

Target image validation modeling using deep neural network algorithm

Naemah Mubarakah^{1,2}, Poltak Sihombing³, Syahril Efendi³, Fahmi²

¹Doctoral Program in Computer Science, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Medan, Indonesia

²Department of Electrical Engineering, Faculty of Engineering, Universitas Sumatera Utara, Medan, Indonesia

³Department of Computer Science, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Medan, Indonesia

Article Info

Article history:

Received Jul 1, 2024

Revised Nov 24, 2024

Accepted Dec 2, 2024

Keywords:

Activation functions

Batch sizes

Deep neural network

Image validation

Neurons

ABSTRACT

Research on image validation models is an interesting topic. The application of deep learning (DL) for object detection has been demonstrated to effectively and efficiently address the challenges in this field. Deep neural networks (DNN) are deep learning algorithms capable of handling large datasets and effectively solving complex problems due to their robust learning capacity. Despite their ability to address complex problems, DNN encounter challenges related to the necessity for intricate architectures and a large number of hidden layers. The objective of this research is to identify the most effective model for achieving optimal performance in image validation. This study investigates target image validation using DNN algorithms, examining architectures with 3, 4, 5, and 6 hidden layers. This study also evaluates the performance of image validation across various activation functions, batch sizes, and numbers of neurons. The results of the study show that the best performance for image validation is achieved using the Leaky-ReLU and Sigmoid activation functions, with a batch size of 64, and an architecture consisting of 3 hidden layers with neuron sizes of 256, 128, and 64. This model is capable of providing real-time target image validation with an accuracy of up to 94.31%.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Naemah Mubarakah

Doctoral Program in Computer Science, Faculty of Computer Science and Information Technology,

Universitas Sumatera Utara

Medan, Sumatera Utara, Indonesia

Email: naemah@usu.ac.id

1. INTRODUCTION

Validation is still a fascinating area of study. Researchers are actively looking for the best ways to provide robust validation in order to minimize corrective errors, thanks to developments in machine learning and internet of things (IoT) sensors. Non-linear and non-stationary data are analyzed using a variety of data-driven techniques, including machine learning and signal processing. However, inadequate information for real-time applications frequently results in a reduction in performance. Numerous studies in the validation sector have used a variety of techniques, such as architectural framework alterations [1], genetic algorithm (GA) [2], super learner algorithm [3], and differential evolution (DE) [4]. Real-time model applicability is still quite limited despite a large number of studies.

Deep learning methods for object detection are acknowledged for their efficiency, owing to their capacity to utilize diverse learning strategies and train on extensive datasets [5], [6]. This efficiency is

evidenced by significant advancements in segmentation [7], detection [8], and classification [9]. Historically, image object exploration techniques relied on color descriptors [10] and image descriptors [11], processed through unsupervised algorithms such as K-means clustering [12], spectral clustering [13], pooling clusters [14], and spanning trees [15], as well as less robust supervised algorithms like deep metric learning [16] and subspace learning [17]. Deep neural networks (DNNs), which fall under the deep learning category, are characterized by their multi-layered structures-typically comprising three or more interconnected layers. DNNs excel in addressing complex problems and have been instrumental in driving significant innovations across various societal [18] and industrial domains [19]–[22]. However, despite their efficacy in solving complex challenges, DNNs necessitate sophisticated architectures with numerous hidden layers, which results in prolonged training durations [23].

The primary challenge in deep neural networks (DNN) algorithms is determining the optimal model to achieve the best performance in target image validation. This study focuses on target image validation using DNN algorithms in real-time sensors. DNNs, which feature numerous hidden layers, are evaluated by comparing configurations with 3 to 6 hidden layers. The choice of activation function is critical to DNN performance, making the selection of the appropriate activation function essential. Additionally, this research assesses the impact of batch size and the number of neurons on model performance. The goal is to identify the optimal model architecture for target image validation. The model will be tested in real-time contexts to evaluate its effectiveness. The findings of this study can make a significant contribution to the advancement of data mining techniques.

2. MATERIAL AND METHOD

2.1. Deep neural network

A network made up of layers of neurons is called a DNN, and each neuron is connected to the others by random number biases [22]. Through channels with values called weights, neurons in one layer communicate with neurons in the next layer. The information that is shared between neurons is determined by these weights and biases. The network generates an output that reflects the prediction of the processed input in the last layer, referred to as the output layer [24]. An artificial neuron is seen in Figure 1.

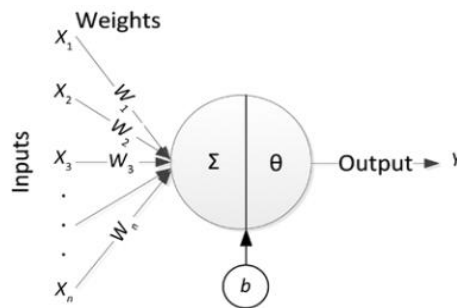


Figure 1. An artificial neuron

The input X_i is connected to the neuron through weighted connections, and the sum of all inputs, each multiplied by its corresponding weight W_i , is computed. This summation is then added to a bias (b), and the result is subsequently processed using an activation function (θ). Mathematically, the output of a perceptron unit can be formulated as (1) [23].

$$Y = \theta(\sum_{i=1}^n W_i X_i + b) \quad (1)$$

The following is another way to model it with matrix notation:

$$Y = \theta(W \cdot X + b) \quad (2)$$

Where $W = [W_1 \ W_2 \ \dots \ W_n]$ dan $x = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}$

To put it simply, a neural network is a group of layers made up of many neurons. Each layer has an output vector, a bias vector, and a weight matrix, generating columns of neurons that work in parallel. When processing an input vector N in a layer of neurons M , W_{ij} is the weight of the connection between the j^{th} input and the i^{th} neuron in that layer, and Y_i and b_i are the j^{th} neuron's output and bias, respectively [25]. As a result, the following matrix notation can be used to represent a neuron layer:

$$W = \begin{bmatrix} W_{11} \cdots W_{1m} \\ \vdots \quad \quad \quad \vdots \\ W_{n1} \cdots W_{nm} \end{bmatrix} \quad (3)$$

In matrix notation, the column index indicates the connection's input source. While the row index indicates the destination neuron for the corresponding connection. As a result, the layer's output Y can be written like this:

$$Y \begin{bmatrix} Y_l \\ \vdots \\ Y_i \\ \vdots \\ Y_N \end{bmatrix} = \begin{bmatrix} \theta(\sum_{j=1}^M W_{1j} X_j + b_l) \\ \vdots \\ \theta(\sum_{j=1}^M W_{ij} X_j + b_i) \\ \vdots \\ \theta(\sum_{j=1}^M W_{Nj} X_j + b_N) \end{bmatrix} = \theta(W \cdot X + b) \quad (4)$$

Where $b = \begin{bmatrix} b_l \\ \vdots \\ b_n \end{bmatrix}$.

In neuron layers, superscript indices are also used. For example, W_{ij}^k indicates the weight between the i^{th} neuron in layer k and the j^{th} neuron in layer $(k - 1)$, while Y_i^k indicates the output of the i^{th} neuron in layer k . Furthermore, N_k is intended to symbolize the quantity of buried neurons in layer k . Therefore, the function that can be derived from this network is as (5):

$$\begin{aligned} Y^3 &= \begin{bmatrix} Y_l^3 \\ \vdots \\ Y_i^3 \\ \vdots \\ Y_{N_3}^3 \end{bmatrix} = \theta(W^3 Y^2 + b^3) = \theta(W^3 \theta(W^2 Y^1 + b^2) + b^3) \\ &= \theta(W^3 \theta(W^2 (\theta(W^1 X + b^1) + b^2) + b^3)) \end{aligned} \quad (5)$$

One kind of artificial neural network with several layers is called a DNN. An input layer, $N > 2$ hidden layers, and an output layer are the three layers that are typically present in a DNN. 'Deep' describes the comparatively high number of layers. Deep learning is the term for the learning process that takes place inside a DNN. A deep neural network is the name given to the neural network in a DNN [24]. Formula (6) is used to calculate the final output of a DNN with four layers, where σ is the activation function and β , γ , and λ stand for noise or bias.

$$f_i = \sigma(\sum_{j=1}^{H_2} u_{j,i} \sigma(\sum_{k=1}^{H_1} v_{k,j} \sigma(\sum_{m=1}^M x_m w_{m,k} + \beta_k) + \gamma_j + \lambda_i)) \quad (6)$$

Deep neural networks can be trained using the back-propagation process. It can be difficult to estimate parameters in deep neural networks because of their complexity, which includes several layers and a large number of synaptic weights. Neural network scalability is strongly related to the back-propagation technique, which is frequently used for training neural networks. The method makes iterative modifications to get optimal weight configurations. Three-layer DNN is seen in Figure 2.

2.2. DNN algorithm performance measurement

Performance measurement is critical in the field of machine learning. The area under the curve (AUC) of the receiver operating characteristic (ROC) curve is a commonly used performance statistic. An important metric for evaluating the effectiveness of classification models is the AUC [26]. In particular, the area under the ROC curve is quantified by the AUC. Plotting the true positive rate (TPR) versus the false positive rate (FPR) across various classification thresholds allows the ROC curve, a graphical tool, to assess a classification model's performance. The following are the main elements of the ROC curve:

- a. The ratio of accurately anticipated positive cases to all actual positives is known as the TPR, and it is defined as (7):

$$TPR = \frac{TP}{TP+FN} \tag{7}$$

- b. The ratio of falsely projected positive cases to all actual negatives is known as the FPR, and it is defined as (8):

$$FPR = \frac{FP}{FP+TN} \tag{8}$$

where *TP* is true positive, *TN* is true negatives, *FP* is false positive and *FN* is false negatives. The *TPR* is represented by the Y-axis in the ROC curve, whereas the *FPR* is represented by the X-axis. The AUC values can be interpreted as follows:

- a. AUC = 1: The model exhibits flawless categorization capabilities.
- b. 0.5 < AUC < 1: The model outperforms random guessing; the closer the AUC value is to 1, the better the model performs.
- c. AUC = 0.5: The model's performance is on par with guesswork.
- d. AUC < 0.5: The model performs worse than random guessing, which may imply that the model is inverted

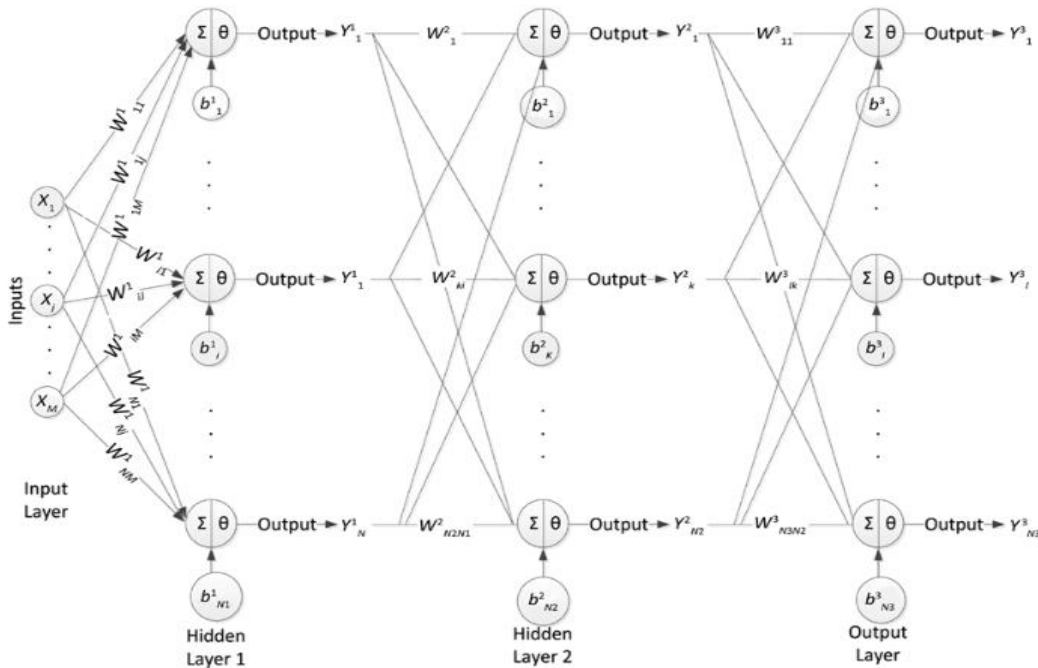


Figure 2. Three-layer DNN

2.3. Architecture research

Various schemes were applied with different configurations of hidden layers, activation functions, batch sizes, and numbers of neurons. After determining the optimal model for object validation using the DNN algorithm, this model was directly applied to objects captured by a camera for real-time analysis. Figure 3 shows the DNN-based machine learning modeling procedure. The study flowchart is displayed in Figure 3(a). Several machine learning models are viewed in the crate, train, and assessment phase in order to select the best model, as illustrated in Figure 3(b). Table 1 lists the many parameters used in the search for the optimal model.

To identify the best model, reference parameters will be chosen based on the outcomes of parameter modifications. The ROC curve's AUC serves as the foundation for this evaluation. The best model will be chosen for target picture validation after it has been determined and evaluated in real-time to assess its validation performance.

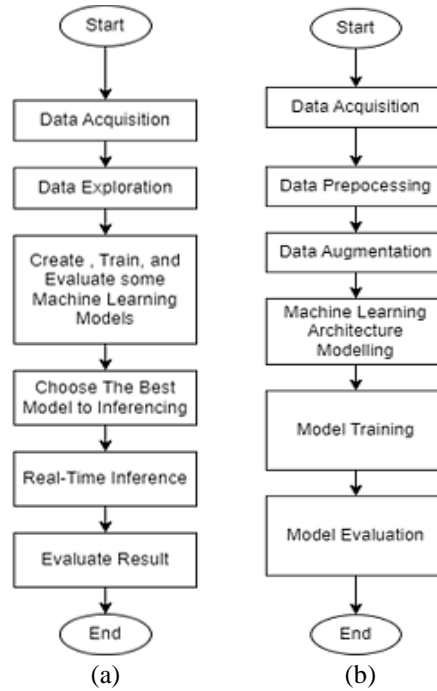


Figure 3. Research flowchart: (a) main process and (b) model development process in machine learning with DNN

Table 1. The research's parameters

No.	Parameters used in the research
1.	Number of hidden layers: 3, 4, 5, 6
2.	Activation function: Rectified linear unit (ReLU), Sigmoid, Leaky ReLU, Tanh, Linier, Scaled exponential linear unit (SELU) and SoftMax.
3.	Number of batch sizes: 16, 32, 64, 128, 256
4.	Number of neurons: 64, 128, 256, 512

3. RESULTS AND DISCUSSION

3.1. Activation functions variation

Deep learning methods rely on activation functions to generate efficient system performance. Knowing which activation function is appropriate to use with the DNN method is therefore essential. Several activation function modifications, such as rectified linear unit (ReLU), Lucky ReLU, SELU, Sigmoid, Tanh, Linear, and Softmax, are used in this work. Changes in the number of hidden layers are used to test the ReLU, Sigmoid activation function. In this instance, three, four, five, and six hidden levels are implemented. Figure 4 displays the confusion matrix results of the ReLU and Sigmoid activation function on 3, 4, 5, and 6 hidden layers, while Figure 5 displays the training and validation loss graphs. Figures 4 and 5 demonstrate how the DNN method performs poorly when ReLU and Sigmoid are used as activation functions. The AUC findings from the ROC for the application of the ReLU are derived from the results of Figures 4 and 5, as shown in Figure 6. The ROC of the other activation function's AUC is displayed in Figure 7.

The performance of the suggested DNN model with various activation functions is displayed in Figure 7. The ROC of the DNN model with three to six hidden layers that uses Sigmoid in the output layer and Leaky-ReLU activation layers in the hidden layers is displayed in Figure 7(a). The DNN model's performance with Tanh and Sigmoid activation layers is shown in Figure 7(b). The DNN model's performance utilizing the linear and sigmoid activation functions is displayed in Figure 7(c). The AUC for the DNN model with sigmoid in the output layer and SELU in the hidden layer is then shown in Figure 7(d). Additionally, Figure 7(e) illustrates how Sigmoid is used in all layers, including the output and hidden layers. Lastly, the model's performance employing the SELU and SoftMax as the activation function is displayed in Figure 7(f). According to the outcome, the model's AUC rises from 0.5 to 0.82 when Sigmoid is used in the output layer. This is because the Sigmoid activation function is appropriate for binary classification problems because it produces values between 0 and 1. Furthermore, the SELU, Sigmoid with four hidden layers, yields the best AUC of any studied activation function, at 0.82. However, Table 2 and Figure 8 demonstrate that the

Sigmoid model performs best when evaluating the stability of the model's performance across varying numbers of hidden layers, with an average AUC of 0.805 for the Leaky-ReLU.

Figure 8 shows the average AUC scores for various activation functions. Based on the graph, the average AUC varies depending on the activation function combinations used. The combination of Leaky-ReLU with Sigmoid and Linier with Sigmoid demonstrates the best performance, with an average AUC 0.805 and 0.79, indicating better model classification capabilities. Overall, activation functions such as Leaky-ReLU, Linier and Sigmoid tend to produce good performance when paired with the appropriate activation function. Therefore, the combination of Leaky-ReLU with Sigmoid is recommended to achieve the best performance.

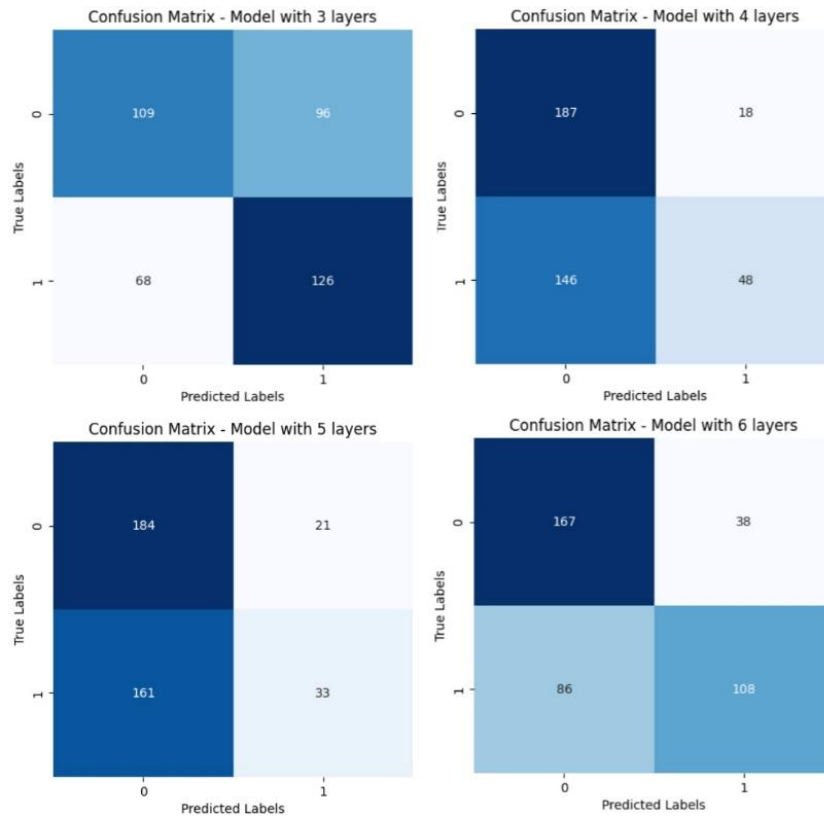


Figure 4. Confusion matrix of ReLU, Sigmoid activation functions at 3, 4, 5, and 6 hidden layers

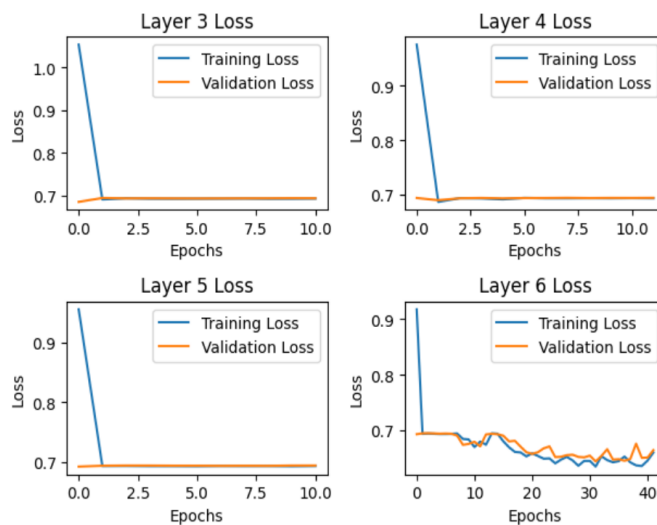


Figure 5. Training and validation loss graph with ReLU and Sigmoid activation functions

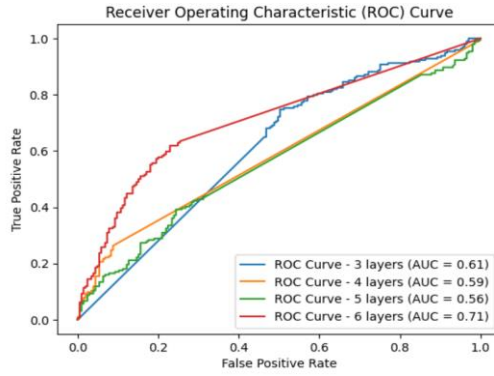


Figure 6. ROC graph of the ReLU-Sigmoid activation function

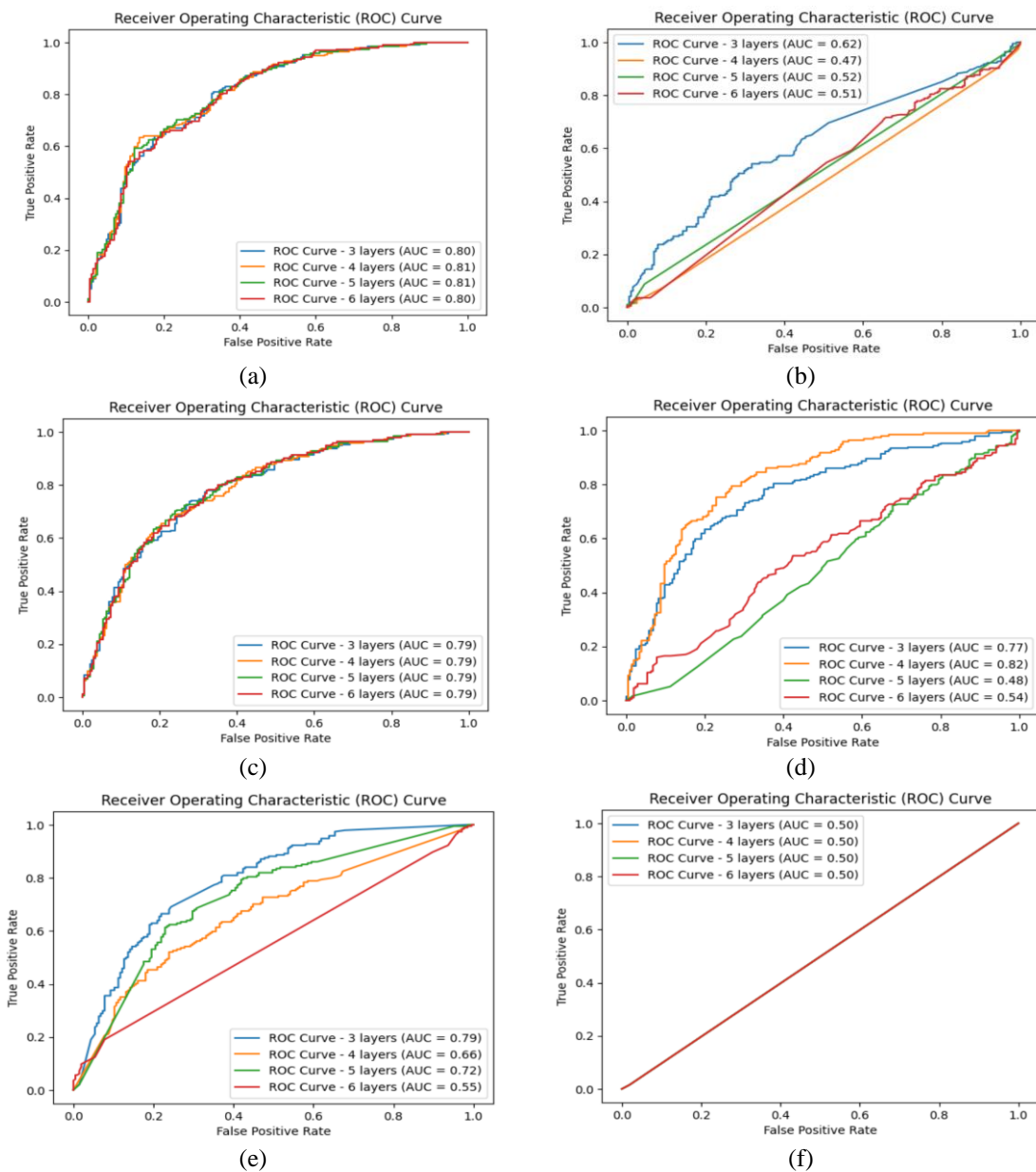


Figure 7. ROC graph of: (a) Leaky ReLU, Sigmoid, (b) Tanh, Sigmoid, (c) Linier, Sigmoid, (d) SELU, Sigmoid, (e) Sigmoid, Sigmoid, and (f) SELU, SoftMax activation functions

Table 2. Area under the curve from various activation functions

No.	Activation functions	AUC from different number of hidden layers				Average AUC score
		3	4	5	6	
1	ReLU, Sigmoid	0.61	0.59	0.56	0.71	0.6175
2	Leaky ReLU, Sigmoid	0.80	0.81	0.81	0.80	0.805
3	Tanh, Sigmoid	0.62	0.47	0.52	0.51	0.53
4	Linier, Sigmoid	0.79	0.79	0.79	0.79	0.79
5	SELU, Sigmoid	0.77	0.82	0.48	0.54	0.6525
6	Sigmoid, Sigmoid	0.79	0.66	0.72	0.55	0.68
7	SELU, SoftMax	0.5	0.5	0.5	0.5	0.5

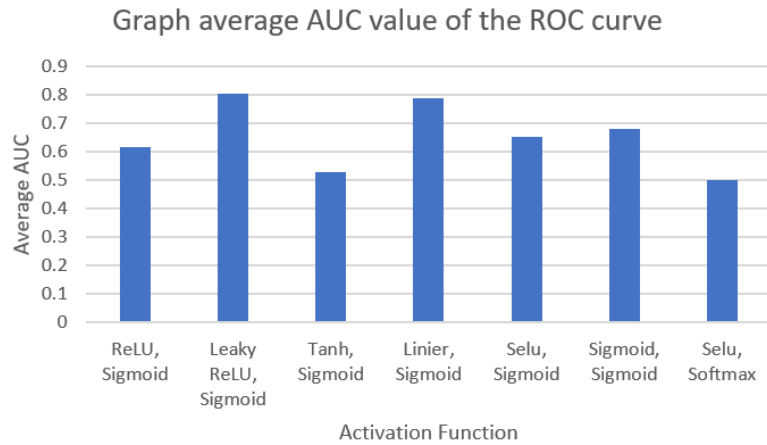


Figure 8. Graph of the average AUC value from various activation functions

3.2. Batch sizes variation

From section 3.1 it is obtained that the usage of Leaky-ReLU, sigmoid activation function returns the best performance. Therefore, in the batch size variation tests, the Leaky-ReLU, sigmoid activation function is chosen. In this test, varying batch size 16, 32, 64, 128, and 256 are used for the DNN model with various number of hidden layers. Figure 9 displays the model's training and validation losses for batch sizes of 16. Additionally, Figure 10 displays the model's AUC score with a batch size of 16, with an average AUC score of 0.82. Using five hidden layers yields the greatest results, with an AUC score of 0.82. The model's AUC score is also displayed in Figure 11 for batch sizes of 32, 64, 128 and 256. Figure 11(a) displays the AUC of the ROC when using a batch size of 32, Figure 11(b) displays the AUC when using a batch size of 64, Figure 11(c) displays the AUC of the model using a batch size of 128 and Figure 11(d) displays the AUC when using a batch size of 256.

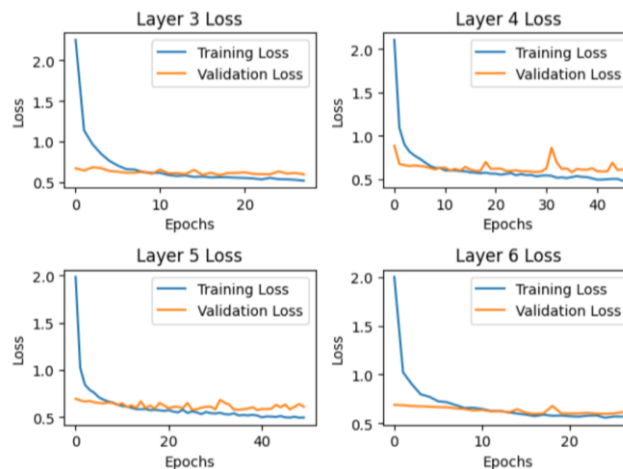


Figure 9. Training and validation loss using Leaky-ReLU and Sigmoid activation function for batch sizes of 16

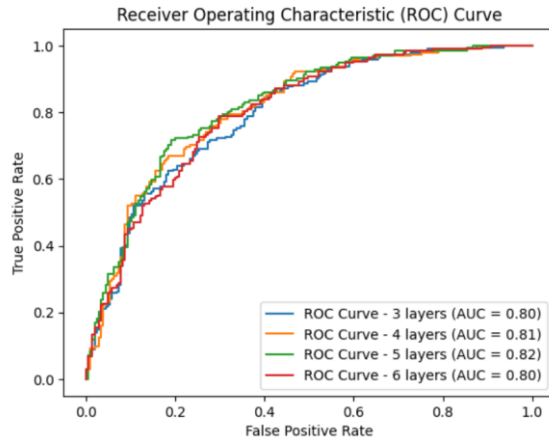


Figure 10. ROC graph with Leaky-ReLU, Sigmoid activation function for batch sizes of 16

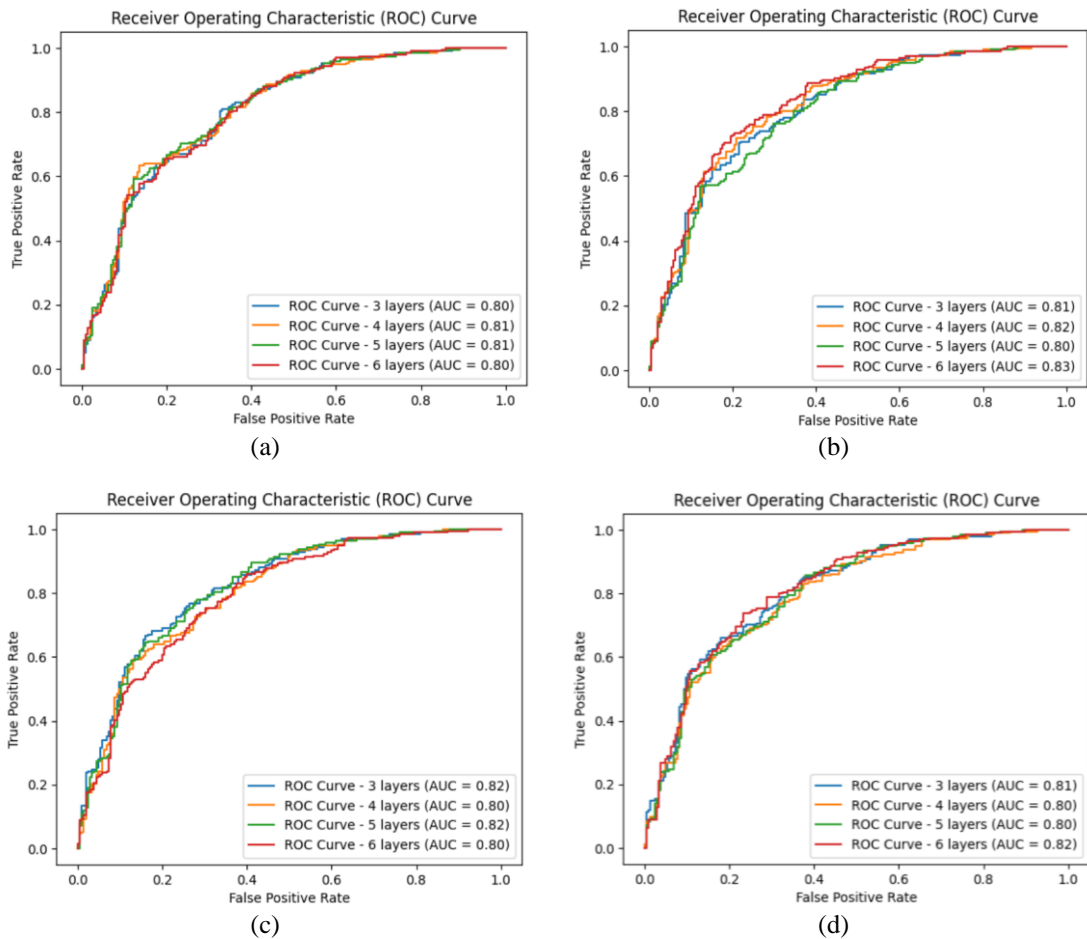


Figure 11. Graph of ROC curve with batch sizes of: (a) 32, (b) 64, (c) 128, and (d) 256

Based on the batch size variation test, all the batch sizes produce the AUC score no lower than 0.8 with the highest AUC score of 0.82. However, using different batch sizes for training the proposed DNN model does not significantly improve the model's performance. The overall performance of the batch size variation test is shown in Table 3. From Table 3, it can be seen that the highest average AUC score of 0.815 is achieved when using batch size is 64. Therefore, it is recommended to use batch size of 64 when training the DNN model.

Table 3. AUC score from ROC with various batch size

No.	Batch size	AUC from different number of hidden layers				Average AUC score
		3	4	5	6	
1	16	0.80	0.81	0.82	0.80	0.8075
2	32	0.80	0.81	0.81	0.80	0.805
3	64	0.81	0.82	0.80	0.83	0.815
4	128	0.82	0.80	0.82	0.80	0.81
5	256	0.81	0.80	0.80	0.82	0.8075

3.3. Number of neurons variation

After obtaining the best activation function and batch size, further investigation is made to find the best number of neurons for the DNN model. The number of neurons in the DNN layers has a big impact on how well the DNN model performs. The model's accuracy and likelihood of generalizing successfully would both be enhanced by adding additional neurons, which would let it to learn more about the intricate underlying patterns in the data. Underfitting, in which the model is too basic to capture the link between the input and output data, results from using too few neurons, which prevents the model from comprehending the patterns in the data. Therefore, determining the best number of neurons in each layer of the DNN model is essential for improving the model's accuracy in data validation.

Figure 12 shows the graph for the training and validation loss when applying 64 neurons for each layer of the DNN model. It is observed that the validation loss for each layer is below 0.6 where the training loss is around 0.5. Based on the graph, hidden layer 4 demonstrates better performance compared to the other layers. This is evident from training loss and validation loss, which consistently decrease at the beginning and stabilized without significant fluctuations, indicating that the model neither overfits nor underfits.

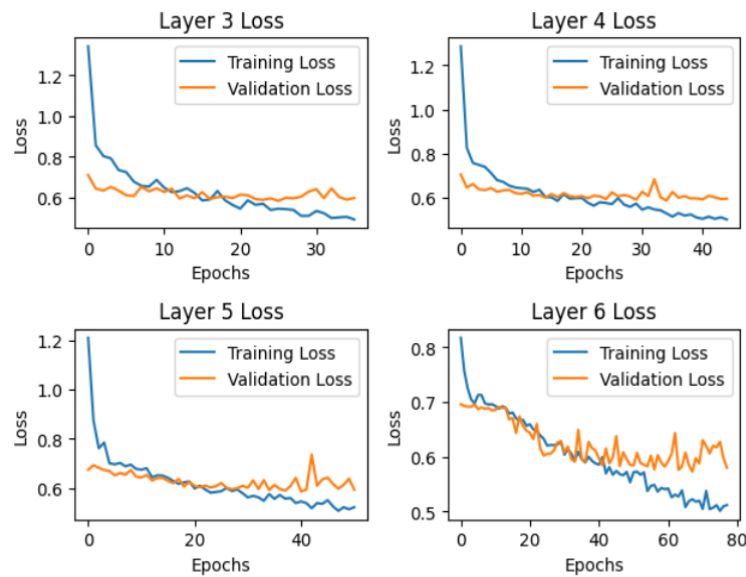


Figure 12. Training and validation loss for Leaky-ReLU, Sigmoid activation layer with 64 neurons per layer

The model's AUC score is displayed in Figure 13 for different numbers of neurons in each hidden layer. Figure 13(a) depicts the AUC score when the model employed 64 neurons in each layer for 3, 4, 5, and 6 hidden layer architecture. Next, Figure 13(b) shows the model performance when 128 neurons are used, Figure 13(c) shows the performance when 256 neurons are used and finally Figure 13(d) shows the performance of the model when 512 neurons are used in each hidden layer. The highest average AUC score is obtained when 512 neurons are employed in each layer as shown in Figure 13(d) and Table 4.

Table 4 depicts the average AUC score for various number of neurons where the highest average is achieved when using 512 neurons in each hidden layer. However, the AUC score obtained when using 3-hidden layer architecture with 256 neurons in each layer was also notably high, reaching a value of 0.82. This makes it a good contender to be employed in the DNN model as lower number of neurons may potentially reduce the computational cost and the training time of the model.

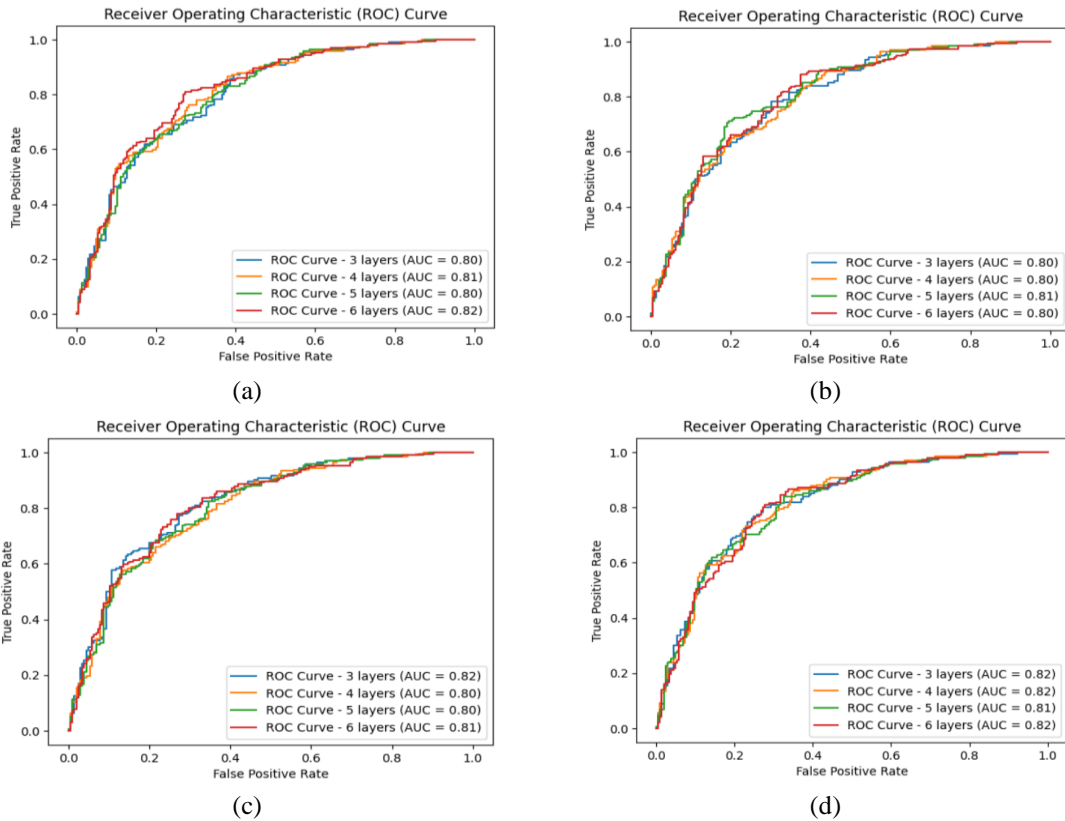


Figure 13. ROC curve for Leaky-ReLU, Sigmoid activation function with the number of neurons per layer: (a) 64, (b) 128, (c) 256, and (d) 512

Table 4. AUC score of the DNN model for various number of neurons

No.	Number of neurons in each layer	AUC from different number of hidden layers				Average AUC score
		3	4	5	6	
1	64	0.80	0.81	0.80	0.82	0.8075
2	128	0.80	0.80	0.81	0.80	0.8025
3	256	0.82	0.80	0.80	0.81	0.8075
4	512	0.82	0.82	0.81	0.82	0.8175

3.4. Image validation with DNN algorithms

After testing various scenario for the DNN model with varying number of hidden layers, activation functions, batch sizes and number of neurons, the best performance is obtained when employing a 3-hidden layer architecture with Leaky-ReLU, sigmoid activation function, 64 batch size and 256, 128, and 64 neurons in each respective hidden layer. This model is then applied to an image sensor in real-time for validating cat images where the model was able to achieve a validation accuracy of 94.31%. This result can be seen in Figure 14.

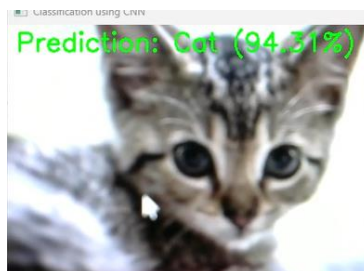


Figure 14. Validation result from applying the best DNN model

The DNN model developed in this research provides satisfactory results when compared with other research on real-time object validation, such as the work of Tan *et al.* [26], which achieved validation accuracy of up to 90%. There is an increase in validation accuracy using the proposed model of 4.31% compared to the performance obtained by Tan *et al.* [26]. Therefore, this research can be a valuable input for the progress of data mining, especially in the development of image recognition technology and image sensor-based devices.

4. CONCLUSION

The choice of activation functions, batch size, and the number of neurons significantly affects the performance of the DNN algorithm. A well-designed model is required for DNN algorithms, which excel at solving complex problems. This research developed a model within the DNN algorithm that achieves high validation accuracy. The use of a 3-hidden layer architecture with Leaky-ReLU and Sigmoid activation functions, a batch size of 64, and neurons set to 256, 128, and 64, was able to achieve a validation accuracy of 94.31% in real-time. These results can provide valuable insights for data mining, particularly in the application of DNN algorithms.





REFERENCES

- [1] H. Yi, S. Shiyu, X. Duan, and Z. Chen, "A study on deep neural networks framework," in *Proceedings of 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference, IMCEC 2016*, 2017, pp. 1519–1522, doi: 10.1109/IMCEC.2016.7867471.
- [2] J. Li, W. Liang, Y. Li, Z. Xu, X. Jia, and S. Guo, "Throughput maximization of delay-aware DNN inference in edge computing by exploring DNN model partitioning and inference parallelism," *IEEE Transactions on Mobile Computing*, vol. 22, no. 5, pp. 3017–3030, 2023, doi: 10.1109/TMC.2021.3125949.
- [3] H. Kwon, M. Pellauer, A. Parashar, and T. Krishna, "Flexion: a quantitative metric for flexibility in DNN accelerators," *IEEE Computer Architecture Letters*, vol. 20, no. 1, 2021, doi: 10.1109/LCA.2020.3044607.
- [4] S. Sindhu and M. Saravanan, "Architectural framework for multi sensor data fusion and validation in IoT based system," in *Proceedings - 2021 IEEE 10th International Conference on Communication Systems and Network Technologies, CSNT 2021*, 2021, pp. 448–452, doi: 10.1109/CSNT51715.2021.9509613.
- [5] C. Szegedy *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 1–9, doi: 10.1109/CVPR.2015.7298594.
- [6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 2, pp. 1097–1105, 2012.
- [7] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: semantic image segmentation with deep convolutional nets, Atrous convolution, and fully connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018, doi: 10.1109/TPAMI.2017.2699184.
- [8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.
- [9] He K, Zhang X, Ren S, and Sun J, "Identity mappings in deep residual networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9909 LNCS, 2016.
- [10] K. Van De Sande, T. Gevers, and C. Snoek, "Evaluating color descriptors for object and scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1582–1596, 2010, doi: 10.1109/TPAMI.2009.154.
- [11] K. Hammoudi, M. Melkemi, F. Dornaika, T. D. A. Phan, and O. Taoufik, "Computing multi-purpose image-based descriptors for object detection: powerfulness of LBP and its variants," *Advances in Intelligent Systems and Computing*, vol. 797, pp. 983–991, 2019, doi: 10.1007/978-981-13-1165-9_90.
- [12] K. P. Sinaga and M. S. Yang, "Unsupervised K-means clustering algorithm," *IEEE Access*, vol. 8, pp. 80716–80727, 2020, doi: 10.1109/ACCESS.2020.2988796.
- [13] F. M. Bianchi, D. Grattarola, and C. Alippi, "Spectral clustering with graph neural networks for graph pooling," in *37th International Conference on Machine Learning, ICML 2020*, 2020, vol. PartF16814, pp. 851–860.
- [14] Y. Wang, D. Chang, Z. Fu, and Y. Zhao, "Seeing all from a few: nodes selection using graph pooling for graph clustering," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 5, pp. 7231–7237, 2024, doi: 10.1109/TNNLS.2022.3210370.
- [15] P. C. Pop, "The generalized minimum spanning tree problem: an overview of formulations, solution procedures and latest advances," *European Journal of Operational Research*, vol. 283, no. 1, pp. 1–15, 2020, doi: 10.1016/j.ejor.2019.05.017.
- [16] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9370, pp. 84–92, 2015, doi: 10.1007/978-3-319-24261-3_7.
- [17] N. Vaswani, T. Bouwmans, S. Javed, and P. Narayanamurthy, "Robust subspace learning: robust PCA, robust subspace tracking, and robust subspace recovery," *IEEE Signal Processing Magazine*, vol. 35, no. 4, pp. 32–55, 2018, doi: 10.1109/MSP.2018.2826566.
- [18] S. R. Padupanambur, M. R. Ahmed, and B. P. Divakar, "Optimal state estimation techniques for accurate measurements in internet of things enabled microgrids using deep neural networks," *International Journal of Electrical and Computer Engineering*, vol. 12, no. 4, pp. 4288–4301, 2022, doi: 10.11591/ijece.v12i4.pp4288-4301.
- [19] A. Shrivastava, A. Kundu, C. Dhir, D. Naik, and O. Tuzel, "Optimize what matters: Training DNN-HMM keyword spotting model using end metric," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2021, vol. 2021-June, pp. 4000–4004, doi: 10.1109/ICASSP39728.2021.9414797.
- [20] X. Feng, B. Richardson, S. Amman, and J. Glass, "On using heterogeneous data for vehicle-based speech recognition: A DNN-based approach," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2015, vol. 2015-Augus, pp. 4385–4389, doi: 10.1109/ICASSP.2015.7178799.





- [21] B. Khabbazan, M. Riera, and A. González, "QeiHaN: an energy-efficient DNN accelerator that leverages log quantization in NDP architectures," in *Parallel Architectures and Compilation Techniques - Conference Proceedings, PACT*, 2023, pp. 325–326, doi: 10.1109/PACT58117.2023.00036.
- [22] I. Nurhaida, V. Ayumi, D. Fitriana, R. A. M. Zen, H. Noprisson, and H. Wei, "Implementation of deep neural networks (DNN) with batch normalization for batik pattern recognition," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 2, pp. 2045–2053, 2020, doi: 10.11591/ijece.v10i2.pp2045-2053.
- [23] M. Awad and R. Khanna, *Efficient learning machines*. Berkeley, CA: Apress, 2015, doi: 10.1007/978-1-4302-5990-9.
- [24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.
- [25] K. Saednia, A. Jalalifar, S. Ebrahimi, and A. Sadeghi-Naini, "An attention-guided deep neural network for annotating abnormalities in chest X-ray images: visualization of network decision basis," in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, 2020, vol. 2020-July, pp. 1258–1261, doi: 10.1109/EMBC44109.2020.9175378.
- [26] W. K. Tan, Z. Husin, and M. A. Hakim Ismail, "Feasibility study of beef quality assessment using computer vision and deep neural network (DNN) algorithm," in *2020 8th International Conference on Information Technology and Multimedia, ICIMU 2020*, 2020, pp. 243–246, doi: 10.1109/ICIMU49871.2020.9243353.

BIOGRAPHIES OF AUTHORS







Naemah Mubarakah     was born Mei 6, 1979, in Medan, North Sumatera Utara, Indonesia. She did his undergraduate work at Universitas Sumatera Utara in Medan, North Sumatera, Indonesia. She received a Bachelor of Electrical Engineering in 2003. She continued taking a master's program in electrical engineering at Institut Teknologi Sepuluh Nopember, Surabaya, East Java, Indonesia, from 2007-2009. Currently, she is doing the doctoral study in the Department of Computer Science, Faculty of Computer Science and Information Technology, Universitas Sumatera Utara, Indonesia. She is also a lecturer at Universitas Sumatera Utara. She can be contacted at naemah@usu.ac.id.







Poltak Sihombing     born in Sidikalang (Dairi). He obtained a bachelor's degree from the Department of Physics (computer concentration), Universitas Sumatera Utara in 1988, then completed a Master's degree in computer science (Universitas Indonesia) in 1999. He then obtained a doctorate in computer science from the University of Sains Malaysia in 2010. He is a lecturer who actively contributes and participates in several organizations. He once served as Chair of the Association of Computer Informatics Higher Education Region 1 North Sumatra-Aceh (APTİKOMWIL1). He can be contacted at email: poltak@usu.ac.id.



Syahril Efendi     was born November, 1967. He obtained a bachelor's degree from Universitas Sumatera Utara in 1994, then completed a master's degree Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia (UKM) in 1999. He then obtained a doctorate from the Universitas Sumatera Utara in 2013. Currently he serves as Chair of the Association of Computer Informatics Higher Education Region 1 North Sumatra-Aceh (APTİKOMWIL1), Medan. He can be contacted at syahril1@usu.ac.id.



Fahmi     was born December, 1979. He did his undergraduate work at the Bandung Institute of Technology in Bandung, West Java, Indonesia. He received a Bachelor of Electrical Engineering in 2002. He continued taking a master's program in sensor system technology, FH Karlsruhe, Germany, from 2003-2005. He completed his PhD in biomedical engineering at Universiteit van Amsterdam, Netherlands in 2010-2015. Currently, he is a lecturer at the Universitas Sumatera Utara. He can be contacted at fahmimn@usu.ac.id.