A Discussion of Migration of Common Neural Network Regularization Methods on SNNs

Yilin Lyu
The University of Melbourne
Grattan Street, Parkville Victoria
Email: yillyu1@student.unimelb.edu.au

Bo Yin
South China University of Technology
Guangzhou, Guangdong
Corresponding: yb0927@outlook.com

***Abstract*—SNN models offer notable advantages in terms of memory efficiency and energy consumption when compared to traditional artificial neural network models. However, their unique impulse propagation characteristics often result in lower prediction accuracy and limited generalization abilities. Consequently, SNNs are not widely adopted in many application domains.In light of this current state of affairs, we have undertaken an investigation into the application of various dropout techniques, including standard dropout, dropout2d, Feature dropout, and alpha dropout, which are commonly employed in traditional artificial neural networks. Our objective is to assess the impact of these dropout techniques on the training loss and test accuracy of SNN models.To conduct our experiments, we have focused on fully connected SNN models using two benchmark datasets: MNIST and CIFAR-10. Our findings indicate that SNNs subjected to multiple dropout techniques exhibit subpar performance in both training loss and test accuracy evaluations. This suggests that the straightforward application of dropout and its variants does not yield the desired improvement in the performance of SNN models.**

Keywords- Spiking Neural Networks, Standard-dropout, Dropout2d, Feature-dropout, Alpha-dropout

I. INTRODUCTION

In the field of neural network modeling, modern artificial neural networks typically process continuous input values and produce continuous outputs through fully connected structures. While these networks have achieved significant breakthroughs across various domains, they fall short of replicating the intricate operations observed in biological neurons within the human brain.

Inspired by the brain's information processing principles, Spiking Neural Networks (SNNs) have gained attention for their potential in efficient event-driven computations. Unlike traditional artificial neural networks, SNNs use discrete temporal pulses for communication and computation. This unique approach allows SNN models to be deployed in specific scenarios, such as embedded devices. SNNs offer distinct advantages, including reduced memory usage and computational overhead, critical for minimizing memory footprint, operational latency, and energy consumption compared to traditional ANN models. Furthermore, SNNs can enhance overall computational efficiency by streamlining processing steps.[1]

However, due to their impulse-driven nature and the scarcity of labeled impulse data, impulse neural networks face formidable challenges related to overfitting during training. Consequently, training SNNs to achieve high prediction accuracy remains a challenging endeavor.

To address these challenges, improve SNN performance, and emphasize their advantages, we explored the application of dropout — a common regularization technique in artificial neural networks — and its variants to SNNs. The goal was to mitigate overfitting issues and enhance their generalization capabilities. We conducted an extensive series of experiments using two publicly available datasets. Unfortunately, our findings revealed that incorporating these four dropout techniques did not yield improvements in prediction accuracy or generalization ability for SNNs. In fact, the results were less favorable than those of the original SNN model. Subsequent experiments on neurons with active impulse information confirmed that the application of dropout techniques disrupted impulse data, ultimately compromising SNN performance.

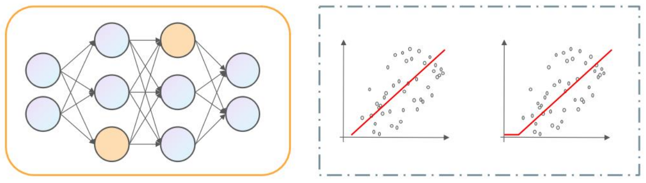


Fig. 1. Schematic diagram of dropout and activation methods

In the figure provided above, we can observe the foundational model of dropout along with an accompanying flowchart. Figure 2 illustrates the schematic representation of the activation in the two neural networks.

II. METHODOLOGY

In our study, we have employed various dropout techniques, including standard dropout, dropout2d, Feature-Dropout, and Alpha-Dropout, within the context of Spiking Neural Networks (SNNs). Below, we provide an overview of the foundational model we utilized in our research.

A. SNN

SNNs convey and process information through the timing of impulses. Our experimental setup involved a three-layer fully-connected SNN, which included a hidden layer featuring weighted delayed synapses between layers. This model is rooted in a spike response model, where the neurons in the

[1] We are co-first in this paper.

hidden layer receive spikes from the preceding layer and emit spikes when their membrane potential initially crosses the threshold.

In this particular experiment, the input data for the SNN were transformed into the first firing time of the input neuron. This firing time served as an integral component of the input vector for the hidden layer neuron. The postsynaptic potential of the kth synaptic connection from input neuron i to hidden layer neuron h was contingent on the strength of the synaptic connection. Subsequently, the membrane potential could be calculated using the following equation:

$$\begin{aligned} \upsilon_h(t) &= \sum_i \sum_k \omega_{ih}^k \sigma(t - t_i - d_{ih}^k) \\ &= \sum_{\text{i}} \sum_k \omega_{ih}^k y_{ih}^k(t) \end{aligned} \tag{1}$$

Here, the symbol ω denotes the synaptic weight, with each synapse delayed by a time period of d, and σ represents the spike response kernel.

$$\sigma(\alpha) = \begin{cases} \frac{\alpha}{\tau} e^{1-\frac{s}{\tau}} & if \;\; a > 0 \\ 0 & if \;\; a \le 0 \end{cases} \tag{2}$$

To facilitate the discussion, the equations (1) and (2) can be expressed in matrix form:

$$V_H(t) = W_H Y_H(t) \tag{3}$$

Here, $U_H = \{U_1, \cdots, U_H\}$(where H represents the number of neurons in the hidden layer), W_H denotes the weight matrix connecting the input layer to the hidden layer, and Y_H represents the kernel vector of the spike response in the hidden layer.[1-4]

B. Standard-dropout

In standard dropout, during the forward propagation process for each neuron layer, let's consider a neuron with its output denoted as *Y,* which is influenced by input *X* and weight *W*, this relationship can be represented as:

$$Y = f(Wx) \tag{4}$$

Here, f represents the activation function. Now, introducing the dropout probability p during the training phase, we apply a mask to each neuron in the neuron vector layer, as follows:

$$Y = f(Wx) \circ m \quad (m_i \sim Bernoulli) \tag{5}$$

After adjusting the weights through a loss function, during the testing phase, all neurons become active, and the network operates using the weights obtained during training. However, each neuron is retained with a probability of $1-p$. Consequently, the output is predicted as:

$$Y = (1-p) f(Wx) \tag{6}$$

C. dropout2d

Dropout2d is a regularization technique employed in convolutional neural networks, setting itself apart from standard dropout by its application to the output of convolutional layers. Its primary purpose is to mitigate overfitting by randomly discarding portions of channels within the output feature map of a convolutional layer.

During training, the input feature map of a convolutional layer is denoted as X, comprising multiple channels represented as $x_{i,j,k}$. The channel index i and spatial location indices j and k are associated with the feature map. The Dropout2d operation is expressed as follows:

$$Y_{i,j,k} = \begin{cases} 0 & with \ probabily \ p \\ \frac{X_{i,j,k}}{1-p} & with \ probabily \ 1-p \end{cases} \tag{7}$$

Here, $Y_{i,j,k}$ signifies the output after applying Dropout2d, $X_{i,j,k}$ denotes the input, and p represents the dropout probability. In this equation, if a channel is dropped, all its values at spatial locations are set to zero. Otherwise, its values are normalized by division with $(1-p)$.[5]

D. Feature-dropout

Feature-dropout resembles regular dropout but is typically a variant applied to convolutional neural networks (CNNs) that operate in the channel dimension. During each training iteration, Feature-dropout randomly sets specific channels of the input data to zero with a probability 'p'. This practice can be considered as feature-level regularization, promoting network independence from specific feature channels and thereby enhancing generalization performance.

E. Alpha-dropout

Standard-Dropout typically assigns a probability of $1-p$ to randomly set the activation g to zero during training. To maintain the mean, the activation g is divided by p to offset the portion that was set to zero.

On the other hand, Alpha-dropout is a regularization method designed for the $SeLU$ activation function, aimed at keeping the mean and variance constant to support the self-normalization property of the network. For the $SeLU$ activation function, the default and low variance value as x tends to negative infinity is given by:

$$\lim_{x \to -\infty} SeLU(x) = -\lambda\alpha = \alpha' \tag{8}$$

However, unlike Standard-Dropout, Alpha-dropout randomly sets the input to α' instead of 0. Through affine transformations, this ensures that the network ultimately achieves a zero mean and unit variance.[6-8]

The following are four common neural network regularization algorithms:

Algorithm: Main algorithm of Dropout on SNN

```
Input: X represents the input random variable,
Y represents the output random variable, H(x) represents
entropy, and H(X|Y) represents conditional entropy.

# Initialization
Randomly initialize data space X
Randomly initialize data label Y
input = Image(X, Y)

# Define a forward propagator function
for t in range(simulation_time_steps):
    weighted_sum=numpy.dot(input, weighteds)+self.biases
    spike_train = dropout algorithm # include dropout,
    dropout2d,alpha-dropout,feature-dropout;
end for

# Simulate the pulse propagation of neurons
for each input synapse in each neuron
    calculate the Membrane potential $U_H$(t) of each neuron
        with function $U_H$(t) = $W_H Y_H$(t)
    if $U_H$(t) > threshold then
        neuron.spike = true;
        Check if the neuron is firing a pulse
        Transmit the pulse to subsequent neurons
    end if
end for

Implement back propagation and weight updating

# Result Output
output_result = network_neurons[-1].spike
```

F. *complexity analysis*

The time complexity of standard dropout is calculated as $O(\sum_{j=1}^{j}\sum_{i=1}^{i} n_{ij}^2)$. The space complexity depends on storing the dropout mask and intermediate results during forward propagation, both of which are typically constant.

In contrast, dropout2d, Feature-dropout, and Alpha-dropout all have a time complexity of $O(N \cdot C \cdot H \cdot W)$, where N represents the batch size, C represents the number of channels in the input feature map, H denotes the height of the feature map, and W denotes the width of the feature map. For dropout2d, the space complexity is $O(1)$, while the complexity of Feature-dropout and Alpha-dropout is $O(N \cdot C \cdot H \cdot W)$.

III. EXPERIMENT

Based on the SNN model, we conduct experiments using two benchmark datasets, some of the details of which are as follows:

A. *experimental basis*

Dataset	Trainning Samples	Test samples	Input Dimention
Minist	60000	10000	28×28
CIFAR-10	50000	10000	32×32×3

Table1:Statistic Of Image classification Datasets

The MNIST dataset is one of the most commonly used and straightforward datasets in classification tasks. It comprises approximately 70,000 images in which the numbers 0-9 have been artificially handwritten. These images are grayscale maps, meaning they possess only a single image channel.

On the other hand, the CIFAR-10 dataset is somewhat more complex compared to MNIST. While MNIST is artificially generated, CIFAR-10 is a real-world dataset. CIFAR-10 consists of 10 distinct classes, and there are a total of 60,000 color images, each measuring 32x32 pixels.

Our experiments were conducted on a machine equipped with an NVIDIA Tesla V100 GPU boasting 32GB of RAM. To ensure the reliability of our results, we conducted each experiment five times, reporting both the mean performance and standard deviation. Our evaluation was based on categorization accuracy and loss performance metrics.

B. *Assessment of results*

In this experiment, we individually apply dropout, dropout2d, Feature-dropout, and alpha-dropout to SNNs, and compare the training loss against test accuracy with the base SNN (i.e., the SNN without any dropout operation). The results are visualized in Fig 2 to Fig 5.

From the four figures it can be seen that both the CIFAR dataset and the MINIST dataset, the training loss of the SNN model after applying the dropout and dropout2d methods is greater than that of the SNN base model, and the test accuracy is less than that of the SNN base model, while the application of Feature-dropout and alpha-dropout makes the SNN lose its training and prediction capabilities. In further research, we sorted the weights of neurons that were not dropout before and after training to the following results:

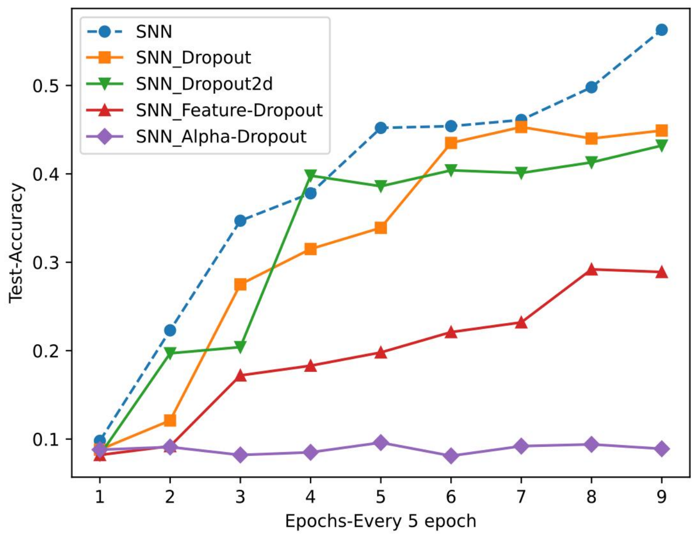
	Top 1-3			Top 4-10						
Without dropout	9	8	3	2	0	1	4	5	6	7
With dropout	112	127	121	122	116	118	113	9	1	5

Table2: Neuron ID Sorting

In Table 2, the numbers represent neuron numbers, and the higher the weight, the higher the ranking and the greater the importance. It can be seen that the important neurons with active pulse information change significantly before and after dropout. However, according to the existing experience, the overall importance of neurons should not change significantly with the same parameter settings, which is the reason why the neural network model has relative reproducibility. The reason for this dramatic change is due to the fact that simply applying dropout to SNNs will cause their impulse information to be disturbed, which will reduce the training and prediction effect of the SNN network, or even prevent the impulse neural network from working properly.[9-14]

C. *conclusion*

In this paper, we have implemented various dropout techniques on spiking neural networks (SNNs) and conducted training and testing on two datasets. After visualizing and analyzing the results, we have observed that the straightforward application of dropout and its variations to SNNs does not mitigate overfitting and does not effectively enhance their generalization abilities. Instead, it appears to compromise the performance of the SNN model.



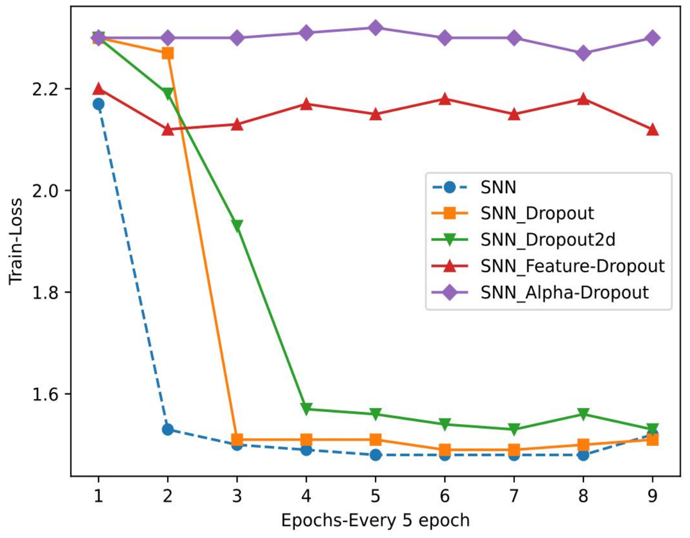
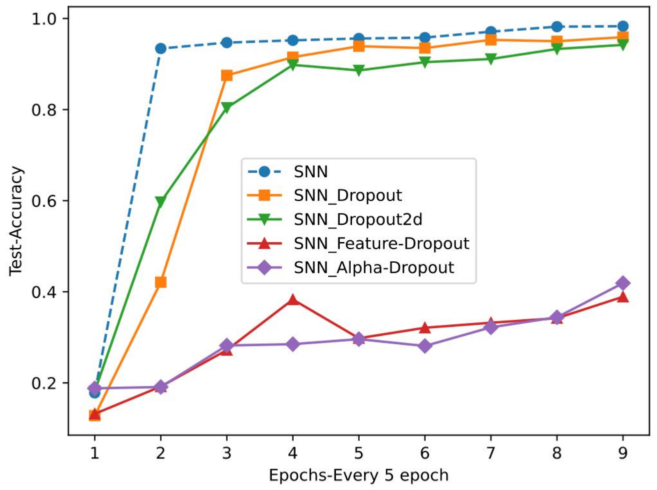


Fig2-3: Based on CIFAR-10 dataset



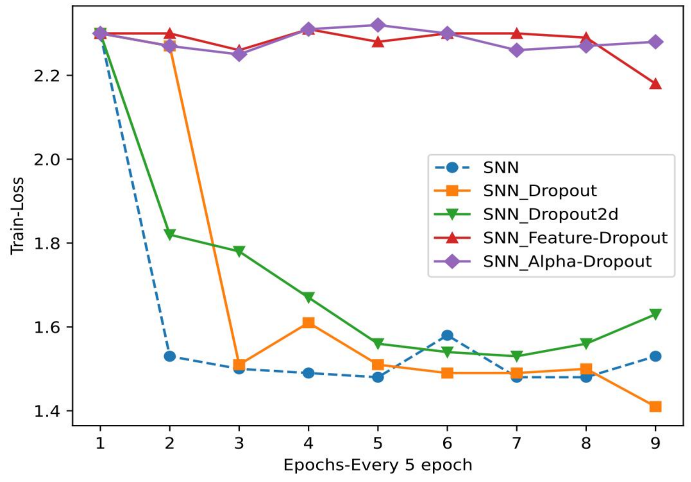


Fig4-5: Based on MINIST dataset

IV. References

[1] Zhao J, Yang J, Wang J, et al. Spiking neural network regularization with fixed and adaptive drop-keep probabilities[J]. IEEE Transactions on Neural Networks and Learning Systems, 2021, 33(8): 4096-4109.

[2] Liu Y, Tian M, Liu R, et al. Spike-based approximate backpropagation algorithm of brain-inspired deep SNN for sonar target classification[J]. Computational Intelligence and Neuroscience, 2022, 2022.

[3] Singh S, Sarma A, Lu S, et al. Skipper: Enabling efficient SNN training through activation-checkpointing and time-skipping[C]//2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2022: 565-581.

[4] Rathi N, Srinivasan G, Panda P, et al. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation[J]. arXiv preprint arXiv:2005.01807, 2020.

[5] Zahid U, Gambardella G, Fraser N J, et al. FAT: Training neural networks for reliable inference under hardware faults[C]//2020 IEEE International Test Conference (ITC). IEEE, 2020: 1-10.

[6] Klambauer G, Unterthiner T, Mayr A, et al. Self-normalizing neural networks[J]. Advances in neural information processing systems, 2017, 30.

[7] Wang S, Cheng T H, Lim M H. LTMD: Learning Improvement of Spiking Neural Networks with Learnable Thresholding Neurons and Moderate Dropout[J]. Advances in Neural Information Processing Systems, 2022, 35: 28350-28362.

[8] Sun T, Yin B, Bohte S. Efficient Uncertainty Estimation in Spiking Neural Networks via MC-dropout[J]. arXiv preprint arXiv:2304.10191, 2023.

[9] Zhao J, Yang J, Wang J, et al. Spiking neural network regularization with fixed and adaptive drop-keep probabilities[J]. IEEE Transactions on Neural Networks and Learning Systems, 2021, 33(8): 4096-4109.

[10] Y. Goldberg, ′′ Neural network methods for natural language processing,′′ Synth. Lectures Hum. Lang. Technol., vol. 10, no. 1, pp. 1–309, 2017.

[11] T. Young, D. Hazarika, S. Poria, and E. Cambria. (2017). ′′Recent trends in deep learning based natural language processing.′′ [Online]. Available: https://arxiv.org/abs/1708.02709

[12] Y. LeCun, Y. Bengio, and G. Hinton, ″Deep learning,″ Nature, vol. 521, pp. 436–444, May 2015.

[13] J. Bayer and C. Osendorfer. (2014). ′′Learning stochastic recurrent networks.″ [Online]. Available: https://arxiv.org/abs/1411.7610

[14] E. Neuman and J. Sandor. On the Ky Fan inequality and related inequalities i. ´ MATHEMATICAL INEQUALITIES AND APPLICATIONS, 5:49–56, 2002.