Indonesian Journal of Electrical Engineering and Computer Science
Vol. 44, No. 1, October 2026, pp. 177~185
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v44.i1.pp177-185

Min-dropout: deterministic dropout via low-activation neuron removal

Tjong Wan Sen[1], Tahir Mehmood[2], Pajri Aprilio[1]
[1]Faculty of Computer Science, President University, Bekasi, Indonesia
[2]Faculty of Artificial Intelligence and Frontier Technologies, UNITAR International University, Petaling Jaya, Malaysia

Article Info

Article history:

Received Nov 23, 2025
Revised Aug 28, 2026
Accepted Sep 10, 2026

Keywords:

Deep learning
Dropout algorithm
Performance improvement
Selective neuron deactivation
Small neuron value

ABSTRACT

The dropout algorithm is a widely utilized technique in neural networks to mitigate overfitting. However, the conventional dropout method randomly deactivates neurons. This research introduces a novel dropout algorithm that selectively deactivates k neurons with the lowest values within each layer, rather than employing random deactivation. The underlying hypothesis is that neurons with low values contribute minimally to the model's performance. The objective of this research is to improve model performance by removing neurons with low activation. Experimental validation was conducted using the iris and heart disease datasets from the UCI machine learning repository, containing 150 and 303 data entries, respectively. The proposed min-dropout algorithm achieved the highest test accuracies of 75.34% on the Iris dataset and 56.67% on the heart disease dataset at a dropout rate of 0.5. While standard dropout achieved higher test accuracy at moderate dropout rates on the Iris dataset, min-dropout produced the best results at the higher dropout rate on both datasets. These findings suggest that removing low-activation neurons can be an alternative to standard dropout. It may achieve better performance under certain dropout configurations. Further studies on larger and more diverse datasets are required to assess the general applicability of the proposed method.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Tahir Mehmood
Faculty of Artificial Intelligence and Frontier Technologies
UNITAR International University, Faculty of Computer Science
Petaling Jaya, Selangor, Malaysia
Email: tahir.mehmood@unitar.my

1. INTRODUCTION

Deep learning is a subfield of machine learning (ML) that enables models to learn complex patterns from data through multiple processing layers [1]. These layers are capable of achieving outstanding performance across a wide range of tasks, including image classification and structured data analysis [2]. As the number of layers or neurons increases, neural networks can capture richer feature representations and achieve higher model performance. Neural networks can achieve strong performance across many tasks, but they are also prone to overfitting, where a model performs well on training data but fails to generalize to unseen data [3]–[7]. In such cases, the network memorizes noise and specific details in the training set rather than learning the underlying data distribution [8]. This case limits the model's ability to perform reliably in real-world scenarios [9], [10]. To address this problem, various regularization techniques have been developed to improve generalization. Dropout is commonly used to reduce this problem by randomly masking activation values during training [11], [12]. By introducing this randomness, dropout encourages the network to learn redundant, robust representations and reduces the dependency on any single neuron.

However, standard dropout has limitations. It deactivates neurons randomly in each layer during training [13]. However, random masking may also remove informative activations during training, thereby disrupting the learning process and reducing model performance. Consequently, many researchers have proposed alternative dropout strategies to achieve more controlled or data-dependent neuron selection.

Several dropout variants have been proposed to improve neuron selection during training. Salehinejad and Valaee [14] proposed EDropout, an energy-based dropout/pruning framework. Lu *et al*. [15] proposed LocalDrop, a hybrid regularization method based on local Rademacher complexity that jointly optimizes the keep-rate matrix and network weights to improve the generalization of fully connected and convolutional neural networks. R-Drop introduces consistent regularization between parallel subnetworks [16], AS-Dropout uses adaptive activation-based probabilities [17], and CorrDrop removes neurons according to feature correlation information [18]. Although these methods improve generalization performance, they often require additional computations and complexity. In contrast, the original dropout method remains computationally simple but relies on random neuron deactivation [12]. Therefore, a simple dropout strategy that performs more informed neuron selection without introducing substantial computational overhead is still needed. However, deterministic neuron removal may reduce the randomness that helps prevent co-adaptation between neurons [12]. This potentially creates a trade-off between deterministic neuron selection and the random neuron selection used in standard dropout. This study introduces a new dropout variant called min-dropout. Unlike standard dropouts, min-dropout does not use random masking. Instead, it applies deterministic low-activation filtering. This research is supported by activation-based pruning studies by Hu *et al.* [19], which report that neurons with low activations may contribute less to the subsequent layers and to the model's decision-making process. The main objective of this research is to design, implement, and evaluate the min-dropout algorithm as an alternative to the standard dropout method. The performance of min-dropout is compared against both standard dropout and no-dropout models.

2. METHOD

2.1. Research stages

The general workflow of this research is three sequential stages, from dataset preparation to model evaluation. The study used mainly the UCI ML repository iris and heart disease datasets [20], [21]. The research process involved dataset preparation, pre-processing, model development, training and evaluation, and comparison of performance. We tested three model settings: without dropout, with standard dropout, and with the proposed min-dropout. Then the results were compared to evaluate the effect of deterministic neuron selection on model performance and generalization.

2.2. Data pre-processing

Prior to training, both datasets were pre-processed to ensure consistency and compatibility with the neural network models. The steps of preprocessing are normalization (the process of scaling all the numerical features in the range of 0 to 1 to stabilize the training process), label encoding (the process of converting the non-numerical categorical attributes to integer codes), one-hot encoding (the process of converting the target classes to one-hot encoded vectors for the iris and heart disease dataset labels), and train-test split (the process of dividing each dataset into 80% training and 20% testing subsets with the help of a fixed random seed to make the results reproducible).

2.3. Model development and training

Three model architectures were developed to compare different dropout configurations: a model with no dropout (baseline model), standard dropout, and a min-dropout layer. The model architecture of the evaluated model is summarized in Table 1. Each model was trained using identical hyperparameter settings to ensure a fair comparison, including the Adam optimizer, 500 epochs, a batch size of 150, a 0.001 learning rate, categorical cross-entropy for the Iris dataset, binary cross-entropy for the heart disease dataset, and accuracy as the evaluation metric. All models consisted of two dense hidden layers with ReLU activation functions and an output layer with sigmoid activation. The no-dropout model did not use any dropout layer, while the standard dropout and min-dropout models applied a dropout layer after each dense hidden layer. The number of units in each dense hidden and output layer was set equal to the number of input features and classes, respectively.

2.4. Min-dropout algorithm development

The method for dropping minimum values is inspired by denoising techniques utilizing wavelets [22] where salient patterns (large values) encode the majority of the information, whereas small values primarily represent noise, based on our investigation on robust automatic speech recognition [23], [24].

Table 1. Model architecture comparison

No	Layer	No dropout	Standard dropout	Min dropout
1	Input	Number of input features	Number of input features	Number of input features
2	Dense	Units: *n*, ReLU	Units: *n*, ReLU	Units: *n*, ReLU
3	Dropout	-	Standard Dropout, rate = r	Min-dropout, rate = r
4	Dense	Units: *n*, ReLU	Units: *n*, ReLU	Units: *n*, ReLU
5	Dropout	-	Standard dropout, rate = r	Min-dropout, rate = r
6	Output	Units: number of classes, Sigmoid	Units: number of classes, Sigmoid	Units: number of classes, Sigmoid

Note: n denotes the number of input features

The algorithm performs six main operations,

a) Determine the number of neurons to drop: the algorithm first calculates how many activation values should be set to zero based on the predefined dropout rate (r) that represents the proportion of neurons to deactivate. The number of neurons to be dropped is given by,

$$N_{drop} = \lfloor r \times N_{total} \rfloor \quad (1)$$

where N_{total} is the total number of activations in the current layer.

b) Flatten the input tensor: The multi-dimensional activation tensor is reshaped into a one-dimensional vector to efficiently identify the indices of the lowest activation values.

c) Identify the lowest activation values: The indices of the N_{drop} smallest activation values are identified using a top-k operation on the negative activations. These indices represent neurons that contribute the least to the layer's output.

d) Generate the dropout mask: a binary dropout mask is created with zeros assigned to the selected indices and ones assigned elsewhere. This mask indicates which neurons are deactivated and which remain active.

e) Reshape mask to original dimensions: the one-dimensional mask is reshaped to the original tensor's dimensions, preserving the structural alignment of neurons across the layer.

f) Apply the mask to the layer output: the mask is applied through element-wise multiplication,

$$Y = X \odot M \quad (2)$$

where X is the input tensor, M is the dropout mask, and Y is the output tensor. This is further presented in Algorithm 1.

Algorithm 1. The algorithm performs six main operations

```
// 1. Determine the total number of neurons (N_total)
       N_total = SIZE(X)
// 2. Determine the number of neurons to drop (N_drop)
       N_drop = FLOOR(r * N_total)
// Check if N_drop is zero, if so, return original tensor
IF N_drop == 0 THEN
 RETURN X
END IF
// 3. Flatten the input tensor, store the original shape for later reshaping
       Original_Shape = SHAPE(X)
       X_flat = FLATTEN(X) // Reshapes X into a 1D vector
// 4. Identify the indices of the lowest activation values
// A. Negate the flattened tensor
       X_neg = -X_flat
// B. Find the indices of the N_drop largest values in X_neg.
// This is equivalent to finding the indices of the N_drop smallest values in X_flat.
// TOP_K_INDICES returns the indices of the largest K elements.
       Indices_to_Drop = TOP_K_INDICES(X_neg, N_drop)
// 5. Generate the dropout mask (M_flat)
// Initialize a 1D mask of ones with the same size as X_flat
       M_flat = ONES(SIZE(X_flat))
// Set the elements at the identified indices to zero
FOREACH index IN Indices_to_Drop DO
 M_flat[index] = 0
END FOR
// 6. Reshape the mask to the original dimensions
       M = RESHAPE(M_flat, Original_Shape)
// 7. Apply the mask to the layer output (Element-wise multiplication)
       Y = X * M // Element-wise multiplication (Hadamard product)
// 8. Output the resulting tensor
       RETURN Y
```

The min-dropout algorithm was implemented by extending keras.layers.Layer to allow integration with existing architectures [25], [26]. The min-dropout algorithm during training performs the dropout operation, while during inference, it returns the unmodified input tensor to preserve the model's learned structure.

For example, given an input tensor X,

$$X = [2,4,1,3,5,2,3,1] \tag{3}$$

If the dropout rate is set to r=0.25, two lowest activations are dropped, as shown in (4).

$$N_{drop} = \lfloor 0.25 \times 8 \rfloor = 2 \tag{4}$$

The two smallest activation values (1 and 1) are identified, and a binary mask M is created as follows,

$$M = [1,1,0,1,1,1,1,0] \tag{5}$$

Applying the mask produces the output tensor,

$$Y = X \odot M = [2,4,0,3,5,2,3,0] \tag{6}$$

It should be noted that min-dropout differs from standard dropout in its masking mechanism. Standard dropout applies random masking to activation values, where the keep/drop decision is generated independently based on a predefined probability. In contrast, the current min-dropout implementation applies deterministic masking by flattening the activation tensor in the current batch, selecting the k-lowest activation values according to the dropout rate, and setting their values to zero. Therefore, min-dropout is best interpreted as a dropout-inspired activation filtering method rather than a direct equivalent of standard dropout. This distinction is important because the current implementation performs batch-level activation filtering, where the selected activations depend on the activation distribution within the current batch.

2.5. Accuracy comparison

Accuracy was compared for no dropout, standard dropout, and min-dropout. The same parameters were used five times for each experiment, and the average accuracy was reported. Generalization was measured using the test-train gap, which shows the difference between test and train accuracy [27]. A positive or near-zero gap implies similar training and testing performances, while a large negative gap may indicate overfitting. The gap is calculated as,

$$Gap = Test\ Accuracy - Training\ Accuracy \tag{7}$$

3. RESULTS AND DISCUSSION

3.1. Result

This section presents the empirical results of the three model configurations (no-dropout, standard random dropout, and min-dropout) on the iris and heart disease datasets. Each experiment was repeated five times, and the reported values are the averages across runs. The training and test accuracy were reported for dropout rates of r = {0.2, 0.3, 0.5}.

3.1.1. Iris dataset

Table 2 and Figure 1 report the average outcomes (over five independent runs) for the Iris experiments. For each configuration, the training accuracy, test accuracy, average training time per run (seconds), and the test-train gap were reported. According to the experimental results on the Iris dataset, several observations can be made:

– At a dropout rate of r = 0.2, the standard dropout model achieved the highest test accuracy (71.33%), outperforming both the baseline model (56.67%) and min-dropout model (63.34%). The test-train gap is negative but close to zero, indicating that the model performance is still similar on both the model and training datasets. At r = 0.3, the no-dropout baseline obtained the best performance (86.00% test accuracy). Standard dropout also produced competitive results (74.00%), and min-dropout 65.34%. At r = 0.5, the min-dropout model outperformed the other configurations, with a test accuracy of 75.34% compared to 72.67% for the baseline and 46.00% for standard dropout.

- Training times were similar across all methods (approximately 28-31 seconds), indicating that the inclusion of min-dropout does not add significant computational overhead.
- The test-train gap values were generally close to zero, indicating similar performance on the training and testing datasets. Positive gaps of 11.83% and 20.83% were observed for standard dropout at dropout rates of 0.2 and 0.3, suggesting that test accuracy exceeded training accuracy under these settings.
- At r = 0.2, standard dropout achieved the highest average of test accuracy (71.33%), but its high standard deviation (±23.99%) indicates considerable variation across repeated runs.

Table 2 Iris dataset results across dropout rates over five runs

Rate	Model	Train acc. (mean ± SD)	Test acc. (mean ± SD)	Training time (s)	Gap (Test-Train)
0.2	ND	60.00 ± 20.27 %	56.67 ± 15.99 %	30.886	-3.33 %
	SD	59.50 ± 17.25 %	71.33 ± 23.99 %	28.940	11.83 %
	MD	62.17 ± 19.98 %	63.34 ± 23.92 %	29.755	1.17 %
0.3	ND	86.50 ± 12.41 %	86.00 ± 13.21 %	29.577	-0.50 %
	SD	53.17 ± 17.63 %	74.00 ± 25.86 %	29.131	20.83 %
	MD	68.33 ± 18.24 %	65.34 ± 16.09 %	29.870	-3.00 %
0.5	ND	73.17 ± 14.58 %	72.67 ± 13.42 %	28.579	-0.50 %
	SD	38.17 ± 10.23 %	46.00 ± 20.33 %	28.298	7.83 %
	MD	75.33 ± 10.01 %	75.34 ± 11.92 %	28.747	0.00 %

Note: ND = No Dropout, SD = Standard Dropout, MD = Min-dropout.

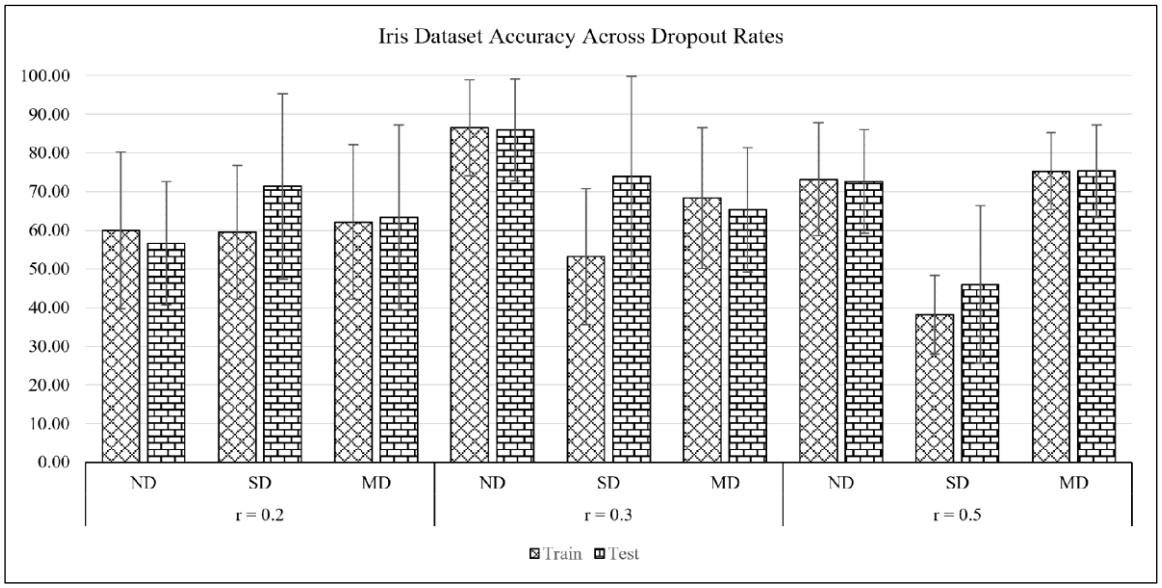

Figure 1. Iris dataset accuracy across dropout rates

3.1.2. Heart disease dataset

Table 3 and Figure 2 summarize the average results over five independent runs of the heart disease experiment. For each dropout configuration, the table provides training accuracy, test accuracy, average training time per run, and test-train accuracy gap. According to the experimental results on the heart disease dataset, several observations can be made:

- At r = 0.2, the no-dropout baseline achieved the highest test accuracy (55.33%), slightly outperforming the min-dropout (53.67%) and standard dropout (52.67%) configurations. At r = 0.3, both the baseline and min-dropout produced similar test accuracy (55.67% and 55.00%, respectively), while standard dropout remained lower (53.00%). At r = 0.5, the min-dropout model achieved the highest test accuracy (56.67%), marginally outperforming both the baseline (54.33%) and standard dropout (52.66%).
- Across all dropout rates, standard dropout consistently showed the lowest training and testing accuracies, suggesting that random deactivation of neurons may remove relevant activations in smaller tabular datasets.

- Training times were generally comparable across models, although the no-dropout configuration at r = 0.5 showed a noticeably longer training time. This variation may be related to runtime or training process variability rather than the dropout method itself.
- The test-train gap values remained small and negative across all configurations. The largest gap observed was 6.51%, which is still relatively close to zero. This suggests that no significant overfitting occurred in any of the tested models.
- The heart disease results showed lower standard deviations than the Iris results, indicating more consistent performance across runs. However, because the accuracy differences were small, the advantage of min-dropout at r = 0.5 should be interpreted cautiously

Table 3. Heart disease dataset results across dropout rates over five runs

Rate	Model	Train acc.	Test acc.	Training time (s)	Gap (Test-Train)
0.2	ND	58.82 ± 5.69 %	55.33 ± 2.17 %	29.173	-3.49 %
	SD	53.50 ± 0.63 %	52.67 ± 0.91 %	29.225	-0.83 %
	MD	59.83 ± 2.49 %	53.67 ± 1.39 %	29.793	-6.16 %
0.3	ND	60.09 ± 4.39 %	55.67 ± 3.25 %	36.183	-4.42 %
	SD	53.76 ± 0.48 %	53.00 ± 0.74 %	30.322	-0.75 %
	MD	60.59 ± 4.03 %	55.00 ± 2.04 %	31.223	-5.59 %
0.5	ND	60.84 ± 3.51 %	54.33 ± 2.79 %	84.265	-6.51 %
	SD	53.84 ± 0.23 %	52.66 ± 1.49 %	36.389	-1.17 %
	MD	61.01 ± 5.03 %	56.67 ± 2.64 %	34.063	-4.35 %

Note: ND = No Dropout, SD = Standard Dropout, MD = Min-dropout.

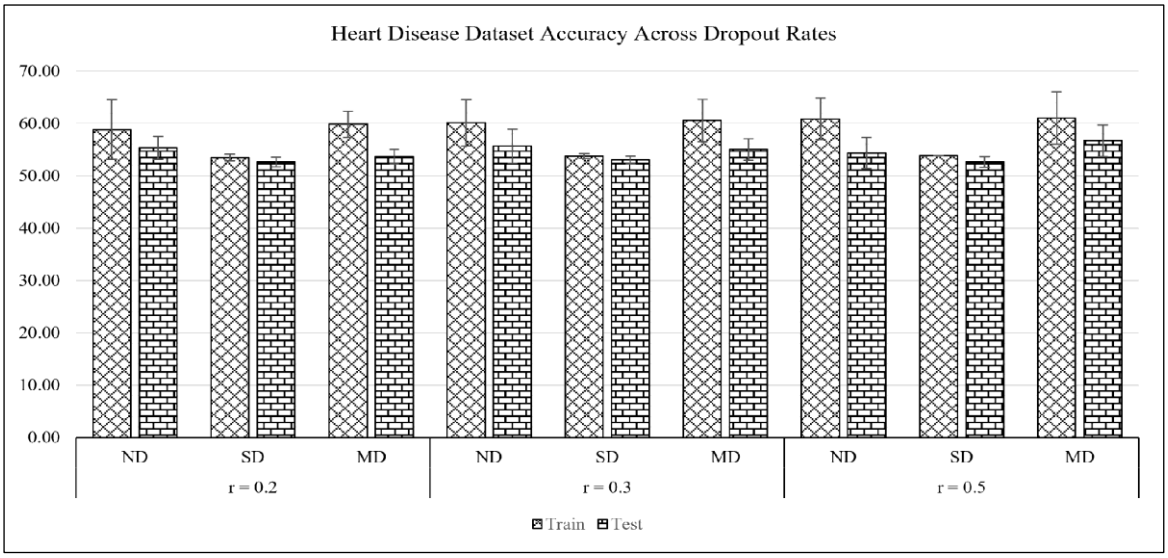

Figure 2. Heart disease dataset accuracy across dropout rates

3.2. Discussion

The experimental findings from the iris and heart disease datasets show that the effectiveness of min-dropout depends on the dropout rate and dataset characteristics. Overall, min-dropout did not consistently outperform the other methods across all configurations, but it showed competitive performance, especially at the higher dropout rate. The following discussion focuses on the effect of dropout rate, generalization behavior, training stability, and computational cost.

The first observation relates to training stability and generalization consistency. The results show that the test-train gap was small for most experiments indicating similar test and training performance. In contrast, standard dropouts occasionally produced larger variations, including cases where test accuracy exceeded training accuracy. This may occur in small datasets due to random effects from data splitting, model initialization, and random neuron masking. Therefore, min-dropout's deterministic behavior may offer more predictable learning dynamics. The second observation is that training time remained consistent across all methods and dropout rates. The additional computation required for ranking and masking low activation values did not cause noticeable overhead. This confirms that min-dropout can be integrated into standard

neural network models without significantly affecting training efficiency, making it practical for real-world applications. The third finding is the influence of dropout rate and dataset characteristics. On the Iris dataset, standard dropout achieved higher test accuracy at moderate dropout rates (r = 0.2-0.3), while min-dropout achieved the highest test accuracy at r=0.5. A similar pattern was observed on the heart disease dataset, where min-dropout also produced the best result at r=0.5, although the improvement over the baseline was modest. These findings suggest that min-dropout may be more beneficial at a higher rate. In standard dropout, at higher rates, random neuron deactivation may remove informative neuron while min-dropout removes lower activation values. It potentially reduces the loss of useful information during training. This variation may also be influenced by dataset characteristics. On the simple Iris dataset, moderate random dropout may improve generalization, but at r=0.5, it may remove more informative activations. For the more heterogeneous heart disease dataset, deterministic low-activation masking may provide more stable behavior, although the improvement over the baseline remained modest.

To provide an initial evaluation on an image dataset, a preliminary experiment was also conducted on CIFAR-10 using a convolutional neural network. The no-dropout model achieved the highest test accuracy of 81.61%, followed by standard dropout at 66.77% and min-dropout at 65.97%. Min-dropout produced the smallest test-train gap of -1.84%, but it did not improve test accuracy compared with the other configurations. The use of shallow architecture also made the experiment easier to repeat and the training behavior easier to observe, which is useful during early algorithm development [28]. However, because the experiments were limited to small datasets and a simple model structure, the findings should be interpreted as preliminary evidence of min-dropout's behavior rather than a general conclusion about its effectiveness across deep architectures.

4. CONCLUSION

This study proposed min-dropout, a deterministic dropout-inspired algorithm that masks the lowest activation values instead of applying random masking. The objective was to evaluate whether activation-based masking could provide an alternative strategy to standard dropout. Experiments on the iris and heart disease datasets showed that min-dropout did not consistently outperform standard dropout across all configurations. Standard dropout achieved higher mean test accuracy on the Iris dataset at moderate dropout rates of r = 0.2 and r = 0.3. However, min-dropout achieved the highest mean test accuracy at r = 0.5 on both datasets, with 75.34% on Iris and 56.67% on heart disease. These findings suggest that deterministic low-activation masking may be useful under stronger dropout conditions, although the improvement on the heart disease dataset was modest. This study has several limitations. The experiments were conducted on relatively small datasets using shallow neural network architectures. In addition, the proposed method was evaluated only against standard dropout, and the gradient behavior and learning dynamics were not analyzed. Another limitation is that the current implementation performs batch-level activation filtering, which differs from the independent random masking mechanism used in standard dropout. Therefore, the findings should be interpreted as preliminary evidence for a dropout-inspired activation filtering method rather than a general replacement for standard dropout.

Future research should evaluate min-dropout on larger datasets and deeper architectures, such as convolutional neural networks and transformers. More specifically, min-dropout could be integrated with L2 regularization to examine whether activation-based masking provides complementary regularization effects. It could also be applied to pre-trained transformer layers to evaluate its behavior in high-dimensional embedding spaces. Further studies should also compare min-dropout with modern dropout variants, such as R-Drop and AS-Dropout, examine gradient distribution, gradient norm, and learning dynamics, and investigate per-sample or adaptive activation filtering mechanisms. Overall, min-dropout provides a simple activation-based alternative that shows competitive performance at higher dropout rates, but broader validation is required before stronger conclusions can be drawn about its effectiveness as a general regularization method.

FUNDING INFORMATION

Funding is partially provided by President University and UNITAR International University. Contract number 022/SK/YPUP-PresUniv/A.Dir.LRPM/BSS/II/2024.

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
Tjong Wan Sen	✓	✓	✓	✓	✓	✓	✓	✓		✓		✓		✓
Tahir Mehmood	✓			✓			✓			✓			✓	✓
Pajri Aprilio		✓	✓	✓	✓	✓		✓	✓		✓			

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 stated no conflict of interest.

DATA AVAILABILITY

Experimental results, per-run accuracy, and implementation files are available at: https://docs.google.com/spreadsheets/d/19YNbsE0AEUaHPaLoIwO-gG2vj4kWzy1L1gYyZeIDI4E/ and https://drive.google.com/file/d/1GbBvdHg iJSvleJZ5 Vbcotx_qzl6GlsEE/view?usp=sharing

REFERENCES

[1] M. S. Sunarjo, H.-S. Gan, and D. R. I. M. Setiadi, "High-performance convolutional neural network model to identify COVID-19 in medical images," *Journal of Computing Theories and Applications*, vol. 1, no. 1, pp. 19–30, 2023, doi: 10.33633/jcta.v1i1.8936.

[2] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus, "Regularization of neural networks using DropConnect," in *30th International Conference on Machine Learning, ICML 2013*, 2013, no. PART 3, pp. 2095–2103.

[3] L. Rice, E. Wong, and J. Z. Kolter, "Overfitting in adversarially robust deep learning," in *37th International Conference on Machine Learning, ICML 2020*, 2020, vol. PartF168147-11, pp. 8049–8074.

[4] A. C. Müller and S. Guido, *Introduction to machine learning with python: a guide for data scientists*. Sebastopol: O'Reilly Media, Inc., 2016.

[5] M. M. Bejani and M. Ghatee, "A systematic review on overfitting control in shallow and deep neural networks," *Artificial Intelligence Review*, vol. 54, no. 8, pp. 6391–6438, 2021, doi: 10.1007/s10462-021-09975-1.

[6] Y. Cao, Z. Chen, M. Belkin, and Q. Gu, "Benign overfitting in two-layer ReLU convolutional neural networks," *Proceedings of Machine Learning Research*, vol. 202, pp. 17615–17659, 2023.

[7] X. Ying, "An overview of overfitting and its solutions," *Journal of Physics: Conference Series*, vol. 1168, no. 2, 2019, doi: 10.1088/1742-6596/1168/2/022022.

[8] M. Uzair and N. Jamil, "Effects of hidden layers on the efficiency of neural networks," *Proceedings - 2020 23rd IEEE International Multi-Topic Conference, INMIC 2020*, 2020, doi: 10.1109/INMIC50486.2020.9318195.

[9] K. Zhang, J. Chen, T. Zhang, and Z. Zhou, "A compact convolutional neural network augmented with multiscale feature extraction of acquired monitoring data for mechanical intelligent fault diagnosis," *Journal of Manufacturing Systems*, vol. 55, pp. 273–284, Apr. 2020, doi: 10.1016/j.jmsy.2020.04.016.

[10] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed. Sebastopol: O'Reilly Media, 2023.

[11] C. C. Aggarwal, "Neural networks and deep learning: a textbook," *Neural Networks and Deep Learning: A Textbook*, pp. 1–497, 2018, doi: 10.1007/978-3-319-94463-0.

[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[13] H.-C. Kim and M.-J. Kang, "A comparison of methods to reduce overfitting in neural networks," *International journal of advanced smart convergence*, vol. 9, no. 2, pp. 173–178, 2020, [Online]. Available: http://dx.doi.org/10.7236/IJASC.2020.9.2.173.

[14] H. Salehinejad and S. Valaee, "EDropout: energy-based dropout and pruning of deep neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 10, pp. 5279–5292, 2022, doi: 10.1109/TNNLS.2021.3069970.

[15] Z. Lu, C. Xu, B. Du, T. Ishida, L. Zhang, and M. Sugiyama, "LocalDrop: a hybrid regularization for deep neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3590–3601, 2022, doi: 10.1109/TPAMI.2021.3061463.

[16] L. Wang and Q. Hu, "R-Block: regularized block of dropout for convolutional networks," 2023, [Online]. Available: http://arxiv.org/abs/2307.15150.

[17] Y. Chen and Z. Yi, "Adaptive sparse dropout: Learning the certainty and uncertainty in deep neural networks," *Neurocomputing*, vol. 450, pp. 354–361, 2021, doi: 10.1016/j.neucom.2021.04.047.

[18] Y. Zeng, T. Dai, and S. T. Xia, "Corrdrop: correlation based dropout for convolutional neural networks," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2020-May, pp. 3742–3746, 2020, doi: 10.1109/ICASSP40776.2020.9053605.

[19] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: a data-driven neuron pruning approach towards efficient deep architectures," 2016, [Online]. Available: http://arxiv.org/abs/1607.03250.

[20] R. A. Fisher, "Iris," *UCI Machine Learning Repository*, 1936.

[21] A. Janosi and W. Steinbrunn, "Heart disease," *UCI Machine Learning Repository*, 1989.

[22] S. Sardy, P. Tseng, and A. G. Bruce, "Robust wavelet denoising," *IEEE Transactions on Signal Processing*, vol. 49, no. 6, pp. 1146–1152, 2001, doi: 10.1109/78.923297.
[23] T. Sen, B. Riyanto Trilaksono, A. Akhmad Arman, and R. Mandala, "Robust automatic speech recognition features using complex wavelet packet transform coefficients," *ITB Journal of Information and Communication Technology*, vol. 3, no. 2, pp. 123–134, 2009, doi: 10.5614/itbj.ict.2009.3.2.4.
[24] T. W. Sen, B. R. Trilaksono, and A. A. Arman, "Evaluation of wavelet transform coefficients for robust speech recognition feature vectors," in *Proceedings of the International Conference on Electrical Engineering and Informatics*, 2007, pp. 1018–1021, [Online]. Available: http://research.mercubuana.ac.id/proceeding/H-48.pdf.
[25] F. Chollet, "Keras," *GitHub*, 2015, [Online]. Available: https://github.com/fchollet/keras%0A%0A.
[26] A. Zollanvari, "Machine learning with Python: Theory and implementation," *Machine Learning with Python: Theory and Implementation*, pp. 1–452, 2023, doi: 10.1007/978-3-031-33342-2.
[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, vol. 29, no. 7553. Cambridge, 2016.
[28] T. W. Sen, "Voice activity detector for device with small processor and memory," *ICSECC 2019 - International Conference on Sustainable Engineering and Creative Computing: New Idea, New Innovation, Proceedings*, pp. 212–217, 2019, doi: 10.1109/ICSECC.2019.8907081.

BIOGRAPHIES OF AUTHORS



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.



Tahir Mehmood completed his Ph.D. in Information Engineering from the University of Brescia (Italy) in 2021 and did M.Sc in Information Technology from Universiti Teknologi PETRONAS (Malaysia). He was part of the Italian National Project, "Infrastructures and services for Sustainable and Resilient Mobility - MoSoRe@Unibs" total budget of 10,943,963.97€, where he worked on the topic, "Analysis and development of innovative artificial intelligence, learning and automated planning techniques for autonomous systems" during his post-doc fellowship. As an exchange student during his Ph.D., he was involved as visiting scholar in Natural Language Processing group at Bruno Kessler Foundation (R&D Institute), Italy. His Ph.D. work mainly focused on the biomedical named entity recognition task in the biomedical text data using deep learning approach. Currently, he is serving as a senior lecturer at UNITAR International University, Malaysia. His research interests include natural language processing, deep learning, generative AI, image processing, and machine learning. He can be contacted at email: tahir.mehmood@unitar.my.



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.