Efficient Converted Spiking Neural Network for 3D and 2D Classification

Yuxiang Lan[1], Yachao Zhang[2 ‡], Xu Ma[3], Yanyun Qu[1 ‡], Yun Fu[3]
[1]School of Informatics, Xiamen University
[2] Tsinghua Shenzhen International Graduate School, Tsinghua University
[3] Department of ECE, Northeastern University
lanyuxiang@stu.xmu.edu.cn, yachaozhang@sz.tsinghua.edu.cn, yyqu@xmu.edu.cn

Abstract

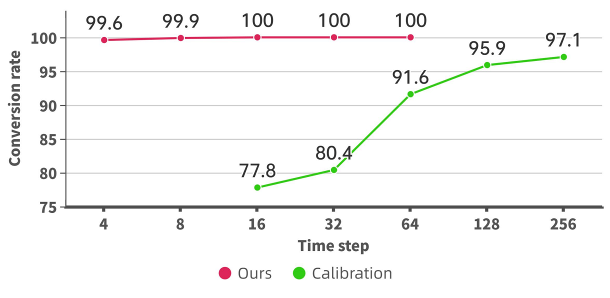
Spiking Neural Networks (SNNs) have attracted enormous research interest due to their low-power and biologically plausible nature. Existing ANN-SNN conversion methods can achieve lossless conversion by converting a well-trained Artificial Neural Network (ANN) into an SNN. However, converted SNN requires a large amount of time steps to achieve competitive performance with the well-trained ANN, which means a large latency. In this paper, we propose an efficient unified ANN-SNN conversion method for point cloud classification and image classification to significantly reduce the time step to meet the fast and lossless ANN-SNN transformation. Specifically, we first adaptively adjust the threshold according to the activation state of spiking neurons, ensuring a certain proportion of spiking neurons are activated at each time step to reduce the time for accumulation of membrane potential. Next, we use an adaptive firing mechanism to enlarge the range of spiking output, getting more discrimination features in short time steps. Extensive experimental results on challenging point cloud and image datasets demonstrate that the suggested approach significantly outmatches state-of-the-art ANN-SNN conversion based methods.

1. Introduction

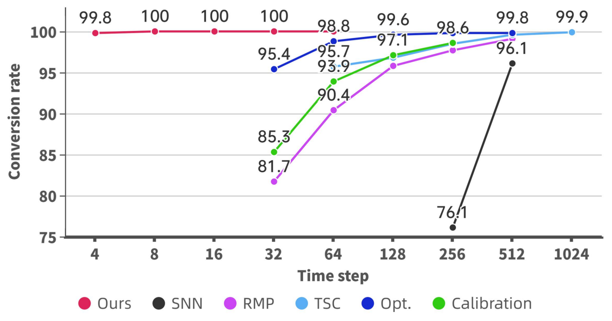
Artificial Neural Networks (ANNs) have achieved a phenomenal triumph in many Artificial Intelligence (AI) tasks such as 3D understanding [18, 30, 28], and image recognition [25, 6, 11, 8, 31]. However, the trend of significant growth in computational cost has also emerged with the rapid development of this field. State-of-the-art ANN models usually have billions of parameters, which consumes such a huge amount of computation that renders performing on-device inference challenging [1, 16]. Such a situation stimulates the demand for deploying more power-efficient networks in real-world applications [20]. Therefore, recent works have paid more attention to SNNs due to their low-power and biologically plausible nature [7, 9].

SNN transmits information by spike trains, which extend in the additional time dimension, and each spiking neuron in SNN triggers a spike when its accumulated membrane potential reaches the given upper bound, otherwise, it would

‡Corresponding Author



(A)



(B)

Figure 1. The inference conversion rate (%) for 3D classification and 2D classification. Conversion rate refers to the ratio of the performance of SNN to ANN. (A) presents the experimental results of Pointnet++ framework on ModelNet-10 data set. SNN Calibration is our implementation of point cloud classification using the official open-source code. (B) shows the experimental results using ResNet20 as the framework on CIFAR-10 dataset.

remain inactive for the current time step, which leads to more power-efficient computation by substituting expensive multiplication with addition [13]. Finally, SNN accumulates the output in all time steps to get the final output. Existing studies have shown that SNNs can save several orders of magnitude more energy than ANNs, on some assigned hardware, [21, 2].

Because of the discontinuity of spikes, we cannot use the backpropagation algorithm like ANN to train SNN. Two promising methods for obtaining SNN are the backpropagation with the surrogate gradient method and the weight conversion from ANNs. Firstly, SNN can be trained directly by the surrogate gradient, which is realized through the customized activation function [24, 10]. Based on this method, SNN can achieve close or even better performance compared with ANNs. But it is limited to the shallow SNN structure (usually containing a few hidden layers) and can not provide consistent results with ANN on complex large-scale datasets. Another way is by directly transferring the network parameters of pre-well-trained ANN into SNN, dubbed as ANN-SNN conversion methods [10, 4, 3, 14]. This method requires that the same network structure of SNN and the source ANN, and the parameters of SNN are transformed from ANN through a certain linear transformation. As shown in Figure 1, some state-of-the-art ANN-SNN conversion methods [23, 5, 4, 3, 14] achieve competitive accuracy with source ANN methods on complex datasets, but require huge simulation length, *i.e.,* a large time step to achieve near-lossless conversion performance [3, 15]. The time step is proportional to the inference time. A larger time step means that they need to consume more time in inference, that is, they have a larger latency.

Recently, SNN has been widely studied in image classification tasks [27, 23]. However, no relevant work has been done on 3D understanding tasks. 3D understanding is increasingly being used in many scenarios, including remote sensing, AR/VR, robotics, and automatic driving, these tasks also have the demand for energy saving in practical applications. Moreover, 2D and 3D are now used in multi-modal tasks, so it is necessary to introduce spiking neural networks into 3D understanding tasks. We directly transfer ANN-SNN conversion methods in image classification to point cloud classification tasks, but as shown in Figure 1, these methods have greater latency due to the unstructured and sparse for point cloud data.

There are two main reasons why SNN requires a large time step to fit the output of ANN. First, the initial value of the membrane potential is 0, which means that most spiking neurons need a long time to accumulate the membrane potential to reach the activation threshold from the initial membrane potential. This results in that almost all spiking neurons are inactive for a long time step, and the inactive neurons were more inhibited as the number of layers increased. Therefore, it takes a long period of potential accumulation to make most of the spiking neurons into an active state. Moreover, the sparsity of point cloud data makes the output of spiking more sparse, which increases the time of membrane potential accumulation and leads to greater latency. Secondly, SNN adopts binary output in each time step and accumulates the output in all time steps to get the final output. This means that after the spiking train of length T, the output result of the spiking can only be an integer in the range of $[0, 1]$, that is, only $T + 1$ values, which undoubtedly greatly reduces the discrimination between features compared with the float output of ANN. Therefore, it is essential to utilize a large time step to expand the value range of spiking neural network output to increase the discrimination between features and improve the SNN's final performance.

To remedy these problems, we propose an efficient unified point cloud classification and image classification method based on the SNN, which includes the adaptive dynamic activation threshold and the adaptive firing mechanism adapted to the dynamic threshold. To the best of our knowledge, it is the first spiking neural network for 3D point cloud classification. Specifically, we first propose to utilize KL divergence to initialize the activation threshold to reduce the difference in output distribution between ANN and SNN. Then, an adaptive dynamic threshold based on the activation state is used to dynamically adjust the threshold according to the activation state of the spiking neuron and the transformation of the time step. The dynamic threshold will converge quickly to the optimal threshold within a few time steps. Finally, in order to enlarge the range of spiking output value and increase the discrimination between features, we propose an adaptive firing mechanism that fully fires according to the current dynamic threshold.

The contributions of this work can be summarized as:

- We propose a unified ANN-SNN conversion method with low latency for point cloud classification and image classification. It is the first spiking neural network for 3D point cloud understanding.

- We introduce an adaptive dynamic activation threshold method, which adaptively adjusts the threshold, enabling spiking neurons to activate earlier. And an adaptive firing mechanism is established to enlarge the range of spiking output, ensuring that there is greater discrimination between features in short time steps.

- Extensive experimental results demonstrate the effectiveness of the proposed classification method based on ANN-SNN conversion. Compared with state-of-the-art ANN-SNN conversion methods in image classification tasks, the proposed method also has lower latency.

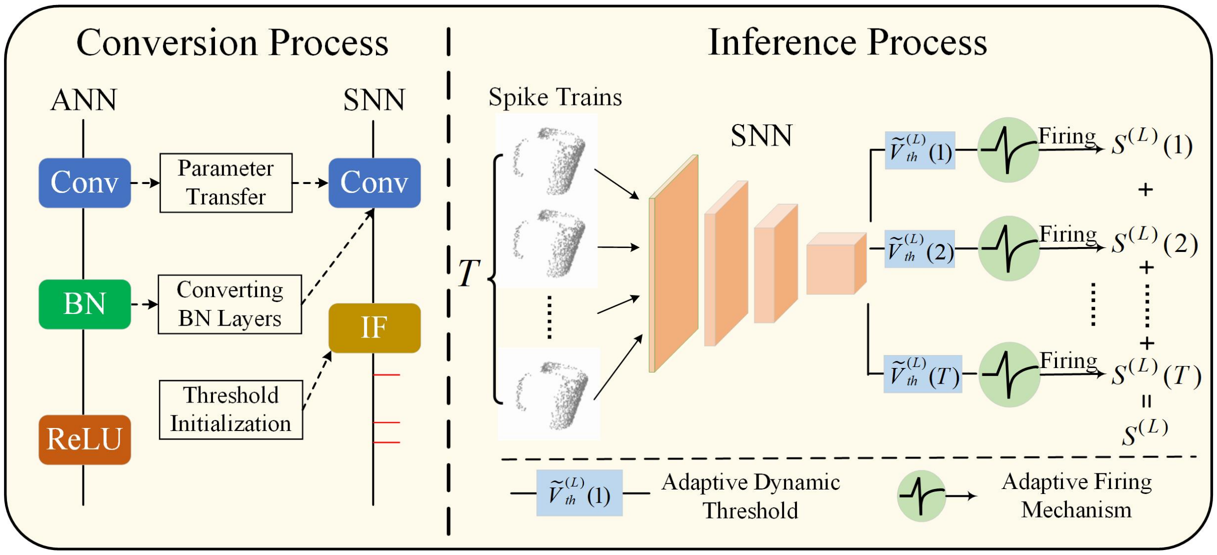


Figure 2. The framework of efficient converted SNN. The left part is the ANN to SNN conversion process, including the transfer of convolution layer parameters, the BN layer transformation, and the threshold initialization using KL divergence. The right part is the inference process of SNN, including the proposed adaptive adjustment of dynamic threshold according to the activation state of spiking neurons and adaptive firing mechanism.

2. Preliminaries

Integrate-and-Fire Mechanism. In this paper, we use the SNN model based on the Integrate-and-Fire (IF) [17] mechanism. Specifically, we suppose the input of spiking neurons in l^{th} layer at the time step t is $s^{(l-1)}(t) \in \{0,1\}$. Then, the temporary membrane potential $\widetilde{v}_{mem}^{(l)}(t)$ of neuron will be accumulated by

$$\widetilde{v}_{mem}^{(l)}(t) = v_{mem}^{(l)}(t-1) + W^{(l)} s^{(l-1)}(t), \tag{1}$$

where $W^{(l)}$ contains the weights and biases of the network in layer l, respectively. $v_{mem}^{(l)}(t-1)$ denotes the membrane potential after firing by the IF mechanism at time step $t-1$. Then the spiking output $s^{(l)}(t)$ will be produced by:

$$s^{(l)}(t) = \begin{cases} 1, & if\ \widetilde{v}_{mem}^{(l)}(t) \geq V_{th}^{(l)} \\ 0, & otherwise \end{cases}, \tag{2}$$

where $V_{th}^{(l)}$ is a pre-set activation threshold of l^{th} layer. The final membrane potential $v_{mem}^{(l)}(t)$ at the time step t is updated by soft-reset mechanism as:

$$v_{mem}^{(l)}(t) = \widetilde{v}_{mem}^{(l)}(t) - s^{(l)}(t) \cdot V_{th}^{(l)}. \tag{3}$$

Converting BN Layers. Because SNN generates a binary output, so there is no corresponding module in SNN for Batch Normalization (BN) layers. To better migrate the presentation capability of ANN and reduce the effect of removing the BN layer, Rueckauer et al. [22] propose to absorb the BN parameters to the weight and bias, which can be represented by:

$$\boldsymbol{W} \leftarrow \boldsymbol{W}\frac{\gamma}{\sigma}, \boldsymbol{b} \leftarrow \beta + (\boldsymbol{b} - \mu)\frac{\gamma}{\sigma}, \tag{4}$$

where μ, σ are the running mean and standard deviation, and γ, β are the transformation parameters in the BN layer.
Output Calculation of SNN. Compared with ANN, SNN employs binary activation (*i.e.* spikes) at each layer. To compensate for the loss in representation capacity, researchers introduce the time dimension to SNN by repeating the forwarding pass T times to get final results $S^{(l)}$ as:

$$S^{(l)} = \frac{1}{T}\sum_{1}^{T} s^{(l)}(i). \tag{5}$$

The Adjustment of Bias. To convert ANN into SNN, if only an appropriate threshold is found, there may still be a large deviation between the two outputs. Therefore, inspired by [14], we further adjust the network bias. First, we define a reduced mean function:

$$\bar{x} = \frac{1}{BN}\sum_{i=1}^{B}\sum_{j=1}^{N} x_{ij}, \tag{6}$$

where B and N represent the batch size and the number of points in the point cloud respectively. Then we can define the output error of ANN and SNN as:

$$\bar{e} = \bar{o}^{(l)} - S^{(l)}. \tag{7}$$

Finally, the bias is adjusted directly according to the error as $b^{(l)} \leftarrow b^{(l)} + \bar{e}$.

Algorithm 1 Conversion process.

```
Input: Pretrained ANN model; simulation length T
1: Folding BN Layers into Conv Layers
2: for all $l = 1, 2, ..., L^{th}$ layers in ANN do
3:     Sample 10 batches training data to initialize threshold $V_{th}^{(l)}$ with Eq. (8)
4:     Sample 10 batches training data to compute output error $\bar{e}$ with Eq. (7)
5:     Adjust bias $b^{(l)}$
6: end for
Output: Converted SNN model
```

Algorithm 2 Inference process.

```
Input: Converted SNN model; simulation length T; Initialize threshold $V_{th}^{(l)}$ for each layer
1: for each data do
2:     Set $v_{mem}^{(l)}(0) = 0$, $s^{(l)}(0) = 0$, $S^{(L)} = 0$
3:     for $t = 1, 2, ..., T$ do
4:         for all $l = 1, 2, ..., L^{th}$ layers in SNN do
5:             Cumulative membrane potential with Eq. (1)
6:             Compute dynamic threshold $\widetilde{V}_{th}^{(l)}(t)$ with Eq. (9) - Eq. (11)
7:             Compute spiking output with Eq. (13) and (14)
8:         end for
9:         Cumulative spiking output $S^{(L)} \leftarrow S^{(L)} + s^{(L)}(t)$
10:    end for
11: end for
Output: $S^{(L)}$ for each training data
```

3. Method

3.1. Overall Framework

The framework of our proposed unified ANN-SNN conversion method is shown in Figure 2. To obtain an SNN, it is necessary to transfer the parameters of a well-trained ANN, and fold BN layers into convolution layers. Then, initialize the threshold and adjust the bias parameters of each layer, the specific steps can be shown in Algorithm 1. Following the previous methods, in order to reduce the calculation cost, we only select 10 batches of samples to represent the entire dataset for initializing the threshold and computing output error. In the inference stage, the threshold of each layer is adjusted adaptively in real-time according to the activation state of spiking neurons of each layer for different samples, and the number of firing is determined by the dynamic threshold. The specific inference process is shown in Algorithm 2.

3.2. Adaptive Dynamic Activation Threshold

Initialize Threshold by KL Divergence. Although ANN and SNN have the same network structure and related network parameters, the output results of ANN and SNN are very different due to different inputs and activation mechanism. The output of ANN at each layer is a feature map, and the output of SNN is the activation of the spike. For classification tasks, we do not need to ensure that the outputs of ANN and SNN after activation are consistent, we only need to pay attention to the output distribution of the two networks. As long as the two networks can maintain a consistent output distribution, the same classification results can be obtained. In an effort to better enable the output of ANN and SNN to be distributed consistently, we use KL divergence to obtain the initialized layer-wise threshold $V_{th}^{(l)}$. Here we use a layer-wise optimization method like [3] to initialize our layer-wise thresholds. First, we normalize the output results of ANN and SNN by channel-wise respectively to get $\widetilde{o}^{(l)} = norm(o^{(l)})$ and $\widetilde{S}^{(l)} = norm(S^{(l)})$. Then, we calculate the KL divergence for both distributions to find the best initial threshold as:

$$\min_{V_{th}^{(l)}} \sum \widetilde{o}^{(l)} log \frac{\widetilde{o}^{(l)}}{\widetilde{S}^{(l)}}, \tag{8}$$

where $o^{(l)} = Relu(W^{(l)}x^{(l)} + b^{(l)})$ is the output of the i^{th} layer of ANN.

Equ. (8) takes a lot of time to accurately initialize the threshold layer by layer in the way of backpropagation, but in fact we do not need to be so precise for threshold initialization. Given that there is a clear upper and lower bound on the activation threshold, we sample some training samples and use grid search to find the $V_{th}^{(l)}$ that minimizes the objective function. Specifically, we uniformly sample N grids between [0.05$\cdot max(\widetilde{S}^{(l)})$, $max(\widetilde{S}^{(l)})$] to find the lowest KL divergence, where N is set to 95 empirically.

Adaptive Dynamic Threshold Based on Activation State. In order to reduce the time step for each spiking neuron to reach the activation threshold from the initial membrane potential and ensure that a certain number of spiking neurons are in the activated state in each time step, we introduce an adaptive dynamic threshold based on the activation state, which dynamically adjusts the threshold according to the activation state of the spiking neuron and the transformation of the time step.

At the l^{th} layer spiking neural network at time step t, we first estimated the unactivated proportion $\tau^{(l)}(t)$ of spiking neuron according to the temporary membrane potential:

$$\tau^{(l)}(t) = 1 - \frac{|\widetilde{v}_{mem}^{(l)}(t) \geq \widetilde{V}_{th}^{(l)}(t)|_{num}}{|\widetilde{v}_{mem}^{(l)}(t)|_{num}}. \tag{9}$$

where $|\widetilde{v}_{mem}^{(l)}(t) \geq \widetilde{V}_{th}^{(l)}(t)|_{num}$ and $|\widetilde{v}_{mem}^{(l)}(t)|_{num}$ represent the number of spiking neurons activated at the current threshold and the total number of spiking neurons, respectively. We first lower the threshold to induce some spiking neurons to activate in advance, and then we expected the threshold to return to the optimal threshold solved by KL divergence as the spiking sequence progressed. At the same time, we hope to determine the amplitude of threshold adjustment according to the firing state of the spiking neuron. So, we integrated the temporal variation with the spiking neuron activation state to get our activation threshold adjustment factor ω:

$$\omega^{(l)}(t) = 1 - \alpha \cdot e^{-(t-1)} \cdot \tau^{(l)}(t), \tag{10}$$

where α is the lower bound of the threshold adjustment factor, *i.e.*, $\min \omega^{(l)}(t) = \alpha$, which is a quantitative parameter set according to the memory size of the spiking activation value storage, and it will be explained in detail in the experiment section. According to the adjustment factor of Formula 10, we can get the dynamic threshold $\widetilde{V}_{th}^{(l)}(t)$ of each time as:

$$\widetilde{V}_{th}^{(l)}(t) = \omega^{(l)}(t) \cdot V_{th}^{(l)}, \tag{11}$$

In Formula 10, since α is a fixed value, and $\tau^{(l)}(t) \in [0,1]$ is a variable with upper and lower bounds according to Formula 9, we can deduce:

$$\lim_{t\to\infty} \omega^{(l)}(t) = 1 - \lim_{t\to\infty} \alpha \cdot e^{-(t-1)} \cdot \tau^{(l)}(t) = 1. \tag{12}$$

Therefore, we can prove theoretically that the dynamic threshold $\widetilde{V}_{th}^{(l)}(t)$ will eventually converge to the optimal threshold $V_{th}^{(l)}$ solved by KL divergence, that is, we can reduce the threshold adaptively at the beginning of the spiking sequence according to the activation of the spiking neuron, but with the passage of the sequence, the dynamic threshold will eventually return to our initial threshold. In this way, when the initial membrane potential is 0, the threshold can be reduced adaptively through the activation state of spiking neurons, inducing some spiking neurons to activate in advance, reducing the accumulation time of membrane potential from 0 to the activation limit, and improving the activation efficiency of spiking neurons. At the same time, after inducing SNN to enter the active state in advance, we rapidly converge the dynamic threshold to the optimal threshold, which ensures the stability of the SNN conversion performance. The dynamic threshold adaptive based on the activation state also makes the method more generalized.

3.3. Adaptive Firing Adapted to Dynamic Threshold

As shown in Figure. 3, after using the dynamic threshold, we can adaptively lower the threshold according to the activation state, but using the original IF firing mechanism will result in the insufficient firing of some spiking neurons, resulting in a backlog of membrane potential. At the same time, in order to expand the output range of spiking neurons and make them have a larger value range in a short

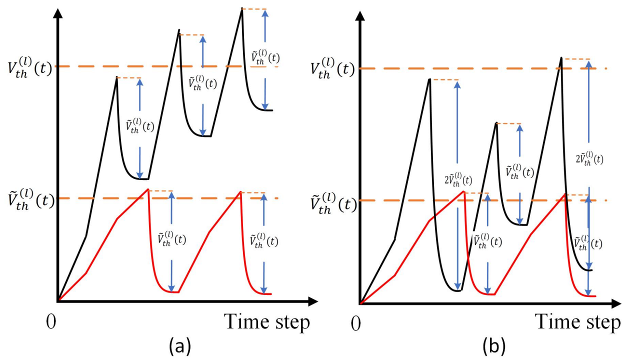


Figure 3. Fig(a) shows the membrane potential as a function of the spiking using IF activation, and Fig(b) shows the membrane potential as a function of the spiking train using the adaptive firing mechanism. The black and red lines represent two different spiking neurons

time step to increase the discrimination between features, we introduce an adaptive firing mechanism adapted to the dynamic threshold.

The purpose of using the adaptive firing mechanism is to enable the spiking neuron to fire several times according to the current dynamic threshold until the membrane potential is below the dynamic threshold. In this way, it can break binary output limit, and reduce the time step required for the spiking network to achieve lossless conversion. Specifically, the firing process can be expressed as:

$$s^{(l)}(t) = \begin{cases} \lfloor \frac{\widetilde{v}_{mem}^{(l)}(t)}{\widetilde{V}_{th}^{(l)}(t)} \rfloor, \; if \; \widetilde{v}_{mem}^{(l)}(t) \geq \widetilde{V}_{th}^{(l)}(t) \\ 0, \quad otherwise \end{cases}. \tag{13}$$

Considering that the original spiking neural network has the hardware advantage of low memory consumption in practical applications, expanding the spiking output range will increase the memory overhead, so we further limit the spiking output range to reduce the memory usage as much as possible on the premise of not affecting the performance of the adaptive firing mechanism. According to Equ.(13), we perform a truncation of the spiking output:

$$s^{(l)}(t) = clip\left(s^{(l)}(t), 0, 2^n - 1\right), \tag{14}$$

where *clip* (a, b, c) denotes that limiting the range of a to between b and c, n is a positive integer. In this way, the output of the spiking only needs n bits of space to be stored. In order to minimize the triggering of the truncation mechanism and allow the spiking neurons to fully fire, we use the parameter α in Equ.(10) to limit the adjustment factor, so that the original value of $s^{(l)}(t)$ falls within the interval $[0, 2^n - 1]$ as much as possible. According to Equ. (10), $\min \omega^{(l)}(t) = 1 - \alpha$. So, $\max s^{(l)}(t) \geqslant \dfrac{1}{1-\alpha}$. We assume

Table 1. Accuracy loss (%) and time steps due to ANN-SNN conversion on ModelNet-10 dataset. T means time step, which is proportional to inference time, *i.e.,* the larger the T, the greater the latency. Loss (%) represents the difference between the classification accuracy of SNN and ANN, *i.e.,* the classification accuracy of ANN minus the classification accuracy of SNN. Rate (%) means conversion rate, *i.e.,* the classification accuracy of ANN divided by the classification accuracy of SNN, which refers to the ratio of the performance of SNN to ANN

Arch.	Method	ANN	T	SNN	Loss	Rate
PointNet	Baseline	92.81	4	78.81	14.00	84.9
	Ours	92.81	4	92.38	0.43	99.5
	Baseline	92.81	8	88.52	4.29	95.4
	Ours	92.81	8	92.59	0.22	99.8
	Baseline	92.81	16	91.08	1.73	98.1
	Ours	92.81	16	**92.75**	**0.06**	**>99.9**
PointNet++	Baseline	93.54	4	74.02	19.52	79.1
	Ours	93.54	4	93.18	0.36	99.6
	Baseline	93.54	8	82.04	11.50	87.7
	Ours	93.54	8	**93.54**	**<0.01**	**100**
	Baseline	93.54	16	82.31	11.23	88.0
	Ours	93.54	16	**93.54**	**<0.01**	**100**

that most spiking output $s^{(l)}(t)$ does not exceed $\frac{1}{1-\alpha}+1$, so to satisfy that most spiking output falls in the interval $[0, 2^n-1]$, α can be set as $1-\frac{1}{2^n-2}$.

Using this adaptive firing mechanism adapted to the dynamic threshold, sufficient firing can be achieved at each time step, reducing the gap between the total input and total output in the entire time series, thereby effectively reducing the loss of information. At the same time, in each time step, we expand the output range of the spiking, which can increase the discrimination between the outputs in a finite time step.

4. Experiments

4.1. Experimental Details

Datasets and ANN. To evaluate the effectiveness and the efficiency of the suggested method in this paper, we adopt five challenging benchmarks: (1) ModelNet-10, ModelNet-40 [29] and ScanObject[26] for 3D point cloud classification, (2) CIFAR-10 and CIFAR-100 [12] for 2D image classification with extremely low simulation length. We selected two representative point cloud classification networks PointNet [18] and Pointnet++ [19] and two representative image classification networks VGG-16 [25] and ResNet-20 [6] as ANN of the experiment. In the ANN-SNN conversion method, using Maxpooling will not work. To better match SNN conversion, the Maxpooling layer in ANN needs to be replaced with Avgpooling layer.

Implementation Details. In the experiment part, all experiments are performed on a single NVIDIA GeForce GTX 2080 with Pytorch 1.8.1. The SGD optimizer is adopted for training ANN with an initial learning rate of 0.05 and momentum of 0.9 while the batch size is kept fixed at 16. In the process of converting ANN to SNN, we sample 5 batches of training samples and compute the KL divergence to initialize the threshold. And 5 batch point cloud samples are selected for adjustment. In all experiments, we set the hyperparameters $\alpha = 0.5$ and $n = 2$. The Baseline is the vanilla version without the adaptive dynamic threshold and adaptive firing mechanism.

Table 2. Accuracy loss (%) and time steps due to ANN-SNN conversion on ModelNet-40 dataset

Arch.	Method	ANN	T	SNN	Loss	Rate
PointNet	Baseline	88.17	4	64.96	23.22	73.7
	Ours	88.17	4	87.95	0.22	99.8
	Baseline	88.17	8	79.98	8.19	90.7
	Ours	88.17	8	87.99	0.18	99.8
	Baseline	88.17	16	84.17	4.00	95.5
	Ours	88.17	16	**88.17**	**<0.01**	**100**
PointNet++	Baseline	89.45	4	62.10	27.35	69.4
	Ours	89.45	4	89.37	0.08	99.9
	Baseline	89.45	8	72.87	16.58	81.5
	Ours	89.45	8	89.43	0.02	>99.9
	Baseline	89.45	16	78.26	11.19	87.5
	Ours	89.45	16	**89.45**	**<0.01**	**100**

4.2. Experiment Results in 3D Classification

In this section, we compare our proposed method with baseline under different time steps, and the results are shown in Table 1-3. It should be noted that it is the first time for us to introduce spiking neural network in 3D point cloud classification. Compared with the spiking neural network in 2D image classification, we have lower conversion loss and lower latency of network inference. In all experiments, our method only needs 8 steps to achieve a lossless or nearly lossless performance compared to ANNs.

We can see that in ModelNet-10, when PointNet was used as ANN to convert SNN, the performance of our SNN basically reached that of ANN at the time step of $T = 8$. More surprisingly, for the more complex network PointNet++, we achieve a complete lossless conversion at the time step of $T = 8$. It can be seen that the Baseline, under the extreme condition of $T = 4$, only has a conversion rate of about 80%, while the method we proposed has achieved a conversion rate of more than 99%.

ModelNet-40 has more classification categories than ModelNet-10, so the classification difficulty is improved, which results in the decline of ANN's classification performance, and the conversion rate of Baseline at $T = 4$ is further reduced to about 70%. However, it can be witnessed

Table 3. Accuracy loss (%) and time steps due to ANN-SNN conversion on ScanObject dataset

Arch.	Method	ANN	T	SNN	Loss	Rate
PointNet	Baseline	66.56	4	43.34	23.22	65.1
	Ours	66.56	4	65.16	1.40	97.9
	Baseline	66.56	8	58.37	8.19	87.7
	Ours	66.56	8	65.94	0.62	99.1
	Baseline	66.56	16	62.56	4.00	94.0
	Ours	66.56	16	**66.56**	**<0.01**	**>99.9**
PointNet++	Baseline	69.22	4	49.70	19.52	71.8
	Ours	69.22	4	68.91	0.31	99.6
	Baseline	69.22	8	57.72	11.50	83.4
	Ours	69.22	8	69.06	0.16	99.8
	Baseline	69.22	16	57.99	11.23	83.8
	Ours	69.22	16	**69.22**	**<0.01**	**100**

that under the setting of $T = 4$, the presented method can achieve a conversion rate of 99.8% or above for both kinds of ANN. This demonstrates the effectiveness and robustness of our proposed method.

As can be seen from ANN's performance, the ScanObject dataset has higher classification difficulty. However, the good news is that our method can still maintain a high conversion rate at a time step of 4 while ensuring lossless conversion at $T = 16$.

4.3. Experiment Results in 2D Classification

From Table 4, we can see that in CIFAR-10, our method achieves comparable performance to SpikeConverter [15] in conversion loss using the same time step, but our inference accuracy can be higher than it. Furthermore, compared to other methods, we only need 1/4 or 1/128 time steps to achieve higher inference accuracy and lower conversion loss than them. Our proposed adaptive ANN-SNN conversion method can achieve a lossless or nearly lossless conversion performance at $T = 8$. Compared to the Baseline, after using the adaptive dynamic threshold and adaptive firing mechanism the conversion performance achieves significantly improved with the same time step.

The experimental results in the CIFAR-100 dataset are shown in Table 5. Under ResNet20 and VGG16 frameworks, lossless conversion can be basically achieved by using time step 8. We have achieved a slight advantage over SpikeConverter in terms of conversion losses, as well as a lead in the final inference performance of SNN, especially in the VGG16 framework. We get better performance with much fewer time steps than other methods except SpikeConverter. Likewise, we have achieved a great improvement over the Baseline in a small time step.

Table 4. Accuracy loss (%) and time steps due to ANN-SNN conversion of the state-of-the-art SNNs on CIFAR-10 dataset

Arch.	Method	ANN	T	SNN	Loss
ResNet-20	Spike-Norm [23]	89.10	2048	87.46	1.64
	RMP [5]	91.47	2048	91.36	0.11
	TSC [4]	91.47	2048	91.42	0.05
	Opt. [3]	92.14	128	90,89	1.25
	SpikeConverter [15]	91.47	16	91.47	<0.01
	Baseline	94.23	8	85.63	8.60
	Ours	94.23	8	94.16	0.07
	Baseline	94.23	16	90.52	3.71
	Ours	94.23	16	**94.19**	0.04
VGG-16	Spike-Norm [23]	91.70	2048	91.55	0.15
	RMP [5]	93.63	2048	93.63	<0.01
	TSC [4]	93.63	2048	93.63	<0.01
	SpikeConverter [15]	93.63	16	93.71	$\uparrow$**0.08**
	Baseline	94.31	8	87.31	7.00
	Ours	94.31	8	94.26	0.05
	Baseline	94.31	16	91.11	3.20
	Ours	94.31	16	94.30	0.01

Table 5. Accuracy loss (%) and time steps due to ANN-SNN conversion of the state-of-the-art SNNs on CIFAR-100 dataset

Arch.	Method	ANN	T	SNN	Loss
ResNet-20	Spike-Norm [23]	68.72	2048	64.09	4.63
	RMP [5]	68.72	2048	67.82	0.90
	TSC [4]	68.72	2048	68.18	0.54
	SpikeConverter [15]	68.72	16	68.69	0.03
	Baseline	68.75	8	47.46	21.29
	Ours	68.75	8	68.67	0.08
	Baseline	68.75	16	60.28	8.07
	Ours	68.75	16	**68.73**	**0.02**
VGG-16	Spike-Norm [23]	71.22	2048	70.77	0.45
	RMP [5]	71.22	2048	70.93	0.29
	TSC [4]	71.22	2048	70.97	0.25
	SpikeConverter [15]	71.22	16	71.22	<0.01
	Baseline	73.40	8	53.11	20.29
	Ours	73.40	8	73.36	0.04
	Baseline	73.40	16	62.45	10.95
	Ours	73.40	16	**73.39**	**<0.01**

4.4. Ablation Study

We verify the design threshold initialization by KL divergence, the adaptive dynamic threshold based on the activation state, and the adaptive firing adapted to the dynamic threshold mechanism. In all ablation experiments, we test PointNet and PointNet++ on ModelNet-10 and set the time step as 16. The results are shown in Table 6.

Effectiveness of Threshold Initialization (Initia.). For the threshold initialization, we compare two initialization methods: the initialization method using the max value of the feature map as the initialization threshold, and the initialization method based on the KL divergence proposed in this paper. From #1 $\rightarrow$ #2 and #5 $\rightarrow$ #6, we can find that using the KL-based method, the performance of SNN has been

Table 6. Result of ablation study on ModelNet-10 dataset using PointNet and PointNet++ with time step as 16

	Baseline	Initia.	DT	Firing	PN	PN++
#1	✓	Max			88.52	82.04
#2	✓	KL			89.56	83.31
#3	✓	KL	✓		86.59	77.48
#4	✓	KL		✓	91.67	83.54
#5	✓	Max	✓	✓	92.18	91.90
#6	✓	KL	✓	✓	92.75	**93.54**

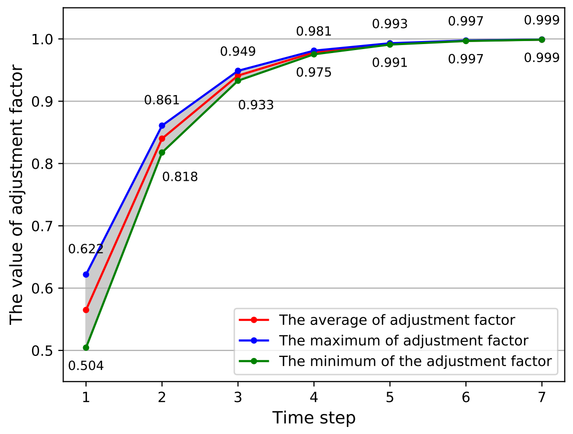


Figure 4. In inference process, the average, maximum, and minimum of adjustment factor in each time step

improved to a certain extent in both network structures.

Effectiveness of Adaptive Dynamic Threshold (DT) and Adaptive Firing Machine (Firing). Compare with #2 $\rightarrow$ #3, #2 $\rightarrow$ #4, #3 $\rightarrow$ #6 and #4 $\rightarrow$ #6, it can be seen that the performance is improved to a certain extent when the adaptive firing mechanism is used alone, but the conversion performance deteriorates when the adaptive dynamic threshold is used alone. The combined use of the dynamic threshold and the adaptive firing mechanism can significantly improve performance. This indicates that using of dynamic threshold alone can activate the spiking in advance by reducing the threshold, if there is no matching firing mechanism, the membrane potential cannot be fully released and will continue to accumulate, resulting in a large gap between the total input and total output after the end of the sequence, which leads to the degradation of performance. On the contrary, the output range of the spiking neural network can be extended by using the adaptive firing mechanism, which can also reduce the delay to a certain extent. This fully verifies that dynamic threshold and adaptive firing machine are complementary, the best performance is achieved when the two are used together.

4.5. Analysis

The Convergence of the Adjustment Factor. We randomly selected 20 batches of training samples for the experiment. In the experiment, we set the hyperparameters $\alpha = 0.5$ and $n = 2$. Finally, the threshold of the first layer in the network was selected for statistical analysis. Figure 4 shows the trend of the maximum, minimum, and average values of the adjustment factors overtime during the inference process. As can be seen from Figure 4, as time progresses, the maximum and minimum values tend to be close, and the value of the adjustment factor finally converges to 1. This verifies the convergence of the adjustment factor and also shows that after the threshold value is reduced briefly to promote the early activation of the spiking neuron, it can quickly recover to the optimal threshold obtained by the KL divergence, which can reduce the delay and ensure the conversion effect.

Table 7. Results on different parameters α and n on ModelNet-10 dataset using PointNet architecture and PointNet++ architecture.

Arch.	α	n	ANN	T			
				2	4	8	16
PointNet	0.10	1	92.81	91.44	92.19	92.56	92.73
	0.50	2		91.57	92.38	92.79	92.80
	0.83	3		91.76	92.68	92.81	92.81
	0.93	4		91.98	92.75	92.81	92.81
PointNet++	0.10	1	93.54	92.70	92.90	93.06	92.38
	0.50	2		93.06	93.18	93.54	93.54
	0.83	3		93.21	93.54	93.54	93.54
	0.93	4		93.27	93.54	93.54	93.54

Table 8. The comparison of computational costs of theoretical hardware-deployed between ANN and SNN

Operations	ANN	SNN
Addition	$N \times (D-1)$	$N \times (D-1) \times T$
Multiplication	$N \times D$	0

Parameter Sensitivity Experiment. We further verify parameter α of Formula 10 and parameter n of Formula 14. We change the value of n and calculate the corresponding value of α according to the method in Section 3.3. We use PointNet as ANN to conduct experiments on the ModelNet-10 dataset and set the time step as 8 during the experiment. The experimental results are shown in Table 7.

Inference Computation Cost. In most Artificial Neural Network workloads, the computational effort is focused on general matrix-matrix multiplication, which often occurs in forward processes. For a more intuitive comparison, in this section, we use vector-matrix multiplication (VMM) to evaluate the computational cost, since vectors are a special case of matrices. The size of the point cloud of each scene is $N \times D$, where N is the number of points in one scene, D is the feature dimension of each point cloud, and the size of the convolution kernel is $D \times 1$. Therefore, the dimensions of vector-matrix multiplications is $(N \times D) \times (D \times 1)$. In the ANN, $N \times (D-1)$ additions and $N \times D$ multiplications are performed to compute a VMM. In the converted SNN model, it is unnecessary to perform multiplication anymore, but only $N \times (D-1) \times T$ additions, where T denotes the number of time steps. The comparison of computation between ANN and converted SNN is shown in Table 8.

5. Conclusion

We propose a unified and efficient adaptive SNN conversion method, which can significantly reduce latency for SNN to achieve lossless performance compared to source ANN. This huge boost comes from two mechanisms: the adaptive dynamic threshold and the adaptive firing mechanism. The former reduces the time required for membrane potential to accumulate by adaptively adjusting the threshold based on the activation state of the spiking neuron at each time step. The latter enlarges the range of the spiking output and makes the greater discrimination of features in a short time step. Extensive experimental results on point cloud and image datasets demonstrate the effectiveness of our proposed method.

6. Acknowledgments

Yuxiang Lan, Yachao Zhang, and Yanyun Qu were supported by the National Natural Science Foundation under Grant No. 6217224, the China Postdoctoral Science Foundation No. 2023M731957, the CCF-Lenovo Blue Ocean Research Fund.

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[2] Lei Deng, Yujie Wu, Xing Hu, Ling Liang, Yufei Ding, Guoqi Li, Guangshe Zhao, Peng Li, and Yuan Xie. Rethinking the performance comparison between snns and anns. *Neural networks*, 121:294–307, 2020.

[3] Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks. *ICLR*, 2021.

[4] Bing Han and Kaushik Roy. Deep spiking neural network: Energy efficiency through time based coding. In *ECCV*, pages 388–404. Springer, 2020.

[5] Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In *CVPR*, pages 13558–13567, 2020.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[7] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

[8] Runze Hu, Yutao Liu, Ke Gu, Xiongkuo Min, and Guangtao Zhai. Toward a no-reference quality metric for camera-captured images. *IEEE Transactions on Cybernetics*, 2021.

[9] Eugene M Izhikevich. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6):1569–1572, 2003.

[10] Saeed Reza Kheradpisheh, Mohammad Ganjtabesh, Simon J Thorpe, and Timothée Masquelier. Stdp-based spiking deep convolutional neural networks for object recognition. *Neural Networks*, 99:56–67, 2018.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 25, 2012.

[12] Nair V. Krizhevsky, A. and G Hinton. The cifar-10 dataset. `http://www.cs.toronto.edu/kriz/cifar.html`.

[13] Hunjun Lee, Chanmyeong Kim, Yujin Chung, and Jangwoo Kim. Neuroengine: a hardware-based event-driven simulation system for advanced brain-inspired computing. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 975–989, 2021.

[14] Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In *ICML*, pages 6316–6325. PMLR, 2021.

[15] Fangxin Liu, Wenbo Zhao, Yongbiao Chen, Zongwu Wang, and Li Jiang. Spikeconverter: An efficient conversion framework zipping the gap between artificial neural networks and spiking neural networks. In *AAAI*, 2022.

[16] Fangxin Liu, Wenbo Zhao, Zhezhi He, Yanzhi Wang, Zongwu Wang, Changzhi Dai, Xiaoyao Liang, and Li Jiang. Improving neural network efficiency via post-training quantization with adaptive floating-point. In *ICCV*, pages 5281–5290, 2021.

[17] Ying-Hui Liu and Xiao-Jing Wang. Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron. *Journal of computational neuroscience*, 10(1):25–45, 2001.

[18] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017.

[19] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, pages 5099–5108, 2017.

[20] Nitin Rathi, Amogh Agrawal, Chankyu Lee, Adarsh Kumar Kosta, and Kaushik Roy. Exploring spike-based learning for neuromorphic computing: Prospects and perspectives. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 902–907. IEEE, 2021.

[21] Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. *ICLR*, 2019.

[22] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of

continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

[23] Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures. *Frontiers in neuroscience*, 13:95, 2019.

[24] Sumit B Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *Advances in neural information processing systems*, 31, 2018.

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1588–1597, 2019.

[27] Stanisław Woźniak, Angeliki Pantazi, Thomas Bohnstingl, and Evangelos Eleftheriou. Deep learning incorporating biologically inspired neural dynamics and in-memory computing. *Nature Machine Intelligence*, 2(6):325–336, 2020.

[28] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *CVPR*, pages 9621–9630, 2019.

[29] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015.

[30] Yachao Zhang, Yanyun Qu, Yuan Xie, Zonghao Li, Shanshan Zheng, and Cuihua Li. Perturbed self-distillation: Weakly supervised large-scale point cloud semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15520–15528, 2021.

[31] Yachao Zhang, Yuan Xie, Cuihua Li, Zongze Wu, and Yanyun Qu. Learning all-in collaborative multiview binary representation for clustering. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.