mathematics

MDPI

Article

IC-SNN: Optimal ANN2SNN Conversion at Low Latency

Cuixia Li [1,2], Zhiquan Shang [2], Li Shi [1,3], Wenlong Gao [2] and Shuyan Zhang [2,*]

[1] School of Electrical Engineering, Zhengzhou University, Zhengzhou 450001, China
[2] School of Cyber Science and Engineering, Zhengzhou University, Zhengzhou 450001, China
[3] Department of Automation, Tsinghua University, Beijing 100084, China
* Correspondence: syzhang@zzu.edu.cn

Abstract: The spiking neural network (SNN) has attracted the attention of many researchers because of its low energy consumption and strong bionics. However, when the network conversion method is used to solve the difficulty of network training caused by its discrete, too-long inference time, it may hinder the practical application of SNN. This paper proposes a novel model named the SNN with Initialized Membrane Potential and Coding Compensation (IC-SNN) to solve this problem. The model focuses on the effect of residual membrane potential and rate encoding on the target SNN. After analyzing the conversion error and the information loss caused by the encoding method under the low time step, we propose a new initial membrane potential setting method and coding compensation scheme. The model can enable the network to still achieve high accuracy under a low number of time steps by eliminating residual membrane potential and encoding errors in the SNN. Finally, experimental results based on public datasets CIFAR10 and CIFAR100 also demonstrate that the model can still achieve competitive classification accuracy in 32 time steps.

Keywords: spiking neural networks; conversion error; initial membrane potential

MSC: 74A40; 74J05

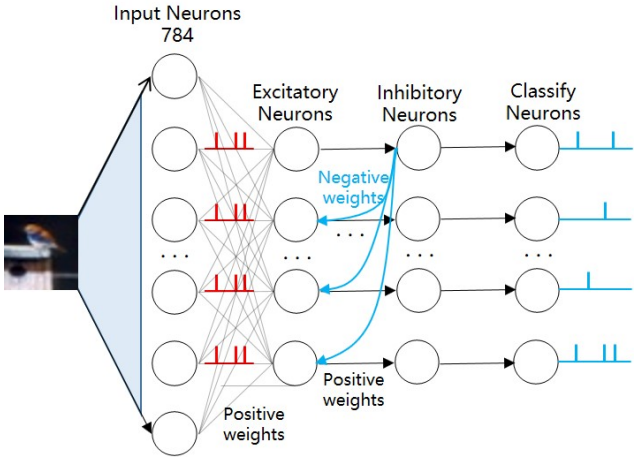
Citation: Li, C.; Shang, Z.; Shi, L.; Gao, W.; Zhang, S. IC-SNN: Optimal ANN2SNN Conversion at Low Latency. *Mathematics* **2023**, *11*, 58. https://doi.org/10.3390/math11010058

Academic Editor: Jonathan Blackledge

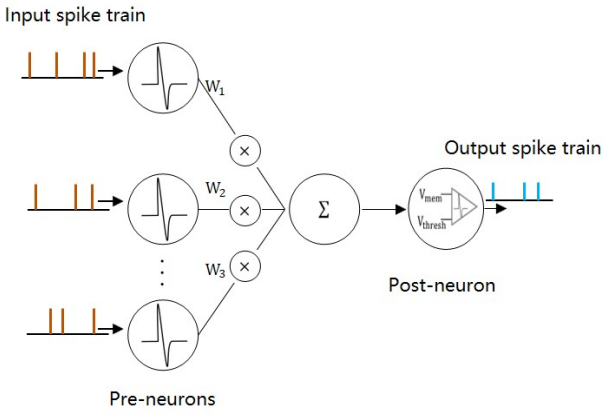
Received: 14 November 2022
Revised: 11 December 2022
Accepted: 19 December 2022
Published: 23 December 2022

1. Introduction

Artificial neural networks (ANNs) have achieved great success in the past few years, although they differ structurally from the brain in terms of calculation methods and learning rules. Although some technologies similar to pruning, quantization, and compression have been proposed, ANN still faces problems of high power consumption and complex calculations [1–3]. The spiking neural network (SNN) is a third-generation neural network, and it has gained widespread attention due to its low power consumption. Benefiting from its biological inspiration, SNNs use spike-based binary values to pass information between neurons [4]. Figure 1 represents the overall structure of SNN and the workflow on spiking neurons. Davd et al. [5] carried out detailed calculations on the energy consumption of a single neuron, proving that the brain's energy consumption is low. Maass et al. [6] analyzed the capabilities of cortical microcircuits for real-time computing and demonstrated that SNNs have powerful computing capabilities. The binary nature of SNN enables it to use accumulated operations instead of expensive multiply accumulated operations during network propagation, thereby reducing the amount of network computation. In the application of edge devices, the low calculation load of SNN has a great advantage. In addition, in some specific neuromorphic chips, SNN can run in an event-driven manner. When not receiving spikes, neurons will remain dormant, saving a lot of power consumption. However, SNNs struggle to achieve similar accuracy in traditional machine learning tasks. On the one hand, there is currently no conclusive understanding about the learning process in the brain, and it is far from enough to simply simulate brain neurons. On the other hand, the learning algorithms of SNNs are still in the developmental stage and lack effective learning algorithms.



(**a**) The network structure of spiking neural network (SNN).



(**b**) Workflow of spiking neurons.

Figure 1. Spike neuron network. (**a**) represents the structure of the spiking neural network. The image is input into the network and propagated forward in the form of spike sequences. Then, a set of spike sequences for classification is obtained through excitatory neurons or inhibitory neurons; (**b**) represents the network in the structure of neurons in a certain layer. The spike sequence enters the neurons and integrates and fires the spikes, which are transmitted to the next layer of neurons as output.

Research on learning algorithms for SNNs can be roughly divided into three categories: learning algorithms based on synaptic plasticity, surrogate gradient-based backpropagation, and ANN-to-SNN conversion. As SNN is biologically credible, some researchers believe that it should follow biological mechanisms to train SNNs. They tend to apply the neuronal mechanism in the biological brain to SNNs. The network is trained by using Hebbian, Spike-Timing-Dependent Plasticity (STDP), Self-backpropagation (SBP), and other neuron mechanisms [7–9]. Inspired by backpropagation (BP), which excels in ANNs, researchers try to use BP in the training of SNNs [10]. They convert non-differentiable pulse trains into differentiable values, and build surrogate gradient functions to achieve backpropagation in space and time [11,12]. Furthermore, inspired by transfer learning, the researchers trained a topological and homogeneous non-spiking neural network in advance and transferred the weights to the spiking neural network to achieve approximate accuracy. There are differences between different networks and neurons, so it is necessary to weigh thresholds and weights in the conversion process, as well as eliminate errors such as clipping and quantization [13–15].

In this paper, we analyze the error of the network in the conversion process and propose an algorithm to initialize the membrane potential. This algorithm can sufficiently reduce the error in the conversion process. To cope with neural networks at low time steps, we use the ANN-SNN transformation as an initialization step and initialize the membrane potentials of neurons in each layer based on the transformed network and the training set. In addition, we propose a compensation tactic for the adopted rate coding. The tactic is based on the time step and combines the compensation part of each layer with the initialized membrane potential, which reduces the information loss to a certain extent.

The main contributions of this work are as follows:

- We theoretically analyze the error of ANN2SNN, which explains why information is lost at low time steps to a certain extent. We propose an algorithm to initialize membrane potential to eliminate the error term at low latency.
- We analyze the coding methods and propose a coding compensation method combined with the membrane potential. The novel coding compensation method can reduce information loss.
- Experiments are conducted on datasets based on two traditional network models. Experimental results demonstrate the validity of the new algorithm. The results

show that our approach can effectively reduce the conversion error and obtain a high accuracy at low latency.

2. Related Work

The learning algorithm of SNN is one factor that restricts its network performance (e.g., accuracy, time step). Current learning algorithms can roughly be divided into three categories. We mainly introduce the related research work in recent years.

2.1. Learning Algorithms Based on Synaptic Plasticity

Based on the biological credibility of SNNs, it is feasible to transfer the neuron learning mechanism in biology to the network. Masquelier et al. [16] used STDP rules in feedforward spiking neural networks, which showed that temporal encoding may be the key to understanding the fantastic processing speed achieved by the visual system and that STDP can lead to fast and selective responses. By analyzing the learning effect of STDP, Legenstein et al. [17] realized that neurons that have learned through STDP can classify the firing patterns in space and time. Ponulak et al. [18] proposed a remote supervised method (ReSuMe). They demonstrated that the technique could train neurons to reproduce spike trains with controllable temporal offsets relative to a target template. Furthermore, they noted that spiking neurons may predict behavior through firing timing. Wade et al. [19] combined the Bienenstock Cooper Munro (BCM) learning rule with STDP and proposed the Synaptic Weight Association Training (SWAT) algorithm. Diehl et al. [20] used excitatory and inhibitory neurons and opted to use a lateral inhibition mechanism to reduce computational complexity. Tavanaei et al. [21] trained a biologically plausible convolutional filter and learned detector. The detector employs probabilistic neurons to extract independent features. Zhang et al. [22] first trained the network to a balanced membrane potential state, then maintained the network's balance based on excitatory and inhibitory neurons. The neural networks adopted short-term plasticity (STP) to ensure overall balance to speed up convergence. Influenced by the self-organized propagation mechanism of hippocampal neurons, Zhang et al. [9] introduced a non-local synaptic modification method termed "self-BP" (SBP) for synaptic potentiation and depression. However, the various improvements described above for synaptic plasticity are often applied to shallow networks with five, four, or even three layers. To a certain extent, this indicates that the improvement based on synaptic plasticity pays too much attention to local information. It lacks the acquisition of global information, and is difficult to apply in deeper networks.

2.2. Surrogate Gradient-Based Backpropagation

The BP and the optimization algorithm of gradient descent commonly used in ANN are a good choice. Bohte et al. [11] tried to apply the BP algorithm to SNNs for the first time and proposed the Spike Propagation (SpikeProp) algorithm, which is suitable for SNN. The proposed of SpikeProp has led some researchers to focus on converting discrete, non-differentiable spike trains into continuous values and using backpropagation to train the network. Wu et al. [12,23] proposed space-time-based backpropagation. The method employs gradient approximation to avoid discontinuities in spiking activation functions and applies backpropagation through time (BPTT) on SNNs. Zhang et al. [24] used Temporal Spike Sequence Learning via Backpropagation(TSSL-BP) to bypass the non-differentiability of the spike activation function and improve the temporal learning accuracy. Fang et al. [25] improved the neuron model and added hyper-parameters to the backpropagation learning to enhance the expressive power of neurons. Xiao et al. [26] proposed that the average firing rate of SNN evolves to an equilibrium state over time and follows a fixed point equation. Equation-based implicit differentiation, computing the parameters' gradient, avoids the non-differentiable spike function problem. The current research on such algorithms generally focuses on adjusting hyperparameters to make the gradient function more suitable for the spike firing rate. The study focuses on constructing the gradient function rather than

the SNN itself. In addition, the gradient function also accumulates approximation errors. Thus, it is difficult to obtain a high-performance SNN.

2.3. ANN-to-SNN Conversion

Transfer learning can not only achieve higher initial performance, but it can also reduce training time significantly. Inspired by transfer learning, the researchers share the weight parameters of the source neural network to the target the neural network, thus saving the training time on the target neural network. Cao et al. [13] proposed an indirect training approach, which trained a homogeneous topological neural network and transferred trained neural network parameters to the spiking neuron network. Diehl et al. [27] explained that the conversion principle is a correlation between the value of the activation function of traditional neurons and the firing rate of spiking neurons, and proposed weight normalization for the difference in the output range of ANN and SNN. Rueckauer et al. [28] proposed a weight standardization based on the P quantile considering the influence of the singular point in the weight normalization process. Sengupta et al. [29] further analyzed the conversion principle. They pointed out that threshold balance and weight normalization can be regarded as the same operation, that is, to adjust the relationship between threshold and weight. Then, the existing standardization is improved by the standardization technology Spike Normalization based on SNN. For the linear regression problem, Kim et al. [30] proposed a more fine-grained channel normalization and neurons with negative thresholds. Park et al. [31,32] proposed time-based coding of Time-To-First-Spike (TTFS) and Temporal-Switch-Coding (TSC). Stöckl et al. [33] adopted dynamic neurons to fit the activation function used by the ANN network, which relieved the constraints of the activation function. Han et al. [34] analyzed the reset mechanism of neurons and refined the threshold balance. Deng et al. [35] analyzed the transformation error by recursively reducing it to a hierarchical sum. Ding et al. [36] proposed a best-fit curve to quantify the error of the conversion process. Rathi et al. [37] combined ANN2SNN with spike-based backpropagation and used the spike time for secondary training based on the transformed SNN to reduce inference time. Bu et al. [14] realized that the transformed SNN suffers from severe performance degradation with shorter time steps. By analyzing and deriving the estimated SNN activation function, a quantization clip-floor-shift activation function is proposed to replace the ReLU activation function in the source ANN. Bu et al. [38] set a trainable clipping upper bound in the ANN and obtained the optimal initial membrane potential based on the derivation results and clipping upper-bound. Wang et al. [15] employed quantization noise to train thresholds in networks with quantized clipping functions and employed BPTT in SNNs to calibrate weights, biases, and initial membrane potential to minimize residual potential. Early switching research focused on how to switch two different types of neurons, and each proposed a switching strategy. As this area of research progresses, most researchers are focusing on how to reduce the conversion loss and achieve accuracy as close to the ANN as possible. At present, researchers are gradually realizing that the long inference time brings a huge obstacle to the application of SNN. They have begun to analyze the error term at low latency and proposed various strategies to eliminate the error, thereby improving the network accuracy of SNN at low latency.

3. Method

In this section, we introduce the activation function and network output of the source ANN, as well as the neuron working mechanism and network output of the target SNN. After analyzing the network output of the target SNN, we obtain the error term produced during the transformation. For this conversion error, an algorithm for initializing the membrane potential is proposed. Furthermore, coding compensation is proposed to eliminate the errors caused by the coding method. Combining coding compensation with initial membrane potential, a neuron with initial membrane potential is obtained.

3.1. Neuron Model

This paper uses ReLU as the activation function in the source neural network. During the conversion, the threshold balancing technique is used to solve the problem of inconsistent input and output ranges of different neurons [27,28]. The meanings of symbols that are used in the following paper are provided in Table 1.

Table 1. The definition of the symbols in this paper.

Symbol	Definition
l	Layer index
a^l	Input in ANN
W^l	Weight in ANN and SNN
b^l	Bias in ANN and SNN
$max(a, 0)$	Activation function in ANN
t	Time step
T	Simulation time step
$x^l(t)$	The spike sequence, consisting of 0 and 1
$v^l_{temp}(t)$	Temporary membrane potential at time t
$v^l(t)$	Membrane potential at time t
V^l_{th}	Threshold
$\theta^l(t)$	The spiking state of neurons at time t
r^l	Spike fire rate

The output of ReLU at l-th layer can be described by:

$$a^{l+1} = max\left(W^l a^l + b^l, 0\right) \tag{1}$$

where $max(x, 0)$ refers to the ReLU activation function. The symbols a^l and a^{l+1} denote the input of the l-th layer and $(l + 1)$-th layer, respectively.

SNN employs Integrate-And-Fire (IF) neurons, which integrate stimuli from presynaptic neurons and fire spikes at postsynaptic neurons [39]. The IF model has a better fit for the output value of the activation function of the ANN than the Leaky Integrate-And-Fire (LIF) model [27]. This neuron can be described as

$$V_{temp}(t) = V(t-1) + X(t) \tag{2}$$

$$V(t) = \begin{cases} V_{temp}(t) - V_{th}, & \text{if } V(t) > V_{th} \\ V_{temp}(t), & \text{otherwise} \end{cases} \tag{3}$$

where $V(t-1)$ is the temporary membrane potential at the previous moment, $X(t)$ is the input at the current moment, and $V_{temp}(t)$ is the temporary membrane potential at time t. The neuron fires and produces a spike when $V_{temp}(t)$ exceeds its threshold V_{th}. Then, $V(t)$ is obtained by performing "soft reset" operations on $V_{temp}(t)$. Otherwise, no spike is fired, and $V(t)$ is equal to $V_{temp}(t)$. Here, "soft reset" can reduce the loss in the conversion process [28,34].

Figure 2 depicts the differences between ANN and SNN concerning neuron models. Unlike the direct input of actual values in ANN, SNN takes spikes as input. After integration, SNN makes the membrane potential change and emits spikes. Then, the emitted spike sequence is used as the output of this layer. The values in the spike sequence are only 0 and 1, where 0 indicates no spikes are fired, and 1 indicates spikes are fired.

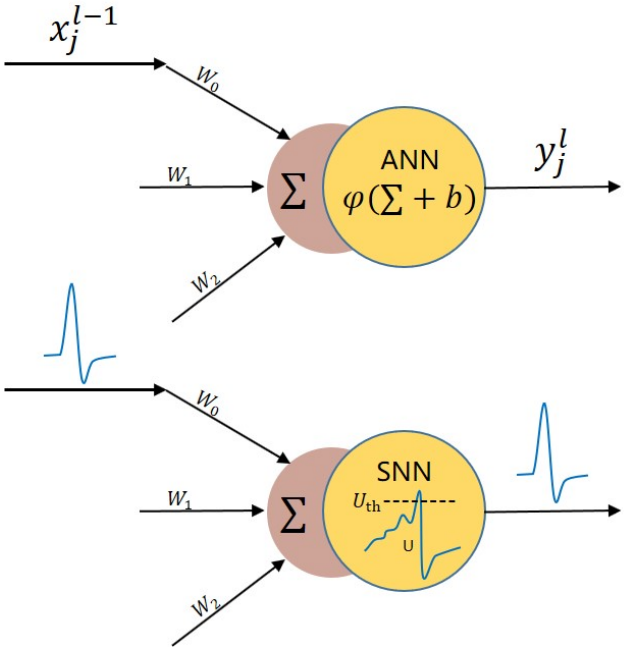


Figure 2. Source artificial neural networks (ANN) and target SNN.

3.2. Error Analysis

From Equations (1) and (2), the change of membrane potential in SNN can be obtained as follows:

$$v_{temp}^l(t) = v^l(t-1) + W^l x^l(t) + b^l \tag{4}$$

where $v_{temp}^l(t)$ is the temporary membrane potential at time t of the l-th layer, which is the sum of the membrane potential at time $t-1$ and the membrane potential received at time t. Based on the spike fire mechanism in Equation (3), the final membrane potential change is described as:

$$v^l(t) = v^l(t-1) + W^l x^l(t) + b^l - \theta^l(t) V_{th}^l \tag{5}$$

The temporary membrane potential needs to be judged to determine whether it emits a spike, and then the membrane potential is reset by a "soft reset". The $\theta^l(t)$ is used to determine whether to emit spikes and is described as:

$$\theta^l(t) = \begin{cases} 1, \text{ if } v_{\text{temp}}^l(t) > V_{th}^l \\ 0, \text{ otherwise} \end{cases} \tag{6}$$

where $\theta(t)$ is a piece-wise function, such that if the value of $v_{temp}^l(t)$ is greater than the threshold V_{th}^l, it is regarded as a spike and has a value of 1, and otherwise the value is 0. $\theta^l(t)$ indicates whether a spike is sent at time t. Therefore, θ^l can be regarded as the output spike sequence of the l-th layer and the input of the $(l+1)-th$ layer.

Based on the explicit iterative Equation (5), the values are given by integrating the input sequence and the spike fire sequence during $t \in (0, T)$. The equation of membrane potential change at time T can be described as follows:

$$v^l(T) = v^l(0) + W^l \sum_{t=1}^{T} x^l(t) + Tb^l - \sum_{t=1}^{T} \theta^l(t) V_{th}^l \tag{7}$$

where $v^l(0)$ is the membrane potential of the l-th layer at time $t = 0$, and $v^l(T)$ is the membrane potential at time $t = T$. Of course, θ^l is the output of the l-th layer and also the input of the $(l+1)$-th layer. Here, the spike fire rate r^l be led into:

$$r^l = \frac{\sum_{t=1}^{T} \theta^l(t)}{T} \tag{8}$$

Since we use the threshold balance mechanism in the conversion process, here is the equation: $x^{l+1}(t) = \theta^l(t) V_{th}^l$. Therefore, the new spike fire rate r^{l-1} can be described as:

$$r^{l-1} = \frac{\sum_{t=1}^{T} x^l(t)}{T V_{th}^{l-1}} \tag{9}$$

Substituting (8) and Equation (9) into Equation (7), we obtain:

$$v^l(T) = v^l(0) + W^l T r^{l-1} V_{th}^{l-1} + T b^l - T r^l V_{th}^l \tag{10}$$

Arranging Equation (10) and extracting TV_{th}^l, we can obtain the following equation:

$$r^l = \frac{W^l r^{l-1} V_{th}^{l-1} + b^l}{V_{th}^l} + \frac{v^l(0) - v^l(T)}{T V_{th}^l} \tag{11}$$

The right-hand side of Equation (11) has two subterms. The first sub-item $\frac{W^l r^{l-1} V_{th}^{l-1} + b^l}{V_{th}^l}$ is the correspondence between the spike firing rate r^l at the l-th layer and the spike firing rate r^{l-1} at the $(l-1)$-th layer. The second sub-term $\frac{v^l(0) - v^l(T)}{T V_{th}^l}$ is the error term obtained during the derivation process.

3.3. Initialize Membrane Potential

From Equation (11), it can be determined that there will be a error term in the process of conversion: $\frac{v^l(0) - v^l(T)}{T V_{th}^l}$. The error term $\frac{v^l(0) - v^l(T)}{T V_{th}^l}$ can be seen as $\frac{v^l(0) - v^l(T)}{V_{th}^l} \frac{1}{T}$, where $\frac{v^l(0) - v^l(T)}{V_{th}^l}$ is strictly limited to (-1, 0), and $\frac{1}{T}$ is used as a scaling factor of $\frac{v^l(0) - v^l(T)}{V_{th}^l}$. The common processing method is to set the initial membrane potential $v^l(0)$ to 0 and discard the other part of $v^l(T)$. Traditional conversion scenarios generally use a larger simulation duration T, such as 1024, 2048, or even 4096. In this case, the scaling factor $\frac{1}{T}$ becomes so small that the entire error term shrinks to be negligible. Therefore, setting $v^l(0)$ to 0 and discarding $v^l(T)$ brings almost no loss of information. Although long simulation time can bring the accuracy closer to the original network, it also leads to the problem of too-long inference time.

Large time steps could become a huge obstacle to the practical application of SNN. When considering small time steps, as T gradually decreases, the scaling factor gradually increases, so that the error term can no longer be ignored. At this time, the common processing method is no longer applicable. The analysis of Equation (11) helps explain why the accuracy of the SNN obtained by the conversion is low in the case of low latency.

In the process of analyzing the conversion error, Bu et al. [38] regarded $v^l(T)$ in it as being caused by the uneven distribution of the output pulse, and pointed out that assuming the potential $v^l(t)$ falls into [0, V_{th}^l] will be able to estimate the activation function of SNNs, ignoring the effect of unevenness error. Unlike Bu, who abandons $v^l(T)$, in this paper, we regard $v^l(T)$ as a non-negligible error, which is the residual membrane potential because the membrane potential does not reach the neuron threshold at t = T. Wang et al. [15] also considers the residual membrane potential, which cannot be ignored, but the overall training steps use secondary training, more training steps may have a certain impact on the training time and training volume. Here, we propose a strategy that only requires forward

propagation on the transformed SNNs to obtain the initial membrane potential, thereby eliminating this part of the error. Unlike the first two, which require various changes and adaptations on ANN, this strategy only relies on SNN. In the face of other types of ANNs, this strategy does not need to consider the adaptability of ANN to the strategy and only focuses on the target SNN.

This paper, the values $v^l(0)$ and $v^l(T)$ are processed together. If the value can be given of $v^l(T)$ and used as the initial membrane potential, these two terms can be canceled simultaneously. From Equation (7), $v^l(T)$ is the membrane potential remaining at the last moment. If all membrane potentials from $t = 0$ to $t = T$ are accumulated and regarded as the dividend, then the threshold of the trigger spike can be regarded as the divisor. Therefore, $v^l(T)$ is equivalent to the remainder obtained by dividing the two. Therefore, a method is proposed to obtain the residual membrane potential $v^l(0)$ based on the trained ANN.

In Algorithm 1, the fire distribution of the spike sequence is more extreme. That is, neurons emit spikes at every moment. In this case, the sum of the membrane potentials of the $t = 1$ to $t = T$ time steps in each neural network layer are approximately the accumulation of the weights for the entire T period. Then, the sum of the obtained membrane potentials is taken as the dividend; the threshold after equilibrium is taken as the divisor. Finally, the remainder is given to obtain the approximate residual membrane potential. This algorithm reduces the conversion loss in the ANN2SNN process and improves the network's accuracy.

Algorithm 1 Initialized membrane potential based on a trained ANN.

Input: ANN that has been trained; Simulation duration T; threshold V_{th}^l that has been threshold-balanced;
Output: Initialized membrane potential *init_mem*;

The threshold is modulated based on the weights in the trained ANN;
$init\_mem = []$
for *l in layers* **do**
    $sum\_mem = ANN.weight[l] * T + ANN.bias[l]$
    $mem = sum\_mem \% V_{th}^l$
    $init\_mem[l] = mem.mean()$
end for
return *init_mem*;

Data often play an essential role in the training process of neural networks. Therefore, the training set is introduced in the algorithm. The initial membrane potential of neurons is obtained through the training set, and the extreme firing of neurons is discarded. Dataset-based initialized membrane potentials are not only not limited by spiked discharge conditions and more versatile, but also perform better in practical applications, allowing the network to achieve higher accuracy.

Algorithm 2 firstly transfers the weight parameters of the trained ANN to the SNN and adopts the threshold balancing operation. Then, the training set is used for training on the SNN. The forwarding propagation process and the record of remaining membrane potential at time $t = T$ in each neural network layer must be paid attention to in the training process. Then, based on the record, the initial membrane potential is approximately given. While this is not a strong guarantee that the network can maintain performance on the test set, the training set should represent the test set. Results show this approach to be highly effective.

Algorithm 2 Initialized membrane potential based on training set.

Input: ANN that has been trained; SNN ready for transformation; Simulation duration T; training datasets;

Output: Initialized membrane potential *init_mem*;

The weights of the trained ANN are converted into the SNN; then the SNN is used to train on the training set, and the value of the remaining membrane potential is recorded.

$init\_mem = []$
$mem\_sum = []$
for l *in layers* **do**
    $SNN.weight[l] = ANN.weight[l]$
    $SNN.bias[l] = ANN.bias[l]$
end for
for *dataset in datasets* **do**
    for t *in* T **do**
        for l *in Layer* **do**
            if $t = T$ **then**
                $mem = l.mem.mean()$
                $mem\_sum[l] = mem\_sum[l] + mem$
            end if
        end for
    end for
end for
for l *in Layer* **do**
    $init\_mem[l] = mem\_sum[l]/len(datasets)$
end for
return *init_mem*;

3.4. Coding Compensation

A very important point in brain research at the moment is how the brain encodes incoming information. SNN can simulate the use of spikes to transmit information between neurons in the brain. However, encoding the original information into a spike sequence is still an ongoing research problem. This is one of the reasons why an SNN is challenging to train.

In the choice of encoding, most researchers will choose rate encoding, which is also the most commonly used encoding method. The rate at which spikes are fired over time can be used to replace the actual values passed into the neural network. In the case of a relatively large time step, the network can achieve almost lossless conversion. If the time step is shortened, the information loss during the conversion process will be greatly increased. In Figure 3, rate-encoded pictures at different time steps are shown. Among them, 2.a is the original image from the dataset CIFAR10, and the image size is 32×32. Furthermore, 2.b, 2.c, and 2.d are pictures encoded with 32-time steps, 128 time steps, and 1024 time steps, respectively. Although the difference between the image obtained under the 1024 time steps encoding and the source image is minimal, there is still a visible difference at low latency.

In Section 3.3, an algorithm is given about the initial membrane potential for SNN. So here, the second half of Equation (11) can be approximately zero. Then, simplify Equation (11), which can be described as:

$$r^l = \frac{W^l r^{l-1} V_{th}^{l-1} + b^l}{V_{th}^l} \tag{12}$$

Moving the threshold of the l-th layer to the left side of the equation, we can obtain

$$r^l V_{th}^l = W^l r^{l-1} V_{th}^{l-1} + b^l \tag{13}$$

where $x^l = r^l V_{th}^l$ is introduced, representing the average membrane potential of the post-synaptic and also the decoding of the spike train. Substituting this into Equation (13), we can obtain:

$$x^l = W^l x^{l-1} + b^l \tag{14}$$

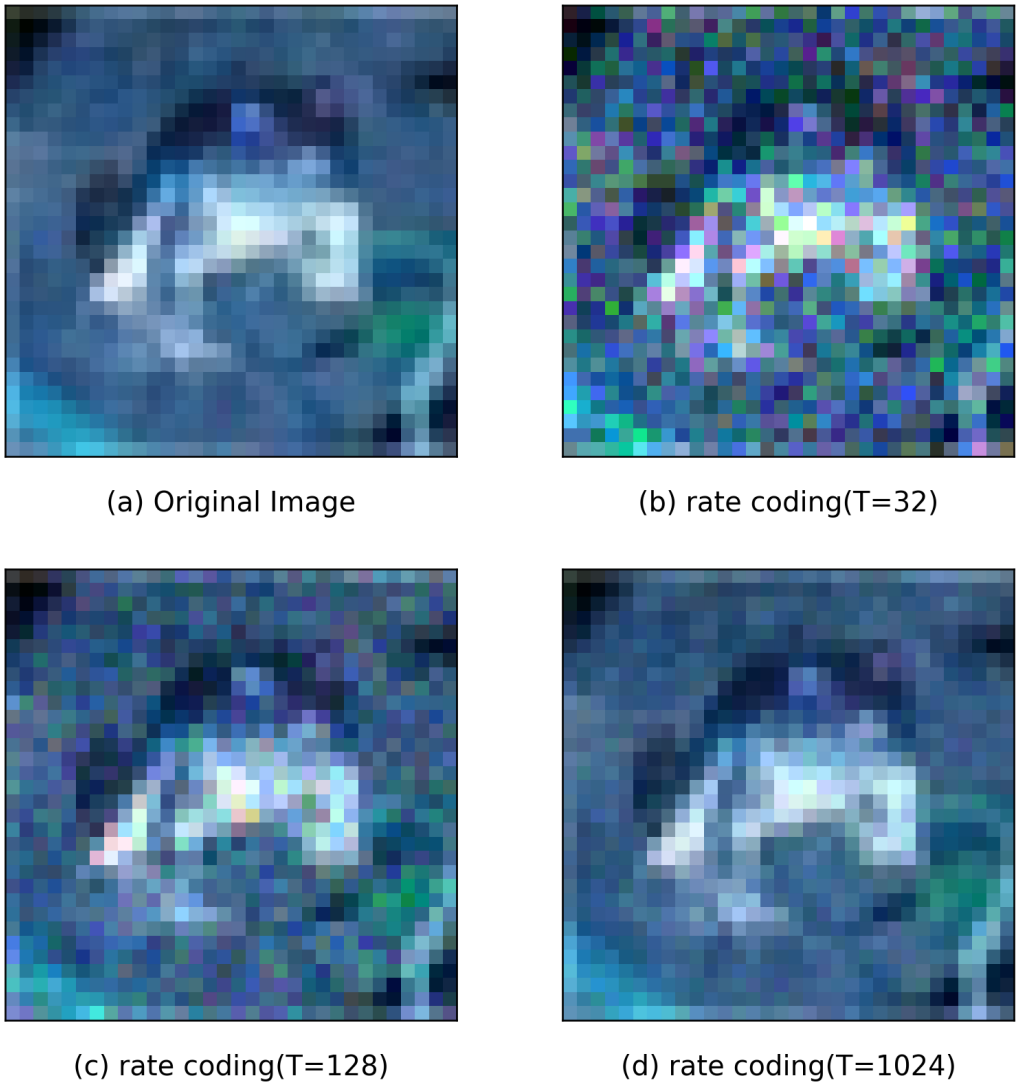


Figure 3. Differences between encoded pictures at different time steps.

In addition, with Equation (1), we can obtain the error of the output of the two neural networks:

$$\begin{aligned} \Delta a^l &= a^l - x^l \\ &= W^l a^{l-1} + b^l - \left(W^l x^{l-1} + b^l \right) \\ &= W^l \left(a^{l-1} - x^{l-1} \right) \end{aligned} \tag{15}$$

From Equation (15), it is known that, from the first layer's encoding, each layer accumulates the information loss generated during the encoding process. Consider the following equation when using rate coding: $a^l \approx \frac{V_{th}^l}{T} \sum_{t=1}^{T} x(t) = x^l$. Some information is discarded during this coding process. Furthermore, unlike traditional networks, SNNs acquire less information about each layer. After multiple layers of accumulation, the information loss caused by encoding will accumulate rapidly.

From the equation $a^l \approx \frac{V_{th}^l}{T} \sum_{t=1}^{T} x(t) = x^l$, it is clear that there are potential information loss items in the process of encoding: $\Delta a^l \in (0, \frac{V_{th}^l}{T})$. Of course, the distribution of this part of the coding error is unknown; if we assume that its distribution is uniform, then the expectation of Δa^l is $\frac{V_{th}^l}{2T}$. Bu et al. [14] also assumes that the distribution of the input z^l is uniform, and the expectation of the conversion error is minimum, and zero when φ is 0.5. The final result is also $\frac{V_{th}^l}{2T}$, which is called a shift operation. The traditional way of processing can be regarded as a lack of firing of a spike. Then, the activation of a spike is increased by adjusting the bias of each layer of the network. Here, combined with the content of Section 3.3, a method for handling this situation is proposed.

The Algorithm 3 uses SNN as the source network, and the SNN does not perform the initializing membrane potential operation. First, the compensation value is used as the initial membrane potential in each layer of the network, and then the network uses the algorithm of Section 3.3 to initialize the membrane potential. Finally, the obtained initialized membrane potential is combined with the compensation value to obtain the final membrane potential. Compared with directly combining the compensation value with the initialized membrane potential, this algorithm can improve the effect of the algorithm in Section 3.3 while performing the compensation operation, thereby improving the overall accuracy of the network.

Algorithm 3 Coding compensation.

Input: SNN before initializing membrane potential; Simulation duration T; Initializing membrane potential function $init\_mem()$;

Output: Initialized membrane potential $final\_mem$;

A compensation value is added to each layer of the SNN, and then the initial membrane potential is obtained by training on the SNN after coding compensation, and finally, the initial membrane potential is added to the coding compensation to obtain the final membrane potential:

$init\_mem = []$
$final\_mem = []$
for l *in layers* **do**
    $SNN.init\_mem[l] = V_{th}^l/2T$
end for
$init\_mem = init\_mem(SNN)$
for l *in layers* **do**
    $final\_mem = init\_mem[l] + V_{th}^l/2T$
end for
return $final\_mem$;

To sum up, the overall architecture of the network is shown in Figure 4. First, the initially trained network is an ANN, and then the weights after training are transferred to the SNN, and the threshold-balancing operation is used in this process. Subsequently, the suitable initial membrane potential is obtained based on the remaining membrane potential training at time T. Considering the error of the data brought about by the encoding process, the encoding compensation is combined with the initial membrane potential to complete the training. A neuron with a suitable initial membrane potential is finally obtained.

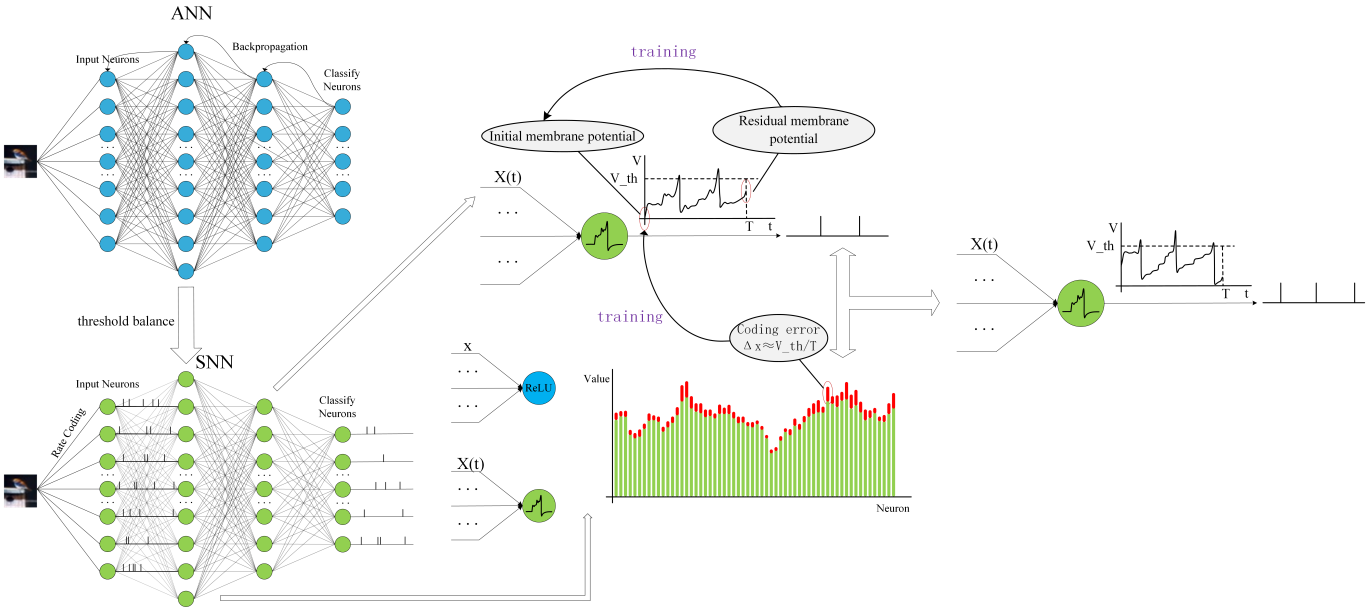


Figure 4. Network structure diagram of the SNN with Initialized Membrane Potential and Coding Compensation (IC-SNN).

4. Experiments

In this section, we use different datasets to conduct multiple experiments and analyze the improved network type in detail from various point of views. We also compare the experimental results with other networks to demonstrate the characteristics of our model and verify the accuracy of the model.

4.1. Experiment Preparation

Before conducting experiments, we must preprocess the dataset and initialize the hyperparameters using related techniques to ensure the best experimental results.

4.1.1. Experiment Environment

Two public image classification datasets, CIFAR10 and CIFAR100, were used in our experiments. For the source ANN for training, VGG16 and ResNet20, which are more common in ANNs, were chosen to be used, with a slight modification [40,41]. Batch normalization and dropout in neural networks were discarded, AvgPool replaced MaxPool, and the ReLU function was used as an activation function. The experiment was run using Pytorch as the deep learning framework, which runs on RTX2060 (12 GB). During the whole conversion process, only the threshold balancing technique was used [27,28].

4.1.2. Pre-Processing

The dataset used for training needed to be preprocessed. CIFAR10 and CIFAR100 were flipped horizontally and randomly cropped to prevent overfitting. The image size was set to 32×32 to fit the input size of the neural network. The final image was normalized to ensure a mean of 0 and a variance of 1.

4.1.3. Hyper-Parameters

The ANN training method generally does not affect the error term of the network conversion, so in our experiments, we adopted a common method to train ANN. During the training of the ANN, the initial learning rate was set to 0.1, the momentum was set to 0.9, the weight decay was set to 5×10^{-4}, and the optimizer we chose was SGD. The batch size was set to 128, and the epoch was set to 300. At epochs 70, 130, 190, and 240, the learning rate decayed by 5. In the ANN-to-SNN conversion threshold balance, the p-quantile was set to [0.9–0.99].

4.2. Validity Verification

We will verify the effectiveness of our improvements from multiple perspectives and analyze the specific reasons for the improvement.

4.2.1. Accuracy

We experimentally verified the two improvement strategies mentioned in Sections 3.3 and 3.4. We used VGG16 as the source neural network and CIFAR10 as the dataset and divided the experiments into five groups: the source neural network (ANN), the non-improved SNN (SNN), the SNN with initialized membrane potential (I-SNN), the SNN with coding compensation(C-SNN), and the SNN with initialized membrane potential and coding compensation(IC-SNN). The results are shown in Figure 5.

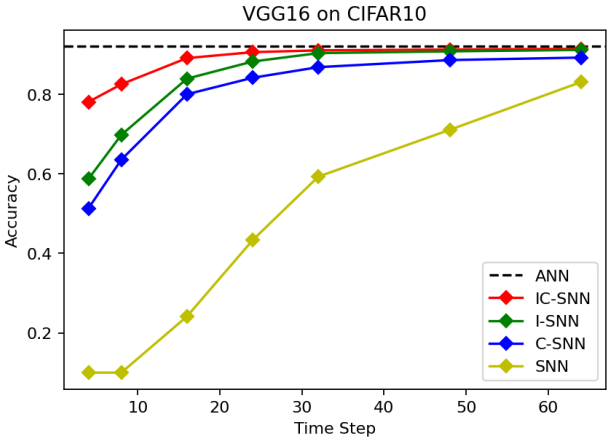


Figure 5. VGG16 on CIFAR10 accuracy.

Compared with the conventional ANN2SNN, the proposed two improved strategies improve the accuracy. The SNN can achieve higher accuracy at lower time steps with both improvements. It can be seen in the figure that the conventional ANN2SNN has inferior performance at low latency, with an accuracy rate of only 20% to 30%. Adopting the improved strategy can easily achieve an accuracy rate of more than 70%. As the time step increases gradually, although the magnitude of the improvement will become progressively smaller, the overall progress will still be ongoing.

4.2.2. Fire Time

The spiking neuron needs to accept the previous layer's output for a long time and accumulates until it reaches the threshold, and then the spike is fired. We analyzed the timing of spiking neurons firing and explored the impact of our improvements on the firing timing. At low time steps, neurons often do not receive enough output from the previous layer to fire spikes and remain inactive, which seriously hinders the firing of spiking neurons. A large amount of information is lost in the transmission process, resulting in a significant drop in the accuracy of the neural network. The two improved strategies in this paper essentially give the neuron an initial membrane potential, which enables the neuron to activate faster and fire spikes to transmit information to the next layer. Then, to a certain extent, this avoids situations where neurons are not activated or under-activated for a sustained period.

Figure 6 shows the time of the first burst of each layer of the IC-SNN after introducing the improved strategy compared to the traditional transformation network SNN. The abscissa is the corresponding network layer number, and the ordinate is the average first firing time of neurons. It can find that after the introduction of the initial membrane potential, the average firing time of neurons in each layer of the IC-SNN is earlier than that in the SNN. Especially in deeper networks, IC-SNN can significantly speed up the time of firing

spikes, which is conducive to transmitting more information and improving the accuracy of the network.

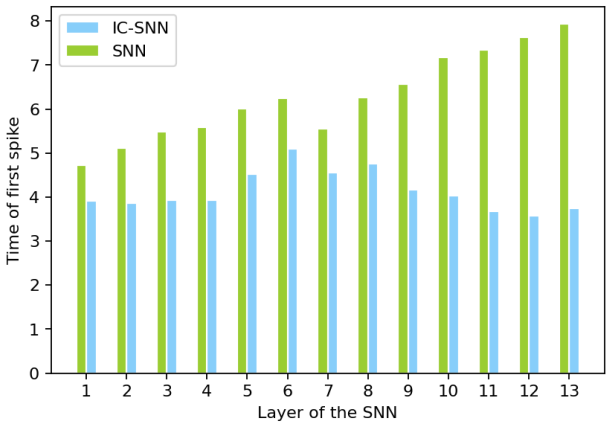


Figure 6. Spike firing time of SNN.

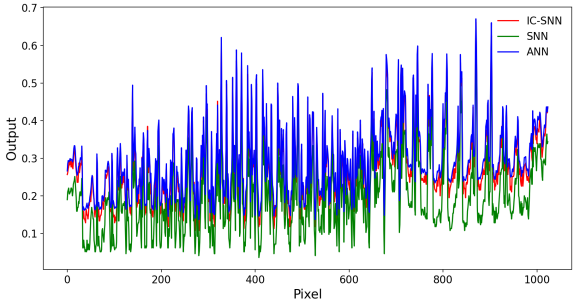
4.2.3. Output

Figure 7 shows the output of the intermediate process of the neural network, which explains to a certain extent why the improved strategy leads to an increase in accuracy. The images on the left are the output of the first layer of the neural network. From top to bottom, the network output is under eight time steps, 16-time steps, and 32-time steps. The images on the right are the output of the third layer network.

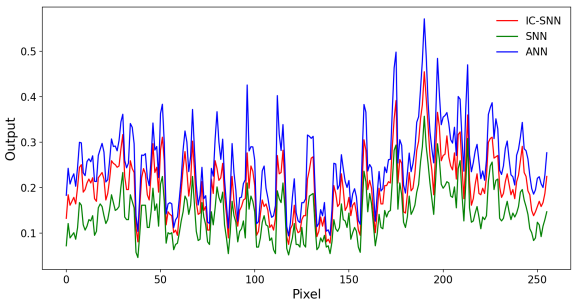
From Figure 7, at 32 time steps, the IC-SNN with the improved strategy almost coincides with the ANN, while there are still some large differences between the conventional SNN and the ANN. This situation is more pronounced at 16 time steps and especially at eight time steps. As the network deepens, conversion errors begin to accumulate. At this point, IC-SNN and ANN begin to separate gradually, but are still closer to the original value than SNN. Therefore, adopting the initialization membrane potential and coding compensation (IC) strategy can effectively reduce the conversion error and make the output of each layer in the SNN closer to the source neural network, thereby improving the accuracy of the SNN.

4.3. Contrast Experiment

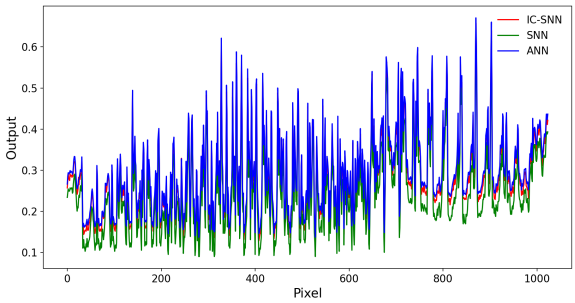
We compare the proposed improvements with other networks, and the specific comparison results are shown in Tables 2–5. This paper only compares the experimental results using the same network model and dataset in the original literature. The first column lists previous research and improvements; ANN_{ACC} in the second column represents the accuracy of the source neural network. Each subsequent column represents the accuracy loss of ANN and the converted SNN in different time steps. We define accuracy loss as $T_t\_\mathcal{L} = ANN_{ACC} - SNN_{ACC}$, where t represents the time step. For example, the column of $T_8\_\mathcal{L}$ represents the value of the accuracy loss when converting ANN to SNN at eight time steps. The smaller value indicates a more effective algorithm. In the comparison item, the table sets the SNN that only uses the initialized membrane potential (I-SNN), the SNN that only uses the coding compensation (C-SNN), and the SNN that uses both the initialized membrane potential and the coding compensation (IC-SNN).



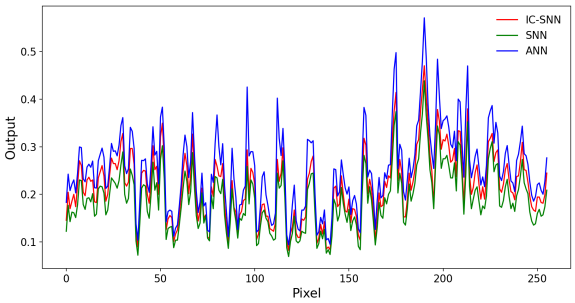
(**a**) 8 time steps at first layer.



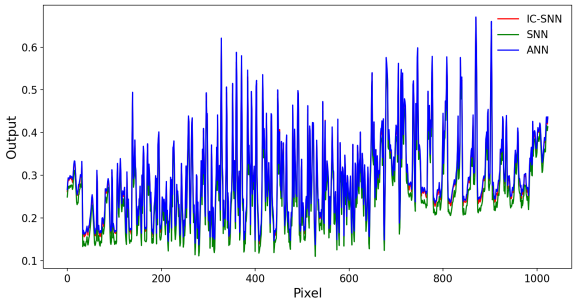
(**b**) 8 time steps at third layer.



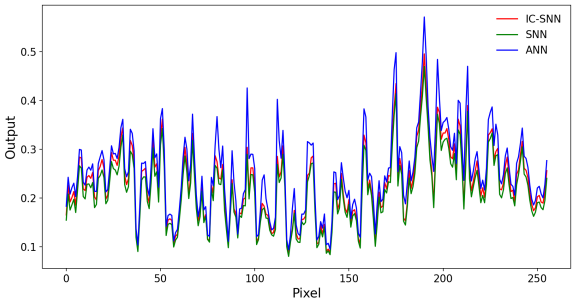
(**c**) 16 time steps at first layer.



(**d**) 16 time steps at third layer.



(**e**) 32 time steps at first layer.



(**f**) 32 time steps at third layer.

Figure 7. The output of the neural network at different time steps. Among them, (**a**,**c**,**e**) are the outputs of the first-layer network using 8, 16, and 32 time steps, respectively; (**b**,**d**,**f**) are the outputs of the third-layer network using 8, 16, and 32 time steps, respectively.

Table 2. Results of the test on the CIFAR10 dataset using VGG16.

Method	ANN_{ACC}	$T_8\_\mathcal{L}$	$T_{16}\_\mathcal{L}$	$T_{32}\_\mathcal{L}$	$T_{64}\_\mathcal{L}$	$T_{128}\_\mathcal{L}$	$T_{256+}\_\mathcal{L}$
Robust Norm [28]	92.82	–	82.71	49.79	11.30	2.02	0.07
Spike Norm [29]	91.70	–	–	–	–	–	0.15
Hybrid Train [37]	92.81	–	–	–	–	1.68	0.33
RMP [34]	93.63	–	–	33.33	3.28	1.22	**0.01**
TSC [32]	93.63	–	–	–	0.84	0.36	0.01
Opt. [35]	**95.72**	–	–	19.48	5.08	1.61	0.01
RNL [36]	92.86	–	34.96	7.46	1.71	0.35	0.01
$I-SNN(Our\ work)$	92.09	22.28	8.14	1.72	0.91	0.75	0.08
$C-SNN(Our\ work)$	92.09	28.52	12.06	5.25	2.84	1.37	0.13
$\boldsymbol{IC-SNN(Our\ work)}$	92.09	**9.95**	**2.97**	**1.03**	**0.61**	**0.29**	0.05

Table 3. Results of the test on the CIFAR100 dataset using VGG16.

Method	ANN_{ACC}	$T_8\_\mathcal{L}$	$T_{16}\_\mathcal{L}$	$T_{32}\_\mathcal{L}$	$T_{64}\_\mathcal{L}$	$T_{128}\_\mathcal{L}$	$T_{256+}\_\mathcal{L}$
Spike Norm [29]	71.22	–	–	–	–	–	0.45
RMP [34]	71.22	–	–	–	–	7.46	0.29
TSC [32]	71.22	–	–	–	–	1.36	0.25
Opt. [35]	**77.89**	–	–	70.25	50.05	16.85	**0.18**
$I-SNN(Our\ work)$	70.59	33.79	10.22	4.03	1.37	0.83	0.31
$C-SNN(Our\ work)$	70.59	47.38	20.42	8.31	2.85	1.89	0.41
$\boldsymbol{IC-SNN(Our\ work)}$	70.59	**15.72**	**7.11**	**3.74**	**1.24**	**0.77**	0.20

As shown in Tables 2 and 3, we used the VGG16 model and the CIFAR10 and CIFAR100 datasets to compare with other researchers' experiments. In the CIFAR10 dataset, IC-SNN achieved more than 80% accuracy at eight time steps, whereas other networks require about 32 time steps to achieve similar accuracy. At 16 time steps, the network achieves close to 90% accuracy, but other networks require at least 64 time steps. The results of VGG16 running on the CIFAR100 dataset further demonstrate the effectiveness of our improvements. The accuracy rate has exceeded 60% at 16-time steps, reaching the mark that other networks require 64 time steps. The conversion error loss is reduced to 1.24% at 64-time steps and to within 1% at 128-time steps.

Thanks to the proposed initializing membrane potential strategy and coding compensation strategy, our network has fewer conversion errors than other networks at low time steps, especially at eight time steps and 16 time steps. From the error analysis in Section 3.2, it can be found that the error equation we want to eliminate is negatively related to the time step. Therefore, the lower the time step, the more noticeable our improvements. In the process of gradually increasing the time step, the performance improvement of our network starts to decrease slowly. However, it can still achieve relatively high accuracy at the time step of 128.

We also used the ResNet20 model to conduct experiments on CIFAR10 and CIFAR20. The structure of ResNet is more complicated than that of VGG. Here, we adopt the conventional processing for the residual block, regarding the residual block as a whole layer, and provide a corresponding balance threshold for one residual block. Similarly, when initializing membrane potential and coding compensation, we adopt a similar method and only introduce the related improvement strategy for the first layer in the residual block.

Tables 4 and 5 show the comparative results of our networks. Even on the more complex ResNet, our improvements still lead to huge improvements. IC-SNN achieves 80% accuracy on CIFAR10 at 16 time steps, and the error of the network transformation shrank to 1% at 128 time steps. The performance on CIFAR100 is also excellent. At 32 time steps, the accuracy error with the source network was reduced to less than 10%, and the error was only 1.77% at 128 time steps.

Table 4. Results of the test on the CIFAR10 dataset using ResNet20.

Method	ANN_{ACC}	$T_8\_\mathcal{L}$	$T_{16}\_\mathcal{L}$	$T_{32}\_\mathcal{L}$	$T_{64}\_\mathcal{L}$	$T_{128}\_\mathcal{L}$	$T_{256+}\_\mathcal{L}$
Spike Norm [29]	89.10	–	–	–	–	–	1.64
Hybrid Train [37]	**93.15**	–	–	–	–	–	0.21
RMP [34]	91.47	–	–	–	–	3.87	0.11
TSC [32]	91.47	–	–	–	22.09	2.90	**0.05**
$I-SNN(Our\ work)$	91.46	43.21	21.75	4.72	3.87	1.06	0.21
$C-SNN(Our\ work)$	91.46	57.18	30.39	8.25	5.34	2.39	0.27
$\boldsymbol{IC-SNN(Our\ work)}$	91.46	**37.31**	**10.23**	**3.64**	**2.14**	**0.81**	0.09

Table 5. Results of the test on the CIFAR100 dataset using ResNet20.

Method	ANN_{ACC}	$T_8\_\mathcal{L}$	$T_{16}\_\mathcal{L}$	$T_{32}\_\mathcal{L}$	$T_{64}\_\mathcal{L}$	$T_{128}\_\mathcal{L}$	$T_{256+}\_\mathcal{L}$
Spike Norm [29]	69.72	–	–	–	–	–	5.63
RMP [34]	68.72	–	–	41.08	21.89	11.03	0.90
TSC [32]	**71.22**	–	–	–	–	10.30	0.54
$I-SNN(Our\ work)$	66.05	39.68	22.41	15.77	4.62	2.03	0.59
$C-SNN(Our\ work)$	66.05	46.71	27.19	21.03	7.93	3.31	0.63
$\boldsymbol{IC-SNN(Our\ work)}$	66.05	**35.59**	**17.70**	**9.28**	**3.13**	**1.61**	**0.45**

During the experiment, we tried 16 time steps and even eight time steps, which hardly anyone other than us has used. This is because, in such a short time step, the firing of spikes is seriously affected, which will cause a large amount of information loss, resulting in a significant drop in the accuracy of the neural network. After the improved strategy was introduced, our network still showed excellent performance at such a low time step. It also confirms that our improvement can effectively speed up the firing of spikes, thereby reducing inference time.

5. Conclusions

This paper analyzes errors in the conversion process to help explain the loss of information at low latency. For the case of low delay, a method to initialize the membrane potential was proposed to eliminate the error. Weight-based initialization takes into account more intense spike sequences; data-based initialization is more general and better improves the accuracy of the network. In further analysis after removing errors, a rate-coding-based compensation method was proposed, which, combined with the initialized membrane potential, enabled the network to perform better. This paper conducts experiments on the IC-SNN on different network models and datasets. The results show higher accuracy and minor transformation error at low latency.

In future work, we will focus on the source neural network to improve its performance using batch normalization and dropout techniques. In addition, we will also try to combine biological mechanisms, which further enhance the accuracy at low time steps.

Author Contributions: Conceptualization, C.L. and Z.S.; formal analysis, Z.S., L.S. and W.G.; methodology, Z.S.; project administration, C.L. and S.Z.; validation, Z.S. and W.G.; visualization, L.S. and W.G.; writing—original draft, C.L.; writing—review and editing, Z.S., L.S., W.G. and S.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Key Technologies R&D Program (2020YFB1712401, 2018YFB1701400, 2018*****4402), the 2020 Key Project of Public Benefit in Henan Province of China (201300210500), and the Nature Science Foundation of China (62006210, 62206252), Key scientific research projects of colleges and universities in Henan Province (23A520015).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Guo, Y.; Yao, A.; Chen, Y. Dynamic network surgery for efficient dnns. *arXiv* **2016**, arXiv:1608.04493
2. Gong, Y.; Liu, L.; Yang, M.; Bourdev, L. Compressing deep convolutional networks using vector quantization. *arXiv* **2014**, arXiv:1412.6115.
3. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv* **2015**, arXiv:1510.00149.
4. Gerstner, W.; Kistler, W.M. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*; Cambridge University Press: Cambridge, UK, 2002. [CrossRef]

5. Attwell, D.; Laughlin, S.B. An energy budget for signaling in the grey matter of the brain. *J. Cereb. Blood Flow Metab.* **2001**, *21*, 1133–1145. [CrossRef]
6. Maass, W.; Markram, H. On the computational power of circuits of spiking neurons. *J. Comput. Syst. Sci.* **2004**, *69*, 593–616. [CrossRef]
7. Song, S.; Miller, K.D.; Abbott, L.F. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. Neurosci.* **2000**, *3*, 919–926. [CrossRef]
8. Hebb, D.O. *The Organization of Behavior: A Neuropsychological Theory*; Psychology Press: New York, NY, USA, 2005. [CrossRef]
9. Zhang, T.; Cheng, X.; Jia, S.; Poo, M.m.; Zeng, Y.; Xu, B. Self-backpropagation of synaptic modifications elevates the efficiency of spiking and artificial neural networks. *Sci. Adv.* **2021**, *7*, eabh0146. [CrossRef]
10. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
11. Bohte, S.M.; Kok, J.N.; La Poutre, H. Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* **2002**, *48*, 17–37. [CrossRef]
12. Wu, Y.; Deng, L.; Li, G.; Zhu, J.; Shi, L. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Front. Neurosci.* **2018**, *12*, 331. [CrossRef]
13. Cao, Y.; Chen, Y.; Khosla, D. Spiking deep convolutional neural networks for energy-efficient object recognition. *Int. J. Comput. Vis.* **2015**, *113*, 54–66. [CrossRef]
14. Bu, T.; Fang, W.; Ding, J.; Dai, P.; Yu, Z.; Huang, T. Optimal ANN-SNN Conversion for High-accuracy and Ultra-low-latency Spiking Neural Networks. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 30 April–3 May 2021.
15. Wang, Z.; Lian, S.; Zhang, Y.; Cui, X.; Yan, R.; Tang, H. Towards Lossless ANN-SNN Conversion under Ultra-Low Latency with Dual-Phase Optimization. *arXiv* **2022**, arXiv:2205.07473.
16. Masquelier, T.; Thorpe, S.J. Unsupervised learning of visual features through spike timing dependent plasticity. *PLoS Comput. Biol.* **2007**, *3*, e31. [CrossRef] [PubMed]
17. Legenstein, R.; Pecevski, D.; Maass, W. A learning theory for reward-modulated spike-timing-dependent plasticity with application to biofeedback. *PLoS Comput. Biol.* **2008**, *4*, e1000180. [CrossRef]
18. Ruf, B.; Schmitt, M. Learning temporally encoded patterns in networks of spiking neurons. *Neural Process. Lett.* **1997**, *5*, 9–18. [CrossRef]
19. Wade, J.J.; McDaid, L.J.; Santos, J.A.; Sayers, H.M. SWAT: An unsupervised SNN training algorithm for classification problems. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–6 June 2008; pp. 2648–2655. [CrossRef]
20. Diehl, P.U.; Cook, M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.* **2015**, *9*, 99. [CrossRef]
21. Tavanaei, A.; Maida, A.S. Bio-inspired spiking convolutional neural network using layer-wise sparse coding and STDP learning. *arXiv* **2016**, arXiv:1611.03000.
22. Zhang, T.; Zeng, Y.; Zhao, D.; Xu, B. Brain-inspired Balanced Tuning for Spiking Neural Networks. In Proceedings of the IJCAI, Stockholm, Swede, 13–19 July 2018; pp. 1653–1659.
23. Wu, Y.; Deng, L.; Li, G.; Zhu, J.; Xie, Y.; Shi, L. Direct training for spiking neural networks: Faster, larger, better. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 8–12 October 2019; Volume 33, pp. 1311–1318. [CrossRef]
24. Zhang, W.; Li, P. Temporal spike sequence learning via backpropagation for deep spiking neural networks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 12022–12033. [CrossRef]
25. Fang, W.; Yu, Z.; Chen, Y.; Masquelier, T.; Huang, T.; Tian, Y. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 10–17 October 2021; pp. 2661–2671. [CrossRef]
26. Xiao, M.; Meng, Q.; Zhang, Z.; Wang, Y.; Lin, Z. Training feedback spiking neural networks by implicit differentiation on the equilibrium state. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 14516–14528.
27. Diehl, P.U.; Neil, D.; Binas, J.; Cook, M.; Liu, S.C.; Pfeiffer, M. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, 12–17 July 2015; pp. 1–8. [CrossRef]
28. Rueckauer, B.; Lungu, I.A.; Hu, Y.; Pfeiffer, M.; Liu, S.C. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci.* **2017**, *11*, 682. [CrossRef]
29. Sengupta, A.; Ye, Y.; Wang, R.; Liu, C.; Roy, K. Going deeper in spiking neural networks: VGG and residual architectures. *Front. Neurosci.* **2019**, *13*, 95. [CrossRef] [PubMed]
30. Kim, S.; Park, S.; Na, B.; Yoon, S. Spiking-yolo: Spiking neural network for energy-efficient object detection. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 11270–11277. [CrossRef]
31. Park, S.; Kim, S.; Na, B.; Yoon, S. T2FSNN: Deep spiking neural networks with time-to-first-spike coding. In Proceedings of the 2020 57th ACM/IEEE Design Automation Conference (DAC), Virtual Event, 20–24 July 2020; pp. 1–6. [CrossRef]
32. Han, B.; Roy, K. Deep spiking neural network: Energy efficiency through time based coding. In *Proceedings of the European Conference on Computer Vision*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 388–404. [CrossRef]

33. Stöckl, C.; Maass, W. Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes. *Nat. Mach. Intell.* **2021**, *3*, 230–238. [CrossRef]
34. Han, B.; Srinivasan, G.; Roy, K. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 13558–13567.
35. Deng, S.; Gu, S. Optimal conversion of conventional artificial neural networks to spiking neural networks. *arXiv* **2021**, arXiv:2103.00476.
36. Ding, J.; Yu, Z.; Tian, Y.; Huang, T. Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks. *arXiv* **2021**, arXiv:2105.11654.
37. Rathi, N.; Srinivasan, G.; Panda, P.; Roy, K. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. *arXiv* **2020**, arXiv:2005.01807.
38. Bu, T.; Ding, J.; Yu, Z.; Huang, T. Optimized Potential Initialization for Low-latency Spiking Neural Networks. *arXiv* **2022**, arXiv:2202.01440.
39. Delorme, A.; Gautrais, J.; Van Rullen, R.; Thorpe, S. SpikeNET: A simulator for modeling large networks of integrate and fire neurons. *Neurocomputing* **1999**, *26*, 989–996. [CrossRef]
40. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
41. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]