# CANONIC SIGNED SPIKE CODING FOR EFFICIENT SPIKING NEURAL NETWORKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Spiking Neural Networks (SNNs) seek to mimic the spiking behavior of biological neurons and are expected to play a key role in the advancement of neural computing and artificial intelligence. The conversion of Artificial Neural Networks (ANNs) to SNNs is the most widely used training method, which ensures that the resulting SNNs perform comparably to ANNs on large-scale datasets. The efficiency of these conversion-based SNNs is often determined by the neural coding schemes. Current schemes typically use spike count or timing for encoding, which is linearly related to ANN activations and increases the required number of time steps. To address this limitation, we propose a novel Canonic Signed Spike (CSS) coding scheme. This method incorporates non-linearity into the encoding process by weighting spikes at each step of neural computation, thereby increasing the information encoded in spikes. We identify the temporal coupling phenomenon arising from weighted spikes and introduce negative spikes along with a Ternary Self-Amplifying (TSA) neuron model to mitigate the issue. A one-step silent period is implemented during neural computation, achieving high accuracy with low latency. We apply the proposed methods to directly convert full-precision ANNs and evaluate performance on CIFAR-10 and ImageNet datasets. Our experimental results demonstrate that the CSS coding scheme effectively compresses time steps for coding and reduces inference latency with minimal conversion loss.

## 1 INTRODUCTION

Spiking Neural Networks (SNNs), recognized as the third generation of neural network models, are inspired by the biological structure and functionality of the brain (Wang et al., 2020). Unlike traditional Artificial Neural Networks (ANNs), which rely on continuous activation functions, SNNs utilize discrete spiking events. This enables SNNs to capture temporal dynamics and process information in a manner that closely resembles brain activity (Taherkhani et al., 2020). The event-driven nature of SNNs aligns with the brain's energy-efficient computational paradigm, offering potential for more efficient and low-power computing systems (Yamazaki et al., 2022).

The two primary learning algorithms for SNNs are gradient-based optimization and ANN-to-SNN conversion. Directly training using supervised backpropagation is challenging due to the non-differentiable nature of spike generation (Lee et al., 2020; 2016). The conversion-based method, however, offers a practical approach to overcome this difficulty and has produced the best-performing SNNs (Deng & Gu, 2021; Bu et al., 2022; Ding et al., 2021).

Encoding the ANN activations into spike trains is a prerequisite for successful ANN-to-SNN conversion. Various coding schemes, such as rate coding and temporal coding, have been proposed to describe neural activity (Guo et al., 2021). Rate coding maps the number of spikes to the corresponding ANN activation (Cao et al., 2015). In contrast, temporal coding focuses on the precise timing or patterns of spikes (Rueckauer & Liu, 2018; Kim et al., 2018; Han & Roy, 2020). For example, Time-to-First-Spike (TTFS) coding maps the the activation value to the time elapsed before the first spike (Stanojevic et al., 2022).

However, both the spike counts in rate coding and the spike timing in TTFS coding are linearly related to the encoded activation. This necessitates a large number of time steps to provide sufficient encoding granularity (Stanojevic et al., 2023; Meng et al., 2022). These methods employ basic Integrate-and-Fire (IF) neurons for neural computation, where the membrane potential accumulation

also follows a linear trend over time. This further contributes to an increased inference latency. Recent works have proposed alleviating these problems by quantizing the ANN activations before conversion (Hu et al., 2023; Bu et al., 2023; Hao et al., 2023). This approach simplifies the encoding process but introduces additional quantizing and training overhead. Our goal is to develop a novel encoding method that can directly convert full-precision ANNs while reducing the number of time steps required.

In the study of the temporal information dynamics of spikes, Kim et al. (2022) discovered a phenomenon of temporal information concentration in SNNs. It is found that after training, information becomes highly concentrated in the first few time steps. This observation led us to hypothesize that the spikes at earlier time steps carry more information and contribute more to the membrane potential. Consequently, we propose a mechanism whereby the neurons amplify its membrane potential with a specific coefficient before processing the subsequent input. As a result, spikes are assigned distinct, predetermined weights and are therefore referred to as canonical. This mechanism breaks the two linear relationships mentioned above and effectively compresses the time steps.

We observed that weighted spikes are prone to temporal coupling during neural computation, which can lead to large encoding errors. This issue can be alleviated by controlling the residual membrane potential within an acceptable range. However, due to the amplification of the membrane potential at each step, the traditional soft-reset method (Han et al., 2020), which reduces the membrane potential by an amount equal to the firing threshold, becomes insufficient. To address this issue, we introduced negative spikes into our coding scheme, which is thus termed as the Canonic Signed Spike (CSS) coding scheme. By leveraging negative spikes to correct the excessively fired information, we allow neurons to reduce their membrane potential by an amount greater than the threshold. This results in a neuron model fires both positive and negative spikes, and is termed as the Ternary Self-Amplifying (TSA) neuron. To better balance the trade-off between coding time steps and inference latency in CSS coding, we introduce a one-step silent period into the TSA neuron, which enables the resulting SNN to achieve improved performance and efficiency.

The main contributions of this paper can be summarized as follows:

- By assigning weights to the spikes, we introduce non-linearity into the coding process and compress the time steps to a logarithmic scale. Neurons amplify the membrane potential at each time step, thereby obtaining more information from the preceding spikes.
- We find that weighted spikes are prone to temporal coupling during neural computation, presenting the biggest challenge when incorporating non-linearity in spike coding. We analyze the underlying reasons and introduces negative spikes along with a corresponding TSA neuron model to address this issue.
- We demonstrate the effectiveness of the CSS coding scheme on CIFAR-10 and ImageNet datasets. The results show that the proposed method effectively reduces the required coding time steps and inference latency. Even in very deep networks, CSS coding scheme can still achieve satisfactory performance with a minimal number of time steps (e.g., 16 time steps for ResNet-50 on ImageNet).

## 2 Related Work

Currently, the mainstream coding schemes in converted SNNs are rate coding and TTFS coding. Rate coding represents different activities with the number of spikes emitted within a specific time window. Due to its simplicity, rate coding is commonly used in deep learning of SNNs. Initially, research focused on reducing conversion loss, leading to methods such as weight normalization (Diehl et al., 2015), threshold rescaling (Sengupta et al., 2019), and soft-reset neuron models (Han et al., 2020). More recent work has shifted towards reducing the number of time steps by optimizing neuron parameters. Meng et al. (2022) introduced the Threshold Tuning and Residual Block Restructuring (TTRBR) method to minimize conversion error in ResNet architectures with fewer time steps. Bu et al. (2022) proposed optimizing the initial membrane potential to reduce conversion loss when using a small number of time steps. Despite these optimizations, deep networks or large datasets still require hundreds of time steps to achieve satisfactory results.

Due to the functional similarity to the biological neural network, SNNs can embrace the sparsity found in biology and are highly compatible with temporal coding. Rueckauer & Liu (2018) were

the first to attempt converting an ANN to a TTFS-based SNN. While this coding method significantly increased sparsity by limiting each neuron to fire at most one spike, they observed large conversion errors, even on MNIST dataset. Stanojevic et al. (2022) demonstrated that accurate mapping from ANN to TTFS-based SNN is feasible but did not specify the number of time steps required. In their subsequent work (Stanojevic et al., 2023), they quantized the spike timing, showing that each layer needed around 64 time steps for accurate conversion. Yang et al. (2023) proposed a TTFS-based conversion algorithm with dynamic neuron threshold and weight regularization. They completed the conversion with 50 time steps per layer. Despite the reduction in the number of time steps per layer, TTFS coding still suffered from high output latency in deep networks for its layerwise processing manner. Han & Roy (2020) introduced the Temporal-Switch-Coding (TSC) scheme, where each input pixel is represented by two spikes, and the time interval between them encodes pixel intensity. However, as this time interval remains linearly related to activation, the issue of long latency persists.

Some recent works have also incorporated non-linearity into the coding process. Stöckl & Maass (2021) and Rueckauer & Liu (2021) used spikes to encode the "1"s in the binary representations of ANN activations. However, both works did not address the temporal coupling issue caused by weighted spikes. Instead, they adopted an approach similar to TTFS coding, where neurons must wait for the arrival of all input spikes before firing. Specifically, non-overlapping time windows are set for each layer, and spikes are only allowed to be emitted during these time windows. In contrast, our approach facilitates the greatest extent of synchronous neural computation, thereby reducing both the coding time steps and output latency.

# 3 Preliminaries

## 3.1 Spiking Neurons

Spiking neurons communicate through spike trains and are interconnected via synaptic weights. Each incoming spike contributes to the postsynaptic neuron's membrane potential, and a spike is generated when the potential reaches a predefined threshold. Generally, a spike sequence $S_i^l[t]$ in the SNN can be expressed as follows:

$$S_i^l[t] = \sum_{t_i^{l,f} \in \mathbb{F}_i^l} \theta^l \delta[t - t_i^{l,f}] \tag{1}$$

where $i$ is the neuron index, $l$ is the layer index, $\theta^l$ is the spike amplitude, $\delta[\cdot]$ denotes an unit impulse[1], $f$ is the spike index, and $\mathbb{F}_i^l$ denotes a set of spike times which satisfies the firing condition:

$$t_i^{l,f} : o_i^l[t_i^{l,f}] \geq \theta^l \tag{2}$$

where $o_i^l[t]$ denotes the membrane potential before firing a spike. Conversion-based works often employ soft-reset IF neuron model, where an amount equal to the spike amplitude is subtracted from the membrane potential for reset. Specifically, its dynamics can be expressed as follows:

$$u_i^l[t] = u_i^l[t-1] + z_i^l[t] - S_i^l[t] \tag{3}$$

where $u_i^l[t]$ denotes the membrane potential after firing a spike and $z_i^l[t]$ denotes the integrated inputs:

$$z_i^l[t] = \sum_j w_{ij}^l S_j^{l-1}[t] + b_i^l \tag{4}$$

where $w_{ij}^l$ is the synaptic weight and $b_i^l$ is the bias. For clarity, definitions of the common symbols are provided in Table 1.

## 3.2 ANN-SNN Conversion

The ANN-SNN conversion typically involves the following two key steps: 1) selecting an appropriate encoding method to represent ANN activations as spike trains, and 2) adopting a suitable neuron

[1] $\delta[t]$ takes the value 1 at $t = 0$, and 0 otherwise

Table 1: Common symbols in this paper.

| Symbol | Definition | Symbol | Definition |
|---|---|---|---|
| $l$ | Layer index | $\beta$ | Amplification factor |
| $i, j$ | Neuron index | $w_{ij}^l$ | SNN weight |
| $S_i^l[t]$ | Spike sequence | $\hat{w}_{ij}^l$ | ANN weight |
| $o_i^l[t]$ | Membrane potential before firing | $b_i^l$ | SNN bias |
| $u_i^l[t]$ | Membrane potential after firing | $\hat{b}_i^l$ | ANN bias |
| $z_i^l[t]$ | Integrated inputs (PSP)[1] | $T$ | Time steps for coding |
| $\theta^l$ | Spike amplitude | $\tilde{\theta}^l$ | Initial spike amplitude |

[1] Postsynaptic potential

model that ensures the generated spike trains accurately encode the outputs of the corresponding ANN neurons.

The most widely used and state-of-the-art approaches employ (signed) soft-reset IF neurons and interprets their output through spike rates (i.e. rate coding). Let $T$ denote the number of time steps, with the initial condition $u_i^l[0] = 0$, we can iteratively update the membrane potential using Eq. (3) until $t = T$. Then substitute $z_i^l[t]$ with Eq. (4), and we can write:

$$\frac{\sum_{t=1}^{T} S_i^l[t]}{T} = \sum_j w_{ij}^l \frac{\sum_{t=1}^{T} S_j^{l-1}[t]}{T} + \sum_{t=1}^{T} \frac{b_i^l}{T} - \frac{u_i^l[T]}{T} \tag{5}$$

See Appendix A.1 for a detailed derivation. Note that both sides of the equation are divided by $T$ to better highlight the interpretation of $\sum_{t=1}^{T} S_i^l[t]/T$ as a "rate". It defines the relationship between neuron's input rate and output rate and can be directly related to the forward pass in a ReLU-activated ANN:

$$a_i^l = \max\left(\sum_j \hat{w}_{ij}^l a_j^{l-1} + \hat{b}_i^l, 0\right) \tag{6}$$

where $a_i^l$ denotes the ANN activation, $\hat{w}_{ij}^l$ and $\hat{b}_i^l$ denote the weight and bias, respectively. Note that in Eq. (5) we have: 1) $\sum S[t]/T > 0$, 2) $u_i^l[T]/T$ becomes negligible as $T$ increases. These observations suggest that mapping ANN activations to SNN spike rates can be achieved by simply using the scaled ANN weights[2] and bias.

However, with fewer time steps, the number of activations that can be encoded by $\sum S_i^l[t]/T$ becomes significantly limited, while the perturbation introduced by $u_i^l[T]/T$ increases. These factors together result in a rapid increase in conversion loss. For any other encoding scheme that relies on a quantity linearly related to the time steps, this issue will theoretically persist. Therefore, our goal is to incorporate nonlinearity into the encoding process to enhance the expressiveness of spike trains. Recent rate-coding-based works have proposed quantizing the ANN before conversion. This approach directly reduces the number of activations that need to be mapped, providing an alternative way to minimize time steps. Notably, this approach is complementary to ours. The proposed encoding scheme can also convert quantized ANNs and further reduce the required number of time steps.

# 4 Methods

## 4.1 Assigning Weights to Spikes

Based on our motivation, spikes that occur earlier in the sequence carry more information and should have a greater impact on the membrane potential. To achieve this, we introduce an amplification factor $\beta > 1$ into the soft-reset IF model:

$$u_i^l[t] = \beta u_i^l[t-1] + z_i^l[t] - S_i^l[t] \tag{7}$$

[2] In SNNs, $\theta^l$ is finally normalized to 1 for simplicity of implementation, which is achieved by absorbing it into the synaptic weights. Consequently, the ANN weights still need to be scaled by a certain factor. Note that $\theta^l$ is typically determined based on the number of time steps and the range of ANN activations in layer $l$.

Following the same derivation as in Eq. (5), we can write:

$$u_i^l[T] = \sum_j w_{ij}^l \sum_{t=1}^{T} \beta^{T-t} S_j^{l-1}[t] + \sum_{t=1}^{T} \beta^{T-t} b_i^l - \sum_{t=1}^{T} \beta^{T-t} S_i^l[t] \tag{8}$$

The detailed derivation can be found in Appendix A.1. As expected, the input at time $t_i^{l,f}$ raises the membrane potential by $\theta^l \beta^{T-t_i^{l,f}}$. In other words, the value encoded by the spike is weighted by $\beta^{T-t_i^{l,f}}$, enabling the use of $\sum_{t=1}^{T} \beta^{T-t} S_i^l[t]$ rather than $\sum_{t=1}^{T} S_i^l[t]$ to map the ANN activations. The spike at time $T$ still encodes $\theta^l$, which is the minimum value a spike can represent and determines the granularity of encoding.

**Definition 1.** Since the number of time steps is finite, the quantization error resulting from limited coding granularity is unavoidable. Let $v$ denote the target value. The encoding is considered successful, denoted as $S_i^l[t] \sim v$, as long as $\left|\sum_{t=1}^{T} \beta^{T-t} S_i^l[t] - v\right| < \theta^l$.

According to Eq. (8), our method can theoretically encode the same number of activations as linear encoding method while log-compressing the number of required time steps. Meanwhile, Eq. (8) serves as the core equation for ANN-SNN conversion. By comparing Eq. (8) with Eq. (6), and noting that $\sum_t \beta^{T-t} S_i^l[t] \geq 0$, we can conclude:

**Observation 1.** Let $S_j^{l-1}[t] \sim a_j^{l-1}$, and set $w_{ij}^l = \hat{w}_{ij}^l$ and $b_i^l = \hat{b}_i^l / \sum_t \beta^{T-t}$, respectively. To reduce encoding errors in layer $l$, the residual membrane potential $u_i^l[T]$ should be minimized.

However, we find that after spike weighting, $u_i^l[T]$ can become vary large (e.g., far greater than $\theta^l$). Before analyzing and solving this issue, we first present the conditions that must be satisfied to limit the value of $u_i^l[T]$:

**Lemma 1.** *Making $u_i^l[T] < \epsilon$ is equivalent to satisfying the following equation:*

$$\forall t_0 \in \{1, 2, \cdots, T\},$$
$$\beta^{T-t_0+1} u_i^l[t_0 - 1] + \sum_j w_{ij}^l \sum_{t=t_0}^{T} \beta^{T-t} S_j^{l-1}[t] + \sum_{t=t_0}^{T} \beta^{T-t} b_i^l < \epsilon + \sum_{t=t_0}^{T} \theta^l \beta^{T-t} \tag{9}$$

The second term on the right-hand side of Eq. (9) represents the maximum value a spike train after $t_0$ can encode. This imposes constraints on both the subsequent input to the neuron and the residual membrane potential carried over from earlier computations (the left-hand side of the equation). Lemma 1 provides the mathematical foundation for the next section, with its detailed derivation available in Appendix A.2.

## 4.2 Incorporating Negative Spikes

### 4.2.1 Temporal Decoupling

Rueckauer et al. (2017) reported that large activation values in ANNs are rare, with most values concentrated within a smaller range. This suggests that when mapped to weighted spike trains, the majority of spikes will occur in the later time steps. As a consequence, Eq. (9) becomes difficult to satisfy as $t_0$ approaches $T$: the left-hand side contains a large amount of input, while the right-hand side provides limited encoding capacity from the subsequent spikes. This mismatch results in an increase in the residual membrane potential, a phenomenon we refer to as temporal coupling.

**Definition 2.** Temporal coupling occurs when the encoding of input information requires output spikes from earlier time steps.

To see this more clearly, consider a two-layer network with two input neurons and one output neuron, where $w_{00}^1 = w_{01}^1 = 1$ and $b_0^1 = 0$. Let $\beta = 2$, $T = 2$, and $\theta^0 = \theta^1 = 1$. If both input neurons fire at $t = 2$, the output neuron will produce a spike at $t = 2$ (which encodes 1) and leave half of the information in $u_0^1[2]$. Our desired outcome is a spike at $t = 1$ (which encodes 2), which is impossible due to the temporal dependency between the input and output. Therefore, a new neuron model is required to enable effective computation based on weighted spikes.

**Algorithm 1:** The forward method of TSA

**Input:** input $X$ of shape [BT, C, H, W], length of silent period $L$, spike amplitude $\theta$
**Output:** output spike train $S$ of shape [BT, C, H, W]
reshape and then pad $X$ with zeros to shape [(T+L), B, C, H, W];
membrane potential $M \leftarrow$ zeros_like $(X[0])$, threshold $v \leftarrow \frac{1}{2}\theta\beta^L$;
**for** $0 \leq i \leq L$ **do**
  $M \leftarrow \beta M + X[i]$; /* silent period */
**end**
**for** $i = 0$ **to** $T-1$ **do**
  $M \leftarrow \beta M + X[i+L]$; /* accumulate input */
  $S[i] \leftarrow (M \geq v).\text{float}() - (M \leq -v).\text{float}()$ ; /* fire ternary spikes */
  $M \leftarrow M - 2v \times S[i]$; /* over firing & soft reset */
**end**

#### 4.2.2 Ternary Self-Amplifying Neuron Model

Based on the above analysis, our approach begins by encouraging spikes to be generated as early as possible. The key idea is to lower the firing threshold and incorporate negative spikes into the encoding scheme to correct the excess information caused by over-spiking.

We set the positive firing threshold to $\frac{1}{2}\theta^l$ and introduce a negative threshold of $-\frac{1}{2}\theta^l$ into the neuron model, which initiates a negative spike when $o_i^l[t]$ falls below it. Intuitively, the neuron predicts that a spike will occur after receiving only half of the required information, and uses negative spikes to correct its prediction when it turns out to be wrong. This adjustment not only facilitates spike generation but also reduces $u_i^l[t_0 - 1]$ in the left-hand side of Eq. (9), making the equation easier to satisfy. The coefficient $\frac{1}{2}$ is selected to confine both positive and negative membrane potential within a narrow and balanced range. Given the above characteristics, we designate the coding method as the CSS coding scheme and the neuron model as the TSA neuron.

Noting that $S_i^l[t]$ can now represent negative activations with negative spikes, we apply additional logic to zero out sequences that encode negative values (a ReLU counterpart). Based on Observation 1, we limit the absolute value of $u_i^l[T]$ for accurate encoding and propose the following theorems:

**Theorem 1.** *Let $S_j^{l-1}[t] \sim a_j^{l-1}$, $w_{ij}^l = \hat{w}_{ij}^l$, and $b_i^l = \hat{b}_i^l / \sum_t \beta^{T-t}$. Then $S_i^l[t] \sim a_i^l$, provided that $\left|u_i^l[T]\right| < \theta^l$.*

Theorem 1 provides the conditions for controlling the encoding error in the hidden layers. Next, we give the method to encode the network input:

**Theorem 2.** *Let the input pixel value be $a_i^0$ and $\beta \leq 2$. By Initializing the membrane potential $u_i^0(0)$ with $a_i^0/\beta^T$, the resulting spike train $S_i^0(t) \sim a_i^0$ with $T$ steps.*

The proofs of the above two theorems can be found in Appendix A.2. By encoding input with Theorem 2 and applying Theorem 1 layer-by-layer, an ANN is then converted to a CSS-coded SNN. Combining Lemma 1 and Theorem 1, we can directly derive the following corollary to ensure conversion accuracy:

**Corollary 1.** *In a converted CSS-coded SNN, the following equation should be satisfied for accurate encoding:*

$$\forall \text{ layer index } l, \forall \text{ neuron index } i, \forall t_0 \in \{1, 2, \cdots, T\},$$
$$\left|\beta^{T-t_0+1} u_i^l[t_0 - 1] + \sum_j w_{ij}^l \sum_{t=t_0}^{T} \beta^{T-t} S_j^{l-1}[t] + \sum_{t=t_0}^{T} \beta^{T-t} b_i^l\right| < \theta^l + \sum_{t=t_0}^{T} \theta^l \beta^{T-t} \tag{10}$$

We refer to this equation as the decoupling condition. Note that a lowered threshold and negative spikes are not sufficient conditions for the above equation to hold, but they effectively keep the left-hand side of the equation within an acceptable range. Next, we shift our focus to the right-hand side of the equation.

**Data-Based Amplitude Adjustment.** Eq. (10) suggests that increasing $\theta^l$ can relax the constraints on the input. Let $\tilde{\theta}^l$ denote the initial spike amplitude. For initialization, we use a strategy similar to rate coding (Rueckauer et al., 2017): after observing the ANN activations over a portion of the training set, we calculate the 99.99th percentile $p^l$ of the activation distribution for each layer $l$ , and then set $\tilde{\theta}^l$ to $p^l / \sum_t \beta^{T-t}$. This setting ensures that the vast majority of the activations remain below the maximum encodable value, $\sum_t \tilde{\theta}^l \beta^{T-t}$, and increases the network's robustness to outlier activations. We then perform forward propagation with the converted SNN, and amplify $\theta^l$ based on the distribution of the residual membrane potential (see Appendix A.3 for more details).

However, an increase in $\theta^l$ makes the decoupling conditions for layer $l+1$ harder to meet[3]. Consequently, in deeper layers, the initial spike amplitude must be amplified by a large factor. This requires a sufficiently small $\tilde{\theta}^l$ to preserve adequate encoding granularity after scaling, which in turn necessitates a larger number of time steps. We address this issue by delaying the TSA output, which eliminates the need for $\tilde{\theta}^l$ amplification.

**One-Step Silent Period.** Inspired by the layerwise processing manner in TTFS coding (Stanojevic et al., 2022), we incorporate a one-step silent period into the TSA neuron model. During this period, neurons integrate input and perform stepwise weighting but are prohibited from firing. This method delays the output by one step, introducing a new term $\theta^l \beta^{T-t_0-1}$ (spike from $t = t_0 - 1$) to the right side of Eq. (10), thereby relaxing the decoupling condition. Since the input information is amplified by $\beta$ after the silent period, the firing threshold is adjusted to $\frac{\beta}{2}\theta^l$ accordingly. Similarly, the membrane potential is reduced by $\beta\theta^l$ for reset. In Algorithm 1, we provide pseudocode for the forward propagation process of TSA neurons. A mathematical description of the TSA neuron model can be found in Appendix A.4.

The silent period partially sacrifices synchronous processing at each time step, leading to increased output latency. For an $n$-layer network, the output layer will start to fire spikes only after $n$ steps. However, this approach effectively reduces the number of time steps for coding. With each layer operating in a pipelined manner, the efficiency gain outweighs the drawback of the increased latency.

### 4.3 Choice of the Amplification Factor

We find that the value of $\beta$ affects the performance of the CSS-SNN. As $\beta$ increases, the weight difference between spikes becomes larger, causing input spikes to cluster at later time steps and increasing the likelihood of temporal coupling. Conversely, a decrease in $\beta$ necessitates a larger $\theta^l$ to encode the same range, which in turn raises quantization error. Taking these observations into account and considering the limitation in Theorem 2, we set $\beta$ to 1.5 for our experiments. This choice also facilitates future hardware implementation, as only shifters and adders are needed for the amplification.

## 5 Experiments

In this section, we converted ANNs to CSS-coded SNNs and conducted experiments on CIFAR-10 and ImageNet datasets. We compared the required time steps and output latency with other coding schemes, including rate coding (Bu et al., 2022; Li et al., 2021; Meng et al., 2022), TTFS coding (Stanojevic et al., 2023; Yang et al., 2023), TSC coding (Han & Roy, 2020), and FS coding (Stöckl & Maass, 2021). We conducted two ablation studies to validate the effectiveness of our proposed method. First, we evaluated the trade-off between coding steps and inference latency introduced by the one-step silent period. Then, we examined the role of negative spikes in breaking temporal coupling and thereby reducing output latency.

### 5.1 Overall Performance

In Table 2, we compared the time steps and inference latency under different coding schemes, which reflect the throughput and latency of the network, respectively. To ensure a fair comparison, all SNNs are converted from full-precision ANNs. Furthermore, the accuracy of the ANNs utilized

[3] In Eq. (10), the spike amplitude of the previous layer is included in $S_j^{l-1}[t]$.

Table 2: Coding time steps and inference latency under different neural coding schemes, evaluated on CIFAR-10 and ImageNet datasets.

| | Methods | Architecture | ANN Accuracy | Coding Scheme | Coding Time Steps | Inference Latency | SNN Accuracy |
|---|---|---|---|---|---|---|---|
| CIFAR-10 | FS-conversion (Stöckl & Maass, 2021) | ResNet-20 | 91.58% | FS | 10 | 200 | 91.45% |
| | TTRBR (Meng et al., 2022) | ResNet-18 | 95.27% | rate | 128 | 128 | 95.18% |
| | TSC (Han & Roy, 2020) | VGG-16 | 93.63% | TSC | 512 | 512 | 93.57% |
| | LC-TTFS (Yang et al., 2023) | VGG-16 | 92.79% | TTFS | 50 | 800 | 92.72% |
| | Exact mapping (Stanojevic et al., 2023) | VGG-16 | 93.68% | TTFS | 64 | 1024 | 93.64% |
| | Calibration (Li et al., 2021) | VGG-16 | 95.72% | rate | 128 | 128 | 95.65% |
| | OPI (Bu et al., 2022) | VGG-16 | 94.57% | rate | 128 | 128 | 94.50% |
| | CSS (ours) | ResNet-20 | 92.10% | CSS | 12 | 32 | 92.06% |
| | | ResNet-18 | 95.24% | | 12 | 30 | 95.30% |
| | | VGG-16 | 95.89% | | 10 | 26 | 95.88% |
| ImageNet | OPI (Bu et al., 2022) | VGG-16 | 74.85% | rate | 256 | 256 | 74.62% |
| | TSC (Han & Roy, 2020) | VGG-16 | 73.49% | TSC | 1024 | 1024 | 73.33% |
| | RMP-SNN Han et al. (2020) | VGG-16 | 73.49% | rate | 2048 | 2048 | 72.78% |
| | Calibration (Li et al., 2021) | VGG-16 | 75.36% | rate | 256 | 256 | 74.23% |
| | TSC (Han & Roy, 2020) | ResNet-34 | 70.64% | TSC | 4096 | 4096 | 69.93% |
| | CalibrationLi et al. (2021) | ResNet-34 | 75.66% | rate | 256 | 256 | 74.61% |
| | FS-conversion (Stöckl & Maass, 2021) | ResNet-50 | 75.22% | FS | 10 | 500 | 75.10% |
| | TTRBR (Meng et al., 2022) | ResNet-50 | 76.02% | rate | 512 | 512 | 75.04% |
| | CSS (ours) | VGG-16 | 75.34% | CSS | 12 | 28 | 75.24% |
| | | ResNet-34 | 76.42% | | 14 | 48 | 76.22% |
| | | ResNet-50 | 80.85% | | 16 | 66 | 80.10% |

in each work is also provided. It's important to note that rather than focusing solely on the SNN accuracy, more attention should be given to the conversion loss.

**Reduction in Coding Time Steps.** The coding time steps refer to the number of time steps required to encode the input into a spike train, as well as the number of time steps needed to encode activations in each layer. This metric reflects the efficiency of the encoding scheme, indicating how well the method can represent information within a given time frame. For simpler classification tasks such as CIFAR-10, CSS coding scheme demonstrated nearly lossless conversion with a significant reduction in the number of required time steps. Compared to linear coding schemes like rate coding, CSS reduces time steps by more than tenfold for both VGG-16 and ResNet-18, while simultaneously reducing the conversion loss. While the FS coding scheme also applied weighted spikes and required fewer time steps for ResNet-20, it experienced greater conversion loss compared to our method. On the more complex ImageNet dataset, the higher precision demands for activation encoding further highlighted the benefits of incorporating non-linearity. Rate coding, even with 256 time steps, resulted in noticeable conversion loss. For example, Li et al. (2021) reported a conversion error exceeding 1% on ResNet-34 with 256 time steps, whereas our method achieved only 0.2% conversion loss with just 14 time steps. Although TTFS coding performed well on CIFAR-10, both works did not extend experiments to the larger-scale ImageNet dataset. FS-coding achieved smaller conversion loss for ResNet-50 with fewer time steps; however, this came at the cost of a latency eight times greater than that of ours.

**Reduction in Inference Latency.** Inference latency refers to the time elapsed from the beginning of input encoding to the receipt of the classification result, and is also measured in time steps. It indicates how efficiently the encoding scheme transmits information through neural computation across the network layers. In the CSS coding scheme, each layer of TSA neurons incorporates a one-step silent period, making the inference latency equal to the sum of layer counts and coding time steps. In contrast, both TTFS coding and FS coding require each layer to wait for the arrival of all inputs before firing spikes. While this approach facilitates lossless conversion with fewer coding time steps, it completely sacrifices the synchronous processing capability of SNNs, leading to increased output latency (i.e. the product of layer counts and coding time steps). For instance, in the CIFAR-10 classification task, the inference latency reported by Stanojevic et al. (2023) on VGG-16 is about 40 times that of our method. FS coding, as a nonlinear encoding scheme, performs well in both coding steps and conversion loss, but its output latency remains a major weakness; on ResNet-20, its latency exceeds that of CSS by over six times. Rate coding allows for synchronous processing but suffers from a higher number of coding time steps, and thus the inference latency

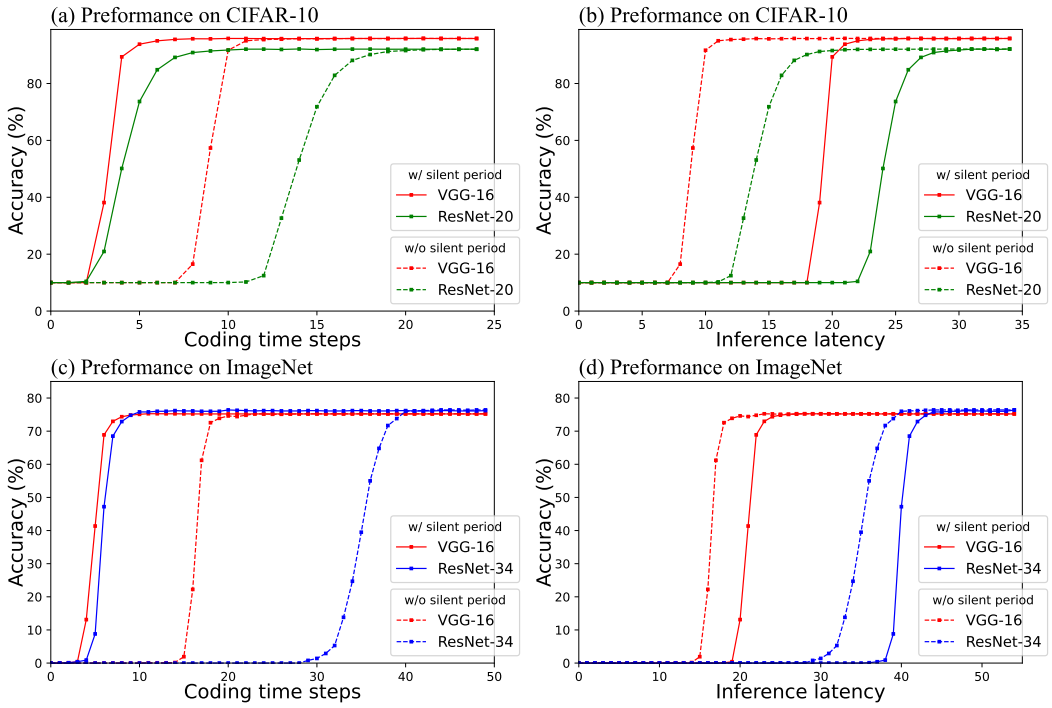


Figure 1: Trade-off between the coding time steps and inference latency. The dashed line represents the results obtained without the silent period, while the solid line represents the results achieved after incorporating the silent period. (a) Coding time steps on CIFAR-10. (b) Inference latency on CIFAR-10. (c) Coding time steps on ImageNet. (d) Inference latency on ImageNet.

is still not ideal. In the ImageNet classification task, for example, rate coding on ResNet-34 has a latency five times greater than our method.

### 5.2 Coding Time Steps vs. Inference Latency

According to our previous analysis, relying solely on $\theta^l$ amplification to break temporal coupling would require smaller $\tilde{\theta}^l$ in deeper layers, which leads to an increase in coding time steps. To address this, we introduce a one-step silent period to achieve a trade-off between coding time steps and inference latency. In this section, we conducted an ablation study to assess the effectiveness of this approach. We performed classification tasks on CIFAR-10 using VGG-16 and ResNet-20, and on ImageNet using VGG-16 and ResNet-34. Fig. 1 (a) and (c) present the relationship between coding time steps and accuracy, while Fig. 1 (b) and (d) show the relationship between inference latency and accuracy. The experimental results indicate that even with no silent period, deeper networks experience larger latency due to increased coding time steps. This can also be understood as neurons in each layer require time to accumulate membrane potential before firing. Thus, incorporating a silent period has a limited effect on increasing inference latency, but plays a significant role in reducing coding time steps. For example, in ResNet-20 on CIFAR-10, the silent period increased latency from 20 to 30 steps but halved the coding time steps, greatly improving throughput. This effect becomes more pronounced with increased network depth or dataset scale. For instance, with ResNet-34 on ImageNet, the silent period added only about 5 steps to inference latency while reducing coding time steps by approximately 30 steps. Overall, incorporating the silent period effectively reduces the required number of time steps for encoding, substantially improving throughput with minimal impact on latency.

### 5.3 Effect of Negative Spikes

The introduction of negative spikes and the corresponding TSA neurons in CSS coding plays a crucial role in breaking temporal coupling, setting our method apart from other coding schemes

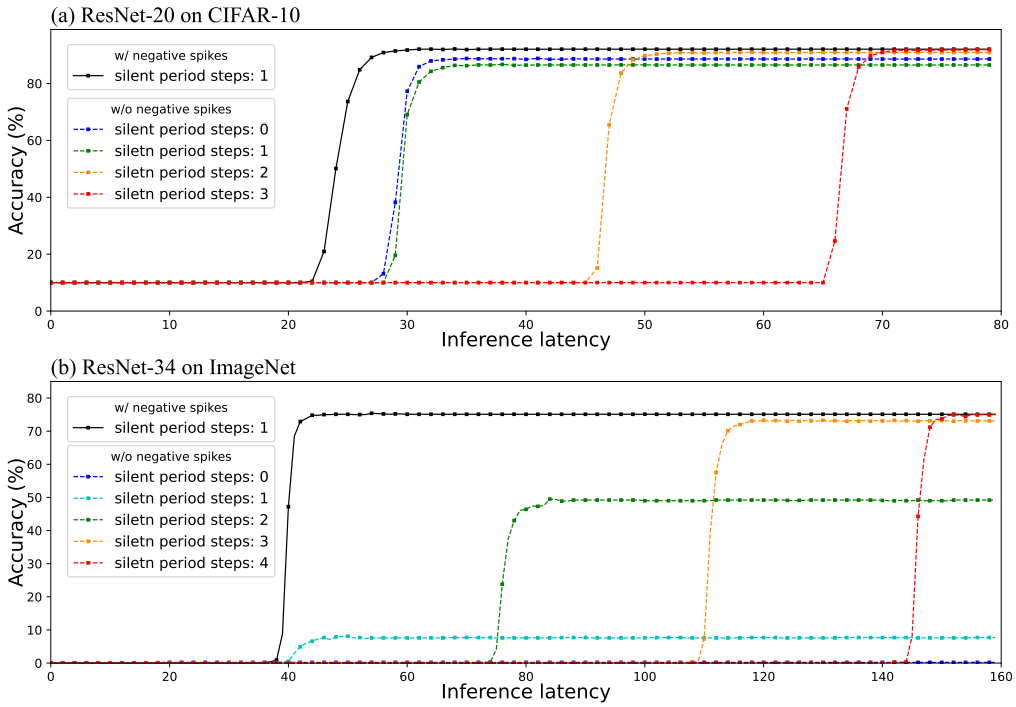


Figure 2: Inference latency with and without negative spikes. The solid line represents the results with negative spikes, while the dashed line indicates the results without negative spikes. The lines of different colors correspond to different lengths of the silent period as shown in the legend. (a) ResNet-20 on CIFAR-10. (b) ResNet-34 on ImageNet.

using weighted spikes (Rueckauer & Liu, 2021; Stöckl & Maass, 2021; Kim et al., 2018). In this section, we validated the importance of negative spikes in achieving low-latency nonlinear encoding. We conducted experiments using ResNet-20 on CIFAR-10 and ResNet-34 on ImageNet, gradually increasing the silent period length from zero in the absence of negative spikes. The results are shown in Fig. 2. For each configuration, data-based amplitude adjustment was applied to help mitigate temporal coupling. The results from CIFAR-10 demonstrated that, without negative spikes, a silent period of at least three steps was required to match the performance gains introduced by negative spikes, which in turn increased inference latency by nearly 50 time steps. As network depth and dataset size grow, the effectiveness of negative spikes becomes more pronounced. For instance, in ResNet-34 on ImageNet, incorporating negative spikes reduced inference latency by over 100 time steps, significantly improving inference speed. It is important to note that although the decoupling condition is satisfied by incorporating silent period and amplitude adjustment on training set, this does not ensure optimal performance on the test set (e.g. the orange dashed line in Fig. 2). By contrast, negative spikes provide a data-independent approach to break temporal coupling, resulting in more stable and consistent performance improvements.

## 6 Conclusion and Discussion

In this work, we compress the coding time steps by assigning weights to spikes, enabling each spike to carry more information. We also introduce negative spikes to break temporal coupling, effectively reducing inference latency. The resulting CSS encoding scheme enhances both the throughput and inference speed of converted SNNs, while minimizing conversion loss.

## REFERENCES


Tong Bu, Jianhao Ding, Zhaofei Yu, and Tiejun Huang. Optimized potential initialization for low-latency spiking neural networks, 2022.

Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks, 2023.

Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, May 2015. ISSN 1573-1405. doi: 10.1007/s11263-014-0788-3. URL https://doi.org/10.1007/s11263-014-0788-3.

Shikuang Deng and Shi Gu. Optimal conversion of conventional artificial neural networks to spiking neural networks, 2021.

Peter U. Diehl, Daniel Neil, Jonathan Binas, Matthew Cook, Shih-Chii Liu, and Michael Pfeiffer. Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2015. doi: 10.1109/IJCNN.2015.7280696.

Jianhao Ding, Zhaofei Yu, Yonghong Tian, and Tiejun Huang. Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks, 2021.

Wenzhe Guo, Mohammed E. Fouda, Ahmed M. Eltawil, and Khaled Nabil Salama. Neural coding in spiking neural networks: A comparative study for robust neuromorphic systems. *Frontiers in Neuroscience*, 15, 2021. ISSN 1662-453X. doi: 10.3389/fnins.2021.638474. URL https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2021.638474.

Bing Han and Kaushik Roy. Deep spiking neural network: Energy efficiency through time based coding. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *Computer Vision – ECCV 2020*, pp. 388–404, Cham, 2020. Springer International Publishing.

Bing Han, Gopalakrishnan Srinivasan, and Kaushik Roy. Rmp-snn: Residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network, 2020.

Zecheng Hao, Jianhao Ding, Tong Bu, Tiejun Huang, and Zhaofei Yu. Bridging the gap between anns and snns by calibrating offset spikes, 2023.

Yangfan Hu, Qian Zheng, Xudong Jiang, and Gang Pan. Fast-snn: Fast spiking neural network by converting quantized ann. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (12):14546–14562, 2023. doi: 10.1109/TPAMI.2023.3275769.

Jaehyun Kim, Heesu Kim, Subin Huh, Jinho Lee, and Kiyoung Choi. Deep neural networks with weighted spikes. *Neurocomputing*, 311:373–386, 2018. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2018.05.087. URL https://www.sciencedirect.com/science/article/pii/S0925231218306726.

Youngeun Kim, Yuhang Li, Hyoungseob Park, Yeshwanth Venkatesha, Anna Hambitzer, and Priyadarshini Panda. Exploring temporal information dynamics in spiking neural networks, 2022.

Chankyu Lee, Syed Shakib Sarwar, Priyadarshini Panda, Gopalakrishnan Srinivasan, and Kaushik Roy. Enabling spike-based backpropagation for training deep neural network architectures. *Frontiers in Neuroscience*, 14, February 2020. ISSN 1662-453X. doi: 10.3389/fnins.2020.00119. URL http://dx.doi.org/10.3389/fnins.2020.00119.

Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation, 2016. URL https://arxiv.org/abs/1608.08782.

Yuhang Li, Shikuang Deng, Xin Dong, Ruihao Gong, and Shi Gu. A free lunch from ann: Towards efficient, accurate spiking neural networks calibration, 2021.

Qingyan Meng, Shen Yan, Mingqing Xiao, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Training much deeper spiking neural networks with a small number of time-steps. *Neural Networks*, 153:254–268, 2022. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2022.06.001. URL https://www.sciencedirect.com/science/article/pii/S0893608022002064.

Bodo Rueckauer and Shih-Chii Liu. Conversion of analog to spiking neural networks using sparse temporal coding. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2018. doi: 10.1109/ISCAS.2018.8351295.

Bodo Rueckauer and Shih-Chii Liu. Temporal pattern coding in deep spiking neural networks. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2021. doi: 10.1109/IJCNN52387.2021.9533837.

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, 2017. ISSN 1662-453X. doi: 10.3389/fnins.2017.00682.

Abhronil Sengupta, Yuting Ye, Robert Wang, Chiao Liu, and Kaushik Roy. Going deeper in spiking neural networks: Vgg and residual architectures, 2019. URL https://arxiv.org/abs/1802.02627.

Ana Stanojevic, Stanisław Woźniak, Guillaume Bellec, Giovanni Cherubini, Angeliki Pantazi, and Wulfram Gerstner. An exact mapping from relu networks to spiking neural networks, 2022. URL https://arxiv.org/abs/2212.12522.

Ana Stanojevic, Stanisław Woźniak, Guillaume Bellec, Giovanni Cherubini, Angeliki Pantazi, and Wulfram Gerstner. High-performance deep spiking neural networks with 0.3 spikes per neuron, 2023. URL https://arxiv.org/abs/2306.08744.

Christoph Stöckl and Wolfgang Maass. Optimized spiking neurons classify images with high accuracy through temporal coding with two spikes, 2021. URL https://arxiv.org/abs/2002.00860.

Aboozar Taherkhani, Ammar Belatreche, Yuhua Li, Georgina Cosma, Liam P. Maguire, and T.M. McGinnity. A review of learning in biologically plausible spiking neural networks. *Neural Networks*, 122:253–272, 2020. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2019.09.036. URL https://www.sciencedirect.com/science/article/pii/S0893608019303181.

Xiangwen Wang, Xianghong Lin, and Xiaochao Dang. Supervised learning in spiking neural networks: A review of algorithms and evaluations. *Neural Networks*, 125:258–280, 2020. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2020.02.011. URL https://www.sciencedirect.com/science/article/pii/S0893608020300563.

Kashu Yamazaki, Viet-Khoa Vo-Ho, Darshan Bulsara, and Ngan Le. Spiking neural networks and their applications: A review. *Brain Sci*, 12(7), June 2022.

Qu Yang, Malu Zhang, Jibin Wu, Kay Chen Tan, and Haizhou Li. Lc-ttfs: Towards lossless network conversion for spiking neural networks with ttfs coding, 2023. URL https://arxiv.org/abs/2310.14978.

# A APPENDIX

## A.1 PROOFS OF EQUATIONS

**Proof of Eq. (8)** (A similar derivation leads to Eq. (5))

$$u_i^l[T] = \sum_j w_{ij}^l \sum_{t=1}^{T} \beta^{T-t} S_j^{l-1}[t] + \sum_{t=1}^{T} \beta^{T-t} b_i^l - \sum_{t=1}^{T} \beta^{T-t} S_i^l[t] \tag{A1}$$

*Proof.* Starting with the initial condition $u_i^l[0] = 0$ and Eq. (7), we can write:

$$u_i^l[1] = z_i^l[1] - S_i^l[1]$$

Next, we derive the expression for $u_i^l[2]$ by substitute the above into Eq. (7):

$$u_i^l[2] = \beta(z_i^l[1] - S_i^l[1]) + z_i^l[2] - S_i^l[2]$$

We can generalize this process to iteratively compute the membrane potential up to $t = T$:

$$u_i^l[T] = \sum_{t=1}^{T} \beta^{T-t}(z_i^l[t] - S_i^l[t])$$

substituting $z_i^l[t]$ from Eq. (4) and rearranging the terms, we get:

$$u_i^l[T] = \sum_{t=1}^{T} \beta^{T-t}(\sum_j w_{ij}^l S_j^{l-1}[t] + b_i^l - S_i^l[t])$$

Reorganizing the terms by summation yields:

$$u_i^l[T] = \sum_j w_{ij}^l \sum_{t=1}^{T} \beta^{T-t} S_j^{l-1}[t] + \sum_{t=1}^{T} \beta^{T-t} b_i^l - \sum_{t=1}^{T} \beta^{T-t} S_i^l[t]$$

□

## A.2 PROOFS OF LEMMAS AND THEOREMS

**Lemma 1.** *Making $u_i^l[T] < \epsilon$ is equivalent to satisfying the following equation:*

$$\forall t_0 \in \{1, 2, \cdots, T\},$$
$$\beta^{T-t_0+1} u_i^l[t_0 - 1] + \sum_j w_{ij}^l \sum_{t=t_0}^{T} \beta^{T-t} S_j^{l-1}[t] + \sum_{t=t_0}^{T} \beta^{T-t} b_i^l < \epsilon + \sum_{t=t_0}^{T} \theta^l \beta^{T-t} \tag{A2}$$

*Proof.* We first prove the forward direction. Given that $u_i^l[T] < \epsilon$, we can express it using Eq. (7) and Eq. (4) as follows:

$$\beta u_i^l[T-1] + \sum_j w_{ij}^l S_j^{l-1}[T] + b_i^l < \epsilon + \theta^l \tag{A3}$$

Continue the above process, and we have:

$$\beta^2 u_i^l[T-2] + \sum_j w_{ij}^l \beta S_j^{l-1}[T-1] + \beta b_i^l + \sum_j w_{ij}^l S_j^{l-1}[T] + b_i^l < \epsilon + \theta^l + \beta\theta^l \tag{A4}$$

The above process can be repeated until we obtain an equation involving $u_i^l[0]$. The left-hand side of each equation regarding $u_i^l[t]$, where $t \in \{1, 2, \cdots, T\}$, can be organized to demonstrate that the forward reasoning is valid.

Then we proceed to prove the backward direction. For any $t_0 \in \{1, 2, \cdots, T\}$, by iteratively updating the membrane potential using Eq. (7) from $t = t_0$ until $t = T$, and substitute $z_i^l[t]$ with Eq. (4), we can get:

$$u_i^l[T] = \beta^{T-t_0+1} u_i^l[t_0 - 1] + \sum_j \sum_{t=t_0}^{T} w_{ij}^l \beta^{T-t} S_j^{l-1}[t] + \sum_{t=t_0}^{T} \beta^{T-t} b_i^l - \sum_{t=t_0}^{T} \beta^{T-t} S_i^l[t] \tag{A5}$$

Note that $\sum_t \beta^{T-t} S_i^l[t] \leq \sum_t \theta^l \beta^{T-t}$. Then we can write:

$$\begin{aligned} u_i^l[T] &\leq \beta^{T-t_0+1} u_i^l[t_0 - 1] + \sum_j w_{ij}^l \sum_{t=t_0}^{T} \beta^{T-t} S_j^{l-1}[t] + \sum_{t=t_0}^{T} \beta^{T-t} b_i^l - \sum_{t=t_0}^{T} \theta^l \beta^{T-t} \\ &< \epsilon \end{aligned} \tag{A6}$$

□

**Theorem 1.** *Let $S_j^{l-1}[t] \sim a_j^{l-1}$, $w_{ij}^l = \hat{w}_{ij}^l$, and $b_i^l = \hat{b}_i^l / \sum_t \beta^{T-t}$. Then $S_i^l[t] \sim a_i^l$, provided that $\left|u_i^l[T]\right| < \theta^l$.*

*Proof.* Eq. (8) can be organized into the following form:

$$\sum_{t=1}^{T} \beta^{T-t} S_i^l[t] = \sum_j w_{ij}^l \sum_{t=1}^{T} \beta^{T-t} S_j^{l-1}[t] + \sum_{t=1}^{T} \beta^{T-t} b_i^l - u_i^l[T] \tag{A7}$$

Given that $S_j^{l-1}[t] \sim a_j^{l-1}$, we use $\sigma_j^{l-1}$ to denote the difference between the encoded value and the activation, defined as $\sigma_j^{l-1} = \sum_t \beta^{T-t} S_j^{l-1}[t] - a_j^{l-1}$. Substituting $a_j^{l-1}$ and $\sigma_i^l$ into Eq. (A7), we can write:

$$\begin{aligned} \sum_{t=1}^{T} \beta^{T-t} S_i^l[t] &= \sum_j w_{ij} (a_j^{l-1} + \sigma_j^{l-1}) + \sum_{t=1}^{T} \beta^{T-t} b_i^l - u_i^l[T] \\ &= \sum_j \hat{w}_{ij} (a_j^{l-1} + \sigma_j^{l-1}) + \hat{b}_i^l - u_i^l[T] \\ &= \sum_j \hat{w}_{ij} a_j^{l-1} + \hat{b}_i^l - u_i^l[T] + \sum_j \hat{w}_{ij} \sigma_j^{l-1} \end{aligned} \tag{A8}$$

According to Definition 1, we have $-\theta^l < \sigma_j^{l-1} < \theta^l$. Considering that $\theta^l$ is typically kept small to provide fine-grained encoding and $\hat{w}_{ij}$ is generally symmetrically distributed around zero, we can ignore the last term on the right-hand side of the equation. Since $S_i^l[T]$ can encode negative values, we implemented a ReLU counterpart to zero out these spike sequences, corresponding to the $\max(\cdot, 0)$ operation in Eq. (6). Combining Eq. (A8) with the condition $\left|u_i^l(T)\right| < \theta^l$ and Definition 1, we can conclude that $S_i^l[t] \sim a_i^l$. □

**Theorem 2.** *Let the input pixel value be $a_i^0$ and $\beta \leq 2$. By Initializing the membrane potential $u_i^0(0)$ with $a_i^0/\beta^T$, the resulting spike train $S_i^0(t) \sim a_i^0$ with $T$ steps.*

*Proof.* We proof this theorem by mathematical induction. Let $\tilde{a}_{i.T}^0$ and $m_T^0$ denote the encoded value and the maximum encodable value, respectively, i.e. $\tilde{a}_{i,T}^0 = \sum_t \beta^{T-t} S_i^0[t]$, $m_T^0 = \sum_t \theta^0 \beta^{T-t}$.

**step 1.** For $T = 1$, it's obvious that:

$$\begin{cases} \left|\tilde{a}_{i,0}^0 - a_i^0\right| < \theta^0 & , \ a_i^0 < m_0^0 \\ \tilde{a}_{i,0}^0 = m_0^0 & , \ a_i^0 \geq m_0^0 \end{cases} \tag{A9}$$

**step 2.** Assume the statement is true for $T = t_0$, i.e. we have:

$$\begin{cases} \left|\tilde{a}_{i,t_0}^0 - a_i^0\right| < \theta^0 & , \ a_i^0 < m_{t_0}^0 \\ \tilde{a}_{i,t_0}^0 = m_{t_0}^0 & , \ a_i^0 \geq m_{t_0}^0 \end{cases} \tag{A10}$$

Note the relationship between $\theta^0$ and $m^0_{t_0}$: $\theta^0 = \frac{m^0_{t_0}}{1+\beta+\cdots+\beta^{t_0-1}}$. Then we prove that for $T = t_0 + 1$, Eq. (A10) still holds.

**case (1).** For $a^0_i < m^0_{t_0+1}$: Consider the first $t_0$ steps. It can be observed that this process is equivalent to encoding $a^0_i/\beta$ with $m^0_{t_0} = \frac{1+\beta+\cdots+\beta^{t_0-1}}{1+\beta+\cdots+\beta^{t_0}} m^0_{t_0+1}$. Then we have:

$$\begin{cases} \left|\tilde{a}^0_{i,t_0} - \dfrac{a^0_i}{\beta}\right| < \dfrac{m^0_{t_0+1}}{1+\beta+\cdots+\beta^{t_0}} & , \dfrac{a^0_i}{\beta} < m^0_{t_0} \\ \tilde{a}^0_{i,t_0} = m^0_{t_0} & , \dfrac{a^0_i}{\beta} \geq m^0_{t_0} \end{cases} \tag{A11}$$

For $\frac{a^0_i}{\beta} < m^0_{t_0}$, we can write:

$$\beta \left|\tilde{a}^0_{i,t_0} - \frac{a^0_i}{\beta}\right| < \frac{\beta m^0_{t_0+1}}{1+\beta+\cdots+\beta^{t_0}} < \frac{2m^0_{t_0+1}}{1+\beta+\cdots+\beta^{t_0}} = 2\theta^0 \tag{A12}$$

For $\frac{a^0_i}{\beta} \geq m^0_{t_0}$:

$$\begin{aligned} \beta \left|\tilde{a}^0_{i,t_0} - \frac{a^0_i}{\beta}\right| &= a^0_i - \frac{1+\beta+\cdots+\beta^{t_0-1}}{1+\beta+\cdots+\beta^{t_0}} m^0_{t_0+1} \\ &\leq m^0_{t_0+1} - m^0_{t_0+1} \frac{1+\beta+\cdots+\beta^{t_0-1}}{1+\beta+\cdots+\beta^{t_0}} = \theta^0 \end{aligned} \tag{A13}$$

According to Lemma 1, $\beta \left|\tilde{a}^0_{i,t_0} - \frac{a^0_i}{\beta}\right| < 2\theta^0$ is equivalent to $\left|u^0_i[t_0+1]\right| < \theta^0$ (as there's neither input nor bias term). Also note $\left|u^0_i[t_0+1]\right| = \left|\tilde{a}^0_{i,t_0+1} - a^0_i\right|$. Then we have $\left|\tilde{a}^0_{i,t_0+1} - a^0_i\right| < \theta^0$.

**case (2).** For $a^0_i \geq m^0_{t_0+1}$:

$$\frac{a^0_i}{\beta} \geq \frac{1}{\beta} m^0_{t_0+1} > \frac{1+\beta+\cdots+\beta^{t_0-1}}{1+\beta+\cdots+\beta^{t_0}} m^0_{t_0+1} = m^0_{t_0} \tag{A14}$$

Then we have:

$$\beta \left|\tilde{a}^0_{i,t_0} - \frac{a^0_i}{\beta}\right| = a^0_i - \beta \frac{1+\beta+\cdots+\beta^{t_0-1}}{1+\beta+\cdots+\beta^{t_0}} m^0_{t_0+1} \geq \frac{m^0_{t_0+1}}{1+\beta+\cdots+\beta^{t_0}} = \theta^0 \tag{A15}$$

which means the neuron will fire a spike at $t = t_0 + 1$, leading to $\tilde{a}^0_{i,t_0+1} = m^0_{t_0+1}$. Combining case (1) and (2), and we have:

$$\begin{cases} \left|\tilde{a}^0_{i,t_0+1} - a^0_i\right| < \theta^0 & , a^0_i < m^0_{t_0+1} \\ \tilde{a}^0_{i,t_0+1} = m^0_{t_0+1} & , a^0_i \geq m^0_{t_0+1} \end{cases} \tag{A16}$$

**step 3.** By the principle of mathematical induction, $\forall T \in \mathbb{N}^\star$:

$$\begin{cases} \left|\tilde{a}^0_{i,T} - a^0_i\right| < \theta^0 & , a^0_i < m^0_T \\ \tilde{a}^0_{i,T} = m^0_T & , a^0_i \geq m^0_T \end{cases} \tag{A17}$$

Considering our initialization strategy for $\tilde{\theta}^0$ and subsequent data-based amplification, we can always ensure that $a^0_i < m^0_T$. Thus, $\left|\tilde{a}^0_{i,T} - a^0_i\right| < \theta^0$, which means $S^0_i[t] \sim a^0_i$. □

### A.3 Spike Amplitude Adjustment

We use a subset of the training set to perform forward propagation for the CSS-based SNN, and then calculate the 99.9th percentile $u^l$ of the distribution of $u^l_i[T]$ for each layer $l$. If $u^l$ exceeds $\theta^l$, we amplify $\theta^l$ by a factor $s^l$. Note that increasing $\theta^l$ raises the firing threshold at the same time, making the change in $u^l_i[T]$ a complex nonlinear process. To determine a suitable $s^l$, we simplify the problem by assuming that $u^l_i[T]$ accumulates uniformly over time. Thus, if the increment of $\theta^l$ is

$\Delta\theta^l$, then $u_i^l[T]$ will decrease by $\Delta\theta^l \sum_t \beta^{T-t}$. Accordingly, $s^l$ is determined using the following equation:

$$s^l = 1 + \begin{cases} \dfrac{u^l - \theta^l}{\sum_t \beta^{T-t}} & , u^l - \theta^l > 0.05 \cdot \sum_t \beta^{T-t} \\ \dfrac{u^l - \theta^l - 0.01 \cdot \sum_t \beta^{T-t}}{\sum_t \beta^{T-t}} + 0.04 & , 0.05 \cdot \sum_t \beta^{T-t} \geq u^l - \theta^l > 0.01 \cdot \sum_t \beta^{T-t} \\ \dfrac{4 \cdot (u^l - \theta^l)}{\sum_t \beta^{T-t}} & , 0.01 \cdot \sum_t \beta^{T-t} \geq u^l - \theta^l > 0 \end{cases} \tag{A18}$$

To ensure that the spike amplitude can still be effectively adjusted when $u^l$ slightly exceeds $\theta^l$, we increase the value of $s^l$ for this range. It is important to note that this adjustment lacks strict mathematical support and serves as a heuristic for fine-tuning the spike amplitude. The introduction of negative spikes remains the core mechanism for breaking temporal coupling.

### A.4 Mathematical Description of the TSA Neuron

To generalize the representation, let $T_s$ denote the length of the silent period. For a TSA neuron in the $l$-th layer, when the time step $t \in \{1, 2, \ldots, T_s l\}$, the membrane potential remains $u_i^l[t] = u_i^l[0] = 0$, and no spikes are generated. After this period, the neuron processes inputs in cycles of length $T + T_s$. Let $T_a^k$ denote the start of the $k$-th cycle, i.e. $T_a^k = T_s l + k(T_s + T) + 1$, and let $T_b^k$ denote the end of the cycle, i.e. $T_b^k = T_s l + (k+1)(T_s + T)$, with $k$ being a natural number. Without loss of generality, we consider the case where $t \in \{T_a^k, T_a^k + 1, \ldots, T_b^k\}$.

The set of its spike times can be expressed as follows:

$$\mathbb{F}_i^l = \left\{ t_i^{l,f} \,\middle|\, \left| o_i^l[t_i^{l,f}] \right| \geq \frac{\theta^l \beta^{T_s}}{2},\ t_i^{l,f} \in \{T_a^k + T_s, T_a^k + T_s + 1, \cdots, T_b^k\} \right\} \tag{A19}$$

The spike sequence it emits, $S_i^l[t]$, can then be written as:

$$S_i^l[t] = \sum_{t_i^{l,f} \in \mathbb{F}_i^l} \operatorname{sgn}\left( o_i^l[t_i^{l,f}] \right) \theta^l \delta[t - t_i^{l,f}] \tag{A20}$$

where $\delta[\cdot]$ denotes an unit impulse, $\operatorname{sgn}(\cdot)$ is the sign function and $\theta^l$ is the spike amplitude. The update process of the membrane potential can be expressed as follows:

$$u_i^l[t] = \beta u_i^l[t-1] + z_i^l[t] - \beta^{T_s} S_i^l[t] \tag{A21}$$

where $z_i^l[t]$ denotes the integrated inputs:

$$z_i^l[t] = \sum_j w_{ij}^l S_j^{l-1}[t] + b_i^l \tag{A22}$$

### A.5 Pesudocode for Conversion Process

See Algorithm 2. The spike amplitude for each layer is determined using the method outlined in Section 4.2.2. Note that this value is absorbed into the weights and bias, so after conversion, the spike amplitudes for TSA neurons are all normalized to 1. For data-based spike amplitude adjustment, we first complete the above conversion process using the initial spike amplitudes. Then, based on the residual membrane potentials observed in the resulting SNN, we update the spike amplitudes. These updated values are then used to re-convert the ANN. The process can be repeated as may times as needed.

### A.6 Implementation of the ReLU Counterpart

In the actual implementation, we fuse the ReLU counterpart into the TSA neuron model to speed up program execution. We refer to this model as TSA-ReLU neuron. Below, we continue the notation

**Algorithm 2:** Algorithm for ANN-SNN conversion

**Input:** ANN model $f_A(\hat{\boldsymbol{W}}, \hat{\boldsymbol{b}})$, encode time steps $T$, amplification factor $\beta$, spike amplitude $\theta^l$ for each layer $l$, total layer number $L$
**Output:** SNN models $f_S(\boldsymbol{W}, \boldsymbol{b})$
reshape and pad $X$ to [T, B, C, H, W] with zeros to shape [(T+L), B, C, H, W];
membrane potential $M \leftarrow$ zeros_like $(X[0])$, threshold $v \leftarrow \frac{1}{2}\theta\beta^L$;
set CSS encoder for the input layer; `/* see Theorem 2 */`
**for** $1 \leq l \leq L$ **do**
  $W^l \leftarrow \frac{\theta^{l-1}}{\theta^l}\hat{W}^l$; `/* norms` $\theta^l$ `in SNN to 1 */`
  $b^l \leftarrow \frac{1}{\sum_{t=0}^{T-1}\beta^t\theta^l}\hat{b}^l$;
  replace ReLU activation with TSA and ReLU counterpart.
**end**

from Appendix A.4 to present the mathematical model of TSA-ReLU. Without loss of generality, we consider the case where $t \in \{T_a^k, T_a^k + 1, \ldots, T_b^k\}$. We use $h_i^l[t]$ and $g_i^l[t]$ to represent the accumulated input and output of TSA-ReLU, respectively:

$$\begin{aligned} h_i^l[t] &= \sum_j w_{ij}^l \sum_{\tau=T_a^k}^{t} \beta^{t-\tau} S_j^{l-1}[\tau] + \sum_{\tau=T_a^k}^{\max(t, T_a^k+T)} \beta^{t-\tau} b_i^l \\ g_i^l[t] &= \sum_{\tau=\min(T_a^k+T_s, t)}^{t} \beta^{t-\tau} S_i^l[\tau] \end{aligned} \tag{A23}$$

Then we set $o_i^l[t]$ according to the following equation:

$$o_i^l[t] = \begin{cases} h_i^l[t] - g_i^l[t] & , h_i^l[t] \geq 0, t \geq T_a^k + T_s \\ -g_i^l[t] & , h_i^l[t] < 0, t \geq T_a^k + T_s \\ 0 & , t < T_a^k + T_s \end{cases} \tag{A24}$$

The firing condition of TSA-ReLU is the same as that of TSA, and is given by Eqs. (A19) and (A20).