the optimal temperature parameter for Softmax function in reinforcement learning," *Applied Soft Computing*, vol. 70, pp. 80-85, Sep. 2018, doi: 10.1016/j.asoc.2018.05.012.
- [23] A. Kulkarni, D. Chong, and F. A. Batareseh, "Foundations of data imbalance and solutions for a data democracy," in *Data Democracy*, 2020, pp. 83-106, doi: 10.1016/B978-0-12-818366-3.00005-8.
- [24] S. S. W. Su and S. L. Kek, "An Improvement of Stochastic Gradient Descent Approach for Mean-Variance Portfolio Optimization Problem," *Journal of Mathematics*, vol. 2021, pp. 1-10, 2021, doi: 10.1155/2021/8892636.
- [25] A. Albadameh and A. Ahmad, *Automated flower species detection and recognition from digital images*, Diss. Princess Sumaya University for Technology (Jordan), pp. 1-6, 2016.

BIOGRAPHIES OF AUTHORS






Qurrotul Aini    holds a Doctor of Electrical Engineering degree from Institut Teknologi Sepuluh Nopember, Indonesia in 2018. She is currently an associate professor at the Department of Information System in Universitas Islam Negeri Syarif Hidayatullah Jakarta Indonesia. Her research interest is in business intelligence, intelligence computation, multimedia application, and ad hoc network. She can be contacted at email: qurrotul.aini@uinjkt.ac.id.






Zulfiandri    is an assistant professor at Department of Information System in Universitas Islam Negeri Syarif Hidayatullah Jakarta Indonesia. His research interests are software engineering, data mining, data warehouse, IT infrastructure, and algorithms & data structures. He can be contacted at email: zulfiandri@uinjkt.ac.id.



Rezky Firmansyah    holds a Bachelor of Computer in Department of Information System at Universitas Islam Negeri Syarif Hidayatullah Indonesia. He is currently working as an IT Support Engineer at Bekasi Municipal Communication and Information Office, Indonesia. His research interest is web development, machine learning, and data analysis. He can be contacted at email: rezkyfmh@gmail.com.



Yunifa Miftachul Arif    received Master's and Ph.D. degrees from the Department of Electrical Engineering, Institut Teknologi Sepuluh Nopember Surabaya, Indonesia, in 2010 and 2022. He is a lecturer in Department of Informatics Engineering, Universitas Islam Negeri Maulana Malik Ibrahim Malang, Indonesia. His research interest includes game technology, blockchain, metaverse, recommender system, and the internet of things. He can be contacted at email: yunif4@ti.uin-malang.ac.id.