



OPEN ACCESS

EDITED BY

Timothée Masquelier,
Centre National de la Recherche Scientifique
(CNRS), France

REVIEWED BY

Malu Zhang,
National University of Singapore, Singapore
Bernard Girau,
Université de Lorraine, France
Seongsik Park,
Korea Institute of Science and Technology
(KIST), Republic of Korea

*CORRESPONDENCE

Siying Liu
✉ siying.liu20@imperial.ac.uk

RECEIVED 24 July 2023

ACCEPTED 11 September 2023

PUBLISHED 02 October 2023

CITATION

Liu S, Leung VCH and Dragotti PL (2023)
First-spike coding promotes accurate and
efficient spiking neural networks for discrete
events with rich temporal structures.
Front. Neurosci. 17:1266003.
doi: 10.3389/fnins.2023.1266003

COPYRIGHT

© 2023 Liu, Leung and Dragotti. This is an
open-access article distributed under the terms
of the [Creative Commons Attribution License
\(CC BY\)](https://creativecommons.org/licenses/by/4.0/). The use, distribution or reproduction
in other forums is permitted, provided the
original author(s) and the copyright owner(s)
are credited and that the original publication in
this journal is cited, in accordance with
accepted academic practice. No use,
distribution or reproduction is permitted which
does not comply with these terms.

First-spike coding promotes accurate and efficient spiking neural networks for discrete events with rich temporal structures

Siying Liu*, Vincent C. H. Leung and Pier Luigi Dragotti

Communications and Signal Processing Group, Department of Electrical and Electronic Engineering, Imperial College London, London, United Kingdom

Spiking neural networks (SNNs) are well-suited to process asynchronous event-based data. Most of the existing SNNs use rate-coding schemes that focus on firing rate (FR), and so they generally ignore the spike timing in events. On the contrary, methods based on temporal coding, particularly time-to-first-spike (TTFS) coding, can be accurate and efficient but they are difficult to train. Currently, there is limited research on applying TTFS coding to real events, since traditional TTFS-based methods impose one-spike constraint, which is not realistic for event-based data. In this study, we present a novel decision-making strategy based on first-spike (FS) coding that encodes FS timings of the output neurons to investigate the role of the first-spike timing in classifying real-world event sequences with complex temporal structures. To achieve FS coding, we propose a novel surrogate gradient learning method for discrete spike trains. In the forward pass, output spikes are encoded into discrete times to generate FS times. In the backpropagation, we develop an error assignment method that propagates error from FS times to spikes through a Gaussian window, and then supervised learning for spikes is implemented through a surrogate gradient approach. Additional strategies are introduced to facilitate the training of FS timings, such as adding empty sequences and employing different parameters for different layers. We make a comprehensive comparison between FS and FR coding in the experiments. Our results show that FS coding achieves comparable accuracy to FR coding while leading to superior energy efficiency and distinct neuronal dynamics on data sequences with very rich temporal structures. Additionally, a longer time delay in the first spike leads to higher accuracy, indicating important information is encoded in the timing of the first spike.

KEYWORDS

spiking neural networks, first-spike coding, firing rate coding, time-to-first-spike, surrogate gradient, event-based data, temporal structures

1. Introduction

The emergence of event-driven neuromorphic devices has given further impetus to the development of spiking neural networks (SNNs) (Anumula et al., 2018). SNNs more closely mimic biological neural systems by processing and transmitting information with sparse and asynchronous binary spikes (Pfeiffer and Pfeil, 2018). By incorporating spike timing in their neuron model, SNNs have become effective tools for acquiring and processing temporal information (Wang et al., 2020). Neuromorphic

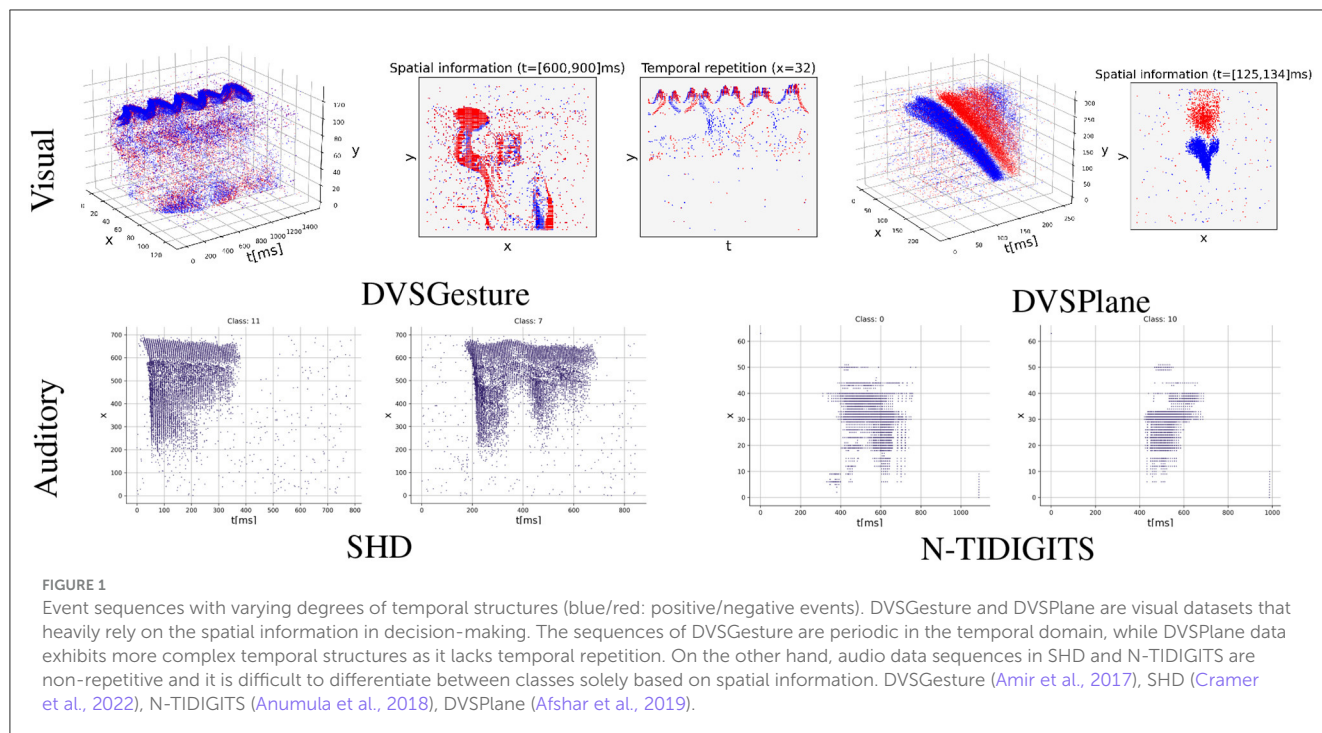
devices such as dynamic vision sensors (DVS) and dynamic audio sensors (DAS) produce asynchronous events which are well-suited to be used as the input of SNNs. Combining SNNs with the output of neuromorphic devices can potentially enable the development of power-efficient systems that more closely mimic biological processing.

SNNs process a sequence of spikes in each layer, which is referred to as spike trains. A spike train is mathematically defined by $s(t) = \sum_{t_i \in \mathcal{I}} \delta(t - t_i)$, where t_i represents the timing of individual spikes in the set \mathcal{I} . In terms of information encoding, rate coding and temporal coding are two distinct approaches in SNNs (Rullen and Thorpe, 2001; Huxter et al., 2003; Brette, 2015; Kiselev, 2016; Liu and Wang, 2022). Rate coding focuses on the firing rate (FR) of neurons, in which information is represented by the average number of spikes within a certain time window. Although it is widely used and effective, rate coding is less efficient because it involves a great number of spikes and ignores the relative timing between spikes, which encodes important information of stimulus in visual (Gollisch and Meister, 2008), auditory (Heil, 2004; Fontaine and Peremans, 2009), and other systems (Panzeri et al., 2001; Huxter et al., 2003). Alternatively, temporal coding schemes rely on the precise timing of individual spikes, and this potentially provides a faster and efficient way of processing and transmitting signals. In particular, the first spike after a stimulus (Panzeri et al., 2001; Johansson and Birznieks, 2004) is capable of reliably conveying considerable information. This inspired methods based on the time-to-first-spike (TTFS) coding, resulting in fewer spikes and efficient computation (Bonilla et al., 2022; Yu et al., 2023). In practice, most of these methods force each neuron to fire at most one spike (Mostafa, 2018; Kheradpisheh and Masquelier, 2020; Göltz et al., 2021; Mirsadeghi et al., 2021; Zhou et al., 2021; Comşa et al., 2022) or assume there is a very long refractory period after a spike (Kotariya and Ganguly, 2021) to allow the computation of exact derivatives of postsynaptic spike times with respect to presynaptic times. This means that these networks can only process static inputs (Mostafa, 2018; Zhou et al., 2021; Comşa et al., 2022; Sakemi et al., 2023), such as spikes converted from intensity of each pixel in images, but not a continuous stream of events. Hence, this type of single-spike encoding is not biologically plausible.

In addition, there have been limited research on investigating the temporal structures in neuromorphic data sequences and SNNs. In the context of spiking signals, temporal structures refer to the patterns, changes, or behaviors that occur over time in the generation and transmission of these signals. Data sequences containing rich temporal structures indicate that useful information is encoded in the temporal domain. As shown in Figure 1, various types of data produce varying degrees of temporal structures, which can yield diverse results when using different coding schemes. However, some widely used datasets lack diverse temporal structures in their sequences because events are generated by repeatedly moving a neuromorphic device around static images, such as neuromorphic MNIST (N-MNIST), N-Caltech101 (Orchard et al., 2015), and CIFAR10-DVS (Li et al., 2017). Results presented by Iyer et al. (2021) illustrated that rate-based SNNs outperform timing-based methods on N-MNIST dataset. The authors argued that spike timings of sequences

in N-MNIST may not contain too much useful information. Moreover, recent evidence (Jiang et al., 2023) indicates that timing-based computation is superior in the task involving abundant temporal information. As for TTFS coding, although some studies focused on event data of static scenes (Park et al., 2020; Kotariya and Ganguly, 2021), there have not been studies applying TTFS coding to real event sequences that exhibit rich temporal structures.

We, therefore, propose a novel decision-making scheme based on first-spike (FS) coding by encoding FS timings of output neurons for real-world event sequences with rich temporal structures, aiming to understand whether the timing of the FS plays a distinct role and whether it exploits the temporal information more effectively than FR. The FS coding differs from traditional TTFS-based studies in that information is conveyed without relying on exact timings, thereby eliminating the restriction of allowing each neuron to fire at most once. Instead, we only encode the timings of the first spike in the output layer and use it in the loss function for supervised learning. However, supervised learning involving precise timing poses several challenges. First, it is impractical to use continuous time encoding due to high computational cost caused by the substantial number of events generated by neuromorphic device. In addition, exact gradients of continuous spike times with respect to spike trains are ill-defined, which prevents the standard backpropagation process used in conventional artificial neural networks (ANNs). Some existing methods overcome this issue by estimating the derivatives of continuous time with respect to membrane potential around the threshold, such as SpikeProp (Bohte et al., 2002) and its variations (Xu et al., 2013; Shrestha and Song, 2016, 2017) and EventProp (Wunderlich and Pehle, 2021). Other methods utilize the relationship between time and membrane potential to achieve supervised learning of precise spike timing, such as probabilistic models of firing intensity (Pfister et al., 2006; Gardner et al., 2015) and implicit differentiation on the equilibrium state (Xiao et al., 2020, 2023). Recent methods have made attempts to calculate exact derivatives of the postsynaptic spike times with respect to presynaptic spike times (Comşa et al., 2022) or potential (Zhang et al., 2022). For example, Zhang et al. (2022) proposed a rectified linear postsynaptic potential function to alleviate problems such as non-differentiable spike function, exploding gradients and dead neurons during backpropagation in deep SNNs utilizing temporal coding. Most methods train the network to learn the timing of desired spike trains, restricting its adaptability to diverse input scenarios. Furthermore, the complicated rules of error propagation and the dependency between spike times in these methods limit their utilization in deep networks. Therefore, to simplify training and alleviate the restrictions for spike times, we exclusively apply discrete temporal coding to the output spikes in the final layer. In this way, we can concentrate on error propagation from output FS timings to subsequent spikes by leveraging the surrogate gradient learning (Wu et al., 2018; Neftci et al., 2019; Yin et al., 2021) for spikes. Specifically, the error of FS time in the output layer is propagated to multiple spikes through a Gaussian window, and then the SuperSpike method (Zenke and Ganguly, 2018), based on surrogate gradient descent, is utilized to achieve the supervised learning of spikes in the network. Additionally, this approach enables a flexible configuration of the network architecture, which



can include a combination of convolutional layers and fully connected (FC) structures with recurrent connections.

Another difficulty in training SNNs is dealing with neurons that fail to generate any spikes within the given time window, commonly referred to as inactive neurons. This issue is particularly predominant in training based on the first-spike time, as the error is only propagated through the first output spike. Consequently, the weights associated with subsequent spikes cannot be updated, leading to a lower firing rate and increasing inactivity in their neurons. Additional strategies are usually necessary to solve this problem, such as weight initialization (Bohte et al., 2002), large penalty on inactive neurons (Mostafa, 2018; Comşa et al., 2022), and synchronization pulses as temporal biases (Comşa et al., 2022). Hence, in this study, we design strategies to facilitate the training based on FS timings, specifically for event sequences. First, we assign the error of inactive neurons across multiple steps rather than just one. Second, the time window is enlarged by adding empty sequences to reduce the number of inactive neurons that generate spikes beyond the observed window due to significant time delay. Finally, to enhance the performance of FS coding, we apply smaller values of time constant and threshold in the initial layers to effectively extract local features, while we use large values in the final layers to facilitate decision-making based on previous stimuli.

In the experiments, we make a comprehensive comparison of FS and FR coding schemes on several commonly used visual and auditory neuromorphic datasets, including DVSGesture (Amir et al., 2017), SHD (Cramer et al., 2022), N-TIDIGITS (Anumula et al., 2018), and DVSPane (Afshar et al., 2019). These data sequences demonstrate different levels of temporal structures, as shown in Figure 1. Results show that FS coding achieves comparable accuracy with FR coding, although typically with a lower temporal delay. There is a trade-off between classification

accuracy and the first-spike latency. An appropriate temporal delay allows the network to make accurate decisions after receiving sufficient information. Furthermore, the FS models exhibit distinct neuronal behavior on different types of data sequences. In particular, the networks based on FS coding demonstrate enhanced performance and superior energy efficiency on audio data sequences with very rich temporal structures. On the other hand, when processing visual data sequences containing repetitive signals and rich spatial information, FS and FR models demonstrate similar neuronal dynamics and produce similar spike counts.

2. Materials and methods

Consider a stream of events emitted by a neuromorphic sensor, $\mathcal{E} = \{e_i, i = 1, 2, \dots, M\}$, over a certain time window. An event e_i in continuous space and time can be represented as a function $e_i(\mathbf{x}, t) = p_i \delta(\mathbf{x} - \mathbf{x}_i, t - t_i)$, which means that an event with polarity p_i is emitted at the location \mathbf{x}_i and at the timestamp t_i . The polarity $p_i = \pm 1$ represents whether the brightness change is positive or negative. To reduce computational cost, we transform \mathcal{E} into a discretized spatio-temporal representation E as the input of SNNs. The input tensor contains T temporal bins by accumulating raw events at a resolution of Δt . Each pixel location takes the number of positive or negative events within each temporal bin. In this setup, every pixel is associated with two channels to indicate the polarity of events. As a result, for a vision sensor with an image plane of dimensions $H \times W$, the input E forms a 4-D tensor of size $2 \times H \times W \times T$. As audio sensors have no polarity applied, the input E is represented as a 2-D tensor with dimensions $F \times T$, where F denotes the number of channels for the audio sensor.

In terms of the coding schemes of SNNs, spike trains can be encoded into different formats to convey information (Guo et al.,

2021). Figure 2A presents the comparison of spike-based coding schemes in decision-making. FR coding focuses on the average spike count within a certain time window. In population coding, several neurons in each population capture different features of input stimuli over time, and their responses are combined to make a decision (Panzeri et al., 2015). In burst coding, a burst of spikes is emitted at one time, in which information is carried in the spike count and the inter-spike interval within the burst (Izhikevich et al., 2003). Traditional TTFS coding restricts each neuron to fire at most once and information is conveyed in the exact timings, while our FS coding only focuses on the first spike of output neurons, since, in FS, the output neuron that fires first determines the classification outcome.

A standard multi-layer SNN architecture is used in this study, and is implemented using either convolutional layers or fully connected layers based on the required task. The overall dynamics is shown in Figure 2B. It takes the spatio-temporal representation of events E as the input, and each neuron generates a spike train of length T . In the output layer, FR or FS coding are used for classification. The models using FR or FS coding are denoted as the FR or FS model in the rest of the article. For the FR model, the predicted class is determined by the highest firing rate. For the FS model, output spike trains are encoded into temporal codes to obtain FS timing for each neuron. The predicted class in this case depends on the earliest spike across all the output neurons.

Our SNN model can be seen as a hybrid system in which multiple spikes are transmitted in hidden layers, but only the first output spike is utilized to make a decision. However, the information within the hidden layers not only depends on the firing rate (FR) of neurons but also considers the order in which the spikes occur. This is because the first-spike timing is emphasized in the output layer, introducing the aspect of spike order as an informative factor. This distinguishes it from standard FR coding, where only the spike count over a certain time window holds significance.

We introduce the SNN model with discrete time encoding in Section 2.1, the error propagation through FS timings in Section 2.2, and strategies facilitating the training based on FS timings in Section 2.3.

2.1. SNN model and time encoding

In this subsection, we introduce the current-based leaky integrate-and-fire (CUBA-LIF) neuron model in Section 2.1.1 and then extend it to a multi-layer SNN for event sequences in Section 2.1.2. In our SNN model, binary spikes are transmitted and processed between layers, and discrete time encoding is applied to the spike trains of the output layer to obtain the FS timings of the system.

2.1.1. Neuron model

One of the most commonly used neuron model is the CUBA-LIF neuron. Consider a set of presynaptic neurons $j = 1, 2, \dots, J$ connected to a postsynaptic neuron i , then the dynamics of the

CUBA-LIF model is as follows:

$$\begin{aligned} \tau_s \frac{dI_i(t)}{dt} &= -I_i(t) + \sum_j w_{ij} \sum_{t_{j,m} < t} \delta(t - t_{j,m}), \\ \tau_m \frac{dU_i(t)}{dt} &= -(U_i(t) - U_r) + I_i(t), \end{aligned} \quad (1)$$

where τ_m and τ_s are the time constant of membrane potential $U(t)$ and synaptic current $I(t)$, respectively, and $\delta(t)$ is the Dirac function representing a spike function. Here, $t_{j,m} < t$ is the firing time of the m th spike generated by the j th presynaptic neuron. Moreover, the synaptic weight between neurons i and j is denoted as w_{ij} , and U_r is the resting potential, where we set $U_r = 0$.

The condition that neuron i fires a spike is when the membrane potential $U(t)$ reaches a threshold θ . After spiking, the potential drops below U_r and then recovers to U_r within a refractory period. In our model, the refractory period is ignored, which means $U_i(t)$ is reset to $U_r = 0$ instantly.

2.1.2. SNN model with discrete time encoding

The CUBA-LIF neuron model is then discretized to construct a multi-layer SNN. Given an SNN model with L layers, the membrane potential $U^{l,n}$ is evolved through layers $l = 1, 2, \dots, L$ and time steps $n = 1, 2, \dots, T$. When the membrane potential of neuron i in layer l at time step n is greater than a threshold: $U_i^{l,n} \geq \theta$, a spike is generated and is denoted as $s_i^{l,n} = 1$. Otherwise $s_i^{l,n} = 0$.

We follow the same discretization scheme used by Neftci et al. (2019). The update of synaptic current $I_i^{l,n}$, membrane potential $U_i^{l,n}$, and spike firing from step n to $n + 1$ are as follows:

$$I_i^{l,n+1} = \beta I_i^{l,n} + \sum_{j=1}^{Q(l-1)} w_{ij} s_j^{l-1,n+1} + \sum_{k=1}^{Q(l)} v_{ik} s_k^{l,n+1}, \quad (2)$$

$$U_i^{l,n+1} = \alpha U_i^{l,n} (1 - s_i^{l,n}) + (1 - \alpha) I_i^{l,n}, \quad (3)$$

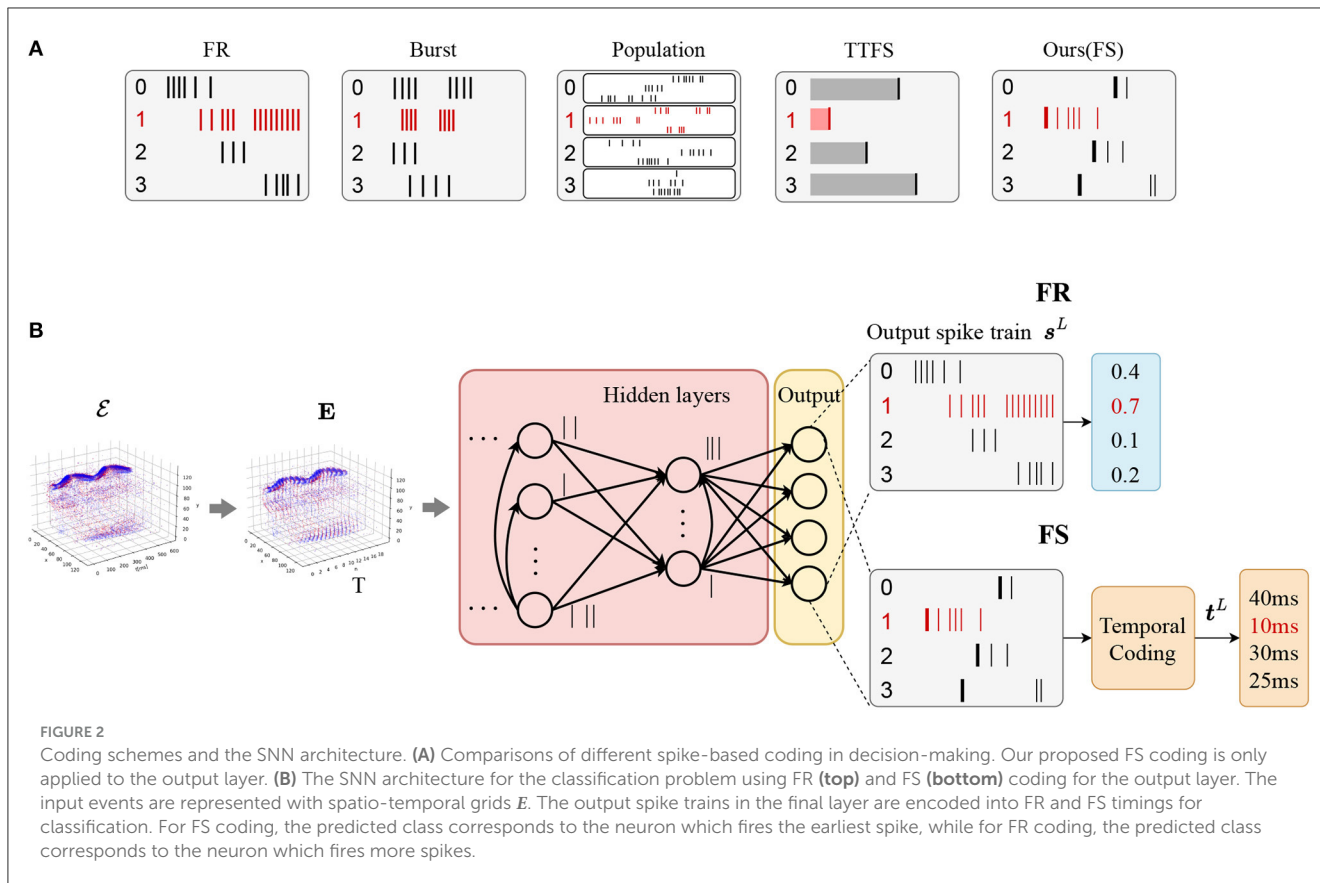
$$s_i^{l,n+1} = f(U_i^{l,n+1}), \quad (4)$$

where $\alpha = e^{-\frac{\Delta t}{\tau_m}}$ and $\beta = e^{-\frac{\Delta t}{\tau_s}}$. Moreover, in our models, $\tau = \tau_s = \tau_m$. The last term in Eq. (2) represents optional recurrent connections in the fully connected layer, where v_{ik} is the weight of a recurrent connection between the k th and i th neuron in the same layer l . The number of neurons in the l th layer is denoted as $Q(l)$. In Eq. (3), the membrane potential $U_i^{l,n}$ is reset by multiplying $1 - s_i^{l,n}$. Finally, the spiking process can be described as a step function of the membrane potential:

$$s = f(U) = \begin{cases} 1, & U \geq \theta, \\ 0, & U < \theta. \end{cases} \quad (5)$$

The forward propagation of a single neuron is shown as solid arrows in Figure 3A. The synaptic current, membrane potential, and spikes are updated in both spatial and temporal domains using Eqs (3), (4).

The output spike trains of the network can be encoded into different formats for classification. To obtain FS timings of neurons, temporal coding is required to obtain the spike timings. We, therefore, apply discrete time encoding for spike trains of neurons



in the last layer L . According to the time sequences, a spike is encoded as its time step, but it is unclear how to encode a silent step that does not generate a spike. Directly encoding it as infinity will cause an error in the computation of the loss. We instead replace it with a fixed large value, denoted as t^{inf} , which should be greater than $T\Delta t$.

Specifically, the output time of the i th neuron at step n is given by:

$$t_i^{L,n} = h(s_i^{L,n}) = \begin{cases} n\Delta t, & s_i^{L,n} = 1, \\ t^{inf}, & s_i^{L,n} = 0, \end{cases} \quad (6)$$

where

$$t^{inf} = (T + 1)\Delta t. \quad (7)$$

The discrete temporal coding process in the last layer L is illustrated in Figure 3B. Spikes are encoded into discrete timestamps in accordance with the time sequence, while other steps are encoded with t^{inf} .

Finally, the time of the first spike from the i th output neuron, denoted as $t_i^{\mathcal{F}}$, is given by the minimum of the temporal codes:

$$t_i^{\mathcal{F}} = \min_n(t_i^{L,n}). \quad (8)$$

However, another concern arises when all the neurons fail to fire within the time window. The network cannot make a prediction by the first spike, because it becomes challenging to determine which neuron fires the first. To solve this problem, we utilize the

maximum membrane potential over time of each neuron $U_i^{max} = \max_n(U_i^{L,n})$ to facilitate the prediction. The higher the membrane potential, the higher the likelihood the neuron would fire earlier. Therefore, the predicted label y^p corresponds to either the neuron that fires the first spike or the one with the highest membrane potential if all the output neurons are inactive.

2.2. Backpropagation through FS timings

In this section, we propose a supervised learning framework for FS coding. First, we define the loss function as the cross-entropy loss based on the FS times of the output neurons. We use this to minimize the FS time of the target neuron and maximize that of non-target neurons:

$$\mathcal{L}_{FS} = - \sum_{i=1}^C y_i \log \frac{\exp(-\alpha_0 t_i^{\mathcal{F}})}{\sum_{j=1}^C \exp(-\alpha_0 t_j^{\mathcal{F}})} + \lambda_t \sum_{\{i|t_i^{\mathcal{F}} > T\Delta t\}} y_i [\exp(\beta_0 t_i^{\mathcal{F}}) - 1], \quad (9)$$

where y_i is the target label (0 or 1) of the i th class, C is the number of output neurons (classes), and α_0, β_0 and λ_t are constant coefficients, where α_0 is used to control the speed of training and prevent the exponential function from taking an excessively high value. The second term is a constraint to penalize a target neuron which never fires. For ease of notation, we use \mathcal{L} to represent \mathcal{L}_{FS} in the rest of the article.

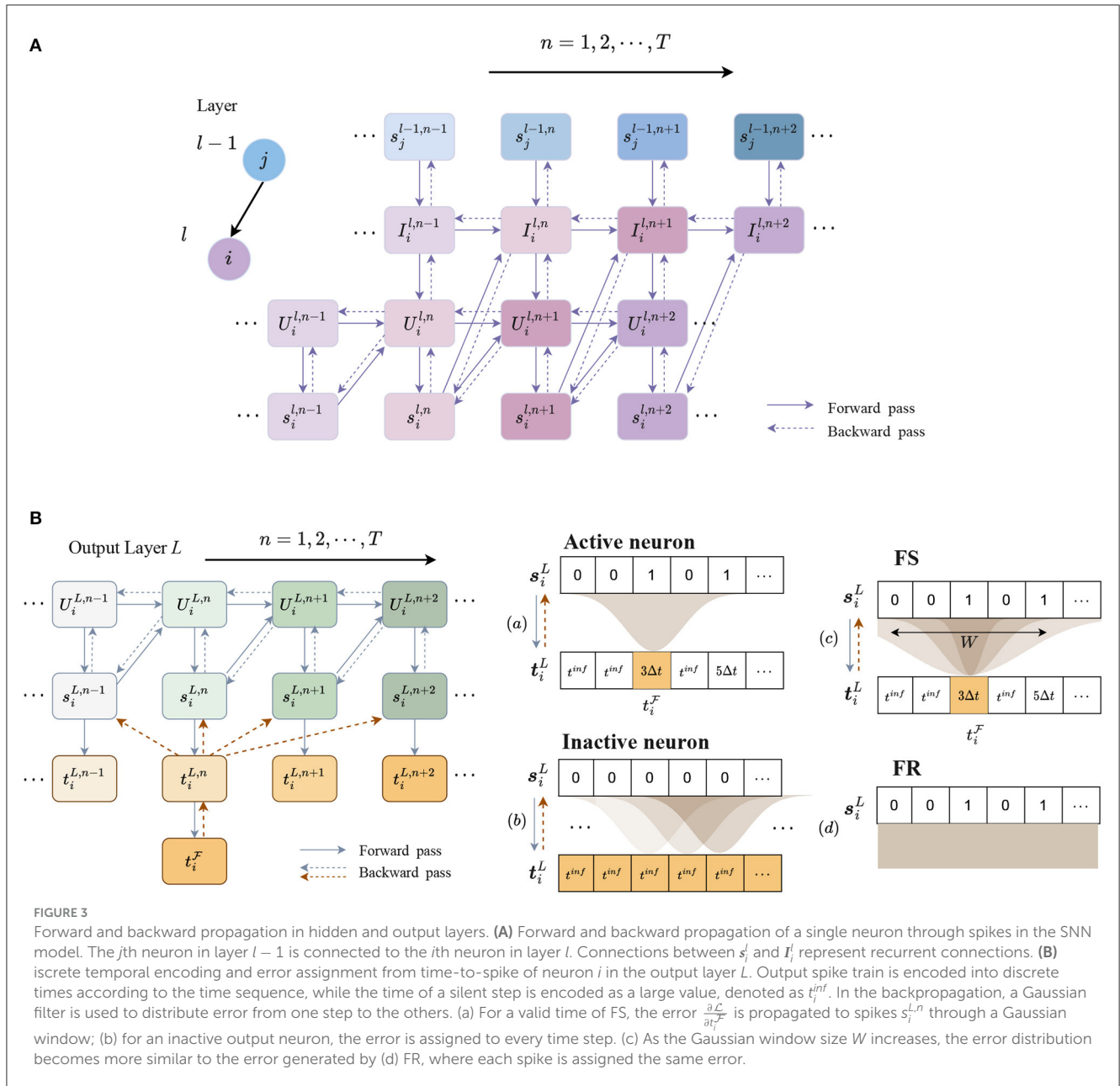


FIGURE 3

Forward and backward propagation in hidden and output layers. (A) Forward and backward propagation of a single neuron through spikes in the SNN model. The j th neuron in layer $l - 1$ is connected to the i th neuron in layer l . Connections between s_j^l and I_i^l represent recurrent connections. (B) discrete temporal encoding and error assignment from time-to-spike of neuron i in the output layer L . Output spike train is encoded into discrete times according to the time sequence, while the time of a silent step is encoded as a large value, denoted as t_i^{inf} . In the backpropagation, a Gaussian filter is used to distribute error from one step to the others. (a) For a valid time of FS, the error $\frac{\partial \mathcal{L}}{\partial t_i^F}$ is propagated to spikes $s_i^{L,n}$ through a Gaussian window; (b) for an inactive output neuron, the error is assigned to every time step. (c) As the Gaussian window size W increases, the error distribution becomes more similar to the error generated by (d) FR, where each spike is assigned the same error.

To compare FS with FR coding, similarly, a cross-entropy loss function maximizing the FR of the target neuron is used and is given by:

$$\mathcal{L}_{FR} = - \sum_{i=1}^C y_i \log \frac{\exp(\alpha_1 f_i)}{\sum_{j=1}^C \exp(\alpha_1 f_j)}, \quad (10)$$

where α_1 is a constant value and $f_i = \frac{1}{T} \sum_{n=1}^T s_i^{L,n}$ is the spike rate within T steps of the i th neuron in the last layer L . Since the training of FR usually does not suffer from inactive neurons, the constraint used in Eq. (9) is not required here.

To learn the weights $W = \{w_{ij}^l\}$ and $V = \{v_{ik}^l\}$, since the discretization described in Eqs (3), (4) effectively leads to an SNN as visualized in Figure 3A, we can simply perform a standard error backpropagation same as in conventional ANNs. However,

there remains two challenges for the learning process based on FS timings. First, we have to propagate the error from the FS time t_i^F to the postsynaptic spikes $s_i^{L,n}$ at the output layer. We, therefore, introduce a novel error assignment scheme in Section 2.2.1. Another obstacle is the non-differentiability of the spike function in Eq. (4), which describes the relationship between membrane potential and postsynaptic spikes. This can be overcome by using a surrogate gradient to approximate that of the step function (see Section 2.2.4).

2.2.1. Error propagation from FS times to spikes

The error of the FS time t_i^F is computed by the loss function, denoted as $\frac{\partial \mathcal{L}}{\partial t_i^F}$. To enable the flow of error throughout the entire network, it is necessary to propagate the error for the single (first)

step to all the associated spikes in the last layer. This process is divided into two steps: first, from the FS time $t_i^{\mathcal{F}}$ to all steps $t_i^{L,n}$, and then from times $t_i^{L,n}$ to spikes $s_i^{L,n}$. According to the chain rule, the error of the spike in the output layer can be computed based on the error of the FS time and related gradients as follows:

$$\frac{\partial \mathcal{L}}{\partial s_i^{L,n}} = \frac{\partial \mathcal{L}}{\partial t_i^{\mathcal{F}}} \sum_{m=1}^T \frac{\partial t_i^{\mathcal{F}}}{\partial t_i^{L,m}} \frac{\partial t_i^{L,m}}{\partial s_i^{L,n}}. \tag{11}$$

The key issue is to compute the gradient $\frac{\partial t_i^{\mathcal{F}}}{\partial t_i^{L,m}}$ and $\frac{\partial t_i^{L,m}}{\partial s_i^{L,n}}$. Figure 3B illustrates the error assignment through these two steps.

2.2.2. From FS times to temporal codes

First, for active neurons, as the FS time is given by the minimum of the temporal codes, the error of FS time is only related to the corresponding time step, which means that the derivative of the FS time with respect to temporal codes $t_i^{L,n}$ is 1 only when $t_i^{L,n}$ is the first-spike time. Specifically,

$$\frac{\partial t_i^{\mathcal{F}}}{\partial t_i^{L,n}} = \begin{cases} 1, & t_i^{L,n} = t_i^{\mathcal{F}}, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

However, for inactive neurons, propagating the error through a single step will make the weights difficult to update. To address this issue, the error $\frac{\partial \mathcal{L}}{\partial t_i^{\mathcal{F}}}$ is assigned to all the other steps for inactive neurons, which means $\frac{\partial t_i^{\mathcal{F}}}{\partial t_i^{L,n}}$ is always equal to 1. These two cases for active and inactive neurons are illustrated in Figures 3B(a, b).

Note that the strategy for inactive neurons has opposite effects on target and non-target neurons. This contrast arises from the fact that the loss function aims to minimize the first-spike time for target neurons while maximize it for non-target neurons. Hence, assigning error to all time steps is equivalent to minimizing (maximizing) the total firing time for target (non-target) neurons. Consequently, this strategy promotes the activation of dormant target neurons, while reinforcing the inactivity of non-target neurons that are already inactive.

2.2.3. From temporal codes to spikes

The second gradient $\frac{\partial t_i^{L,m}}{\partial s_i^{L,n}}$ cannot be computed directly. In inference, temporal codes are exclusively linked to their corresponding spikes at a single step. The timestamp $t_i^{L,n}$ is generated from $s_i^{L,n}$ at the same step n . However, it is essential to involve spikes occurring around step n ($m \neq n$) in the optimization of spike time at step n . During the learning process, the change of spike times results in the change of connections. If the optimization only focuses on a single spike at step n and its corresponding weights, the weight update would be inherently unstable. In the case of target neurons, the learning process optimizes not only the spike at step n but also the spikes and associated weights from earlier steps ($m < n$) to reduce output times. For non-target neurons, optimization should also involve spikes and related weights from later steps ($m > n$). Furthermore, since spike times can only change at most a few time steps at each iteration, the impact of error at step

n diminishes as the time step is far away from n . This means that spikes closer to step n should receive a larger error assignment.

Therefore, a surrogate gradient needs to be designed to distribute error from $t_i^{L,m}$ to $s_i^{L,n}$. A Gaussian window is used to ensure a smooth and gradual weight update. In addition, the error of spikes should have an opposite sign to the error of times. The reason is that decreasing the spike time is equivalent to increasing the probability of spike firing in early steps, in other words, increasing the value of spikes from 0 to 1. Specifically, our approach is to distribute the error of time at step m to the spikes around it through a negative Gaussian window:

$$\frac{\partial t_i^{L,m}}{\partial s_i^{L,n}} = g(m - n), \tag{13}$$

where $g(x)$ is given by:

$$g(x) = -\frac{A}{2\pi\sigma} e^{-\frac{x^2}{2\sigma^2}}, \tag{14}$$

where $A > 0$ is the amplitude and $\sigma = \lfloor \frac{T}{D} \rfloor$ is the standard deviation, determined by the length of the sequence T and a constant factor D .

The width of Gaussian window is determined by 3-sigma limit and the length of whole sequence, $W = \min\{6\sigma + 1, T\}$. As depicted in Figure 3B(c), when the factor D increases, both the standard deviation σ and the window size W decrease, which means that the error assignment only focuses on the time steps surrounding the first spike. When $D \rightarrow \infty$, $\sigma \rightarrow 0$, and $W \rightarrow 1$, the error is only assigned to the current step. On the contrary, when $D \rightarrow 0$, $\sigma \rightarrow \infty$, and $W \rightarrow T$, the error is propagated to all the steps with approximately the same value, which is similar to the error propagation of the loss based on FR [see Figure 3B(d)]. The parameters A and D are determined empirically. The value of A should not be too small, usually around $2T$, otherwise the training is slow with a deteriorated performance. The values of D and window size W have a significant impact on the performance which are discussed further in Section 3.5.

2.2.4. Surrogate gradient descent training through spikes

After propagating error from FS times to spikes in the output layer, the error can be propagated through spikes in the rest of the network. To solve the non-differentiability of the spike function, a surrogate gradient (Zenke and Ganguli, 2018) is used to estimate the derivative of postsynaptic spikes with respect to membrane potential. Specifically, the gradient of the step function in Eq. (5) is estimated using a fast sigmoid function in the backward pass:

$$f'(U) \approx \frac{1}{(1 + \rho|U - \theta|)^2}. \tag{15}$$

Having the error assignment from FS times to spikes and surrogate gradients of spike function, the overall backpropagation pipeline can be constructed. As shown in Figures 3A, B, the error flows from time $\frac{\partial \mathcal{L}}{\partial t_i^{L,n}}$ to spike $\frac{\partial \mathcal{L}}{\partial s_i^{L,n}}$ in the last layer, given by Eqs (12), (13). In each layer, the error of spikes $\frac{\partial \mathcal{L}}{\partial s_i^{L,n}}$ is propagated to membrane potential $\frac{\partial \mathcal{L}}{\partial U_i^{L,n}}$ through the surrogate gradient in

Eq. (15) and then the error of synaptic current $\frac{\partial \mathcal{L}}{\partial I^{l,n}}$ can be calculated. Finally, the derivative of error with respect to weights W^l, V^l in each layer are calculated by taking the derivative of Eq. (2):

$$\frac{\partial \mathcal{L}}{\partial W^l} = \sum_{n=1}^T \frac{\partial \mathcal{L}}{\partial I^{l,n}} \frac{\partial I^{l,n}}{\partial W^n} = \sum_{n=1}^T \frac{\partial \mathcal{L}}{\partial I^{l,n}} s^{l-1,n}, \quad (16)$$

and

$$\frac{\partial \mathcal{L}}{\partial V^l} = \sum_{n=1}^T \frac{\partial \mathcal{L}}{\partial I^{l,n}} \frac{\partial I^{l,n}}{\partial V^n} = \sum_{n=1}^T \frac{\partial \mathcal{L}}{\partial I^{l,n}} s^{l,n}. \quad (17)$$

2.3. Strategies facilitating training based on FS timings

In addition to the computation of gradients, parameter initialization is also crucial to the training of SNNs. Time constant τ , threshold θ , temporal resolution Δt , the length of the sequence T , and weight initialization have a great impact on the results. In addition, training based on FS timings is more challenging because only focusing on the first spike leads to more inactive neurons during training. Thus, apart from the error assignment from time-to-spike described in Section 2.2.1, other strategies can be used in parameter settings to facilitate the training process.

2.3.1. Different time constants and thresholds for feature extraction and decision

An appropriate time constant τ and threshold θ can enhance the performance of the system, which determine the firing rate and neuron activity in the system. Empirically, we found that smaller values of τ and θ in the first few layers but larger values for the final layers leads to better performance on FS coding, which is consistent with our intuition. Figure 4A illustrates the responses of neurons with different τ and θ to the same input sequences. The neuron with a small value of τ has a short memory due to the rapid decay of its membrane potential. Meanwhile, a small θ is used to maintain a high firing rate, thereby facilitating the transmission of sufficient information. Therefore, small values of τ and θ are well-suited for capturing local features in the initial layers.

On the contrary, the final layer requires a longer delay to make a correct decision after enough information is accumulated. We can see from Figure 4A that large values of τ and θ can help keep a longer memory of previous stimuli, ensuring the target output neuron fires the first spike after receiving enough spikes from previous layers. Experiments in Section 3.3 confirm that an appropriate longer time delay of the first spike leads to higher accuracy of prediction.

2.3.2. Extension of time window with empty sequences

One of the challenges in training FS-coded model is dealing with inactive neurons. Having too many inactive neurons in the network will stop the gradient flow and hamper the update of weights, making the training more difficult. In the output layer,

particularly, there is no precise timing for the optimization of inactive neurons. However, we found that some of them are only inactive due to delay between layers which leads to spikes beyond the observed time window, as visualized in Figure 4B.

We thus extend the input window by appending an empty sequence to the end during training. We can see from Figure 4B that the empty sequence allows the output neuron to fire a later spike outside the original window. The length of empty sequence T_E is determined by time constant τ of the last layer empirically. Usually, a network with a larger time constant is more likely to exhibit a longer delay, therefore, a larger T_E should be used in that case. Note that this strategy is only used to facilitate training, and it is useful to enhance accuracy when τ in the last layer is very large and the original window size is relatively small. The downside of this approach is a higher computation workload and a longer delay of the target FS (see Section 3.3). For the experiments in Section 3, we set $T_E = 0$ unless otherwise specified.

3. Results

3.1. Experimental settings

We compare models using FS and FR coding schemes on classification tasks. We avoid using unrealistic datasets, such as N-MNIST (Orchard et al., 2015) and CIFAR10-DVS (Li et al., 2017), in which the first spike is not meaningful because events are generated by moving neuromorphic device around static images and there are no significant temporal differences in their sequences (Iyer et al., 2021). Instead, we test our model on realistic datasets in which important information is encoded in spike timings. For visual datasets DVSGesture (Amir et al., 2017) and DVSPPlane (Afshar et al., 2019), events are generated by real cameras capturing dynamic scenes. For auditory datasets N-TIDIGITS (Anumula et al., 2018) and SHD (Cramer et al., 2022), spikes are derived by converting existing datasets using a neuromorphic device or a realistic simulator. As shown in Figure 1, the temporal structures of event sequences in these datasets are different. For example, auditory signals in SHD and N-TIDIGITS are non-repetitive and the temporal complexity is much higher than visual datasets, since spatial information is crucial in the prediction of visual data. In addition, signals in DVSGesture contain repetitive information, in which a person performs the same gesture several times. By contrast, DVSPPlane exhibits more temporal complexity since the movement is not periodic. We observed that FS and FR models exhibit different behavior on these signals with varying temporal structures.

In our settings, the architecture and parameters vary among different tasks. Note that our aim is to make a comparison between FR and FS codings rather than seeking highest accuracy, hence relatively simple architectures are used in our study. The notation of architecture remains consistent throughout the article. For example, [2,32,32]-32C5S2-P2-64C3-FC128(R)-10 represents a network with a [2,32,32] input, where the first convolutional (C) layer contains 32 kernels (5×5) with a stride (S) of 2, followed by a max pooling (P) (2×2) and 64 convolutional kernels (3×3) with a stride of 1 (default), and finally a fully connected (FC) layer with 128 neurons with 10 recurrent (R) connection and

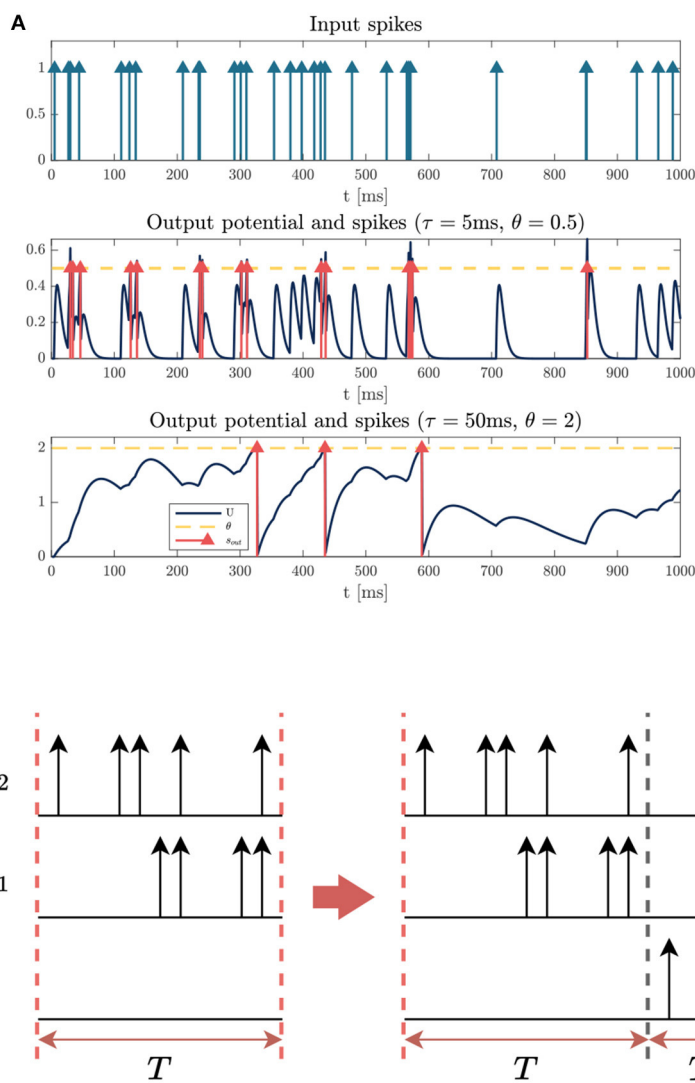


FIGURE 4 Illustrations of strategies to facilitate training with the FS loss. **(A)** Dynamics of neurons with different time constant τ and threshold θ . **(Top)** Input spike sequence (all the weights are equal to 1). **(Middle)** The neuron with $\tau = 5\text{ms}, \theta = 0.5$ responds rapidly to local features. **(Bottom)** The neuron with large parameters $\tau = 50\text{ms}, \theta = 2$ keeps a longer memory of input signals. **(B)** Illustration of how an active neuron with a large time delay can be seen as inactive. We can see that there exists a delay for spikes propagating down the network, causing the neuron to fail to fire a spike within the input window. By extending the time window with an empty sequence, the inactive neuron becomes active.

output classes. Weights are all initialized using Xavier uniform distribution, and the Adam optimizer with a weight decay of $1e^{-4}$ is used. The parameter of surrogate gradient ρ is set to 5 in all the cases. Notations for learning rate, batch size, and the number of epochs are η , B , and N_{ep} , respectively. We denote time constant and threshold as τ_1 and θ_1 for those used in feature extraction, and as τ_2 and θ_2 for those used in decision process. In a convolutional SNN, τ_1/θ_1 are used in convolutional and pooling layers, and τ_2/θ_2 are used in FC layers. For FC architecture, τ_1/θ_1 are used in hidden layers while τ_2/θ_2 are used in the output layer. In our experiments, $\tau_1/\theta_1/\theta_2$ are determined empirically to obtain the optimal results. The value of τ_2 affects time delay and accuracy significantly, whereas the value of θ_2 does not have a great impact on the performance when τ_2 is fixed. We, therefore, focus on τ_2 and test different values with $\tau_2 = \mu\tau_1$. Further details are introduced in the following and in Table 1.

3.1.1. DVSGesture dataset

DVSGesture (Amir et al., 2017) is captured by DVS128 camera, including 11 hand gestures recorded under three different lighting conditions. Since gestures in the last class are random, we only take the first 10 classes. Recordings are split into 1,078 and 264 samples for training and testing, respectively. Each sequence is around 6 s, we clip 1.2 s in training and 2.5 s for testing, with a temporal resolution $\Delta t = 10$ ms. The frame size is 128×128 , which is downsampled by 4 for the input. The architecture is [2,32,32]-64C3-128C3-P2-128C3-P2-FC128(R)-10.

3.1.2. SHD dataset

Spiking Heidelberg Dataset (SHD) is a dynamic audio dataset generated using Lauscher, an artificial cochlea model, including 20 classes of spoken digits from 0 to 9 in both German and English

TABLE 1 Hyperparameters for different tasks.

	Neuron					FS				FR	Training		
	Δt	τ_1	θ_1	θ_2	α_0	λ	β_0	A	D	α_1	η	N_{ep}	B
DVSGesture	10 ms	50 ms	0.5	1	0.1	0.01	0.02	300	4	10	$1e^{-4}$	40	16
SHD	10 ms	50 ms	5	10	0.2	0.01	0.02	200	16	20	$1e^{-3}$	80	128
N-TIDIGITS	5 ms	25 ms	1	2	0.1	0.01	0.02	500	16	20	$1e^{-3}$	200	128
DVSPPlane	2 ms	10 ms	0.5	2	0.1	0.01	0.02	500	8	15	$3e^{-4}$	50	16

languages. A total of 7,736 samples are used for training and 2,264 samples for testing. Each sequence is around 1 s. We clip 0.8 s and 1 s in training and testing, respectively, and $\Delta t = 10$ ms. The architecture is 700-FC256(R)-20, in which the time constant in the hidden layer is initialized with τ_1 but related variables α , β are trainable using the method by Perez-Nieves et al. (2021).

3.1.3. N-TIDIGITS dataset

N-TIDIGITS Dataset (Anumula et al., 2018) transforms TIDIGITS dataset into spikes with the dynamic audio sensor, CochleaAMS1b. It includes 11 classes of spoken numbers 0–9 and a word “oh.” The length of sequences is around 0.08–2.46 s but most of the sequences are <1.2 s. $\Delta t = 5$ ms and $T = 250$ are used in all sequences. Training and testing datasets include 2,463 and 2,486 samples, respectively. The architecture is 64-FC256(R)-FC256(R)-11, in which time constant in hidden layers are trainable.

3.1.4. DVSPPlane dataset

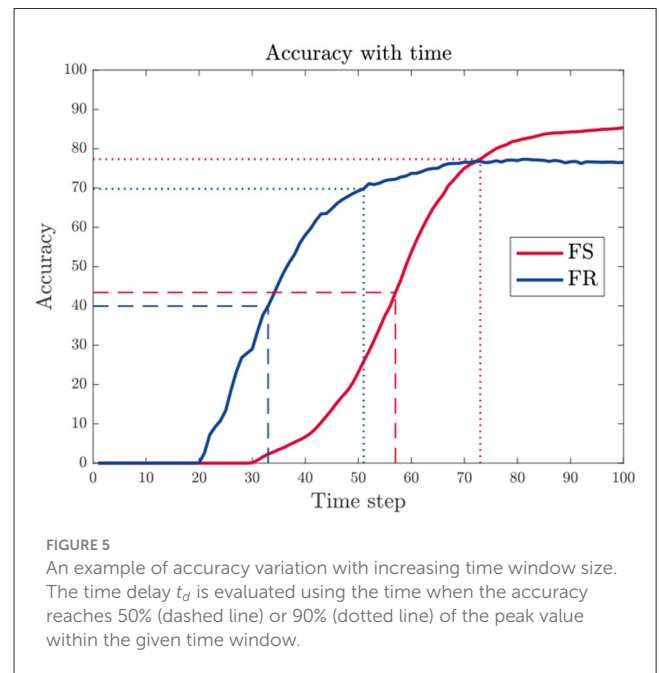
DVSPPlane dataset (Afshar et al., 2019) is captured by an asynchronous time-based image sensor (ATIS). Here, four different airplane models are dropped free-hand from varying heights and distances in front of the camera. The length of sequences is 242 ± 21 ms. We set $\Delta t = 2$ ms and $T = 100$ in training, and $T = 120$ in testing. The 800 samples are split into 640 and 160 samples for training and testing. The image with a size of 304×240 is downsampled by 4 as input. The architecture is [2,76,60]-32C5S2-64C3-P2-128C3-P2-FC256(R)-4.

3.1.5. Evaluation

The results are compared in terms of accuracy, time delay, and spike count. If not specified, the accuracy of the FS or FR model is evaluated in a manner that is consistent with its training.

As shown in Figure 5, the accuracy increases with longer time window. We, therefore, evaluate time delay as time when reaching 50 or 90% of the peak accuracy within the given time window, denoted as $t_d(50\%)$ and $t_d(90\%)$, respectively.

In terms of energy consumption, many studies calculate the number of synaptic operations as a measure. However, these metrics are usually employed when comparing power consumption between SNNs and ANNs, or between different architectures. They usually involve multiplying the number of connections by spikes (Wu et al., 2022), or factoring in firing rate, time step, and FLOPs (Zhou et al., 2023). Nonetheless, in our settings, we only change the coding scheme in the output layer while maintaining



the same architectures. In addition, FS coding scheme does not introduce additional operations during inference. Therefore, energy consumption mainly depends on the spike count in the system. Here, we use the average number of spikes per neuron in the system (denoted as N_s) to evaluate power consumption.

Furthermore, to test the required number of spikes for correct classification, an optional constraint for the spike count is added in the loss function to constrain the total number of spikes in the system, i.e., $\mathcal{L}_s = \lambda_s |N_s - \tilde{N}_s|$, where \tilde{N}_s is the target average spike count per neuron and λ_s is a constant value. The range of λ_s extends from 0.1 to 3, and we empirically adjusted it for various FS and FR models. Results under different spike count constraints are discussed in Sections 3.3 and 3.4.

3.2. Results overview

We compare the results of models using FS and FR codings, respectively, on the four datasets. During the training of FS models, an empty sequence is added with lengths $T_E = 40$ and $T_E = 20$ for DVSGesture and SHD data sequences, respectively. As shown in Table 2, we can see that the FS model demonstrates comparable performance with the FR model. We also tested different time

constant τ_2 for decision layers. For FS models, a larger τ_2 leads to a higher accuracy on both metrics, but the time delay t_d also increases. In contrast, for FR models, τ_2 does not affect the accuracy significantly. It seems that a longer first-time latency encodes more information. The relationship between accuracy and time delay is further discussed in Section 3.3. In addition, the response time t_d of the FR model is shorter than the FS model overall. Note that the disparity between time delays for FR and FS models is large when reaching 50% accuracy, but the gap is reduced when achieving 90% of the best performance in some cases, especially for models with small time constants.

One benefit of using FS coding is the reduction in the number of spikes, thereby leading to better energy efficiency. We can see from Table 2 that the number of spikes N_s in an FS system is usually smaller than the number in an FR system, especially for SHD and N-TIDIGITS classification. Further experiments demonstrate that FS models are robust with fewer spikes, as shown in Section 3.4.

Furthermore, we generated the output spike raster plots to analyze the neuronal activities of the FS and FR systems. As depicted in Figure 6, non-target neurons of an FS model fire fewer spikes than an FR model, which reduces the likelihood of misclassification based on FR. The reason is that the first time of the non-target neuron is optimized to the end of the sequence, which reduces the probability of firing in the whole sequence significantly. This phenomenon allows the FS model to make decisions based on FR as well. Further results on data sequences with different temporal structures are analyzed in Section 3.5.

To validate the effectiveness of FS coding, further experiments are conducted on models using adaptive LIF neurons in Section 3.6.

3.3. Trade-off between accuracy and time delay

During training, we found that it is easier to train FS models when time constant is relatively large and a proper empty sequence is added. Results also indicate that there is a trade-off between accuracy and response time. The two main factors that affect time delay, time constant and length of empty sequences, are analyzed in the following.

3.3.1. Time constant

As observed from Table 2, the FS model with a larger time constant τ_2 in the last layers leads to higher accuracy and fewer spikes in the system overall, but at a cost of longer time delay.

3.3.2. Time window and empty sequence

Another factor that affects time delay is the length of the input time window, which is determined by the original length of data T and the length of added empty sequence T_E . First, we used a fixed T and tested different T_E values on the DVSGesture dataset. The model with $\tau_2 = 12\tau_1$ is tested because the empty sequence is added only when the original window size T is relatively small for a long time delay (large τ_2). As shown in Table 3, an increasing T_E leads to a longer time delay and a higher overall accuracy.

Furthermore, different length T of input data is tested for the model with $\tau_2 = \tau_1$, where $T_E = 0$ for all the trials. The second column in Table 3 shows that the accuracy is higher with a larger window size.

3.3.3. Accuracy vs. time delay under spike count constraints

We further imposed different spike count constraints to FS and FR models with different time constant τ_2 . The time delay and corresponding accuracy of each trial are displayed in Figure 7. We can see that the time delay is mainly determined by time constant. FR models usually exhibit shorter time delay than FS models, except for DVSPPlane. It is clear that FS models for SHD and N-TIDIGITS datasets achieve higher accuracy with longer time delay. The spike raster plots of SHD and N-TIDIGITS classifications in Figure 6 also indicate that FS models achieve higher accuracy with longer time delays and fewer spikes, while FR models respond faster but more spikes produced by non-target neurons interfere with the classification. For DVSGesture and DVSPPlane datasets, although the relationship between FS and FR models is not obvious, the best FS results are generated when we have larger temporal latency.

It is also worth noting that there are points with longer t_d but lower accuracy under the same time constant. These points represent the models with smaller number of spikes, which highlights that time delay tends to be longer with fewer spikes. Overall, fewer spikes in the system usually lead to a lower accuracy and longer time delay. The relationship between accuracy and spike count is further discussed in Section 3.4.

To summarize, appropriate time delay ensures that the first spike makes a correct decision. In other words, the first spike encodes more information with a longer time delay. However, a model with a larger τ_2 has a risk of lacking sufficient spikes to make a decision due to a lower output firing rate. In addition, it becomes more difficult to improve the accuracy as the cost of time delay increases, because the accuracy is not only determined by time delay but also limited by the model itself and other factors.

3.4. Energy efficiency

In SNNs, the power consumption mainly depends on the mean spike activity and the number of synaptic operations (Parameshwara et al., 2021). When deploying SNNs on neuromorphic hardware such as the Intel Loihi (Davies et al., 2018), reducing the number of spikes in the systems could lead to gains in energy efficiency.

Table 2 illustrates that FS systems produce fewer spikes than FR systems overall, especially on SHD and N-TIDIGITS datasets. DVSGesture and DVSPPlane classifications with CNN architecture generate approximately the same number of spikes in both systems. The FS system with a larger time constant is usually more energy efficient, while the FR system generates fewer spikes with a smaller τ_2 .

As mentioned in Section 3.3, reducing the number of spikes through a spike count constraint results in a decline in performance. Figure 8 presents the relationship between accuracy

TABLE 2 Comparison of FR and FS models with different time constant τ_2 for decision layers ($\tau_2 = \mu\tau_1$) in terms of accuracy (Acc), spike count (N_s), and time delay (t_d).

	μ	Trained using FR				Trained using FS			
		Acc(%) \uparrow	N_s \downarrow	$t_d(50\%)$ \downarrow	$t_d(90\%)$ \downarrow	Acc (%) \uparrow	N_s \downarrow	$t_d(50\%)$ \downarrow	$t_d(90\%)$ \downarrow
DVSGesture	1	94.0	10.0	36.2	58.6	90.1	10.2	38.2	56.8
	4	94.2	9.5	41.6	58.8	89.7	9.6	41.8	59.0
	12	95.1	11.0	45.4	60.2	92.8	8.9	61.6	77.6
SHD	1	79.3	26.8	28.8	46.2	78.3	24.9	33.8	47.2
	4	78.8	31.0	30.8	47.2	85.5	12.6	48.4	67.0
	12	77.8	35.7	33.0	49.4	87.6	9.0	58.0	74.8
N-TIDIGITS	1	88.2	10.6	74.4	133.0	87.6	12.1	88.0	129.4
	4	88.7	11.4	76.6	136.8	89.3	5.4	94.4	137.8
	8	88.1	11.5	80.6	139.4	89.3	5.2	98.2	144.0
DVSPPlane	1	83.0	1.3	57.6	84.4	87.6	2.0	52.0	65.2
	4	85.4	1.5	68.0	85.8	90.5	2.0	54.6	67.6
	8	89.3	1.8	67.8	82.8	92.7	1.9	62.0	77.0

The results are the average of 5 trials. The bold values represent the best results obtained when comparing the FR and FS systems for each dataset.

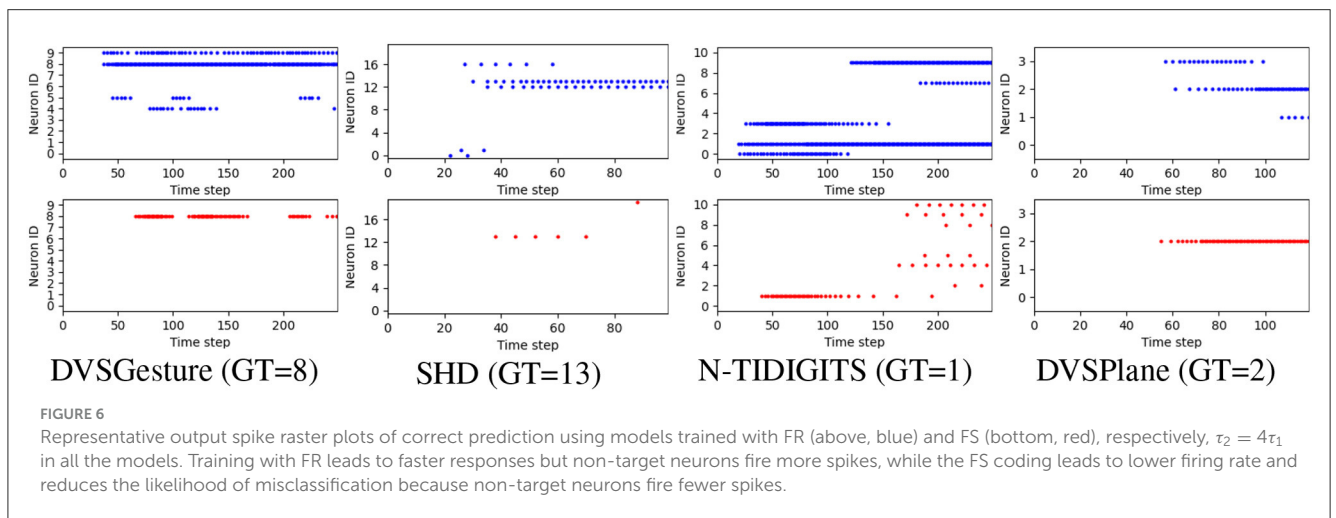


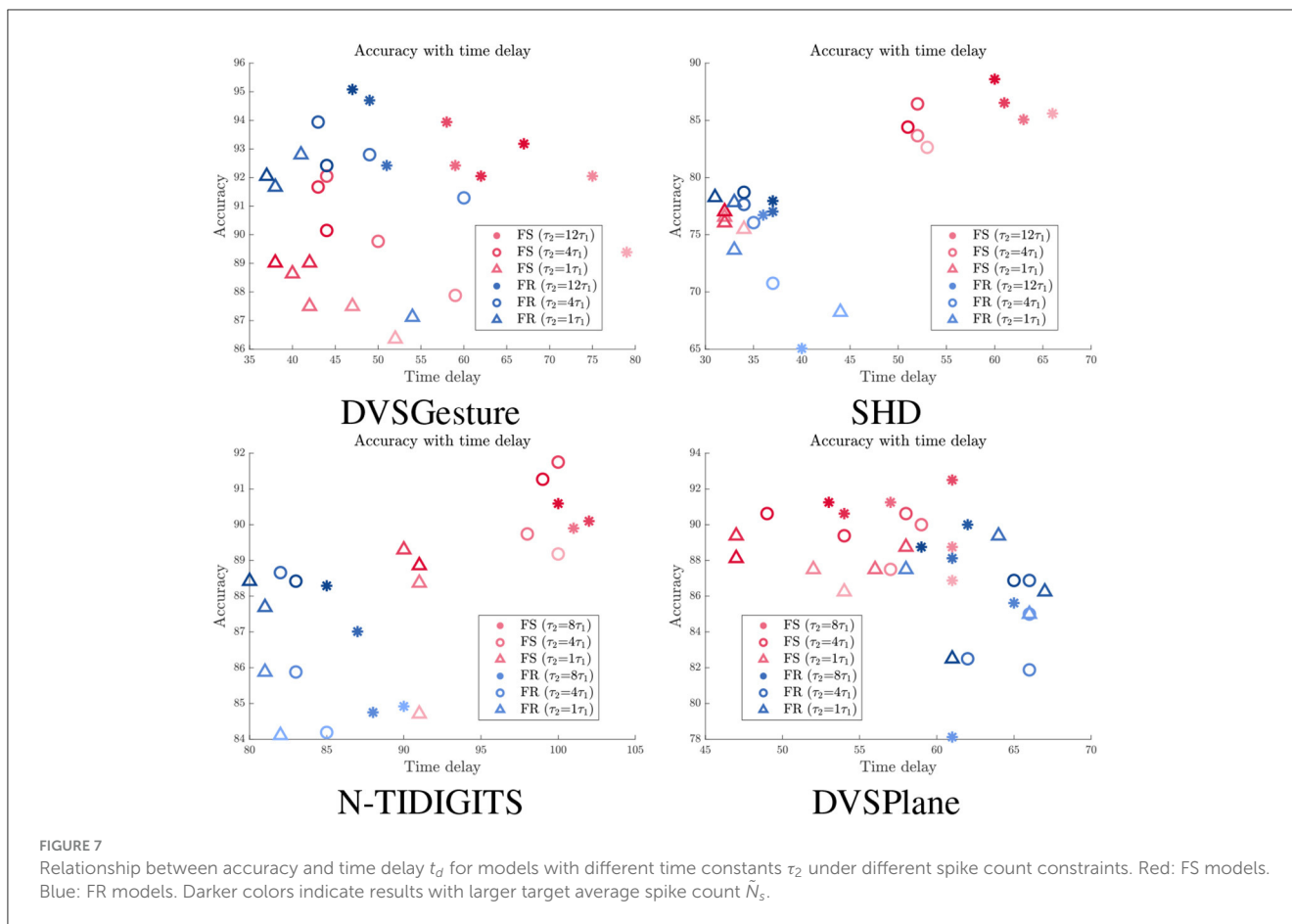
TABLE 3 Results of DVSGesture classification with different length of data sequence T and added empty sequence T_E ($N_{ep} = 50$).

T_E	$T = 120, \tau_2 = 12\tau_1$			$T_E = 0, \tau_2 = \tau_1$			
	Acc(%) \uparrow	$t_d(50\%)$ \downarrow	$t_d(90\%)$ \downarrow	T	Acc (%) \uparrow	$t_d(50\%)$ \downarrow	$t_d(90\%)$ \downarrow
0	92.4	59	82	60	81.1	29	35
20	93.6	64	95	80	86.0	35	44
40	93.6	65	97	100	88.3	36	48
60	93.9	66	99	120	90.9	38	53
80	93.9	77	114	140	89.4	48	82

A larger window size $T + T_E$ leads to higher accuracy but a longer time delay. The bold values represent the best results in each column.

and average spike count N_s for models with different time constants τ_2 , in which red and blue curves represent results of FS and FR models, respectively. We can see that the accuracy of FR models decreases significantly overall, while the FS modes is more stable with fewer spikes, especially for SHD and N-TIDIGITS

tasks. For DVSGesture dataset, the original spike counts of FS and FR models are close, so accuracy of both models drops as the spike count decreases. Note that the original N_s of FS models is larger than FR models in DVSPPlane classification, but the accuracy of FR models decreases more significantly.



In addition, the FS model with a large τ_2 is more robust to reduced number of spikes. Overall, FS models with a large time constant are more energy efficient and more robust to the spike count constraint.

3.5. Performance and behavior on event sequences with different temporal structures

From Table 2, we can see that the performance varies on different datasets. Overall, the FS system model outperforms the FR model on SHD/N-TIDIGITS/DVSPPlane datasets, whereas its performance on DVSGesture task is relatively inferior. These differences are due to different temporal structures in the input sequences. As illustrated in Figure 1, in the DVSGesture dataset, the same gesture is repeated several times. The target neuron is expected to keep firing and make a consistent decision. In contrast, the data pattern of the audio data is non-repetitive. The frequencies of spoken digit numbers are changing over time within a short period, so that the neuron do not have to keep firing after a prediction has been made. Data sequences in DVSPPlane are also non-repetitive, but the spatial features do not change significantly during the dropping of an airplane.

3.5.1. Neuron activities

To better compare the firing patterns between FS and FR models, apart from Figure 6, we aggregated all the output spikes generated in response to different input signals for each class in a single raster plot, as shown in Figure 9. Each color corresponds to the output spikes from a single trial. An ideal case is that each neuron generates spikes of only one color, such as FS-DVSGesture in Figure 9. This indicates that only the target neuron is active while the other neurons remain inactive. On the other hand, mixed colors (such as FR-SHD) indicate that the non-target neurons fire more spikes thereby affecting classification.

As shown in Figures 6, 9, it is notable that FS models exhibit distinct neuronal behavior on different types of data. For the visual signals in DVSGesture and DVSPPlane, FS models produce periodic firing pattern similar to those produced by FR models, in which the target neuron keeps firing to consistently make a prediction. However, for the SHD data sequences without temporal repetition, FS systems generate much fewer spikes than FR systems and the target neuron almost stops firing after the classification has been made.

The distinct neuronal activities of SHD also illustrate why FS coding can outperform FR coding even though the FS coding makes decisions based on only a portion of the given input spikes. First, FS models usually exhibit a longer temporal delay than FR models. This highlights that ensuring high accuracy demands sufficient input information. The time delay indicates the minimum

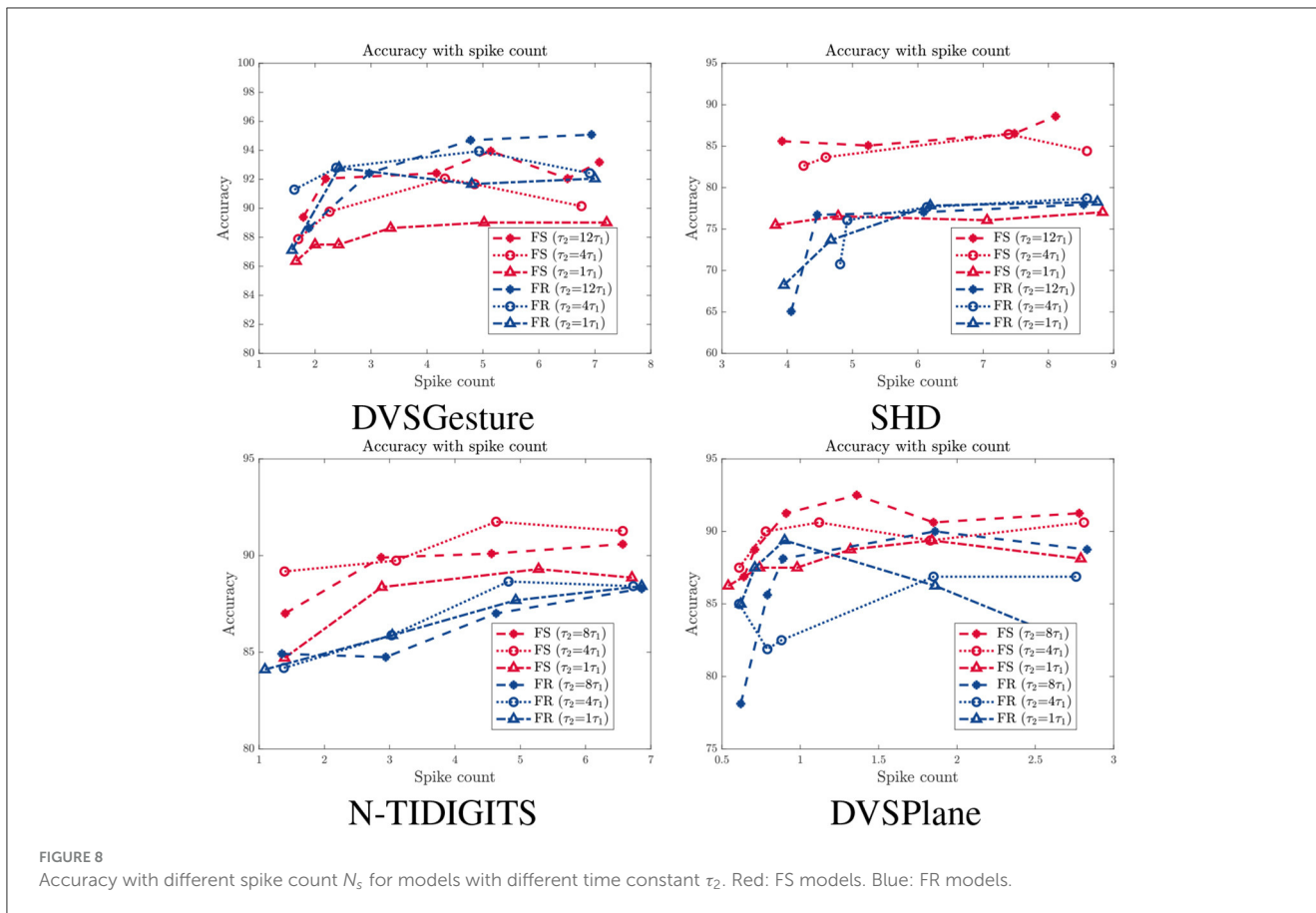


FIGURE 8 Accuracy with different spike count N_s for models with different time constant τ_2 . Red: FS models. Blue: FR models.

required information for accurate decision-making. From FS-SHD in Figure 9, the time delay varies notably across different trials. This indicates that different lengths of input data are necessary for making correct decisions. While FR coding utilizes entire given input data, it can include redundant information. As shown by FR-SHD in Figure 9, output neurons start firing at approximately the same time across different trials, even when the available information is not enough for correct decision-making. Non-target neurons generate more spikes that disrupt the decision process, leading to a lack of precision in decision-making.

The unique output spike pattern suggests that FS models is also capable of achieving accurate classifications based on FR. In Table 4, both FS and FR accuracies are tested on the two types of models. Results demonstrate that FS models performs well on FR and sometimes even better than FR models, although it is trained based on FS timings. However, FR models struggle to predict accurately based on FS.

3.5.2. Gaussian window size in error propagation

Another interesting observation is that the choice of Gaussian window size in the error propagation is related to the data type. We observed that repetitive visual data achieves better performance with a larger Gaussian window W (i.e., with a smaller D), while non-repetitive audio sequences prefer a smaller value of W . Specifically, we obtain optimal results using $D = 16$ for SHD and N-TIDIGITS, and $D = 4, D = 8$ for DVSGesture and

DVSPlane, respectively. A larger window size in error propagation means that the error of FS times is propagated to a wider time range and more spikes are optimized. As a result, the firing patterns are more similar to rate coding with a higher firing rate but the precise timing of spikes is lost. On the contrary, with a smaller window size, the error is propagated to fewer spikes where the precise timing is emphasized. Table 5 shows the results of the FS model on repetitive (DVSGesture) and non-repetitive (SHD) data with different Gaussian window size W . As W decreases, the firing rate of the target neuron decreases since fewer spikes are optimized, leading to a drop in the FR accuracy. Nevertheless, there are distinct behaviors in the FS accuracy of repetitive and non-repetitive signals. The FS accuracy of repetitive data follows a similar trend to FR accuracy, but it improves with a smaller window on non-repetitive data, as the precise timing is more important in this case.

We can, therefore, conclude that FS coding is better suited for the classification of non-repetitive data and when precise timing of the first spike is required. FR neurons should be used for repetitive signals or the cases in which a consistent and stable prediction is required.

3.6. Results of models using AdLIF neurons

To further validate the effectiveness of the FS coding, we conducted further experiments by replacing CUBA-LIF neurons

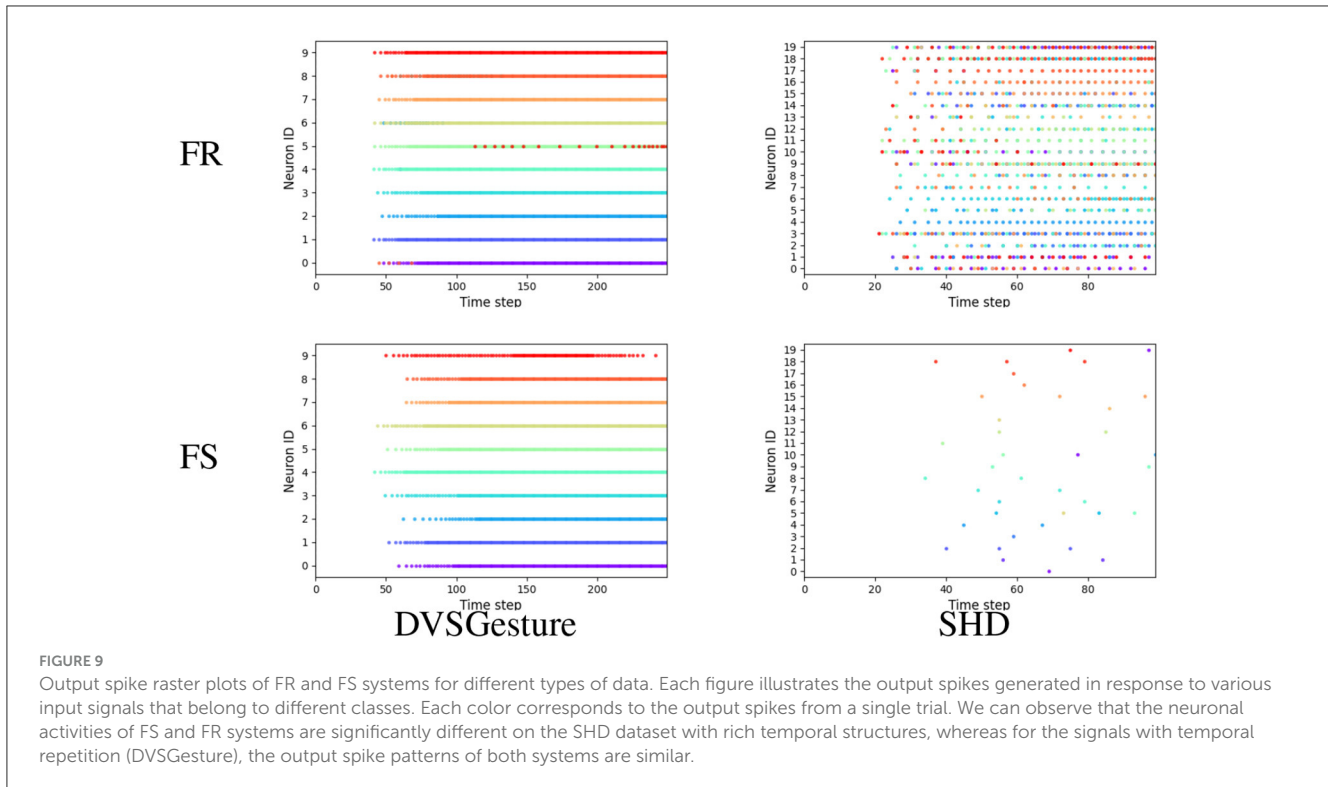


TABLE 4 The accuracy of FS and FR models with different time constants τ_2 for decision layers ($\tau_2 = \mu \tau_1$).

	μ	Trained using FR loss (%)		Trained using FS loss (%)	
		Acc (FS)↑	Acc (FR)↑	Acc (FS)↑	Acc (FR)↑
DVSGesture	1	72.3	94.0	90.1	93.7
	4	81.0	94.2	89.7	94.3
	12	82.6	95.1	92.8	94.6
SHD	1	34.3	79.3	78.3	70.5
	4	39.4	78.8	85.5	79.1
	12	42.8	77.8	87.6	79.4
N-TIDIGITS	1	47.9	88.2	87.6	75.6
	4	47.4	88.7	89.3	87.0
	8	51.0	88.1	89.3	88.0
DVSPane	1	69.5	83.0	87.6	88.5
	4	76.5	85.4	90.5	90.3
	8	82.0	89.3	92.7	90.0

The results are the average of 5 trials. Both accuracy based on FS and FR are tested. Acc (FS) and Acc (FR) indicate accuracy when testing using FS and FR, respectively. The bold values represent the best results obtained when comparing the FR and FS accuracy for each model.

with adaptive LIF (AdLIF) neurons (Bittar and Garner, 2022) in hidden layers. We tested the FC models for the SHD and NTIDIGITS tasks and recurrent connections are removed. Note that CUBA-LIF neurons with fixed parameters are still used in the output layer, because we found that this type of neuron can enjoy longer delay for accurate decision-making. We only tested models with large time constant in the output layer and utilized batch normalization and dropout strategy in hidden layers to obtain the best accuracy. Detailed parameter settings are listed in Appendix.

Table 6 presents comparison results between FS and FR models utilizing AdLIF neurons. The same conclusion can be drawn from these results as those derived from models using CUBA-LIF neurons. The FS coding leads to higher accuracy and superior energy efficiency (fewer spikes) than FR coding on data sequences with rich temporal structures. On the other hand, FS models exhibit longer time delay compared to FR models, but the gap is reduced as the accuracy reaches 90% of its peak value. Furthermore, compared to the results of SHD and NTIDIGITS in Table 2, the models with

TABLE 5 Comparison of results on repetitive visual data (DVSGesture) and non-repetitive audio data (SHD) with different Gaussian window size W in error assignment.

DVSGesture ($\tau_2 = 4\tau_1$)				SHD ($\tau_2 = 4\tau_1$)			
D	W	Acc (%) (FS) \uparrow	Acc (%) (FR) \uparrow	D	W	Acc (%) (FS) \uparrow	Acc (%) (FR) \uparrow
4	120	89.0	94.3	4	120	82.9	81.8
8	91	88.6	93.2	8	73	84.5	80.3
12	61	86.7	88.6	16	37	85.6	77.3
16	46	88.6	90.2	32	19	85.5	76.5

As the window size W decreases, the FR accuracy drops due to a lower firing rate. The FS accuracy of repetitive signals experiences a decline as well, whereas that of non-repetitive signals shows an opposite trend where precise timing is important. The bold values represent the best results in each column.

TABLE 6 Comparison of FR and FS models with AdLIF neurons.

AdLIF	Loss	Acc (%) \uparrow	N_s \downarrow	$t_d(50\%)$ \downarrow	$t_d(90\%)$ \downarrow
SHD	FR	90.18	7.24	28.4	57.8
	FS	94.08	3.65	55.0	77.2
NTIDIGITS	FR	91.11	13.19	64.6	122.0
	FS	92.25	6.19	88.8	129.4

The results are the average of 5 trials. The bold values represent the best results obtained when comparing the FR and FS models for each dataset.

AdLIF neurons exhibit significant accuracy improvement than the models employing CUBA-LIF neurons. This highlights that our approach is flexible and works well with various neuron types. This also demonstrates that the FS coding scheme has potential to achieve higher accuracy for data sequences with complex temporal structures if employing advanced architectures and strategies. The comparison results with other state-of-the-art methods is presented in the [Appendix](#).

4. Conclusion

In this study, we introduce a novel decision-making scheme based on FS coding for realistic event sequences by encoding FS timings of output neurons, and propose a supervised training framework based on FS timings. In the forward pass, discrete temporal coding is applied to the spike trains in the output layer. In the backpropagation, we propose error assignment from FS times to spikes through a Gaussian window and then leverage a surrogate gradient descent method for spikes to achieve supervised learning. Additional strategies are designed to facilitate training and mitigate the influence of inactive neurons, such as adding empty sequences and using different time constants and thresholds for feature extraction and decision layers.

In the experiments, we test the FS coding scheme on classifying various types of event data with rich temporal structures and make a comprehensive comparison with FR coding. Our results provide insights into the distinct mechanisms underlying FS and FR codings. First, FS coding demonstrates comparable performance with FR coding, but there is a trade-off between accuracy and time delay. A relatively longer temporal latency in the first spike helps encode more information, leading to higher FS accuracy. Furthermore, models based on FS and FR codings

demonstrate distinct neuronal behavior on different types of data sequences in terms of firing patterns and sparsity. In particular, FS systems are much more energy efficient than FR systems for non-repetitive audio sequences with highly complex temporal structures. In contrast, for visual data sequences with temporal repetition and spatial information, the behavior of FS and FR models is more aligned. The FS systems tend to exhibit longer response time compared to FR systems. Future research could focus on exploring strategies to reduce the temporal delay of the first spike.

Data availability statement

The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding authors.

Author contributions

SL: Conceptualization, Data curation, Formal analysis, Methodology, Software, Validation, Writing—original draft. VL: Supervision, Writing—review and editing. PD: Project administration, Supervision, Writing—review and editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. SL was supported by the scholarship from the Department of Electrical and Electronic Engineering, Imperial College London, UK.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated

organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or

claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

- Afshar, S., Hamilton, T. J., Tapson, J., Van Schaik, A., and Cohen, G. (2019). Investigation of event-based surfaces for high-speed detection, unsupervised feature extraction, and object recognition. *Front. Neurosci.* 12, 1047. doi: 10.3389/fnins.2018.01047
- Amir, A., Taba, B., Berg, D., Melano, T., McKinstry, J., Di Nolfo, D., et al. (2017). "A low power, fully event-based gesture recognition system," in *Proc. IEEE Comput. Vis. Pattern Recognit. (CVPR)* (Honolulu, HI: IEEE), 7388–7397. doi: 10.1109/CVPR.2017.781
- Anumula, J., Neil, D., Delbruck, T., and Liu, S.-C. (2018). Feature representations for neuromorphic audio spike streams. *Front. Neurosci.* 12, 23. doi: 10.3389/fnins.2018.00023
- Bittar, A., and Garner, P. N. (2022). A surrogate gradient spiking baseline for speech command recognition. *Front. Neurosci.* 16, 865897. doi: 10.3389/fnins.2022.865897
- Bohte, S. M., Kok, J. N., and La Poutré, H. (2002). Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing* 48, 17–37. doi: 10.1016/S0925-2312(01)00658-0
- Bonilla, L., Gautrais, J., Thorpe, S., and Masquelier, T. (2022). Analyzing time-to-first-spike coding schemes: a theoretical approach. *Front. Neurosci.* 16, 971937. doi: 10.3389/fnins.2022.971937
- Brette, R. (2015). Philosophy of the spike: rate-based vs. spike-based theories of the brain. *Front. Syst. Neurosci.* 9, 151. doi: 10.3389/fnsys.2015.00151
- Comşa, I.-M., Potempa, K., Versari, L., Fischbacher, T., Gesmundo, A., Alakuijala, J., et al. (2022). "Temporal coding in spiking neural networks with alpha synaptic function: learning with backpropagation," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33 (IEEE), 5939–5952. doi: 10.1109/TNNLS.2021.3071976
- Cramer, B., Stradmann, Y., Schemmel, J., and Zenke, F. (2022). The Heidelberg spiking datasets for the systematic evaluation of spiking neural networks. *IEEE Trans. Neural Netw. Learning Syst.* 33, 2744–2757. doi: 10.1109/TNNLS.2020.3044364
- Davies, M., Srinivasa, N., Lin, T.-H., Chinya, G., Cao, Y., Choday, S. H., et al. (2018). "Loihi: a neuromorphic manycore processor with on-chip learning," in *IEEE Micro*, vol. 38 (IEEE), 82–99. doi: 10.1109/MM.2018.112130359
- Fang, W., Yu, Z., Chen, Y., Masquelier, T., Huang, T., Tian, Y., et al. (2021). "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)* (Montreal, QC: IEEE), 2641–2651. doi: 10.1109/ICCV48922.2021.00266
- Fontaine, B., and Peremans, H. (2009). Bat echolocation processing using first-spike latency coding. *Neural Netw.* 22, 1372–1382. doi: 10.1016/j.neunet.2009.05.002
- Gardner, B., Sporea, I., and Grüning, A. (2015). Learning spatiotemporally encoded pattern transformations in structured spiking neural networks. *Neural Comput.* 27, 2548–2586. doi: 10.1162/NECO_a_00790
- Gollisch, T., and Meister, M. (2008). Rapid neural coding in the retina with relative spike latencies. *Science* 319, 1108–1111. doi: 10.1126/science.1149639
- Göltz, J., Kriener, L., Baumbach, A., Billaudelle, S., Breitwieser, O., Cramer, B., et al. (2021). Fast and energy-efficient neuromorphic deep learning with first-spike times. *Nat. Mach. Intell.* 3, 823–835. doi: 10.1038/s42256-021-00388-x
- Guo, W., Fouda, M. E., Eltawil, A. M., and Salama, K. N. (2021). Neural coding in spiking neural networks: a comparative study for robust neuromorphic systems. *Front. Neurosci.* 15, 638474. doi: 10.3389/fnins.2021.638474
- Hammouamri, I., Khalfaoui-Hassani, I., and Masquelier, T. (2023). Learning delays in spiking neural networks using dilated convolutions with learnable spacings. *arXiv [Preprint]*. arXiv:2306.17670.
- Heil, P. (2004). First-spike latency of auditory neurons revisited. *Curr. Opin. Neurobiol.* 14, 461–467. doi: 10.1016/j.conb.2004.07.002
- Huxter, J., Burgess, N., and O'Keefe, J. (2003). Independent rate and temporal coding in hippocampal pyramidal cells. *Nature* 425, 828–832. doi: 10.1038/nature02058
- Iyer, L. R., Chua, Y., and Li, H. (2021). Is neuromorphic MNIST neuromorphic? Analyzing the discriminative power of neuromorphic datasets in the time domain. *Front. Neurosci.* 15, 608567. doi: 10.3389/fnins.2021.608567
- Izhikevich, E. M., Desai, N. S., Walcott, E. C., and Hoppensteadt, F. C. (2003). Bursts as a unit of neural information: selective communication via resonance. *Trends Neurosci.* 26, 161–167. doi: 10.1016/S0166-2236(03)00034-1
- Jiang, Z., Xu, J., Zhang, T., Poo, M.-m., and Xu, B. (2023). Origin of the efficiency of spike timing-based neural computation for processing temporal information. *Neural Netw.* 160, 84–96. doi: 10.1016/j.neunet.2022.12.017
- Johansson, R. S., and Birznieks, I. (2004). First spikes in ensembles of human tactile afferents code complex spatial fingertip events. *Nat. Neurosci.* 7, 170–177. doi: 10.1038/nn1177
- Kaiser, J., Mostafa, H., and Neftci, E. (2020). Synaptic plasticity dynamics for deep continuous local learning (DECOLLE). *Front. Neurosci.* 14, 424. doi: 10.3389/fnins.2020.00424
- Kheradpisheh, S. R., and Masquelier, T. (2020). Temporal backpropagation for spiking neural networks with one spike per neuron. *Int. J. Neur. Syst.* 30, 2050027. doi: 10.1142/S0129065720500276
- Kiselev, M. (2016). "Rate coding vs. temporal coding - is optimum between?" in *2016 International Joint Conference on Neural Networks (IJCNN)* (Vancouver, BC: IEEE), 1355–1359. doi: 10.1109/IJCNN.2016.7727355
- Kotariya, V., and Ganguly, U. (2021). "Spiking-GAN: a spiking generative adversarial network using time-to-first-spike coding," in *2022 International Joint Conference on Neural Networks (IJCNN)* (Padua: IEEE). doi: 10.1109/IJCNN55064.2022.9892262
- Li, H., Liu, H., Ji, X., Li, G., and Shi, L. (2017). CIFAR10-DVS: an event-stream dataset for object classification. *Front. Neurosci.* 11, 309. doi: 10.3389/fnins.2017.00309
- Liu, X.-P., and Wang, X. (2022). Distinct neuronal types contribute to hybrid temporal encoding strategies in primate auditory cortex. *PLOS Biol.* 20, e3001642. doi: 10.1371/journal.pbio.3001642
- Mirsadeghi, M., Shalchian, M., Kheradpisheh, S. R., and Masquelier, T. (2021). STiDi-BP: spike time displacement based error backpropagation in multilayer spiking neural networks. *Neurocomputing* 427, 131–140. doi: 10.1016/j.neucom.2020.11.052
- Mostafa, H. (2018). Supervised learning based on temporal coding in spiking neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 29, 3227–3235. doi: 10.1109/TNNLS.2017.2726060
- Neftci, E. O., Mostafa, H., and Zenke, F. (2019). Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Process. Mag.* 36, 51–63. doi: 10.1109/MSP.2019.2931595
- Orchard, G., Jayawant, A., Cohen, G. K., and Thakor, N. (2015). Converting static image datasets to spiking neuromorphic datasets using saccades. *Front. Neurosci.* 9, 437. doi: 10.3389/fnins.2015.00437
- Panzeri, S., Macke, J. H., Gross, J., and Kayser, C. (2015). Neural population coding: combining insights from microscopic and mass signals. *Trends Cogn. Sci.* 19, 162–172. doi: 10.1016/j.tics.2015.01.002
- Panzeri, S., Petersen, R. S., Schultz, S. R., Lebedev, M., and Diamond, M. E. (2001). The role of spike timing in the coding of stimulus location in rat somatosensory cortex. *Neuron* 29, 769–777. doi: 10.1016/S0896-6273(01)00251-3
- Parameshwara, C. M., Li, S., Fermüller, C., Sanket, N. J., Evanusa, M. S., Aloimonos, Y., et al. (2021). "SpikeMS: deep spiking neural network for motion segmentation," in *2021 IEEE/RISJ International Conference on Intelligent Robots and Systems (IROS)* (Prague: IEEE), 3414–3420. doi: 10.1109/IROS51168.2021.9636506
- Park, S., Kim, S., Na, B., and Yoon, S. (2020). "T2FS.NN: deep spiking neural networks with time-to-first-spike coding," in *2020 57th ACM/IEEE Design Automation Conference (DAC)* (San Francisco, CA: IEEE), 1–6. doi: 10.1109/DAC18072.2020.9218689
- Perez-Nieves, N., Leung, V. C. H., Dragotti, P. L., and Goodman, D. F. M. (2021). Neural heterogeneity promotes robust learning. *Nat. Commun.* 12, 5791. doi: 10.1038/s41467-021-26022-3
- Pfeiffer, M., and Pfeil, T. (2018). Deep learning with spiking neurons: opportunities and challenges. *Front. Neurosci.* 12, 774. doi: 10.3389/fnins.2018.00774
- Pfister, J.-P., Toyozumi, T., Barber, D., and Gerstner, W. (2006). Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural Comput.* 18, 1318–1348. doi: 10.1162/neco.2006.18.6.1318
- Rullen, R. V., and Thorpe, S. J. (2001). Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex. *Neural Comput.* 13, 1255–1283. doi: 10.1162/08997660152002852
- Sakemi, Y., Morino, K., Morie, T., and Aihara, K. (2023). A supervised learning algorithm for multilayer spiking neural networks based on temporal coding toward energy-efficient VLSI processor design. *IEEE Trans. Neural Netw. Learn. Syst.* 34, 394–408. doi: 10.1109/TNNLS.2021.3095068
- Shrestha, S. B., and Orchard, G. (2018). "SLAYER: spike layer error reassignment in time," in *Proceedings of the 32nd International Conference on Neural*

- Information Processing Systems* (Montréal, QC: Curran Associates Inc.), 1419–1428. doi: 10.5555/3326943.3327073
- Shrestha, S. B., and Song, Q. (2016). “Event based weight update for learning infinite spike train,” in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)* (Anaheim, CA: IEEE), 333–338. doi: 10.1109/ICMLA.2016.0061
- Shrestha, S. B., and Song, Q. (2017). Robust spike-train learning in spike-event based weight update. *Neural Netw.* 96, 33–46. doi: 10.1016/j.neunet.2017.08.010
- Shrestha, S. B., Zhu, L., and Sun, P. (2022). “Spikemax: spike-based loss methods for classification,” in *2022 International Joint Conference on Neural Networks (IJCNN)* (Padua), 1–7. doi: 10.1109/IJCNN55064.2022.9892379
- Wang, X., Lin, X., and Dang, X. (2020). Supervised learning in spiking neural networks: a review of algorithms and evaluations. *Neural Netw.* 125, 258–280. doi: 10.1016/j.neunet.2020.02.011
- Wu, D., Yi, X., and Huang, X. (2022). A little energy goes a long way: build an energy-efficient, accurate spiking neural network from convolutional neural network. *Front. Neurosci.* 16, 759900. doi: 10.3389/fnins.2022.759900
- Wu, Y., Deng, L., Li, G., Zhu, J., and Shi, L. (2018). Spatio-temporal backpropagation for training high-performance spiking neural networks. *Front. Neurosci.* 12, 331. doi: 10.3389/fnins.2018.00331
- Wunderlich, T. C., and Pehle, C. (2021). Event-based backpropagation can compute exact gradients for spiking neural networks. *Sci. Rep.* 11, 12829. doi: 10.1038/s41598-021-91786-z
- Xiao, M., Meng, Q., Zhang, Z., Wang, Y., and Lin, Z. (2023). SPIDE: a purely spike-based method for training feedback spiking neural networks. *Neural Netw.* 161, 9–24. doi: 10.1016/j.neunet.2023.01.026
- Xiao, R., Tang, H., Ma, Y., Yan, R., and Orchard, G. (2020). An event-driven categorization model for AER image sensors using multispikes encoding and learning. *IEEE Trans. Neural Netw. Learning Syst.* 31, 3649–3657. doi: 10.1109/TNNLS.2019.2945630
- Xu, Y., Zeng, X., Han, L., and Yang, J. (2013). A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks. *Neural Netw.* 43, 99–113. doi: 10.1016/j.neunet.2013.02.003
- Yao, M., Gao, H., Zhao, G., Wang, D., Lin, Y., Yang, Z., et al. (2021). Temporal-wise attention spiking neural networks for event streams classification,” in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (Montreal, QC: IEEE), 10201–10210. doi: 10.1109/ICCV48922.2021.01006
- Yin, B., Corradi, F., and Bohtë, S. M. (2021). Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. *Nat. Mach. Intell.* 3, 905–913. doi: 10.1038/s42256-021-00397-w
- Yu, C., Gu, Z., Li, D., Wang, G., Wang, A., Li, E., et al. (2022). STSC-SNN: spatio-temporal synaptic connection with temporal convolution and attention for spiking neural networks. *Front. Neurosci.* 16, 1079357. doi: 10.3389/fnins.2022.1079357
- Yu, M., Xiang, T., Srivatsa, P., Chu, K. T. N., Amornpaisannon, B., Tavva, Y., et al. (2023). A TTFS-based energy and utilization efficient neuromorphic CNN accelerator. *Front. Neurosci.* 17, 1121592. doi: 10.3389/fnins.2023.1121592
- Zenke, F., and Ganguli, S. (2018). SuperSpike: supervised learning in multilayer spiking neural networks. *Neural Comput.* 30, 1514–1541. doi: 10.1162/neco_a_01086
- Zhang, M., Wang, J., Wu, J., Belatreche, A., Amornpaisannon, B., Zhang, Z., et al. (2022). Rectified linear postsynaptic potential function for backpropagation in deep spiking neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 33, 1947–1958. doi: 10.1109/TNNLS.2021.3110991
- Zhang, W., and Li, P. (2019). “Spike-train level backpropagation for training deep recurrent spiking neural networks,” in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, number 701 (Red Hook, NY: Curran Associates Inc), 7802–7813.
- Zhou, S., Li, X., Chen, Y., Chandrasekaran, S. T., and Sanyal, A. (2021). Temporal-coded deep spiking neural network with easy training and robust performance. *Proc. AAAI Conf. Artif. Intell.* 35, 11143–11151. doi: 10.1609/aaai.v35i12.17329
- Zhou, Z., Zhu, Y., He, C., Wang, Y., YAN, S., Tian, Y., et al. (2023). “Spikformer: when spiking neural network meets transformer,” in *The Eleventh International Conference on Learning Representations*. Available online at: https://openreview.net/forum?id=frE4fUwz_h

Appendix

For the experiments in Section 3.6, we replaced the CUBA-LIF neurons in hidden layers with AdLIF neurons (Bittar and Garner, 2022) and trained with the FS loss. This adjustment introduces parameters related to adaptation, including adaptation time constant τ_w , and recovery variables a and b . According to Bittar and Garner (2022), τ_w is initialized with 100 ms, and a and b are constrained in the range of $a \in [-1, 1]$, $b \in [0, 2]$. However, in our models, we set $a \in [0, 1]$ and $b \in [0, 1]$. We observed that this setting leads to more stable training and improved performance. During training, these parameters are trainable along with τ_m .

In particular, for the SHD task, the architecture is 700-FC256-FC256-20, $\tau_1 = 5ms$, $\tau_2 = 12\tau_1$, $\theta_1 = 0.5$, $\theta_2 = 10$. For the NTIDIGITS task, the architecture is 64-FC256-FC256-11, $\tau_1 = 5ms$, $\tau_2 = 8\tau_1$, $\theta_1 = 0.5$, $\theta_2 = 10$. Other training parameters keep the same with the models using CUBA-LIF neurons. In addition, batch normalization (BN) and dropout (DP) are used in each hidden layer. Specifically, a single layer comprises linear operations + BN + neuron + DP. The dropout rate is set to 0.25.

Table A1 presents comparison results of models utilizing CUBA-LIF and AdLIF neurons and the results of other state-of-the-art methods on DVSGesture/SHD/NTIDIGITS datasets. We can see that the performance of models with CUBA-LIF neurons is generally worse, particularly in the case of DVSGesture, indicating that FS coding is not suited for sequences involving temporal repetition. In contrast, models with AdLIF neurons exhibit significant accuracy improvement, achieving comparable performance with state-of-the-art methods, especially for SHD dataset.

TABLE A1 Result comparison of FS models with other methods.

	Method	Acc (%)
DVSGesture	Hetero. RSNN (Perez-Nieves et al., 2021)	82.9
	SLAYER (Shrestha and Orchard, 2018)	93.64 ± 0.49
	DECOLLE (Kaiser et al., 2020)	95.54
	SLAYER + SpikeMax (Shrestha et al., 2022)	95.83 ± 0.48
	PLIF (Fang et al., 2021) (STBP)	97.57
	STSC-SNN (Yu et al., 2022)	98.96
	Ours (CUBA-LIF)	92.8
SHD	Hetero. RSNN (Perez-Nieves et al., 2021)	82.7 ± 0.8
	RSNN with data aug. + noise (Cramer et al., 2022)	83.2 ± 1.3
	Adaptive SRNN (Yin et al., 2021)	90.4
	TA-SNN (Yao et al., 2021)	91.08
	STSC-SNN (Yu et al., 2022)	92.36
	RadLIF (Bittar and Garner, 2022)	94.62
	SNN with learned delays (Hammouamri et al., 2023)	95.07 ± 0.24
	Ours (CUBA-LIF)	87.6
	Ours (AdLIF)	94.08
NTIDIGITS	GRU-RNN (Anumula et al., 2018)	90.9
	Phased-LSTM (Anumula et al., 2018)	91.25
	ST-RSBP (Zhang and Li, 2019)	93.63 ± 0.27
	SLAYER + spike-rate (Shrestha et al., 2022)	94.19 ± 0.18
	SLAYER + SpikeMax (Shrestha et al., 2022)	93.21 ± 0.32
	Ours (CUBA-LIF)	89.3
	Ours (AdLIF)	92.25

The bold values represent the best results for each dataset.