TELKOMNIKA Telecommunication Computing Electronics and Control
Vol. 24, No. 2, April 2026, pp. 663~675
ISSN: 1693-6930, DOI: 10.12928/TELKOMNIKA.v24i2.27619

Dynamic pooling using average-thresholding to improve image classification performance

Pajri Aprilio, Tjong Wan Sen
Master of Science in Information Technology, Faculty of Computer Science, President University, Bekasi, Indonesia

Article Info

Article history:

Received Oct 20, 2025
Revised Jan 14, 2026
Accepted Jan 30, 2026

Keywords:

Adaptive pooling
Average-thresholding pooling
Convolutional neural networks
Pooling layer
T-Max-Avg pooling

ABSTRACT

Pooling layers are essential in convolutional neural networks (CNNs) for reducing data size while preserving key features. Traditional methods such as Max and Average pooling have limitations. Max pooling is sensitive to noise, while Average pooling treats all activations equally. Although T-Max-Avg pooling addresses these limitations through adaptive top-k selection, its rigid decision rule requires multiple threshold comparisons and limits efficiency, motivating a simpler decision mechanism. This study introduces average-thresholding pooling (ATP), a simplified adaptive method that replaces multiple threshold comparisons with a single decision based on the average of the top-k activations. This design improves computational efficiency and reduces sensitivity to outliers. Experiments on the STL-10 dataset using a LeNet-5 architecture show that the proposed method achieves accuracy comparable to T-Max-Avg pooling (~55.5%) while consistently improving both training efficiency and inference speed. These results indicate that ATP provides a lightweight and practical alternative for CNN-based image classification, offering an improved balance between classification performance and computational efficiency.

Corresponding Author:

Pajri Aprilio
Master of Science in Information Technology, Faculty of Computer Science
President University
Jl. Ki Hajar Dewantara, Kota Jababeka, Cikarang Baru, Bekasi 17550, Indonesia
Email: aprilio.pajri@gmail.com

1. INTRODUCTION

Deep learning is a type of machine learning that uses artificial neural networks to extract patterns from large datasets [1], [2]. Convolutional neural networks (CNNs) have demonstrated outstanding performance in object segmentation and image classification. These capabilities have enabled effective solutions to numerous computer vision problems and successful real-world applications [3]–[6]. Its architecture typically consists of convolution, pooling, and fully connected layers as the primary components [3], [7], [8]. Pooling layer reduces the size of feature maps before sending them to the next layer, thereby lowering the computational cost of subsequent convolutional layers [6], [9]–[11].

The most commonly used pooling methods are max pooling and average pooling. Max pooling reduces dimensionality by selecting the highest activation value within a pooling window, representing the strongest signal [9], [12]. In contrast, average pooling reduces dimensionality by computing the mean of values in each pooling window [9], [13]. However, both methods have drawbacks. Max pooling is highly sensitive to noise and may lead to overfitting in some cases, while average pooling may blur important features and weaken strong signals by treating all values equally [3], [11], [14]. These limitations motivate the development of adaptive pooling strategies that balance robustness and feature preservation.

Despite the emergence of adaptive pooling strategies, existing methods still face a critical trade-off between robustness and efficiency. In particular, adaptive pooling approaches such as T-Max-Avg rely on rigid decision rules and multiple threshold comparisons, which increase computational cost and make pooling decisions sensitive to individual outliers.

One notable adaptive method is T-Max-Avg pooling proposed by Zhao and Zhang [15]. While this approach improves robustness compared to standard max and average pooling, its decision mechanism introduces new limitations. This method selects the top-k strongest values in each region and applies to a strict decision rule. If all values exceed a threshold, the output is the maximum; otherwise, it is the average of the top-k values (TKV). Compared to standard max pooling, this reduces sensitivity to noisy single values, and compared to Average pooling, it gives more weight to strong activations.

However, this rigid decision rule makes T-Max-Avg sensitive to outliers and computationally more demanding due to multiple threshold comparisons per pooling window [15]. A single low activation among the TKV can force the pooling operation to switch from selecting the maximum to computing the average, even when most activations are strong. Although adaptive pooling strategies such as T-Max-Avg improve feature selection compared to conventional methods, their reliance on multiple comparisons increases computational cost. It highlights the need for a more efficient yet effective pooling approach, particularly for lightweight CNNs and resource-constrained deployment scenarios.

This study introduces average-thresholding pooling (ATP), which compares the mean of the TKV to a threshold. This design simplifies decision-making, enhances robustness, and maintains competitive accuracy while making the pooling decision less sensitive to outliers. The objective of this research is to evaluate whether the proposed method can achieve comparable predictive performance with lower computational overhead.

The contributions of this study are as follows: (i) this study proposes ATP with a simplified adaptive pooling strategy that reduces the decision complexity of T-Max-Avg from $O(K)$ to $O(1)$ per pooling window, (ii) theoretical explanation is proposed to show that the proposed decision rule is less sensitive to individual outliers by relying on the mean of TKV activations rather than strict per-value thresholding, and (iii) it is demonstrated that the proposed method achieves accuracy comparable to T-Max-Avg while improving both training and inference efficiency.

2. METHOD

This work compares pooling decision rules under identical parameter settings, rather than performing extensive hyperparameter optimization. The proposed ATP method is evaluated as a drop-in replacement for T-Max-Avg, allowing differences in pooling behavior to be analyzed without confounding effects from parameter tuning. Analysis of parameter sensitivity is beyond the scope of this study and is left for future work.

The performance of ATP is evaluated against three baseline pooling methods: max pooling, average pooling, and T-Max-Avg. All methods are implemented using the same LeNet-5 architecture to ensure a fair comparison. Figure 1 summarizes the overall research workflow, including dataset selection, preprocessing, pooling method implementation, model training, and evaluation.

The experimental procedure of this study can be summarized as follows:

a. Input images are normalized using Min-Max normalization.
b. A LeNet-5 CNN is configured as the baseline model.
c. The pooling layers are replaced with different pooling strategies, including the proposed ATP method.
d. Model performance is evaluated using classification accuracy, training duration per epoch, memory consumption, and inference latency per sample.

The experiments were conducted using Python in the Google Colab environment to ensure accessibility and reproducibility. Google Colab provides a cloud-based platform with consistent computational resources, which helps reduce variability in experimental results. The complete Colab notebook is available at https://colab.research.google.com/drive/14SFi3UsqlKi7lZ37ZoIILrWFrUdSI6A1 enabling independent verification and replication of the experiments.

2.1. Data collection

The experiments were conducted on the STL-10 dataset, which contains 5,000 training and 8,000 testing images across 10 object categories (e.g., airplane, bird, car, cat) [16]. The dataset is designed for evaluating image classification algorithms and provides a balanced representation of various object classes. All images are color images with a fixed resolution of 96×96 pixels, making the dataset suitable for CNN–based models. Figure 2 shows sample images from the dataset.

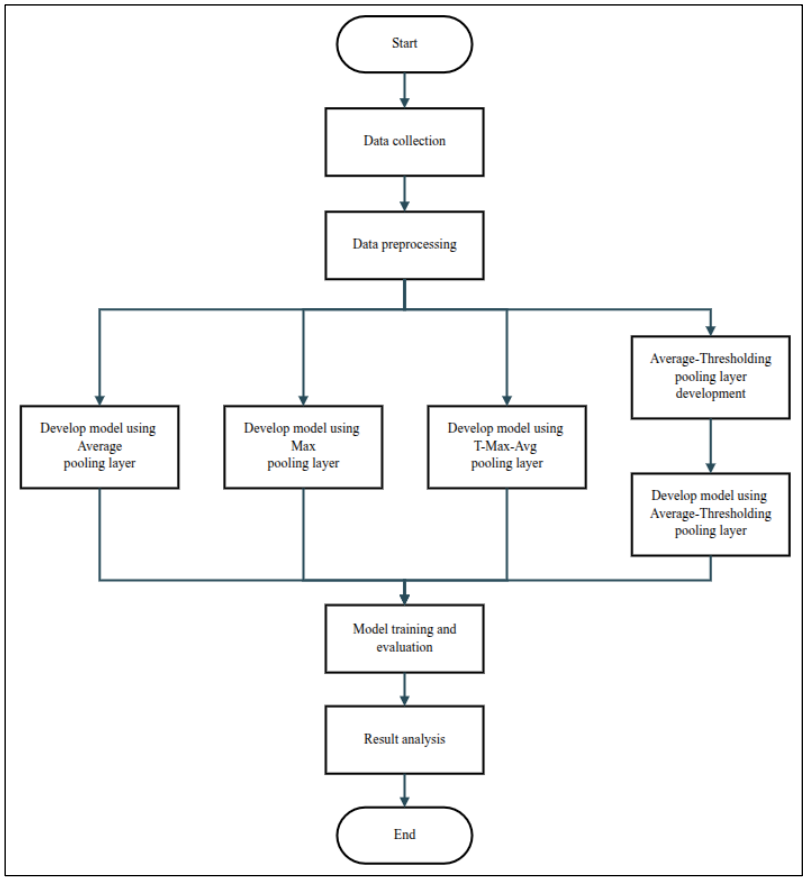


Figure 1. Overview of the experimental workflow used in this study

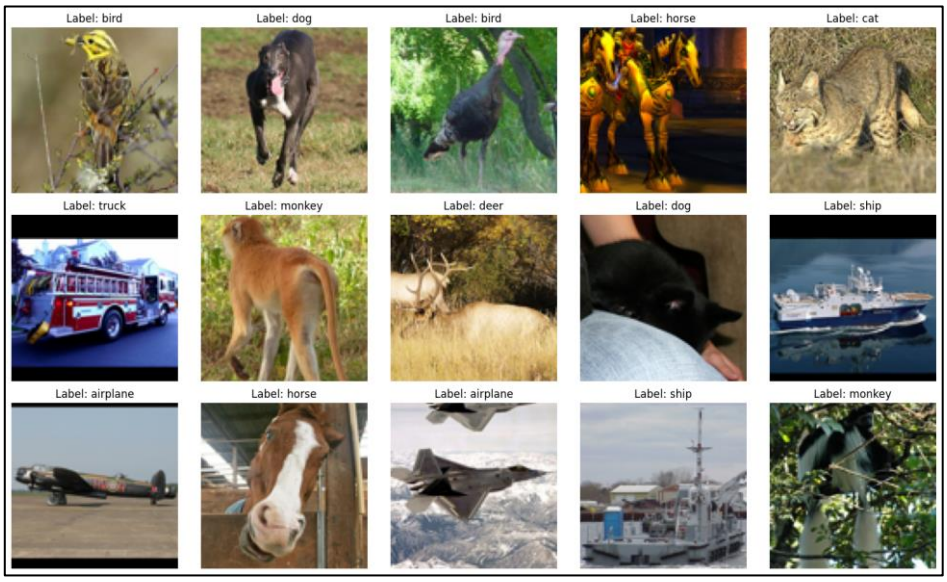


Figure 2. Example images from the STL-10 dataset across different object categories

2.2. Data preprocessing

Min-max normalization was applied to improve model convergence and generalization [17], [18]. This technique scales pixel values from the original range [0.255] to [0.1], ensuring consistent input distributions across all experiments [19], [20]. Normalizing the input data helps stabilize gradient updates

during training and reduces sensitivity to variations in pixel intensity. As a result, the learning process becomes more efficient and less prone to numerical instability.

2.3. Development of the ATP layer

The main methodological contribution of this work is the proposed ATP strategy. ATP computes the mean of the TKV within each pooling window and applies to a single threshold comparison to determine the pooling output. This design reduces computational overhead compared to existing adaptive pooling methods while preserving adaptive feature selection behavior.

The proposed method builds on T-Max-Avg [15] and introduces a simplified and more efficient decision rule. In T-Max-Avg, each of the TKV within a pooling window is compared individually to a threshold T. If all selected values satisfy the threshold condition, the maximum TKV is selected. Otherwise, their average is returned. Although this improves conventional max and average pooling, the decision rule requires multiple comparisons and is sensitive to individual low-valued outliers.

In contrast, ATP evaluates only the mean of the TKV and applies to a single threshold comparison. If the mean activation exceeds the threshold, the maximum TKV is selected; otherwise, the average of the TKV is returned. By relying on an aggregated decision rather than per-value comparisons, ATP reduces decision rigidity while maintaining adaptive pooling behavior. It reduces sensitivity to individual outliers and improves both robustness and efficiency. T-Max-Avg [15] is defined as (1):

$$F_{T\_Max\_Avg}(X) = \begin{cases} max\{Y_i\}_{i=0}^{K}, Y_i \geq T \\ \frac{1}{K}\sum_{i=1}^{K} Y_i\,, Y_i < T \end{cases} \quad (1)$$

In (1), X denotes the set of activations within a pooling window, and Yi represents the i-th largest value among the TKV. The parameter K specifies the number of selected activations, while $T \in [0,1]$ is the threshold. T-Max-Avg outputs the maximum of the TKV only when all selected values satisfy the threshold condition. Otherwise, their average is returned.

$$F_{avg\_thresholding}(X) = \begin{cases} max\{Y_i\}_{i=0}^{K}, \frac{1}{K}\sum_{i=1}^{K} Y_i \geq T \\ \frac{1}{K}\sum_{i=1}^{K} Y_i\,, \frac{1}{K}\sum_{i=1}^{K} Y_i < T \end{cases} \quad (2)$$

The proposed ATP is defined in (2) and follows the same notation as (1). Instead of performing K independent threshold comparisons, ATP computes the mean of the TKV and applies for a single threshold check. This modification reduces the number of threshold comparisons from $O(K)$ to $O(1)$ per pooling window. It improves computational efficiency while maintaining adaptive pooling behavior.

To provide theoretical justification for the proposed decision rule and its computational complexity, let $Y = \{y_1, y_2, \ldots, y_K\}$ denote the selected TKV activations within a pooling window, where one value y_0 is significantly smaller than the others. In T-Max-Avg pooling, each activation y_i must independently satisfy the threshold condition, resulting in K separate threshold comparisons. Consequently, a single low-valued activation is sufficient to force the pooling operation to switch from selecting the maximum to computing the average. In contrast, ATP evaluates only the mean of the TKV and performs a single threshold comparison. Because each activation contributes proportionally to the mean, the influence of an individual low value is reduced by a factor of $1/K$. As a result, the pooling decision reflects the collective response of the feature map rather than being dominated by a single extreme value, providing improved robustness while reducing decision complexity.

Table 1 summarizes the key workflow differences between T-Max-Avg and ATP, highlighting the reduced number of threshold comparisons required by the proposed method. Although both methods operate on the same TKV activations, their decision workflows differ substantially. T-Max-Avg relies on per-value thresholding, whereas ATP applies a single threshold to the mean of the TKV. As a result, the two methods exhibit different sensitivities to the choice of K, which explains why their optimal K values may differ in practice.

Table 1. Comparison of workflows highlighting the reduced threshold comparisons of the proposed ATP

Step	T-Max-Avg pooling	ATP
1	Select TKV	Select TKV
2	Compare each TKV with threshold	Compute mean of TKV
3	Perform K threshold checks	Perform 1 threshold check
4	Output max if all pass; else average	Output max if mean passes; else average
Cost	K comparisons	1 comparison

Figure 3 illustrates how the proposed ATP operates on a pooling window. With $_{K=3}$ and threshold $T = 0.7$, the top-3 values are first selected from each region. Their average is then compared with the threshold to decide the output. In the top-left region, the selected values are [0.9, 0.8, 0.7] with an average of 0.8, which is greater than 0.7, so the maximum value (0.9) is chosen. In contrast, in the bottom-left region the selected values [0.34, 0.33, 0.32] have an average of 0.33, which is below the threshold, so their average (0.32) is used as the output. The process is applied across all pooling windows, resulting in the final pooled output shown on the right side of Figure 3.

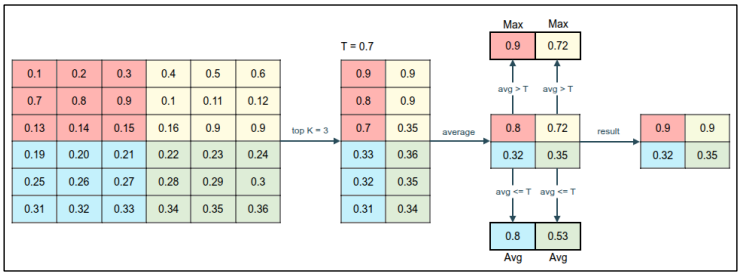


Figure 3. Illustration of the decision process of the proposed ATP, showing how the mean of the top-k activations determines the pooling output

In this study, the pooling method was implemented in Python using the Keras library. Keras supports fast experimentation, easy prototyping, and layer customization [21], [22]. "tf.keras.layers.Layer" class was used to develop an ATP layer and replicate T-Max-Avg from previous research. This study uses T4 GPU from Google Colab and Keras works well on CPU and GPU [22], [23].

In this study, the threshold value was fixed at 0.7 to remain consistent with the original T-Max-Avg proposed by Zhao and Zhang [15]. The reference work adopted this value without detailed parameter sensitivity analysis, and we intentionally followed the same setting to ensure a fair and controlled comparison between pooling strategies. By keeping the threshold constant, the impact of the proposed decision rule can be isolated without introducing confounding effects from parameter tuning.

2.4. Model development

To evaluate the performance of different pooling methods, all experiments were conducted using the same LeNet-5 architecture. LeNet-5 is a CNN originally developed by Lecun *et al*. [24] for handwritten character image classification and has been widely adopted due to its simplicity and effectiveness in image classification tasks [15], [25], [26]. The network consists of three convolutional layers, two pooling layers, and two fully connected layers, forming a lightweight and well-established baseline architecture [27]–[29]. Following the experimental design of T-Max-Avg studies, the same LeNet-5 architecture was retained to isolate the effect of the updated pooling strategy without confounding architectural changes [15]. Unlike prior pooling studies that focused on CIFAR and MNIST datasets, this work evaluates the proposed method on STL-10 to assess its behavior under different image resolution and distribution.

The critical variation across experiments lies in the pooling layers. Pooling plays an important role in reducing the spatial resolution of feature maps while lowering computational cost [30], [31]. In this study, the pooling layers following the first and second convolutional layers were replaced with one of four alternatives: max pooling, average pooling, T-Max-Avg, or the proposed ATP. Apart from this substitution, the network architecture, training setup, and hyperparameters were kept identical to ensure a fair comparison. Figure 4 illustrates the LeNet-5 architecture used in this study, while Table 2 summarizes the network structure and highlights the applied pooling strategies.

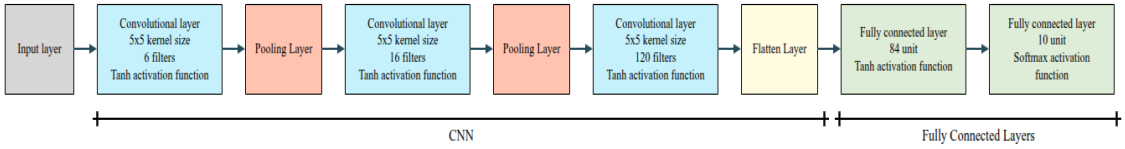


Figure 4. LeNet-5 architecture used in this study, with different pooling strategies applied at the pooling layers

Table 2. LeNet-5 architecture with different pooling methods

No	Layer name	Max pooling	Average pooling	T-Max-Avg	ATP
1	Input	Input	Input	Input	Input
2	Conv1	Conv1	Conv1	Conv1	Conv1
3	Pool1	Max	Average	T-Max-Avg	ATP
4	Conv2	Conv2	Conv2	Conv2	Conv2
5	Pool2	Max	Average	T-Max-Avg	ATP
6	Conv3	Conv3	Conv3	Conv3	Conv3
7	FC1	FC1	FC1	FC1	FC1
8	FC2	FC2	FC2	FC2	FC2

2.5. Hyperparameter settings and selection strategy

This study focuses on comparing pooling decision rules under controlled and identical parameter settings rather than performing extensive hyperparameter optimization. The goal is to focus only on how the pooling decision works, while avoiding confusion from heavy or aggressive parameter tuning. To ensure reproducibility and fair comparison, all pooling methods were evaluated using predefined parameter ranges adopted from prior pooling studies [15].

For both standard and adaptive pooling methods, pooling window sizes of (2×2), (3×3), and (4×4) were evaluated to ensure consistency and fair comparison across all experiments. Standard pooling methods (max pooling and average pooling) depend solely on the pooling window size and do not involve a TKV parameter. In contrast, adaptive pooling methods (T-Max-Avg and the proposed ATP) additionally employ a TKV selection mechanism, where the number of selected activations K was evaluated within the range of 1 to 10, with the valid values determined by the pooling window size. All evaluated pooling window and K combinations are summarized in Table 3 to support reproducibility.

Table 3. Summary of hyperparameters used in this study

Pooling method	Pool size	K	T
Average	(2×2), (3×3), (4×4)	-	-
Max	(2×2), (3×3), (4×4)	-	-
T-Max-Avg	(2×2)	1-3	0.7
	(3×3)	2-6	0.7
	(4×4)	2-10	0.7
ATP	(2×2)	1-3	0.7
	(3×3)	2-6	0.7
	(4×4)	2-10	0.7

The threshold parameter was fixed at $T = 0.7$ for all adaptive pooling experiments. This value was adopted from the original T-Max-Avg study by Zhao and Zhang [15], where it was used consistently across multiple datasets. Although no formal threshold sensitivity analysis was reported in the reference work, retaining this value allows for a controlled evaluation and ensures that observed performance differences arise from the pooling decision strategy rather than threshold tuning.

For each pooling method, all valid parameter combinations were evaluated under identical training and testing conditions. The configuration yielding the highest mean classification accuracy across six independent runs was selected as the representative result reported earlier. This selection strategy was applied consistently across all pooling methods to maintain fairness and methodological consistency.

2.6. Model training and evaluation

All models were trained for 50 epochs with a batch size of 128. The training was performed using the same optimizer, learning rate, and loss function across all pooling methods to ensure fairness. After training, models were evaluated on the STL-10 test set.

To account for variability in training outcomes, each experiment was repeated six times, and the mean accuracy was reported. In addition to classification accuracy, two efficiency measures were also recorded: training time per epoch and inference speed per sample. These metrics allow for a comprehensive comparison of pooling strategies, highlighting their impact on both accuracy and computational efficiency.

3. RESULTS AND DISCUSSION

For each pooling method, multiple parameter configurations were evaluated. The results reported in this section correspond to the configuration that achieved the highest average classification accuracy. All

evaluation metrics, including classification accuracy, training time, memory usage, and inference time, are reported as the mean and standard deviation over six independent runs. This section compares the proposed ATP method with Max pooling, Average pooling, and T-Max-Avg pooling. The analysis focuses on classification performance and computational efficiency under identical experimental conditions.

3.1. Accuracy

Accuracy is used as the primary metric to evaluate the effectiveness of different pooling methods for image classification. Table 4 summarizes the highest classification accuracy achieved by each pooling method under its accuracy-optimal configuration, allowing a direct comparison between conventional and adaptive pooling strategies. All experiments were repeated six times, and the reported results included mean accuracy and standard deviation to capture performance variability. Figure 5 illustrates the relative accuracy of trends across pooling methods.

Table 4. Classification accuracy across pooling methods under their best-performing configurations, illustrating the advantage of adaptive pooling strategies

Pooling method	Pool size	Accuracy (%)	Standard deviation (%)
Average	(4.4)	42.05	0.84
Max	(4.4)	54.61	0.66
T-Max-Avg ($\boldsymbol{K = 3, T = 0.7}$)	(4.4)	55.89	0.26
ATP ($\boldsymbol{K = 4, T = 0.7}$)	(4.4)	55.53	0.27

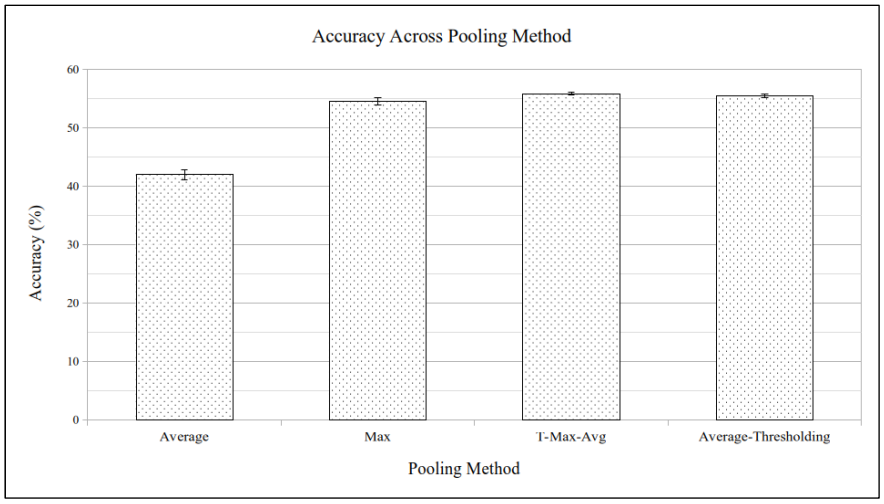


Figure 5. Classification accuracy comparison of pooling methods on STL-10

The results indicate that conventional pooling methods exhibit limited classification capability. Average pooling yields the lowest accuracy, while Max pooling provides a noticeable improvement but remains inferior to adaptive pooling approaches. In contrast, both T-Max-Avg and the proposed ATP achieve higher classification accuracy, exceeding 55% under the evaluated settings.

ATP achieves accuracy comparable to T-Max-Avg, with only a small absolute difference of 0.36%. This difference falls within one standard deviation across repeated runs, indicating that the two adaptive pooling methods exhibit similar classification performance rather than a statistically meaningful separation. These results suggest that modifying the pooling decision rule in ATP does not substantially alter the representational capacity of the network. Since ATP and T-Max-Avg operate on identical intermediate feature maps and differ only in how pooling decisions are triggered within each pooling window [15], large accuracy differences are not expected. Consequently, the observed similarity in accuracy reflects comparable feature preservation behavior. The implications of this design choice in terms of computational efficiency are examined in the subsequent sections through analyses of training time and inference speed. Per-run model performance across all configurations are provided in the supplementary materials (available at: https://docs.google.com/spreadsheets/d/1be27Ru7Nm1Mi9tJlqefkqbruw6M5QmUPEIK09ePOleY/edit?usp=sharing).

3.2. Training time

In addition to classification performance, training time is considered an indicator of the computational cost associated with different pooling strategies. Table 5 reports the average time required to complete 50 training epochs for each pooling method under the same accuracy-optimal configurations identified in section 3.1. The reported values represent the mean and standard deviation over six independent runs. Figure 6 provides a visual comparison of training duration across pooling methods.

The results show that conventional pooling methods require shorter training time. Average pooling achieves the lowest training duration, followed by Max pooling, reflecting their lower computational complexity. However, this efficiency is accompanied by reduced classification accuracy when compared to adaptive pooling methods. Adaptive pooling approaches incur longer training times due to additional operations involved in the pooling decision process. Among these methods, ATP demonstrates a slightly lower training time than T-Max-Avg, with a difference of 1.26 s. This difference lies within the observed variability across repeated runs, as indicated by the reported standard deviations, suggesting that the two adaptive pooling methods exhibit comparable training efficiency. Full per-run training time results across all evaluated configurations are provided in the supplementary materials (available at the following link: https://docs.google.com/spreadsheets/d/1be27Ru7Nm1Mi9tJlqefkqbruw6M5QmUPEIK09ePOleY/edit?gid=1300490738#gid=1300490738).

Table 5. Training time comparison across pooling methods under accuracy-optimal configurations, highlighting the computational cost of adaptive pooling

Pooling method	Pool size	Training time (s)	Standard deviation (s)
Average	(4.4)	39.75	1.16
Max	(4.4)	45.45	3.65
T-Max-Avg ($\boldsymbol{K} = \mathbf{3}, \boldsymbol{T} = \mathbf{0.7}$)	(4.4)	61.90	2.01
ATP ($\boldsymbol{K} = \mathbf{4}, \boldsymbol{T} = \mathbf{0.7}$)	(4.4)	60.64	2.83

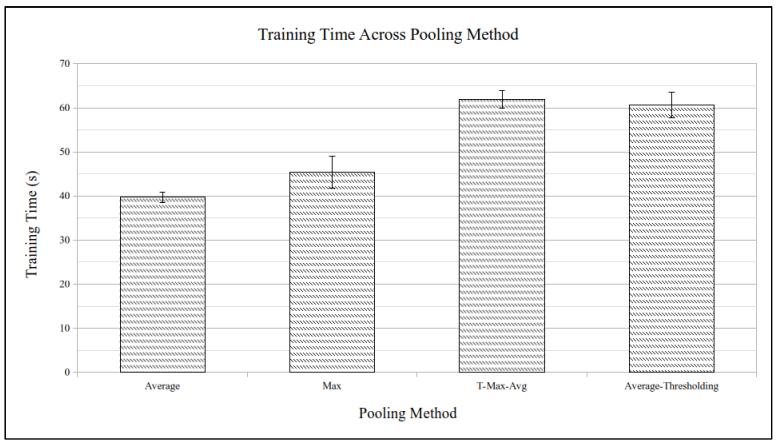


Figure 6 Training time comparison across pooling methods

3.3. Memory usage

Memory usage is an important consideration when deploying CNNs, particularly on hardware platforms with limited resources. Table 6 reports the peak memory usage observed during training for each pooling method under the same accuracy-optimal configurations. Figure 7 provides a visual comparison of memory consumption across pooling methods under identical experimental settings. Since memory allocation is deterministic when the network architecture and training parameters are fixed, standard deviation is not reported for this metric.

Table 6. Peak memory usage across pooling methods under accuracy-optimal configurations

Pooling method	Pool size	Memory usage (MB)
Average	(4.4)	880.15
Max	(4.4)	880.15
T-Max-Avg ($\boldsymbol{K} = \mathbf{3}, \boldsymbol{T} = \mathbf{0.7}$)	(4.4)	880.15
ATP ($\boldsymbol{K} = \mathbf{4}, \boldsymbol{T} = \mathbf{0.7}$)	(4.4)	880.15

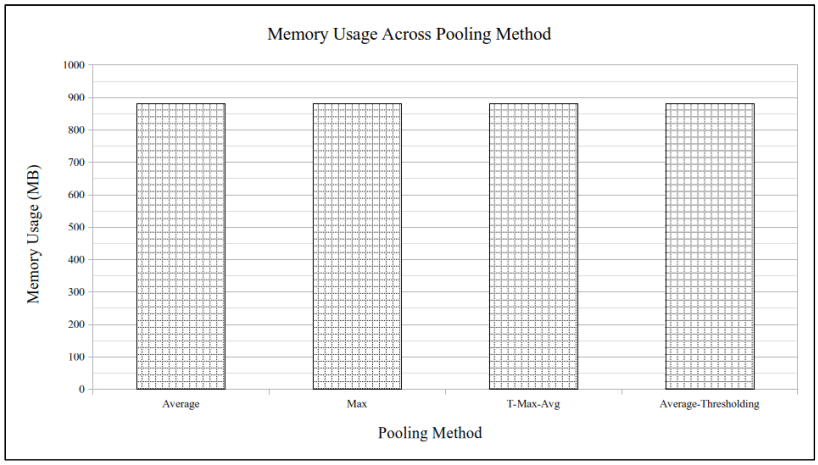


Figure 7. Memory consumption comparison across pooling methods

The results indicate that peak memory consumption remains identical across all evaluated pooling methods under the accuracy-optimal configuration. This outcome is expected because all experiments use the same network architecture, input resolution, and batch size, and the pooling operations do not introduce additional learnable parameters. Although higher memory usage is observed for smaller pooling windows, particularly in T-Max-Avg due to intermediate TKV operations, memory consumption becomes equivalent across pooling methods when larger pooling sizes are used. Consequently, under practical accuracy-optimal settings, adaptive pooling methods such as the proposed ATP do not incur additional memory overhead compared to conventional pooling strategies. These findings suggest that memory usage is primarily determined by architectural and pooling size choices rather than by the pooling decision mechanism itself. Full experimental results, including per-run memory usage across all configurations (pool sizes, thresholds, and K values), are provided in the supplementary materials (available at the following link: https://docs.google.com/spreadsheets/d/1be27Ru7Nm1Mi9tJlqefkqbruw6M5QmUPEIK09ePOleY/edit?gid=1484643535#gid=1484643535).

3.4. Inference speed

Inference speed is evaluated as the average processing time per test sample, measured in milliseconds. Table 7 reports the mean inference time and standard deviation for each pooling method under the same accuracy-optimal configurations. Figure 8 provides a visual comparison of inference latency across pooling strategies, allowing differences in runtime behavior to be examined.

Table 7. Inference time per sample across pooling methods under accuracy-optimal configurations, showing the efficiency advantage of ATP over T-Max-Avg

Pooling method	Pool size	Inference speed (ms/sample)	Standard deviation (ms/sample)
Average	(4.4)	438.68	39.26
Max	(4.4)	418.75	35.68
T-Max-Avg ($\boldsymbol{K = 3, T = 0.7}$)	(4.4)	494.32	70.51
ATP ($\boldsymbol{K = 4, T = 0.7}$)	(4.4)	482.52	65.25

The results show that conventional pooling methods achieve lower inference latency. Max pooling yields the shortest inference time, followed by Average pooling, reflecting their simpler computational operations. In contrast, adaptive pooling methods introduce additional latency due to more complex pooling decision processes.

Among the adaptive approaches, ATP exhibits a slightly lower inference time than T-Max-Avg. The observed difference in inference latency between the two methods falls within the variability across repeated runs, as indicated by the reported standard deviations, suggesting comparable inference efficiency. These results indicate that simplifying the pooling decision rule in ATP can reduce runtime overhead while maintaining classification performance. Full results, including per-run inference speed across all configurations (pool sizes, thresholds, and K values), are provided in the supplementary materials (available at: https://docs.google.com/spreadsheets/d/1be27Ru7Nm1Mi9tJlqefkqbruw6M5QmUPEIK09ePOleY/edit?gid=1812092776#gid=1812092776).

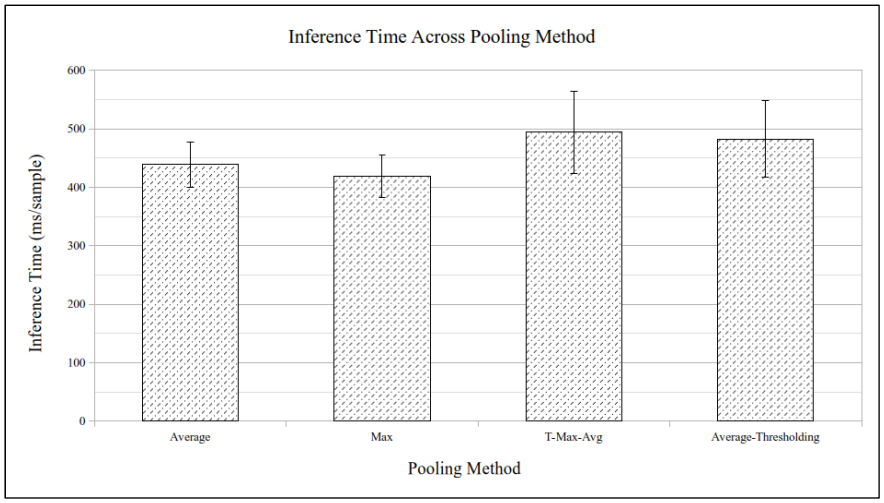


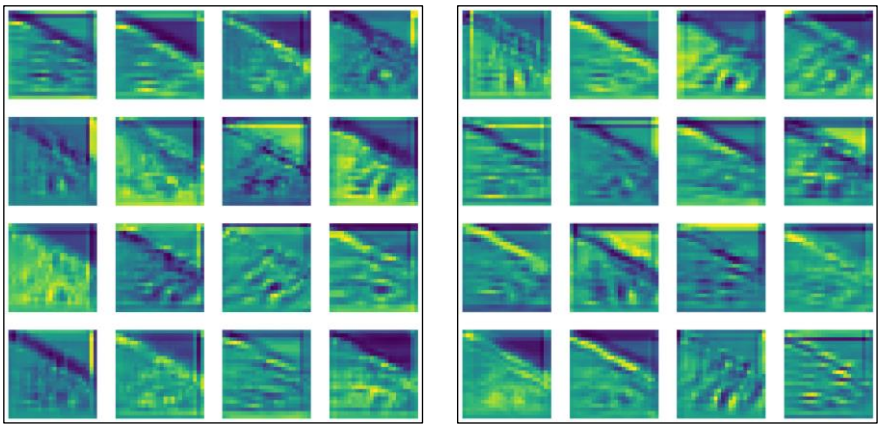
Figure 8. Inference time per sample across pooling methods

3.5. Qualitative feature map analysis

To examine how different pooling decision rules affect intermediate representations, feature maps from the Conv2 layer are visualized for both adaptive pooling methods. Figure 9(a) shows the Conv2 feature maps obtained using T-Max-Avg pooling, while Figure 9(b) presents the feature maps generated by the proposed ATP method. Overall, both methods produce highly similar activation patterns, including comparable edge structures and texture patterns. This similarity is expected, as both pooling strategies operate on identical TKV activations and preserve the same convolutional filters. Therefore, the proposed ATP does not alter the feature extraction process but only modifies the pooling decision rule.

Subtle differences can be observed in activation characteristics. Feature maps generated by T-Max-Avg tend to show sharper and more concentrated activations, whereas ATP produces slightly smoother and more uniformly distributed responses. This behavior is consistent with their respective decision mechanisms: T-Max-Avg relies on strict per-value threshold comparisons, while ATP bases its decision on the mean of the TKV activations, thereby reducing the influence of isolated extreme values.

These visual differences do not lead to a noticeable difference in classification accuracy, as confirmed by the quantitative results presented earlier. Instead, the primary advantage of ATP lies in its reduced computational complexity, which leads to faster inference while preserving discriminative feature representations comparable to T-Max-Avg.



(a) (b)

Figure 9. Comparison of Conv2 feature maps obtained using: (a) T-Max-Avg and (b) ATP.Conv2 outputs are shown to illustrate downstream effects of pooling decision rules

3.6. Summary of findings

The experimental results demonstrate that the choice of pooling strategy has a clear impact on both classification performance and computational efficiency. Conventional pooling methods offer lower computational cost but exhibit reduced classification effectiveness compared to adaptive pooling approaches. Adaptive methods, including T-Max-Avg and the proposed ATP, consistently achieve higher accuracy by better preserving informative feature activations [15].

A key observation from the results is that ATP achieves classification performance comparable to T-Max-Avg while requiring lower computational cost during training and inference. This behavior is expected since both methods operate on identical intermediate feature maps and differ only in their pooling decision mechanisms [15]. By simplifying the decision rule, ATP maintains similar representational capability while reducing computational overhead.

Memory usage remains unchanged across pooling methods under accuracy-optimal configurations, indicating that adaptive pooling does not introduce additional memory requirements in practical settings. Overall, ATP provides a favorable balance between classification performance and efficiency, making it a practical alternative to more complex adaptive pooling methods. These results suggest that ATP is well suited for applications where computational efficiency is an important consideration alongside predictive performance.

The proposed ATP achieved classification accuracy comparable to T-Max-Avg on STL-10 while consistently improving training and inference efficiency. In addition to maintaining competitive accuracy, ATP demonstrated improved efficiency, achieving faster training (60.64 s vs. 61.90 s) and faster inference (482.52 ms/sample vs. 494.32 ms/sample) compared to T-Max-Avg. These results confirm that ATP successfully balances accuracy and computational efficiency.

4. CONCLUSION

The primary contribution of this work is the introduction of a simplified adaptive pooling rule that replaces multiple threshold comparisons with a single comparison based on the average of the TKV activations. This simplified decision mechanism reduces computational complexity and lowers decision rigidity. As a result, ATP becomes less sensitive to individual outliers and offers a practical, lightweight alternative to existing adaptive pooling methods.

This study is limited to experiments conducted on a single dataset (STL-10) and a single CNN architecture (LeNet-5). As a result, the generalizability of the proposed pooling method to deeper networks and more complex datasets remains uncertain. Further validation is required to confirm its effectiveness across diverse architectures and datasets. In addition, the evaluation was conducted using a single threshold and K configuration, following the setting adopted in the original T-Max-Avg. While this choice enables a fair and controlled comparison between pooling decision rules, it does not capture the full sensitivity of the method to parameter variations.

Future work will focus on evaluating the scalability and robustness of ATP by applying it to deeper CNN architectures such as VGG and ResNet. In addition, ATP will be tested on larger and more diverse datasets, including CIFAR-10 and ImageNet. Further studies will investigate adaptive threshold learning mechanisms, including data-driven or learnable threshold formulations, as well as sensitivity analysis of threshold and K values to better understand their influence on performance across different architectures and datasets. Its applicability to other computer vision tasks, such as object detection and image segmentation, will also be investigated.

Overall, ATP provides a lightweight adaptive pooling alternative that preserves classification performance while reducing computational overhead. By simplifying the pooling decision rule, the method offers practical benefits for efficient CNN deployment. This makes ATP particularly suitable for real-time and resource-constrained environments.

ACKNOWLEDGMENTS

The authors would like to thank Faculty of Computer Science, President University, for the academic support and environment provided during the completion of this study. The authors also thank colleagues and peers for their constructive discussions and technical insights that contributed to the development of this work.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Pajri Aprilio	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
Tjong Wan Sen	✓	✓		✓						✓		✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interest in this manuscript

DATA AVAILABILITY

The data that supports the findings of this study will be available in http://cs.stanford.edu/~acoates/stl10.

REFERENCES

[1] C. Janiesch, P. Zschech, and K. Heinrich, "Machine learning and deep learning," *Electronic Markets*, vol. 31, no. 3, pp. 685–695, Sep. 2021, doi: 10.1007/s12525-021-00475-2.

[2] K. Sharifani and M. Amini, "Machine learning and deep learning: A review of methods and applications," *World Information Technology and Engineering Journal*, vol. 10, no. 7, pp. 3897–3904, 2023.

[3] A. Zafar *et al.*, "A comparison of pooling methods for convolutional neural networks," *Applied Sciences*, vol. 12, no. 17, Aug. 2022, doi: 10.3390/app12178643.

[4] T. Turay and T. Vladimirova, "Toward performing image classification and object detection with convolutional neural networks in autonomous driving systems: A survey," *IEEE Access*, vol. 10, pp. 14076–14119, 2022, doi: 10.1109/ACCESS.2022.3147495.

[5] Y. Dogan, "A new global pooling method for deep neural networks: Global average of top-K max-pooling," *Traitement du Signal*, vol. 40, no. 2, pp. 577–587, Apr. 2023, doi: 10.18280/ts.400216.

[6] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, "A review of convolutional neural networks in computer vision," *Artificial Intelligence Review*, vol. 57, no. 4, Mar. 2024, doi: 10.1007/s10462-024-10721-6.

[7] A. Shah *et al.*, "A comprehensive study on skin cancer detection using artificial neural network (ANN) and convolutional neural network (CNN)," *Clinical eHealth*, vol. 6, pp. 76–84, Dec. 2023, doi: 10.1016/j.ceh.2023.08.002.

[8] M. A. Saleem, N. Senan, F. Wahid, M. Aamir, A. Samad, and M. Khan, "Comparative analysis of recent architecture of convolutional neural network," *Mathematical Problems in Engineering*, vol. 2022, pp. 1–9, Mar. 2022, doi: 10.1155/2022/7313612.

[9] G. J. Trivedi and R. Sanghvi, "Medical image fusion Using CNN with automated pooling," *Indian Journal Of Science And Technology*, vol. 15, no. 42, pp. 2267–2274, Nov. 2022, doi: 10.17485/IJST/v15i42.1812.

[10] R. Nirthika, S. Manivannan, A. Ramanan, and R. Wang, "Pooling in convolutional neural networks for medical image analysis: A survey and an empirical study," *Neural Computing and Applications*, vol. 34, no. 7, pp. 5321–5347, Apr. 2022, doi: 10.1007/s00521-022-06953-8.

[11] G. Rangel, J. C. Cuevas-Tello, J. Nunez-Varela, C. Puente, and A. G. Silva-Trujillo, "A survey on convolutional neural networks and their performance limitations in image recognition tasks," *Journal of Sensors*, vol. 2024, no. 1, Jan. 2024, doi: 10.1155/2024/2797320.

[12] L. Hu, L. Gao, Z. Liu, and W. Feng, "Temporal lift pooling for continuous sign language recognition," in *Computer Vision – ECCV 2022: 17th European Conference*, 2022, pp. 511–527. doi: 10.1007/978-3-031-19833-5_30.

[13] U. Nandi, A. Ghorai, M. M. Singh, C. Changdar, S. Bhakta, and R. Kumar Pal, "Indian sign language alphabet recognition system using CNN with diffGrad optimizer and stochastic pooling," *Multimedia Tools and Applications*, vol. 82, no. 7, pp. 9627–9648, Mar. 2023, doi: 10.1007/s11042-021-11595-4.

[14] W. Ayivi, X. Zhang, W. X. Ativi, F. Sam, and F. A. P. Kouassi, "Dynamic–attentive pooling networks: A hybrid lightweight deep model for lung cancer classification," *Journal of Imaging*, vol. 11, no. 8, Aug. 2025, doi: 10.3390/jimaging11080283.

[15] L. Zhao and Z. Zhang, "A improved pooling method for convolutional neural networks," *Scientific Reports*, vol. 14, no. 1, Jan. 2024, doi: 10.1038/s41598-024-51258-6.

[16] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.

[17] M. Shantal, Z. Othman, and A. A. Bakar, "A novel approach for data feature weighting using correlation coefficients and min–max normalization," *Symmetry*, vol. 15, no. 12, Dec. 2023, doi: 10.3390/sym15122185.

[18] J. Shao, K. Hu, C. Wang, X. Xue, and B. Raj, "Is normalization indispensable for training deep neural networks?," in *NIPS'20: Proceedings of the 34th International Conference on Neural Information Processing Systems*, 2020, pp. 13434–13444.

[19] P. J. Muhammad Ali, "Investigating the impact of min-max data normalization on the regression performance of K-nearest neighbor with different similarity measurements," *ARO-The Scientific Journal of Koya University*, vol. 10, no. 1, pp. 85–91, Jun. 2022, doi:

10.14500/aro.10955.
[20] H. A. Ahmed, P. J. Muhammad Ali, A. K. Faeq, and S. M. Abdullah, "An investigation on disparity responds of machine learning algorithms to data normalization method," *ARO-The Scientific Journal of Koya University*, vol. 10, no. 2, pp. 29–37, Sep. 2022, doi: 10.14500/aro.10970.
[21] B. T. Chicho and A. Bibo Sallow, "A comprehensive survey of deep learning models based on Keras framework," *Journal of Soft Computing and Data Mining*, vol. 2, no. 2, Oct. 2021, doi: 10.30880/jscdm.2021.02.02.005.
[22] S. Yousef, "Applying Keras-based deep learning for intelligent analysis in network security and monitoring systems," *Babylonian Journal of Networking*, vol. 2025, pp. 106–115, Jul. 2025, doi: 10.58496/BJN/2025/009.
[23] A. Zollanvari, "Deep learning with Keras-TensorFlow," in *Machine Learning with Python*, Cham: Springer International Publishing, 2023, pp. 351–391. doi: 10.1007/978-3-031-33342-2_13.
[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.
[25] G. Fu *et al.*, "A multi-scale pooling convolutional neural network for accurate steel surface defects classification," *Frontiers in Neurorobotics*, vol. 17, Feb. 2023, doi: 10.3389/fnbot.2023.1096083.
[26] A. K. Singha, M. Jena, S. Zubair, P. K. Tiwari, and A. P. S. Bhadauria, "Deep neural networks performance comparison for handwritten text recognition," in *Mobile Radio Communications and 5G Networks (MRCN 2023)*, 2024, pp. 539–553. doi: 10.1007/978-981-97-0700-3_42.
[27] H. Lee, J. Park, and J. Y. Hwang, "Channel attention module with multi-scale grid average pooling for breast cancer segmentation in an ultrasound image," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, pp. 1–1, 2020, doi: 10.1109/TUFFC.2020.2972573.
[28] Y. Li, M. Lei, Y. Cheng, R. Wang, and M. Xu, "Convolutional neural network with Huffman pooling for handling data with insufficient categories: A novel method for anomaly detection and fault diagnosis," *Science Progress*, vol. 105, no. 4, Oct. 2022, doi: 10.1177/00368504221135457.
[29] M. M. Taye, "Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions," *Computation*, vol. 11, no. 3, Mar. 2023, doi: 10.3390/computation11030052.
[30] I. Shadoul, R. Al-Hmouz, A. Hossen, M. Mesbah, and M. Deveci, "The effect of pooling parameters on the performance of convolution neural network," *Artificial Intelligence Review*, vol. 58, no. 9, Jun. 2025, doi: 10.1007/s10462-025-11273-z.
[31] H. Çetiner and S. Metlek, "Analysis of different pooling functions on a convolution neural network based model," *International Journal of 3D Printing Technologies and Digital Industry*, vol. 8, no. 2, pp. 266–276, Aug. 2024, doi: 10.46519/ij3dptdi.1484354.

BIOGRAPHIES OF AUTHORS

Pajri Aprilio received his associate degree in Informatics from Bandung State Polytechnic, Bandung, Indonesia, in 2013, and his Bachelor's degree in Informatics from Widyatama University, Bandung, Indonesia, in 2023. He is currently pursuing a Master's degree in Information Technology at President University, Bekasi, Indonesia. His research interests include deep learning, computer vision, and artificial intelligence. He can be contacted at email: aprilio.pajri@gmail.com.

Tjong Wan Sen earned his Ph.D. from Institut Teknologi Bandung in 2009. Since 2010, he has been affiliated with the Master of Informatics program within the Faculty of Computing at President University, Jababeka, Indonesia, where he currently serves as a lecturer and researcher. His research endeavors encompass generative adversarial networks, low-powered artificial intelligence, computer vision, the internet of things, and embedded systems. He can be contacted at email: wansen@president.ac.id.