Journal of Applied Informatics and Computing (JAIC)
Vol.10, No.2, April 2026, pp. 1151~1158
e-ISSN: 2548-6861

Performance Comparison of SqueezeNet Implementing a Dendritic Neural Model for Brain Disease Image Classification

Riandika Fathur Rochim [1]*, I GN Lanang Wijayakusuma [2]*, I Putu Winada Gautama [3]*
* Matematika, Universitas Udayana
riandikafathurrochim@gmail.com [1], lanang_wijaya@unud.ac.id [2], winadagautama@unud.ac.id [3]

Article Info

Article history:

Received 2026-01-07
Revised 2026-02-01
Accepted 2026-02-27

Keyword:

Brain Disease,
CNN,
Deep Learning,
Dendritic Neural Model,
SqueezeNet.

ABSTRACT

In this study, a comparative analysis is conducted between a pre-trained SqueezeNet v1.1 model and a modified SqueezeNet architecture integrating a dendritic neural model (DNM) into the classifier layer for eight-class brain disease classification using MRI images. The proposed modification replaces the standard linear classifier with a dendritic-based processing mechanism to enhance nonlinear representation at the classification stage. Experiments are performed on an MRI-based brain medical image dataset, and model performance is evaluated using accuracy, precision, recall, F1-score, and confusion matrix analysis. The results show that the SqueezeNet model integrated with DNM achieves a significant accuracy improvement of approximately 5%, increasing from 90.33% to 95.61%, compared to the standard SqueezeNet model. This performance gain is accompanied by a moderate increase in model complexity, with the parameter count rising by approximately 1.69% (from 726.6K to 738.9K) and a longer convergence time (40 epochs versus 22 epochs). Overall, the findings indicate that incorporating DNM into a lightweight CNN architecture such as SqueezeNet can effectively improve medical image classification performance while maintaining reasonable computational efficiency.

I. INTRODUCTION

Medical image classification plays an important role in supporting fast and accurate disease diagnosis, thereby assisting clinical treatment planning [1]. For example, brain tumor diagnosis can be performed through MRI image analysis to determine the appropriate medical intervention [2]. Other diseases, such as Alzheimer's, also require image-based diagnostic methods with a high level of precision [3]. Recently, convolutional neural networks (CNNs) have been widely used for medical image classification, including MRI images. This is demonstrated by a study by Ahmed *et al*., which shows that CNNs are capable of classifying brain tumor MRI images with an accuracy of up to 98%, higher than large language models (LLMs) such as GPT-4o (60%) and Llama3.2-Vision (52%) [4].

Several popular CNN architectures include AlexNet, ResNet-152, and Inception-ResNet-v2 These three models have been evaluated on the ILSVRC-2012 benchmark dataset, which contains more than one million images with 1,000 object classes. The evaluation results show top-5 accuracies of 80.2%, 94.04%, and 95.1%, respectively [1], [5]. Despite their strong performance, these models are considered large-scale CNNs due to their very high number of parameters. Consequently, they are less suitable for deployment on resource-constrained devices, such as mobile devices or edge devices in medical diagnostic systems [5, 6].

To address these limitations, lightweight CNN architectures have been developed that maintain accuracy while using fewer parameters [7]. One such example is SqueezeNet, which was developed as a modification of AlexNet. SqueezeNet is designed to achieve accuracy comparable to AlexNet but with a much smaller model size through the introduction of the fire module mechanism. On the ILSVRC-2012 dataset, SqueezeNet attains a top-5 accuracy of 80.4%, which is nearly equivalent to AlexNet, while reducing the model size from 240 MB to 4.8 MB, making SqueezeNet more suitable for deployment on devices with limited computational resources [7, 8].

The Dendritic Neural Model (DNM) is a neural network model inspired by the functioning of biological neurons, particularly the dendritic component. Unlike the linear McCulloch-Pitts (MCP) neuron, DNM incorporates dendritic processing mechanisms that enable the modeling of more complex nonlinear relationships [9, 10]. Based on this mechanism, several studies have implemented DNM within CNN architectures for classification tasks. Du *et al*. integrated DNM into ShuffleNetV2 and achieved accuracy improvements on several medical image datasets, albeit with an additional approximately 1.39 million parameters [11]. Furthermore, Wang *et al*. reported that integrating DNM into ResNeXt resulted in an accuracy of around 99.6% on a glaucoma MRI dataset [12]. These findings motivate further research on the use of DNM in other CNN architectures, such as SqueezeNet. Building upon these prior studies, the present work will implement DNM within the SqueezeNet architecture and then compare its performance with standard SqueezeNet for medical image classification of brain diseases.

II. METHOD

The SqueezeNet architecture used in this study is version 1.1, which consists of several components, including 3×3 convolution layers, multiple fire modules, max pooling, global average pooling, and 1×1 convolutions. The processing pipeline from image input to predicted class labels is divided into two stages, namely the feature extraction stage followed by the classifier layer stage.

The feature extraction part begins with a convolution layer called conv1, which consists of 64 filters of size 3×3 with a stride of 2. This layer functions to capture low-level features such as image textures and edges. Next, there are eight fire modules named fire2, fire3, ..., fire8, and fire9. Each fire module consists of two main components, namely a squeeze layer and an expand layer. The squeeze layer is implemented using 1×1 convolutions to reduce channel dimensionality, while the expand layer combines 1×1 and 3×3 convolutions to enrich the learned feature representations. Max pooling is applied after the conv1, fire4, and fire8 layers to reduce the spatial resolution of the feature maps without discarding important information. In the classifier layer, a 1×1 convolution is first applied with the number of filters adjusted to the number of target classes in the dataset, followed by a global average pooling layer and a softmax layer to produce a probability distribution over each class.

The DNM structure is consistently defined as consisting of four layers, namely the synaptic layer, dendritic layer, membrane layer, and soma layer. For each class label in this multiclass classification task, the synaptic layer has an activation function in which each input i-th ($i = 1,2, \dots, N$) is computed with respect to each dendrite j-th ($j = 1,2, \dots, M$), which is defined as

$$Y_{ij} = ReLU(k(w_{ij}x_i - q_{ij})) \quad (1)$$

The activation function used is the ReLU function because it has a time complexity of $O(1)$, where the value Y_{ij} represents all synaptic input information, with a total of N inputs sent to each dendrite j-th, and w_{ij} and q_{ij} are the weight and threshold parameters, respectively, both initialized as random numbers in the interval $[0,1)$. The scalar values k are global learnable parameters (shared across all class labels) with an initial value of 0.5. The multiplication of w_{ij} with x is performed in a column-wise product.

The output values produced by the synaptic layer are then propagated to the dendritic layer by summing them over each dendritic branch j-th, which is denoted as

$$Z_j = \sum_{i=1}^{N} Y_{ij} \quad (2)$$

where N denotes the number of synapses connected to the j-th dendritic branch.

The membrane layer receives the outputs from the dendritic layer by summing all dendritic responses into

$$V = \sum_{j=1}^{M} Z_j \quad (3)$$

Then, the soma layer receives the output from the membrane layer to produce the final output of the model, which is denoted as

$$O = ReLU(k_s(V - q_s)) \quad (4)$$

where k_s and q_s are learnable soma parameters initialized within the interval $[0,1)$.

In the standard SqueezeNet design, the final classifier stage is essentially a linear transformation (1×1 convolution + softmax) applied to the feature vector, which limits its ability to capture complex patterns. By contrast, replacing this stage with a DNM adds explicit nonlinear processing. Each input feature first activates a synaptic nonlinearity (Eq. 1) and is then aggregated by dendrites (Eqs. 2–3) before a final nonlinear soma activation (Eq. 4). This multi-stage, multi-branch structure effectively implements high-order combinations of the input features. Consequently, the DNM-equipped classifier can represent more complex, nonlinear decision boundaries and feature interactions than the simple linear classifier can. This enriched representational capacity helps the model better discriminate among the eight brain disease classes. The implementation of DNM in SqueezeNet is carried out by replacing the original classifier layer of SqueezeNet version 1.1 with a sequence consisting of a global average pooling layer, a flatten layer, followed by a DNM layer, and finally a softmax layer.

The rationale for this implementation is based on the assumption that the final output of the SqueezeNet feature extraction stage takes the form of a set of matrices M which is defined as

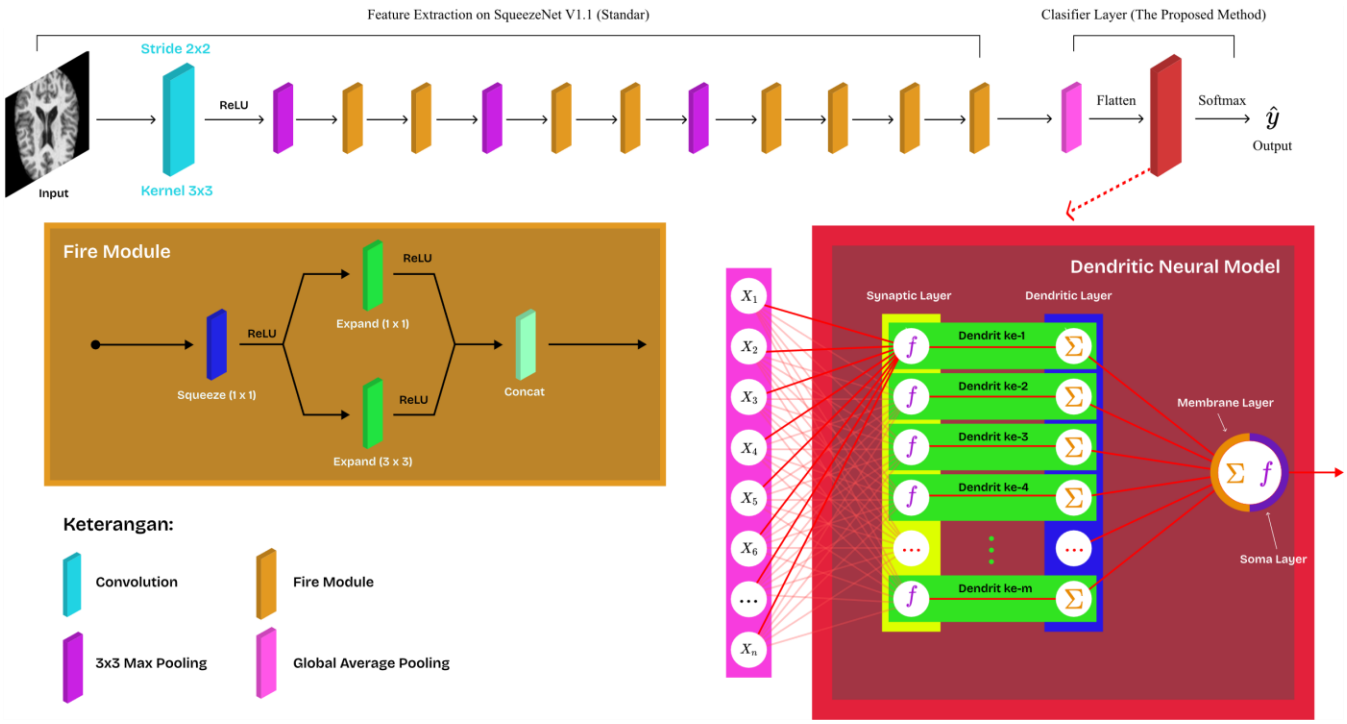

Figure 1. Architecture of the SqueezeNet model integrated with the DNM

$\mathcal{F} := \left\{ M_{\mathcal{H}\times\mathcal{W}}^{[\mathcal{C}]} \mid \mathcal{C} = 1,2,\dots,C; \mathcal{H} = 1,2,\dots,H;\ \mathcal{W} = 1,2,\dots,W \right\}$ where $\mathcal{H}$ and $\mathcal{W}$ denote the height and width of the feature maps, respectively. The global average pooling (GAP) process is then denoted as

$$x_{\mathcal{C}} = \frac{1}{HW} \sum_{\mathcal{H}=1}^{H} \sum_{\mathcal{W}=1}^{W} M_{\mathcal{H}\times\mathcal{W}}^{[\mathcal{C}]} \qquad \mathcal{C} = 1,2,\dots,C$$

Through this process, a vector $\mathbf{x} = (x_1, x_2, \dots, x_C) \in \mathbb{R}^C$ is obtained, which aims to reduce dimensional complexity and summarize the set of entries in each matrix $\mathcal{C}$-th into a single representative value for that matrix. The resulting vector is then fed into the DNM. Since all preceding convolutional layers use the ReLU activation function, the outputs before global average pooling are non-negative. Consequently, the DNM is designed such that dendritic branches in the excitatory state respond strongly to positive input signals, consistent with the characteristics of biological dendrites, which receive the majority of excitatory synapses on spines and transmit excitatory postsynaptic potentials [13, 14, 15].

III. METHOD

In this study, several methods are employed to achieve the final research objectives, as detailed below.

A. Dataset

The data were collected by downloading brain MRI images from Kaggle, specifically from the repository entitled "MCND dataset - Multi-Class Brain MRI Scans" compiled by Ali Fatahi and Zamani Hoda. The dataset comprises 16,400 images in .jpg format, which are categorized into eight classes as summarized in Table 1.

TABLE 1
MULTI CLASS NEUROLOGICAL DISORDER (MCND) DATASET

Type Disease	Class Name	Total Sample
Alzheimer	AD-MildDemented	896
	AD-ModerateDemented	64
	AD-VeryMildDemented	2240
Brain Tumor	BT-glioma	1620
	BT-meningioma	1531
	BT-pituitary	1740
Multiple Sclerosis	MS	1405
Normal	Normal	6904
Total	16400	

The dataset used in this study is available at: https://www.kaggle.com/datasets/alifatahi/multi-class-neurological-disorder-mcnd-dataset. As summarized in Table 1, the dataset contains 16,400 MRI images distributed across eight diagnostic categories. The class distribution is imbalanced: for example, the "Normal" category is the largest

(6,904 images), whereas "AD-ModerateDemented" is the smallest (64 images). The remaining classes include Alzheimer's disease at mild and very mild stages, multiple sclerosis, and three types of brain tumor (glioma, meningioma, and pituitary). This diverse set of conditions ensures that the model must distinguish among multiple neuropathologies, and the pronounced imbalance motivates targeted strategies (e.g., weighted training and augmentation) to ensure robust learning across all classes.

B. Data Preprocessing

To enable the model to effectively learn patterns from the dataset, several data preprocessing steps are applied, beginning with data splitting, in which the dataset is divided into multiple subsets for model training. In this study, the data are divided into three subsets, namely the training set, validation set, and test set, with a ratio of 70:15:15 [16]. The splitting procedure employs a stratified splitting method to preserve the original class proportions in all three subsets, which is crucial given the class imbalance in the dataset. Subsequently, class weights are computed from the training set before performing data augmentation to address class imbalance during model training. Data augmentation is applied to the training set to increase sample diversity through small rotations (±15°), scaling (±10%), horizontal and/or vertical shifts of up to ±10%, and adjustments to brightness and contrast [17]. Finally, each image in the dataset is converted into a tensor and normalized using the mean and standard deviation values of ImageNet-1K, in accordance with the pre-trained weights employed.

To mitigate the class imbalance during training, class weights proportional to the inverse class frequencies were computed on the training subset. Data augmentation was then applied only to the training set, including random small rotations (±15°), scaling (±10%), translations (±10%), and adjustments to image brightness and contrast. These augmentations increase the diversity of training samples without altering the validation or test sets. The 70:15:15 stratified split ensures each subset maintains the original class proportions, allowing a held-out validation set for hyperparameter tuning and early stopping and a separate test set for unbiased evaluation.

C. Class Imbalanced Handling

Class imbalance refers to a condition in which the distribution of samples across classes is significantly uneven. In some cases, one or more classes contain far fewer samples than the others, causing the model to become biased toward the majority class and to exhibit poor predictive performance on the minority class. Therefore, special strategies are required at the data preprocessing stage to ensure that the model can learn from the data in a more balanced manner. Suppose there is a dataset with K classes, where the number of samples in the k-th class is denoted by n_k, and the total number of samples is denoted by N. The class weight assigned to the k-th class using the inverse class frequency method is denoted by

$$\boldsymbol{w_k = \frac{N}{K \cdot n_k}} \tag{5}$$

Then, since one of the classes receives an excessively large weight, the interquartile range (IQR) method is employed by defining the lower bound of the value range as $LB = Q_1 - 1.5 \times IQR$ and the upper bound as $UB = Q_3 + 1.5 \times IQR$. After removing these outlier values, the remaining class weights are subsequently normalized using

$$\widehat{w}_k = \frac{w_k}{\sum_{i=1}^{K} w_k} \tag{6}$$

where this value becomes the final value that will be used as the class weight. The value of $\widehat{w}_k$ is used in the loss function during model training. The loss function employed is the cross-entropy loss, which is denoted as

$$\mathcal{L}_{CE} = -\sum_{i=1}^{K} y_i \log(\mathrm{p}_i) \tag{7}$$

where **y** denotes the ground-truth (one-hot) label of an image, p denotes the predicted probability (softmax output), and K denotes the total number of classes. Equations (6) and (7) constitute the final loss function, which is formulated as

$$\mathcal{L}_{final} = \widehat{w}_k \mathcal{L}_{CE} \tag{8}$$

D. Pre-Trained Model

In this study, the pre-trained weights of SqueezeNet v1.1 are utilized in the feature extraction module, since pre-trained weights for the DNM component are not yet available in existing repositories or prior work. The model parameters are updated using the AdamW optimizer with predetermined hyperparameter configurations, with the objective of minimizing the validation loss through an early stopping mechanism to prevent overfitting. Furthermore, for the baseline SqueezeNet v1.1 model used as a comparator, the same pre-trained weights are applied to the entire network, encompassing both the feature extraction stages and the final classifier layers.

E. Hyperparameter Configuration

The hyperparameter configuration used during model training consists of a maximum of 80 epochs with a batch size of 64, and a fixed random seed of 42 to ensure the reproducibility of the experimental results. The training process is monitored using an early stopping mechanism with a patience of 7 epochs to reduce the risk of overfitting, such

that optimization is terminated when no further improvement in validation performance is observed. The optimization is carried out using the AdamW algorithm with a learning rate of 1×10^{-4}, where the first-moment and second-moment coefficients are set to 0.9 and 0.999, respectively, following common practice in deep learning optimization. In addition, a weight decay of 1×10^{-5} is employed as a form of regularization to enhance the model's generalization capability, while a small constant $\epsilon = 10^{-8}$ is used to maintain numerical stability during the parameter update steps. The loss function adopted is the weighted cross-entropy loss as defined in Equation (8), which is widely used for multi-class classification tasks under class-imbalanced settings.

F. Experimental Setting and Evaluation Metrics

The experiments are conducted using a T4 GPU on a cloud computing platform. The implementation is carried out using Python 3 and the PyTorch framework, which provides flexible support for deep learning model development and training in this study. To empirically assess and compare the performance of the models, a confusion matrix is employed to analyze the correspondence between the model predictions and the ground-truth labels on the testing set. In addition, the evaluation metrics used in this study include accuracy, precision, recall, and F1-score, whose mathematical formulations are defined as

$$\boldsymbol{Acc} = \frac{\boldsymbol{TP} + \boldsymbol{TN}}{\boldsymbol{TP} + \boldsymbol{TN} + \boldsymbol{FN} + \boldsymbol{FP}} \tag{9}$$

$$\boldsymbol{Rec} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FN}} \tag{10}$$

$$\boldsymbol{Prec} = \frac{\boldsymbol{TP}}{\boldsymbol{TP} + \boldsymbol{FP}} \tag{11}$$

$$\boldsymbol{F1} = \frac{\boldsymbol{2(Prec \times Rec)}}{\boldsymbol{Prec + Rec}} \tag{12}$$

In addition, the confusion matrix is used to analyze prediction errors in greater detail by examining the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) for each class, which provides a more granular view of model behavior. This analysis is essential for understanding the model's error patterns and for identifying classes that remain challenging to classify correctly.

The evaluation in this study is performed on a testing set that is completely excluded from the training process to ensure that the reported results reliably reflect the model's generalization capability to previously unseen data.

G. Sensitivity Analysis of Hyperparameter Dendrits M

In this study, various numbers of dendrites are explored in the classification model to investigate how the dendritic configuration influences overall performance. A detailed analysis is conducted on the hyperparameter M, which represents the number of dendrites in the DNM, with the objective of identifying a setting that yields optimal model performance.

The experimental results indicate that the model's performance varies as a function of M, exhibiting a pattern in which performance initially increases and then degrades beyond a certain point. A plausible explanation is that, as the number of dendrites increases, the model is able to capture more informative patterns from the data, but at the same time the amount of noise originating from the medical images also increases, which can adversely affect the classification performance on the evaluated dataset.

IV. RESULT AND DISCUSSION

When revisiting the analysis of how sensitive the model performance is to the number of dendrites used in the architecture, the final outcomes can be examined in Table 2, where M denotes the dendrite-related hyperparameter selected based on the evaluation results obtained using Eqs. (9), (10), (11), and (12).

TABLE 2
PERFORMANCE OF THE MODEL UNDER VARYING NUMBERS OF DENDRITES (WEIGHTED EVALUATION)

M	Acc (%)	Pre (%)	Rec(%)	F1(%)
1	92,11	92,25	92,11	92,02
2	**95,61**	**95,63**	**95,61**	**95,58**
4	90,89	91,75	90,89	90,97
6	85,36	87,64	85,37	85,55
8	90,49	92,06	90,49	90,65
10	94,92	94,94	94,92	94,88
12	91,42	91,82	91,42	91,49
14	87,80	89,30	87,80	88,09
16	91,58	92,10	91,59	91,68
18	91,83	91,98	91,83	91,77

The results indicate that the model in this study achieves its best performance when using two dendrites. The detailed comparison of model performance on the testing set between the SqueezeNet model with and without the DNM module is presented in Table 3, where the standard SqueezeNet is denoted as "model A" and the best-performing SqueezeNet–DNM variant ($M = 2$) is denoted as "model B.".

TABLE 3
COMPARATIVE MODEL PERFORMANCE (WEIGHTED EVALUATION)

Model	Acc(%)	Pre(%)	Rec(%)	F1(%)	Param(K)
A	90,33	91,82	91,54	91,36	**726,6**
B (M=2)	**95.61**	**95,62**	**95,61**	**95,58**	738,88

Table 3 compares the two models in terms of accuracy, precision, recall, F1-score, and parameter count. The SqueezeNet–DNM model (Model B) consistently outperforms the baseline SqueezeNet (Model A) across all evaluation metrics. Specifically, Model B achieves 95.61% accuracy, 95.62% precision, 95.61% recall, and 95.58% F1-

score, whereas Model A attains 90.33%, 91.82%, 91.54%, and 91.36%, respectively. These consistent improvements indicate that the performance gains provided by the DNM extend beyond overall accuracy and enhance class-level prediction balance and robustness.

Although the SqueezeNet–DNM model introduces additional parameters, resulting in an increase of 12.28K parameters (approximately 1.69%), this growth remains relatively modest compared to the achieved accuracy improvement of approximately 5%. In many real-world medical applications, particularly in clinical decision support systems, improved diagnostic accuracy and reliability often outweigh marginal increases in computational cost. Moreover, despite the increase in parameter count and convergence time, the overall complexity of SqueezeNet–DNM remains low, making it suitable for deployment in resource-constrained medical imaging environments while preserving the lightweight nature of the original SqueezeNet architecture.

Fig. 2 illustrates the training loss curves of the standard SqueezeNet and the SqueezeNet-DNM models. Both models exhibit a rapid decrease in loss during the initial training epochs, indicating effective feature learning at early stages; however, clear differences are observed in their convergence behavior. The standard SqueezeNet converges earlier, reaching a stable loss plateau at approximately epoch 22, whereas the SqueezeNet-DNM model requires a longer training process and converges around epoch 40. Although the SqueezeNet-DNM demonstrates slower convergence, its loss curve remains smooth and stable without significant oscillations, indicating a well-behaved optimization process. The extended convergence time can be attributed to the increased model complexity and the additional nonlinear transformations introduced by the dendritic processing mechanism. Nevertheless, the continued reduction in loss beyond epoch 22 suggests that SqueezeNet-DNM enables further refinement of decision boundaries during later training stages, ultimately leading to improved classification performance. These results confirm that the proposed modification trades faster convergence for enhanced representational capacity and higher accuracy.

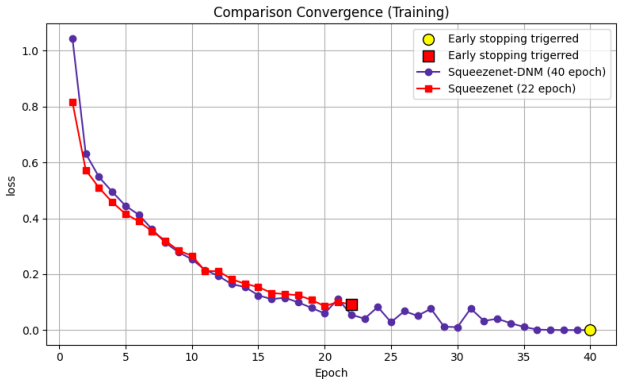

Figure 2. Training loss convergence curves of the standard SqueezeNet and the SqueezeNet-DNM models.

Figure 3 illustrates the comparison of validation accuracy convergence between the standard SqueezeNet and the SqueezeNet–DNM during training, where the horizontal axis represents the number of epochs and the vertical axis denotes the validation accuracy. Based on these results, the standard SqueezeNet model reaches early stopping at the 22-nd epoch with a maximum validation accuracy of approximately 92%, indicating that further training would likely lead to performance stagnation and a higher risk of overfitting. The convergence curve of this model exhibits relatively larger fluctuations in the early training phase, reflecting its limited ability to stably extract more complex feature representations.

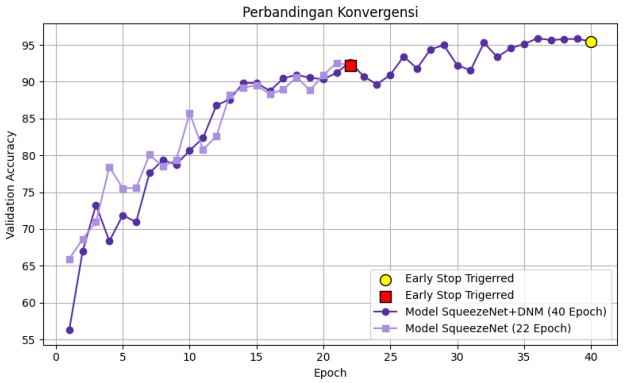

Figure 3. Validation accuracy convergence curves of the two different models.

In contrast, the SqueezeNet model augmented with the DNM exhibits a more stable and sustained convergence trend, with early stopping occurring at the 40th epoch, indicating a longer yet more controlled optimization process.

Figure 4 further supports these findings by depicting the dynamics of the validation loss. The standard SqueezeNet model reaches early stopping at the 22nd epoch with a minimum validation loss of approximately 0.19 but shows relatively sharp fluctuations around the convergence region, suggesting a less stable optimization trajectory. Conversely, the DNM-enhanced SqueezeNet demonstrates a more gradual and stable decrease in loss up to the 40th epoch, achieving a comparable minimum value with smaller inter-epoch variance. From a mathematical perspective, this behavior implies that the loss function of the DNM-based model is associated with a smoother optimization landscape, allowing more consistent parameter updates and leading to solutions with improved generalization capability.

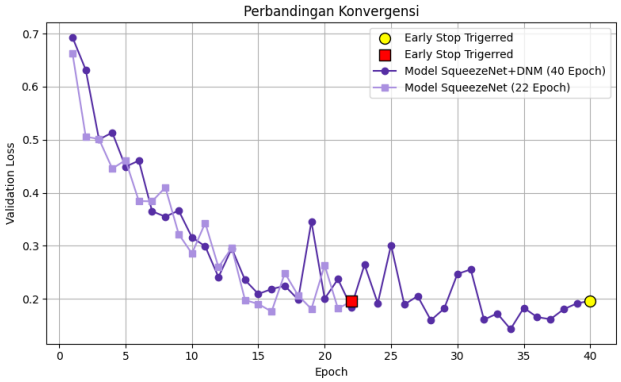

Figure 4. Validation loss convergence curves of the two different models.

Subsequently, an additional evaluation is presented in the form of the confusion matrix for the SqueezeNet model integrated with the DNM, computed on the previously prepared testing set.

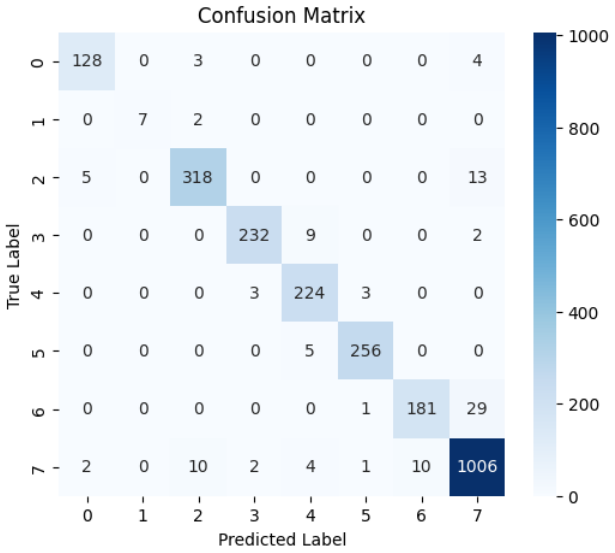

Figure 4. Confusion matrix of the SqueezeNet model integrated with the DNM

Based on Figure 4, which presents the confusion matrix of the DNM-augmented SqueezeNet model, it can be observed that the model is able to correctly classify most classes with a high level of accuracy, as indicated by the dominance of values along the main diagonal. The Normal class (index 7) and the AD_VeryMildDemented class (index 2) exhibit the highest number of correct predictions, suggesting that the model can consistently capture both general patterns and early-stage impairment.

In contrast, misclassifications still occur for classes with overlapping visual characteristics, particularly between AD_MildDemented (index 0) and AD_VeryMildDemented, as well as among several brain tumor subclasses, namely glioma (index 3), meningioma (index 4), and pituitary (index 5). This indicates that, despite the overall strong performance, the model still encounters difficulties in distinguishing between subtle structural or textural differences in certain pathological categories.

TABLE 4
CLASSIFICATION PERFORMANCE FOR EACH CLASS

No.Idx	Class	Precission	Recall	F1
0	AD-MildDemented	0.94	0.94	0.94
1	AD-ModerateDemented	1.00	0.77	0.87
2	AD-VeryMildDemented	0.95	0.94	0.95
3	BT-glioma	0.98	0.95	0.97
4	BT-meningioma	0.92	0.97	0.94
5	BT-pituitary	0.98	0.98	0.98
6	MS	0.94	0.85	0.90
7	Normal	0.95	0.97	0.96

From the confusion matrix, the precision, recall, and F1-score values of the model are obtained for each class. These evaluation metrics are summarized and reported in Table 4.

For future research, the dataset compiled and used in this study has been made publicly available to support further investigation and reproducibility. The dataset can be accessed at https://huggingface.co/datasets/riandika/MCND-dataset and may serve as a benchmark for subsequent studies on multi-class brain disease classification using MRI images. Future work may leverage this dataset to explore alternative lightweight architectures, advanced biologically inspired models, or improved training strategies, as well as to conduct more extensive comparative analyses under unified experimental settings.

V. CONCLUSION

Based on the comparative results obtained in this study, integrating the DNM into the pre-trained SqueezeNet v1.1 architecture yields noticeably improved performance compared to the original pre-trained SqueezeNet v1.1, even after exploring multiple configurations of the required number of dendrites. Compared to the baseline model, the DNM-augmented pre-trained SqueezeNet achieves an approximate performance gain of 5%, at the cost of a modest increase in model size corresponding to an additional 1.69% parameters. Furthermore, the convergence plots for both validation accuracy and validation loss indicate that the incorporation of the DNM contributes to a more stable optimization process, characterized by smoother trajectories and lower inter-epoch variance in the validation metrics, thereby enabling more controlled parameter updates and supporting better generalization. Nevertheless, it is also observed that the standard SqueezeNet model reaches convergence earlier during training (22 epochs versus 40 epochs), highlighting a trade-off between convergence speed and stability. In addition, the confusion matrix results for the DNM-enhanced SqueezeNet demonstrate that the model is capable of predicting both majority and minority classes reasonably well, despite the presence of class imbalance in the dataset. Taken together, these findings suggest that the DNM represents an effective alternative for improving the

performance of SqueezeNet on image classification tasks, offering measurable gains with only a relatively small and still acceptable increase in model complexity.

REFERENCES

[1] S. S. Yadav and S. M. Jadhav, "Deep convolutional neural network based medical image classification for disease diagnosis," *J Big Data*, vol. 6, no. 1, Dec. 2019, doi: 10.1186/s40537-019-0276-2.

[2] M. Aamir, Z. Rahman, U. A. Bhatti, W. A. Abro, J. A. Bhutto, and Z. He, "An automated deep learning framework for brain tumor classification using MRI imagery," *Sci Rep*, vol. 15, no. 1, Dec. 2025, doi: 10.1038/s41598-025-02209-2.

[3] A. Suganya and S. L. Aarthy, "Application of Deep Learning in the Diagnosis of Alzheimer's and Parkinson's Disease: A Review," *Curr Med Imaging*, vol. 20, Jan. 2024, doi: 10.2174/1573405620666230328113721.

[4] S. Ahmed, S. K. Sakib, and A. B. Das, "Can Large Language Models Challenge CNNs in Medical Image Analysis?," Jun. 2025, [Online]. Available: http://arxiv.org/abs/2505.23503

[5] M. Li, Y. Jiang, Y. Zhang, and H. Zhu, "Medical image analysis using deep learning algorithms," *Front Public Health*, vol. 11, 2023, doi: 10.3389/fpubh.2023.1273253.

[6] A. O. A. Deheyab *et al.*, "AN OVERVIEW of CHALLENGES in MEDICAL IMAGE PROCESSING," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Dec. 2022, pp. 511–516. doi: 10.1145/3584202.3584278.

[7] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," Nov. 2016, [Online]. Available: http://arxiv.org/abs/1602.07360

[8] M. Tsivgoulis, T. Papastergiou, and V. Megalooikonomou, "An improved SqueezeNet model for the diagnosis of lung cancer in CT scans," *Machine Learning with Applications*, vol. 10, p. 100399, Dec. 2022, doi: 10.1016/J.MLWA.2022.100399.

[9] X. Wen, M. Zhou, A. Albeshri, L. Huang, X. Luo, and D. Ning, "Improving Classification Performance in Dendritic Neuron Models through Practical Initialization Strategies," *Sensors*, vol. 24, no. 6, Mar. 2024, doi: 10.3390/s24061729.

[10] Y. ; Yang *et al.*, "Yet Another Effective Dendritic Neuron Model Based on the Activity of Excitation and Inhibition," *Mathematics 2023, Vol. 11, Page 1701*, vol. 11, no. 7, p. 1701, Apr. 2023, doi: 10.3390/MATH11071701.

[11] Q. Du, Z. Liu, Y. Song, N. Wang, Z. Ju, and S. Gao, "A Lightweight Dendritic ShuffleNet for Medical Image Classification," *IEICE Trans Inf Syst*, vol. E108D, no. 7, pp. 744–751, Jul. 2025, doi: 10.1587/transinf.2024EDP7059.

[12] N. Wang, Q. Du, Z. Yuan, Y. Gao, R.-L. Wang, and S. Gao, "Dendritic Aggregated Residual Deep Learning for Meningioma MRI Diagnosis," *IEICE Trans Inf Syst*, vol. E108.D, no. 8, pp. 1016–1019, Aug. 2025, doi: 10.1587/transinf.2024edl8049.

[13] Y. Ding, J. Yu, C. Gu, S. Gao, and C. Zhang, "A Multi-In and Multi-Out Dendritic Neuron Model and its Optimization," Sep. 2023, [Online]. Available: http://arxiv.org/abs/2309.07791

[14] L. Fischer *et al.*, "Dendritic Mechanisms for In Vivo Neural Computations and Behavior," in *Journal of Neuroscience*, Society for Neuroscience, Nov. 2022, pp. 8460–8467. doi: 10.1523/JNEUROSCI.1132-22.2022.

[15] A. O. Komendantov and G. A. Ascoli, "Dendritic excitability and neuronal morphology as determinants of synaptic efficacy," *J Neurophysiol*, vol. 101, no. 4, pp. 1847–1866, Apr. 2009, doi: 10.1152/jn.01235.2007.

[16] N. A. Ranggianto, A. P. Segara, D. Wijonarko, A. Andrianto, and M. H. Arief, "Procedural Content Generation pada Level Gim Sokoban Menggunakan Model Hybrid GPT2 dan Algoritma Genetika," *remik*, vol. 9, no. 3, pp. 963–974, Aug. 2025, doi: 10.33395/remik.v9i3.15188.

[17] A. M. Taha, S. A. Aly, and M. F. Darwish, "Detecting Glioma, Meningioma, and Pituitary Tumors, and Normal Brain Tissues based on Yolov11 and Yolov8 Deep Learning Models," Mar. 2025, Accessed: Oct. 10, 2025. [Online]. Available: https://arxiv.org/pdf/2504.00189v1