frontiers | Frontiers in Neuroscience

TYPE Original Research
PUBLISHED 14 February 2024
DOI 10.3389/fnins.2024.1346805

Check for updates

OPEN ACCESS

EDITED BY
Mohammed Fouda,
University of California, Irvine, United States

REVIEWED BY
Shuangming Yang,
Tianjin University, China
Anguo Zhang,
University of Macau, China

*CORRESPONDENCE
Youngeun Kim
youngeun.kim@yale.edu

RECEIVED 29 November 2023
ACCEPTED 30 January 2024
PUBLISHED 14 February 2024

CITATION
Kim Y, Kahana A, Yin R, Li Y, Stinis P, Karniadakis GE and Panda P (2024) Rethinking skip connections in Spiking Neural Networks with Time-To-First-Spike coding. *Front. Neurosci.* 18:1346805. doi: 10.3389/fnins.2024.1346805

COPYRIGHT
© 2024 Kim, Kahana, Yin, Li, Stinis, Karniadakis and Panda. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). 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.

Rethinking skip connections in Spiking Neural Networks with Time-To-First-Spike coding

Youngeun Kim[1]*, Adar Kahana[2], Ruokai Yin[1], Yuhang Li[1], Panos Stinis[2,3], George Em Karniadakis[2,3] and Priyadarshini Panda[1]

[1]Department of Electrical Engineering, Yale University, New Haven, CT, United States, [2]Division of Applied Mathematics, Brown University, Providence, RI, United States, [3]Advanced Computing, Mathematics and Data Division, Pacific Northwest National Laboratory, Richland, WA, United States

Time-To-First-Spike (TTFS) coding in Spiking Neural Networks (SNNs) offers significant advantages in terms of energy efficiency, closely mimicking the behavior of biological neurons. In this work, we delve into the role of skip connections, a widely used concept in Artificial Neural Networks (ANNs), within the domain of SNNs with TTFS coding. Our focus is on two distinct types of skip connection architectures: (1) addition-based skip connections, and (2) concatenation-based skip connections. We find that addition-based skip connections introduce an additional delay in terms of spike timing. On the other hand, concatenation-based skip connections circumvent this delay but produce time gaps between after-convolution and skip connection paths, thereby restricting the effective mixing of information from these two paths. To mitigate these issues, we propose a novel approach involving a learnable delay for skip connections in the concatenation-based skip connection architecture. This approach successfully bridges the time gap between the convolutional and skip branches, facilitating improved information mixing. We conduct experiments on public datasets including MNIST and Fashion-MNIST, illustrating the advantage of the skip connection in TTFS coding architectures. Additionally, we demonstrate the applicability of TTFS coding on beyond image recognition tasks and extend it to scientific machine-learning tasks, broadening the potential uses of SNNs.

KEYWORDS

Spiking Neural Network, temporal coding, image recognition, event-based processing, energy-efficient deep learning

1 Introduction

The communication between spiking neurons in the brain, characterized by its binary, event-driven, and sparse nature, offers significant potential for creating flexible and energy-efficient artificial intelligence (AI) systems (Roy et al., 2019; Christensen et al., 2022). Spiking Neural Networks (SNNs), unlike traditional Artificial Neural Networks (ANNs), leverage binary spikes, thereby offering a unique dimension of time in their operation. Recent studies have shown promising results with SNNs, making them suitable for competitive and energy-efficient applications in neuromorphic hardware (Cao et al., 2015; Diehl and Cook, 2015; Roy et al., 2019; Comsa et al., 2020; Panda et al., 2020).

A primary application of SNNs lies in image recognition (Roy et al., 2019; Christensen et al., 2022). In order to transform a static image into binary spike trains, a range of coding schemes have been introduced (Park et al., 2019; Comsa et al., 2020; Guo et al., 2021). Rate

coding conveys information through the firing rate of spikes (Wu et al., 2019; Fang et al., 2020; Lee et al., 2020; Zhang and Li, 2020; Zheng et al., 2020). Phase coding, meanwhile, embeds temporal information in spike patterns utilizing a global oscillator (Montemurro et al., 2008). In contrast, burst coding transmits spike bursts within brief time periods, which boosts the reliability of synaptic communication between neurons (Park et al., 2019). While these coding schemes have proven successful in training SNNs, they generate a large number of spikes, which presents challenges when applied to ultra-low power devices.

To leverage temporal spike information in ultra-low power environments, researchers have increasingly focused on Time-To-First-Spike (TTFS) coding (Rueckauer and Liu, 2018; Zhang et al., 2019). The core concept involves representing information through spike timing, with each neuron generating a single spike during the forward process. A line of work focuses on training temporal-coded SNNs with backpropagation (Bohte et al., 2000; Xu et al., 2013; Mostafa, 2017; Shrestha and Song, 2017; Comsa et al., 2020; Zhang et al., 2021), which highlights biological plausibility and efficiency of temporal coding. Much of the previous work has centered on developing improved synaptic models capable of effectively processing temporal information. For instance, Mostafa (2017) employed non-leaky integrate-and-fire neurons to compute locally exact gradients for backpropagation, while Comsa et al. (2020) introduced the alpha-synaptic function to enhance SNNs' accuracy. Recently, Zhang et al. (2021) proposed a ReLU-like spike dynamics that effectively mitigates the dead neuron issue caused by the leaky nature of spike functions.

While advances in synaptic modeling have illuminated the understanding of neuronal dynamics, the exploration of network architecture in temporal SNNs has been relatively limited. In this paper, we explore architectural improvements of TTFS coding, focusing on the role of skip connections in neural networks. Skip connections are a widely employed technique in ANNs, facilitating training and enhancing performance by allowing information to bypass certain layers. We examine two types of skip connection architectures: (1) addition-based skip connections, as proposed in ResNet (He et al., 2016), and (2) concatenation-based skip connections utilized in the ShuffleNetV2 architecture (Ma et al., 2018). We find that, when implemented in temporal SNN architectures, addition-based skip connections can introduce extra delays in the time dimension. Conversely, concatenation-based skip connections substantially reduce inference latency, but yield limited performance improvements due to discrepancies between the distributions from the convolution and skip branches. To augment the performance of concatenation-based skip connections, we propose a learnable delay for skip connections, which diminishes the distribution gap between the skip and convolutional branches, allowing for more effective information mixing between the two distributions.

In addition to our exploration of a new architecture for TTFS SNNs, we also investigate applications outside of image recognition, specifically in the realm of scientific machine-learning tasks. We venture into the domain of time-reversal wave localization problems, a significant challenge in physics and engineering. This problem aims to trace back a wave's source given the wave shape at a later time (Bardos and Fink, 2002; Givoli and Turkel, 2012; Kahana et al., 2022). Through these experiments, we aim to demonstrate the versatility and potential of SNNs in various complex tasks, significantly expanding their applicability beyond traditional domains.

In summary, our contributions in this paper are three-fold. (1) First, we explore the network architecture of temporal SNNs, with a particular emphasis on skip connections, examining both addition-based residual connections and concatenation-based skip connections in the context of temporal SNNs. (2) Second, we propose a learnable delay for skip connections to improve the performance of concatenation-based skip connections by reducing the distribution gap between skip and convolutional branches, enabling more effective information mixing. (3) Lastly, we extend the application of Time-To-First-Spike (TTFS) coding beyond image recognition to the time-reversal problem for source localization using wave signal. These contributions not only advance our understanding of network architecture in temporal SNNs but also broaden the potential applications of TTFS coding in various domains.

2 Related work

2.1 Spiking Neural Networks

SNNs, unlike traditional ANNs, operate using temporal spikes, thereby offering a unique dimension of time in their operation (Roy et al., 2019; Christensen et al., 2022). Among their components, the Leaky-Integrate-and-Fire (LIF) neuron is key as it functions as a non-linear activation unit. The LIF neurons stand out due to the "memory" held within their membrane potential, where spikes are incrementally gathered. Once this potential exceeds a certain threshold, the neurons fire output spikes and the potential resets.

Training algorithms have been a primary focus of SNN research. Several methods proposed to address this involve transforming pre-trained ANNs into SNNs using weight or threshold balancing strategies (Diehl et al., 2015; Rueckauer et al., 2017; Sengupta et al., 2019; Han et al., 2020; Li et al., 2021a). While these techniques are generally effective, they require a multitude of timesteps to emulate float activations using binary spikes. A set of recent studies have suggested the use of surrogate functions to circumvent this non-differentiable backpropagation issue (Lee et al., 2016, 2020; Shrestha and Orchard, 2018; Wu et al., 2018, 2020, 2021; Neftci et al., 2019; Li et al., 2021b; Kim et al., 2022a). These methods, accounting for temporal dynamics during weight training, exhibit high performance and short latency.

Recent literature goes beyond enhancing the performance of SNN models, focusing on improving their generalization capability and robustness. The utilization of the information bottleneck theory is a common thread in these efforts. Yang and Chen (2023a) introduce a novel and flexible learning framework called high-order spike-based information bottleneck (HOSIB), making use of the surrogate gradient technique. Yang and Chen (2023b) put forward a spike-based non-linear IB (SNIB) framework with varying orders, resulting in improved performance and robustness. Additionally, Yang et al. (2023) delves into the design space of the information

bottleneck framework, utilizing the membrane potential state for the representation of hidden information.

Another significant aspect of SNN research pertains to the coding scheme. Several schemes have been proposed for image classification with SNNs. Burst coding, for example, communicates a burst of spikes within a short duration, enhancing synaptic communication reliability (Park et al., 2019). Phase coding encodes temporal information into spike patterns based on a global oscillator (Montemurro et al., 2008). Furthermore, rate coding has been applied to large-scale settings and is currently used by state-of-the-art methods (Diehl and Cook, 2015; Lee et al., 2016, 2020). This scheme generates a spike train over T timesteps, where the total spike count reflects the magnitude of the input values. However, this generation of numerous spikes can pose issues for ultra-low power devices. To address this, Time-To-First-Spike (TTFS) coding has gained interest (Mostafa, 2017; Zhang et al., 2019; Comsa et al., 2020), as it generates a single spike per neuron, with spike latency inversely related to information importance. Despite progress in synaptic modeling, the architectural exploration of temporal SNNs remains limited. In this paper, we delve into the architectural enhancement of TTFS coding, specifically emphasizing the significance of skip connections in neural networks.

2.2 Skip connection architecture

The concept of skip connections or shortcut connections has been a cornerstone in the development of deep learning architectures, contributing significantly to the performance enhancement of various models. He et al. (2016) proposed the ResNet architecture that employs skip connections to allow signals to bypass layers, directly flowing from one layer to a layer further into the network. This design helps to alleviate the problem of vanishing gradients in deep networks, thereby enabling the training of networks that are significantly deeper than those previously possible. Furthermore, introduced by Srivastava et al. (2015), Highway Networks utilize gated skip connections, where the data flow is regulated by learned gating functions. This allows the network to learn to control the information flow dynamically. Also, Huang et al. (2017) proposed DenseNet, where each layer receives direct inputs from all preceding layers and passes down its own feature maps to all subsequent layers. This dense connectivity promotes feature reuse and substantially reduces the number of parameters. In the ShuffleNetV2 architectures (Ma et al., 2018), channel shuffle operations and pointwise group convolutions are combined with skip connections to create highly efficient network architectures suitable for mobile devices. The aforementioned architectures have shown the importance of skip connections in enhancing the performance of neural networks. However, most of these architectures have been designed for traditional ANNs, and their application and efficacy in the context of SNNs remain to be thoroughly investigated.

In the SNN domain, a line of work has introduced the residual connection architecture (Fang et al., 2021; Hu et al., 2021), demonstrating state-of-the-art performance. Zhang et al. (2023) propose a Highway Connection module designed for use with residual membrane potential neurons, enhancing the responsiveness of neurons in deep layers to input spikes. Benmeziane et al. (2023) investigates the correlation between the number of skip connections and accuracy in SNN architectures, proposing an algorithm for selecting the optimal number of skip connections. Additionally, Ikegawa et al. (2022) analyzes the impact of batch normalization and residual connection, achieving very deep SNNs (more than 100 layers) with pre-activation residual blocks. While the residual architecture has been explored with rate-coded SNNs, the impact of the residual architecture with temporal considerations has not been thoroughly investigated.

3 Methodology

3.1 Temporal neuron

Our neuron model is based on the non-leaky integrate-and-fire neurons proposed in Mostafa (2017). The neuron employs exponentially decaying synaptic current kernels denoted as ϵ. The influence of a spike occurring at time t_k on the membrane potential can be expressed as

$$\frac{dV^j(t)}{dt} = \sum_i w_{ji} \sum_k \epsilon(t - t_k). \tag{1}$$

Here, $V^j(t)$ represents the membrane potential of neuron j at time t, and w_{ji} is the weight connection between neuron j and neuron i in the preceding layer. The synaptic current kernels, denoted by ϵ, are defined as

$$\epsilon(x) = U(x)exp(-x), \tag{2}$$

where $U(x)$ is the step function. $U(x) = 1$ when $x \geq 0$ and $U(x) = 0$ otherwise, which ensures that the synaptic kernel accounts for the time subsequent to the input spike. By considering both Equations (1, 2), we can derive the equation of the membrane potential. We assume one neuron generates at most one spike, so the membrane potential $V(t)$ increases until there is an output spike. Then, we can write the membrane potential with N input spikes

$$V(t) = \sum_{k=1}^{N} U(t - t_k) w_k (1 - exp(-(t - t_k))). \tag{3}$$

Here, t_k is the spike timing of k-th spike and w_k is the corresponding weight.

The neuron generates an output spike whenever the membrane potential has a higher value than 1, thus, $V(t_{out}) \geq 1$, where t_{out} means the output spike timing. For all spike $t_k < t_{out}$, we refer to the set of input spike index as a casual set C, following the definition from the previous work (Mostafa, 2017).

$$1 = \sum_{k \in C} w_k (1 - exp(-(t_{out} - t_k))). \tag{4}$$

Then, we reorganize Equation (4) for the spike timing.

$$exp(t_{out}) = \frac{\sum_{k \in C} w_k exp(t_k)}{\sum_{k \in C} w_k - 1}. \tag{5}$$

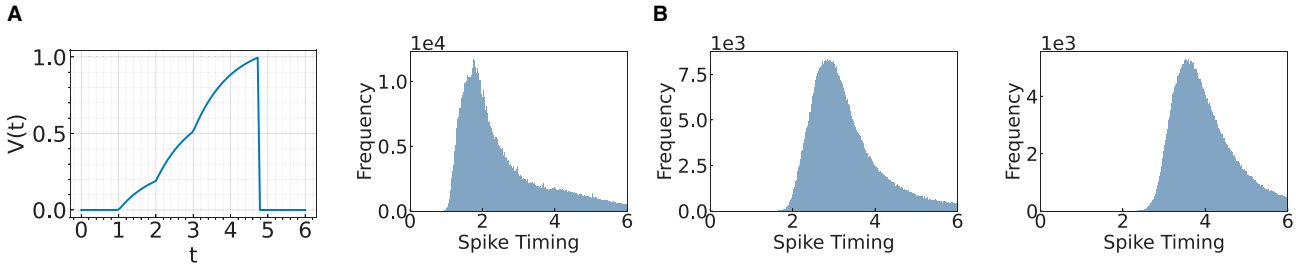

FIGURE 1
Illustration of a spike timing delay through layers. **(A)** Change of membrane potential when there are input spikes at times 1, 2, and 3. **(B)** We visualize the histogram of spike timing after Conv3, Conv4, and Conv5 layers.

For simplicity, we transform $exp(t_i) \rightarrow z_i$, i.e., z-transformation (Weisstein, 2002). Then, Equation (5) can be rewritten as:

$$z_{out} = \frac{\sum_{k \in C} w_k z_k}{\sum_{k \in C} w_k - 1}. \tag{6}$$

With TTFS coding, the networks determine the class of an image based on the neuron in the final layer that fires the earliest spike. For example, consider a scenario involving a 10-class classification problem. If the neuron corresponding to the "dog" category emits the earliest spike in the final layer, the network immediately classifies the given image as a "dog". This allows neural networks to have faster predictions, as the classification is determined as soon as the first spike is generated, without the need to wait for spikes from other neurons.

3.2 Observation: two types of skip connections

3.2.1 Temporal delay in a layer

Our objective is to accelerate the first spike timing in the final layer, without compromising on the accuracy of the model. In this context, one might wonder: *what factors contribute to the delay in temporal coding?* The delay is induced by spiking neurons where each neuron requires time to charge the membrane potential to generate the output spike, as demonstrated in Figure 1A. Figure 1B presents a histogram of spike timings, illustrating that the distribution shifts toward a later time as the layer goes deeper. We aim to reduce this inherent temporal delay by bypassing the convolutional layers.

3.2.2 Architectures

We focus on the skip connection design in the temporal coding. We examine two types of skip connection architectures: (1) addition-based skip connections, as seen in ResNet (He et al., 2016), and (2) concatenation-based skip connections utilized in the ShuffleNetV2 architecture (Ma et al., 2018). Addition-based skip connection architecture adds a skip connection to the main convolutional branch. Let X_l represent the input to l-th block, and $F(X_l)$ represent the non-linear transformation operations (i.e., convolution, batch normalization, and non-linearity) within a residual block. The operation of a residual block can be written as follows:

$$X_{l+1} = F(X_l) + X_l. \tag{7}$$

The overall operation of addition-based skip connection is illustrated in Figure 2A. The major problem with this scheme is that adding two branches (i.e., the skip connection and the convolutional branch) induces a significant delay in spike timing. For example, adding a spike from a skip connection at time t_A and a spike after convolutional operation at time t_B results in the output at time $t_A + t_B$. Therefore, the addition-based skip connection is not appropriate for TTFS coding.

On the other hand, a concatenation-based skip connection utilizes the channel split operation, where an input tensor is split into two parts along the channel dimension. One part is transformed through a series of operations while the other part passes a skip connection. These two parts are then concatenated and shuffled to ensure equal information sharing among channels. Mathematically, for the input activation X_l in l-th block, the operation of a concatenation-based skip connection can be represented as follows:

$$X_l = [X_{l,1}; X_{l,2}]. \tag{8}$$

$$X_{l+1} = \text{Shuffle}(F(X_{l,1}) || X_{l,2}), \tag{9}$$

where $F(X_l)$ represents the non-linear transformation operation, and $X_{l,1}, X_{l,2}$ stand for two input tensors divided through channel dimension, respectively. The overall illustration of the concatenation-based skip connection is shown in Figure 2B.

Different from the addition-based skip connection, the concatenation-based skip connection allows spikes to pass directly through, thereby expediting the timing of spikes. However, this method does come with a notable disadvantage: a timing discrepancy between the spikes in the convolutional branch and those in the skip connection branch. Specifically, we observe the distributions from the convolutional and skip connection branches have less overlap. This lack of overlap can make it difficult to integrate information effectively between the two distributions in the later layers. This is because of the inherent property of temporal neurons in TTFS coding. After a neuron spikes, there is no further activity, which implies a TTFS neuron will not consider any later input after it has output a spike. Consequently, these

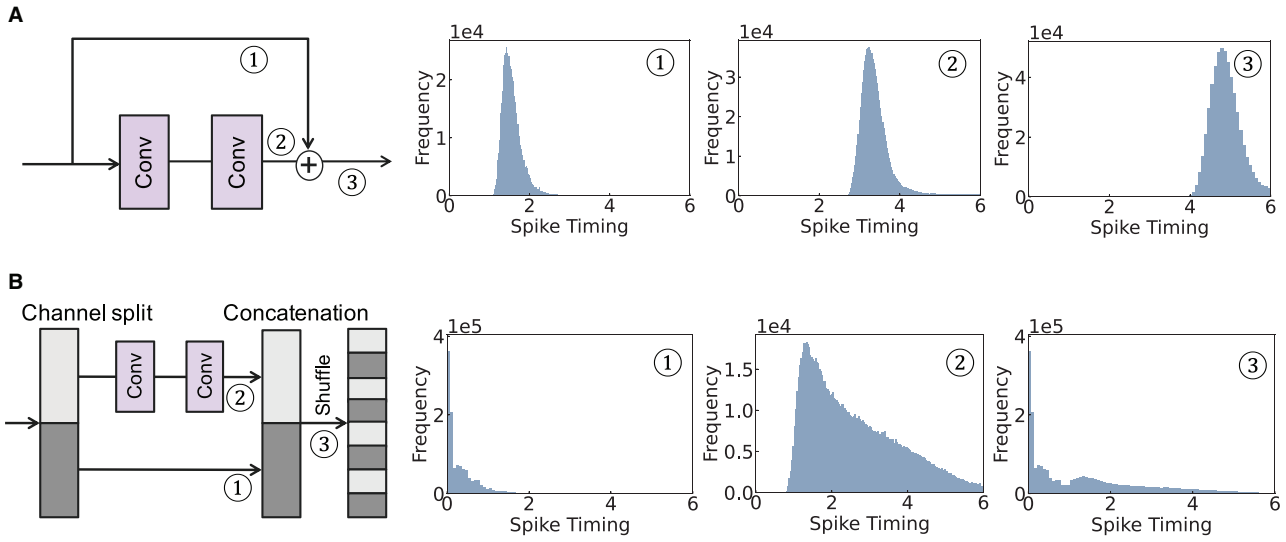

FIGURE 2
Illustration of a spike timing delay of two skip connection architectures. We visualize the histogram of spike timing for ① skip branch, ② convolutional branch, and ③ after combining the two branches. **(A)** Addition-based skip connection. **(B)** Concatenation-based skip connection.

neurons tend to prioritize the earlier input, usually coming from the skip connection. This skewed consideration can potentially lead to a drop in accuracy, as vital information from later inputs could be overlooked. This observation underscores the importance of appropriately managing the timing of inputs in temporal SNNs to ensure effective information integration and high network performance.

To summarize, addition-based skip connections introduce additional timing delays in temporal SNNs. On the other hand, concatenation-based skip connections, despite speeding up the latency during inference, may overlook crucial information from the convolutional branch.

3.3 Adding learnable delay with a skip connection

Hence, a question naturally arises: *how can we improve the accuracy while reducing latency in TTFS?* We focus on the problem of concatenation-based skip connections, i.e., timing discrepancy between the convolutional branch and the skip branch. To address this problem, we introduce a delay to the skip connection, which is designed to minimize the timing disparity between the two branches. Figure 3 provides an illustration of the delay implementation within the skip connection, where a delay is added across each channel. This introduces a slight adjustment to the original concatenation-based skip connection architecture. Initially, we partition the feature map across the channel dimension as follows:

$$X_l = [X_{l,1}; X_{l,2}]. \tag{10}$$

Subsequently, we apply a convolution layer $F(\cdot)$ to $X_{l,1}$ and a delay block $D(\cdot)$ to $X_{l,2}$. Then we concatenate the outputs from those branches:

$$X_{l+1} = \text{Shuffle}(F(X_{l,1})||D(X_{l,2}|\theta_l)), \tag{11}$$

where the delay block can be written as follows:

$$D(X|\theta_l) = X + \theta_l. \tag{12}$$

Here, $\theta_l \in \mathbb{R}^D$ represents the parameters within the delay block in l-th layer, where D is the channel dimension. This delay block applies distinct delays across each channel. We train the parameter θ_l alongside the other weight parameters within the neural network. To further align the distributions from the convolutional layer and the skip connection, we introduce an additional loss constraint during optimization:

$$\mathcal{L}_{overlap} = \sum_l ||Mean(F(X_{l,1})) - Mean(D(X_{l,2}|\theta_l))||_2^2. \tag{13}$$

This loss function encourages the two distributions to converge, enhancing the efficacy of the information mixing between the convolutional and skip branches.

3.4 Overall optimization

For the given image X, we convert the float input to the spike timing. Similar to rate coding (Wu et al., 2018, 2019; Kim et al., 2022b), we add one convolutional layer at the first layer, i.e., *Conv-BN-ReLU*. Then we directly utilize the output of the ReLU layer to the spike timing ($t = ReLU(BN(Conv(X)))$) as the output of the ReLU layer is always higher than zero. We pass through the multiple temporal neuron layers as described in Section 3.1. In the output layer, the class probability is computed from the spike timing. The objective is to train the network such that the neuron associated with the correct class is the first to fire among all neurons. This can be done by applying the cross-entropy loss to the output spike timing $O \in \mathbb{R}^C$ of the last layer:

$$\mathcal{L}_{ce} = \sum_c -y_c ln \frac{exp(-O_c)}{\sum_i exp(-O_i)}. \tag{14}$$

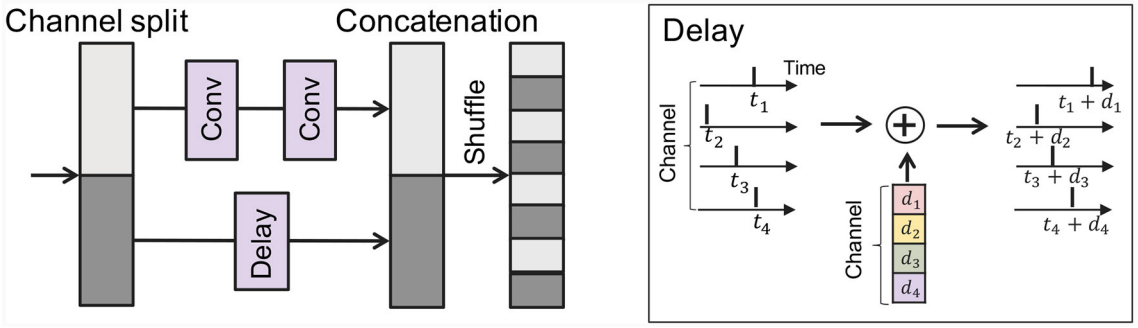

FIGURE 3
Illustration of the concatenation-based skip connection architecture with a delay block.

Here, y_c is a one-hot encoding of a class index, and O_c denotes the c-th index of output neurons. Multiplying -1 with the output neuron is for assigning higher probability weighting for the early spike.

With a cross-entropy loss for classification, following the prior research (Mostafa, 2017), we introduce an additional term to the cost function that penalizes the input weight vectors of neurons whose sum is below 1 (denominator of Equation 6).

$$\mathcal{L}_{weight} = \sum_{l} \sum_{i} max(0, 1 - \sum_{j} w_{ij,l}). \tag{15}$$

Here, i is the neuron index of layer l, and j is the input neuron index from the previous layer to neuron i. Overall, the total loss function is defined as follows.

$$\mathcal{L}_{total} = \mathcal{L}_{ce} + \lambda_1 \mathcal{L}_{weight} + \lambda_2 \mathcal{L}_{overlap}, \tag{16}$$

where λ_1 and λ_2 are the hyperparameters for a trade-off between the losses.

4 Experiments

4.1 Implementation details

We evaluate our method on MNIST (LeCun, 1998) and Fashion-MNIST (Xiao et al., 2017). We train the model with 128 batch samples using Adam optimizer with weight decay 1e-3. The initial learning rate is set to 6e-4 and decayed with cosine learning rate scheduling (Loshchilov and Hutter, 2016). We set the total number of epochs to 100. We set λ_1 and λ_2 to 1 and 1e-6, respectively. We use PyTorch for implementation.

4.2 Experiments on image recognition

Here, we compare the accuracy and latency across standard convolutional networks, addition-based skip connection architecture, and concatenation-based skip connection architecture. More concretely, we construct baseline CNN architecture as follows: *Conv(3,32)-Maxpool(2)-Conv(32,32, stride 2)-Conv(32,32)-Conv(32,64, stride 2)-Conv(64,64)-FC(10)*. For addition-based skip connection architecture, we add a residual connection for *Conv(32,32, stride 2)-Conv(32,32)* block and *Conv(32,64, stride 2)-Conv(64,64)*. In the skip connection, we apply a pooling layer to align the resolution between the skip connection and the convolutional branch. For a concatenation-based skip connection architecture, we also use a similar baseline CNN architecture. Figure 4 shows the details of the architectures.

To compare these architectures, we introduce latency as an additional metric, along with accuracy. In this context, latency is defined as the average time taken for the first spike to occur in the final layer. This is a particularly relevant measure for TTFS coding, as operations can be terminated as soon as the first spike occurs in the last layer. As such, we present both latency and accuracy in our results, offering a comprehensive understanding of the trade-off between speed and precision in these varying architectural designs.

In Tables 1, 2, we present the accuracy and latency results for the MNIST and Fashion-MNIST datasets, respectively. Several key observations can be made from these results: (1) The addition of skip connections significantly improves model accuracy compared to the baseline model. Specifically, the skip connection model improves performance by approximately 4% for both the MNIST and Fashion-MNIST datasets. (2) While the addition-based skip connection enhances accuracy, it also results in increased latency. This additional latency stems from the additive operation between the skip and convolutional branches, as discussed in Section 3.2.2. (3) The concatenation-based skip connection model, without a delay block, achieves a reduction in latency. However, its performance is comparatively lower than that of the addition-based skip connection architecture. (4) Incorporating a delay block into the concatenation-based skip connection model leads to improved performance. Notably, the addition of the delay block does not significantly increase latency for either dataset. In summary, among the tested architectures, the concatenation-based skip connection model with a delay block provides the optimal balance of high performance and low latency.

4.3 Comparison with previous methods

To establish the effectiveness of our proposed architecture, we draw comparisons between our model, Concat-based Skip Connection with Delay, and previous models. We focus on models that have been applied to the MNIST and Fashion-MNIST datasets. Table 3 shows results for the MNIST dataset.

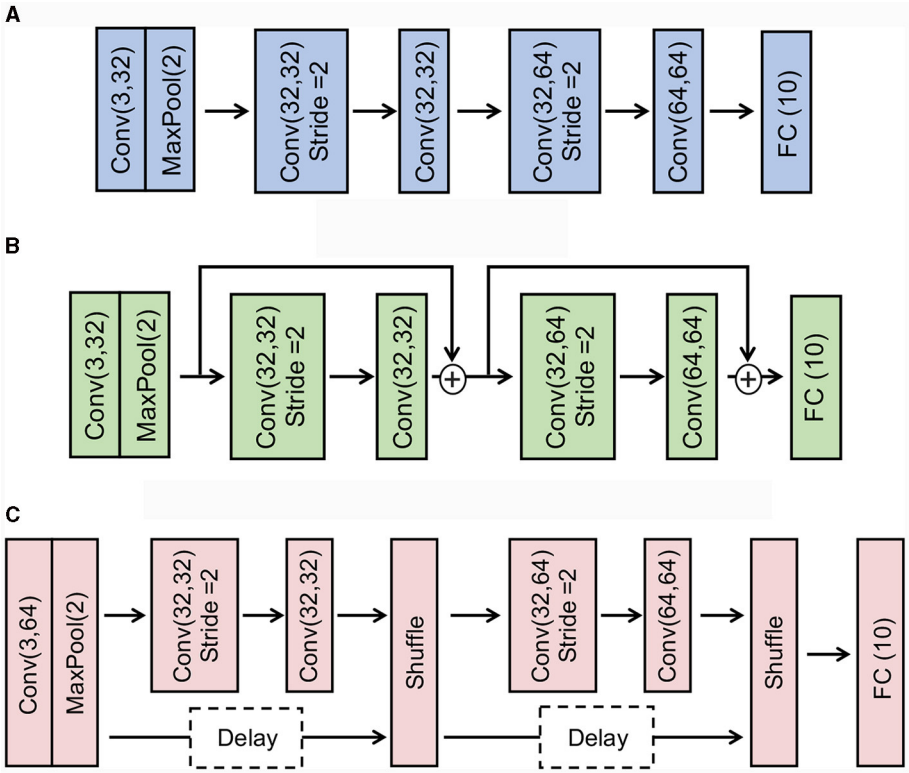

FIGURE 4
Illustration of the architectures of Baseline CNN, Addition-based skip connection, and Concatenation-based skip connection. **(A)** Baseline CNN. **(B)** Addition-based skip connection architecture. **(C)** Concatenation-based skip connection architecture.

TABLE 1 Classification accuracy (%) and latency of skip connection architectures on the MNIST dataset.

Method	Accuracy (%)	Latency
Baseline (Convolution layers without skip connection)	97.9	3.53
Addition-based skip connection	98.4	4.57
Concatenation-based skip connection	98.4	2.16
Concatenation-based skip connection + Delay Block	98.5	1.88

TABLE 2 Classification accuracy (%) and latency of skip connection architectures on the Fashion-MNIST dataset.

Method	Accuracy (%)	Latency
Baseline (Convolution layers without skip connection)	90.7	3.71
Addition-based skip connection	91.1	4.89
Concatenation-based skip connection	90.6	2.28
Concatenation-based skip connection + Delay Block	91.4	2.29

Our model achieves an accuracy of 98.5%, which is competitive with prior work. For the Fashion-MNIST dataset, Table 4 illustrates the superior performance of our model, which achieves an accuracy of 91.4%. The model by Zhang et al. (2021) is the next best performer with an accuracy of 90.1%. In both cases, our Concat-based Skip Connection with Delay architecture outperforms previous models, indicating its effectiveness in enhancing the performance of Spiking Neural Networks. This is particularly noteworthy for tasks that involve Time-To-First-Spike (TTFS) coding. The success of the Concat-based Skip Connection with Delay model may be attributed to its ability to address the timing discrepancy between convolutional and skip branches, thus maintaining high performance while achieving low latency.

4.4 Experimental analysis

4.4.1 Energy efficiency: spike rate comparison

As highlighted in an earlier section, the ability to terminate operation within the networks as soon as the first spike at the last layer occurs is an inherent advantage of TTFS coding. To fully leverage this characteristic, we assess the number of spikes present in each layer when we implement an early exit strategy. In Figure 5, we visualize the spike rate of the baseline, addition-based skip connection architecture, and concatenation-based skip connection architecture. Here, we use the concatenation-based skip connection results with the added delay. In this context, the spike rate refers to the proportion of firing neurons in a given layer. This metric is

instrumental in understanding the speed and efficiency of each model, and consequently, its suitability for real-time or latency-sensitive applications.

The experimental results showcased in Figure 5 provide a clear comparison between the baseline model, the addition-based skip connection model, and the concatenation-based skip connection model in terms of spike rate at different layers of the network. In the baseline model, the spike rates for conv2, conv3, conv4, and conv5 layers are 74.49, 53.45, 64.37, and 26.99% respectively. For the addition-based skip connection model, the spike rates significantly increase, recorded at 85.5, 83.12, 90.61, and 86.02% for conv2, conv3, conv4, and conv5 layers, respectively. These high spike rates demonstrate the model's heightened activity during the computation process. On the other hand, the concatenation-based skip connection model exhibits considerably lower spike rates. Specifically, the rates for conv2, conv3, conv4, and conv5 layers are 61.59, 49.43, 30.67, and 7.59%, respectively. This indicates that fewer neurons are active during computation, thereby leading to lower latency and potentially faster computation times.

In summary, while the addition-based skip connection model tends to enhance activity across the network, the concatenation-based model successfully reduces the spike rate, potentially improving the efficiency of the network, particularly in latency-sensitive scenarios. These results further establish the importance of an appropriate skip connection strategy in designing efficient SNNs.

TABLE 3 Accuracy (%) comparison among the previous work on the MNIST dataset.

Method	Coding	Accuracy (%)
Mostafa (2017)	Temporal	97.5
Comsa et al. (2020)	Temporal	97.9
Kheradpisheh and Masquelier (2020)	Temporal	97.4
Kheradpisheh et al. (2022)	Temporal	97.0
Sakemi et al. (2021)	Temporal	98.0
Ours (Concat-based Skip + Delay)	Temporal	98.5

TABLE 4 Accuracy (%) comparison among the previous work on the Fashion-MNIST dataset.

Method	Coding	Accuracy (%)
Hao et al. (2020)	Rate	85.3
Zhang and Li (2020)	Rate	89.5
Ranjan et al. (2020)	Rate	89.0
Kheradpisheh and Masquelier (2020)	Temporal	88.0
Kheradpisheh et al. (2022)	Temporal	87.3
Zhang et al. (2021)	Temporal	90.1
Ours (Concat-based Skip + Delay)	Temporal	91.4

4.4.2 Comparison with ANN

SNNs are renowned for their energy efficiency in comparison to traditional ANNs. To demonstrate this, we undertake a comparison of the approximate energy consumption between SNNs and ANNs, assuming that both are built using the same architecture. Due to their event-driven nature and binary 1, 0 spike processing, SNNs are characterized by a reduced complexity of computations. More specifically, a multiply-accumulate (MAC) operation reduces in an SNN to a simple floating-point (FP) addition, thus necessitating only an accumulation (AC) operation. In contrast, traditional ANNs still require full MAC operations. In line with previous studies such as Lee et al. (2016) and Park et al. (2020), we estimate the energy consumption for SNNs by quantifying the total MAC operations involved. Using the standard 45 nm CMOS technology as a reference point (Horowitz, 2014), we assign the energy for MAC and AC operations as $E_{MAC} = 4.6pJ$ and $E_{AC} = 0.9pJ$, respectively. This shows that MAC operations consume approximately 5.11 times more energy than AC operations. Since neurons in SNNs only consume energy whenever a neuron spikes, we multiply the spiking rate $R_s(l)$ at layer l with FLOPs to obtain the SNN FLOPs count. The total inference energy of ANNs (E_{ANN}) and SNNs (E_{SNN}) are calculated by: $E_{ANN} = \sum_l FLOPs(l) \times E_{MAC}$ and $E_{SNN} = \sum_l FLOPs(l) \times R_s(l) \times E_{AC}$, respectively. $FLOPs(l)$ represents the number of FLOPs in layer l.

The energy consumption comparison between our method and a conventional ANN is presented in Table 5. Notably, while maintaining competitive performance, our method—which combines concatenation-based skip connections with

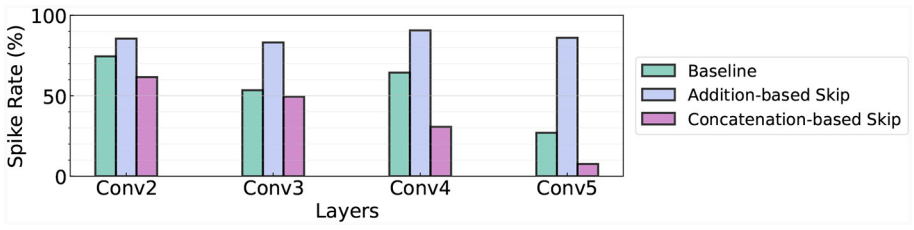

FIGURE 5
Spike rate in different architectures. We use the Fashion-MNIST dataset and measure the spike rate in Conv2-Conv5 layers.

TABLE 5 Energy-efficiency comparison between ANN and SNN (with concatenation-based skip + delay) at inference.

Method	Relative energy cost	Accuracy (%)
ANN	1	92.27
Concat-based Skip + Delay	0.13	91.41

We use the Fashion-MNIST dataset for the experiments.

TABLE 6 Accuracy (%) comparison among the different delay block designs.

Method	Accuracy (%)	Latency
Layer-wise delay	90.71	3.5
Channel-wise delay	91.41	2.29
Pixel-wise delay	91.11	2.39

We use the Fashion-MNIST dataset for the experiments.

a delay block—significantly reduces the relative energy cost. In contrast to the ANN's relative energy cost of 1, our method operates at just 13% of the ANN's energy expenditure, thus demonstrating the superior energy efficiency of our approach.

4.4.3 Delay design

There are several design variants for delay blocks. Here, we propose three types for comparison: (1) Layer-wise delay applies the same delay across all neurons in one layer. (2) Channel-wise delay adds a timing delay for each channel, which is used in our method. (3) Pixel-wise delay adds a timing delay for each spatial location. For intermediate layers where the feature tensor size is $C \times H \times W$, we apply delay for each $H \times W$ location. Table 6 presents the comparative performance of these three delay block design variants, specifically evaluating their impact on accuracy and latency using the Fashion-MNIST dataset. The results suggest that the application of delay varies significantly in effect depending on its implementation level. Specifically, channel-wise delay, as employed in our method, demonstrated the highest accuracy and the lowest latency, indicating its effectiveness for the integration into concatenation-based skip connections. This demonstrates the potential benefits of applying unique delays to each channel, providing an effective balance between performance and computational efficiency.

4.4.4 Analysis on delay value

In this section, we investigate the impact of different initial values within the delay block. For this purpose, we set the initial delay values (θ_l in Equation 12) to [0, 0.25, 0.5, 0.75, 1.0] and train the model accordingly. As depicted in Figure 6 (left), we present the performance corresponding to each initialization time. It is observed that when initialized with a small delay, the resulting latency remains small. Conversely, when the delay value is initialized to be higher, the resultant latency increases, but this also leads to an enhancement in accuracy. Figure 6 (right) visualizes how these delay values fluctuate across epochs. We note that regardless of the initial value, all cases tend to gravitate toward a middle value over time. This results in high initial values decreasing over time, while lower initial values witness an increase.

4.4.5 Advantage of skip connection: improving model stability

Our analysis further explores the benefits of incorporating skip connections within the temporal SNN architecture. Prior research in the field of ANNs suggests that the inclusion of skip connections enhances training stability and accelerates convergence speed (He et al., 2016). We scrutinize this premise in our current context by tracking and visualizing the evolution of training loss and accuracy over successive epochs, as depicted in Figure 7. Our findings corroborate the aforementioned assertions, demonstrating that our optimized SNN architectures foster swift convergence while maintaining high test accuracy. This reinforces the value of skip connections as a significant contribution to the performance and efficiency of temporal SNNs. We also perform a robustness analysis on the temporal skip connection by adding Gaussian noise to the input image. Figure 8 illustrates the accuracy change with respect to different noise levels. As the noise increases, the performance gap between the two models goes larger. Specifically, the two models exhibit an accuracy gap of $\leq$ 1% without noise, which increases to approximately $\sim$ 12% in the case of $\sigma = 0.2$. This observation aligns with prior literature in ANN (Huang et al., 2023), where they demonstrate that the residual connection enhances the robustness of the model. Overall, our results suggest that adding a skip connection in temporal SNN significantly improves the robustness of the model.

4.5 Experiments on wave equation

In broadening the scope of our investigation, we extend the application of TTFS coding to tasks within the domain of scientific machine learning (SciML). An important topic in this field is how to approximate accurately solutions of partial differential equations. In particular, we are interested in solving the inverse (time-reversal) problem of locating sources in an underwater acoustic domain from measurements at a later time. There has been a lot of research in this domain, with or without machine learning, as shown in this survey (Grumiaux et al., 2022). The recent developments of Transformer-based architectures have recently been used for this task as well (Ovadia et al., 2023). Another proposed method refers to using the Time-Reversal method incorporated with Machine learning based inference system (Bardos and Fink, 2002; Givoli and Turkel, 2012; Kahana et al., 2022). Most methods still rely on ANNs that can be expensive to train.

The challenge herein is formulating this problem as a classification problem which aligns with the current implementation of the TTFS. The mathematical formulation of the wave problem we will investigate in this work is given by:

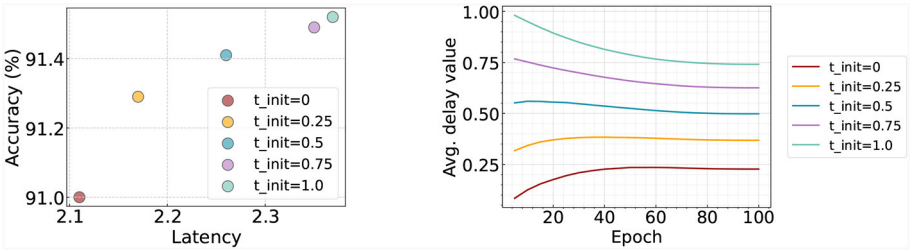

FIGURE 6
Left: Accuracy and latency trade-off with respect to the time initialization in delay block. **Right:** Change of delay as training goes on. We train and test the model on Fashion-MNIST.

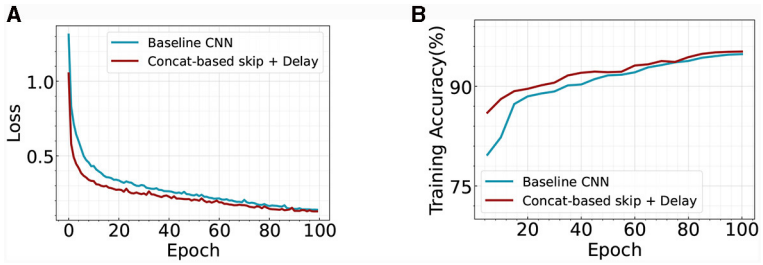

FIGURE 7
Comparison of the **(A)** training loss and **(B)** training accuracy across different architectures on Fashion-MNIST.

$$\begin{cases} \ddot{u}(x,y,t) = c^2 \Delta u(x,y,t) & (x,y) \in \Omega = [x_{min}, x_{max}] \times \\ & [y_{min}, y_{max}], \quad t \in (0,T], \\ u(x,y,0) = u_0(x,y) & (x,y) \in \Omega, \\ \dot{u}(x,y,0) \equiv 0 & (x,y) \in \Omega, \\ u(x,y,t) = 0 & (x,y) \in \partial\Omega, \quad t \in [0,T], \end{cases} \tag{17}$$

where $u(x,y,t)$ is the acoustic wave pressure, and c is the wave propagation velocity (assumed constant in this work and equal to $1484\frac{m}{s}$). A single dot over u denotes a first derivative with respect to time. A double dot denotes a second derivative. The function u_0 is the initial condition for the wave propagation, determined by the source. This initial condition is taken as a small Gaussian eruption ($f(x) = e^{-(\frac{x-\bar{x}}{0.05})^2}$, $\bar{x} = \frac{x_{max}-x_{min}}{2}$) that mimics a localized source (point source that has been smoothed to avoid numerical artifacts). The goal is, from measurements of the wave pressure at some time t, to recover the location of the small initial eruption. This location is defined as a grid point. To turn it into a classification problem, we split the grid into zones, and infer the zone where the source is located. The more zones we use, the more precise localization we can achieve. However, more zones create a harder classification task, with growing computational demands.

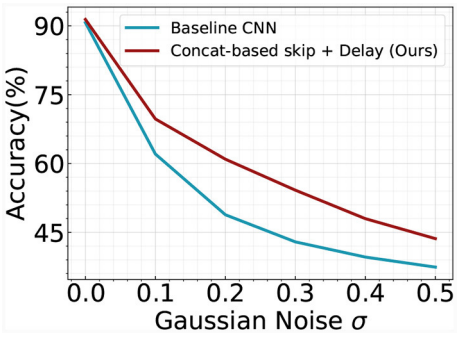

FIGURE 8
Robustness analysis of baseline CNN and our Concatenation-based skip + Delay architecture. We add Gaussian noise to the input image. The experiments are conducted on the Fashion-MNIST dataset.

4.5.1 Dataset configuration

We generate a synthetic dataset based on the wave equation for a domain of $N_x \times N_y$ locations. We use a finite-difference central differences numerical scheme, preserving up to second order accuracy. We choose the ratio between the spatial discretization and the temporal discretization to satisfy the Courant–Freidrichs–Lewy condition (so that the scheme is stable). To create the synthetic dataset, for each sample, we choose a location for the source, use the solver to march in time and compute the wave pressure across the domain at the 100-th time step. Then we create a label based on the location of the source, so the pairs consisting of an image of the pressure at the 100-th time step and its corresponding label (source) form the dataset.

In detail, we posit the existence of a wave source at all locations within the domain, excluding a 10 pixels border around the outer boundary, thus resulting in $(N_x - 10) \times (N_y - 10)$ data samples. As mentioned above, for each wave source location, we compute

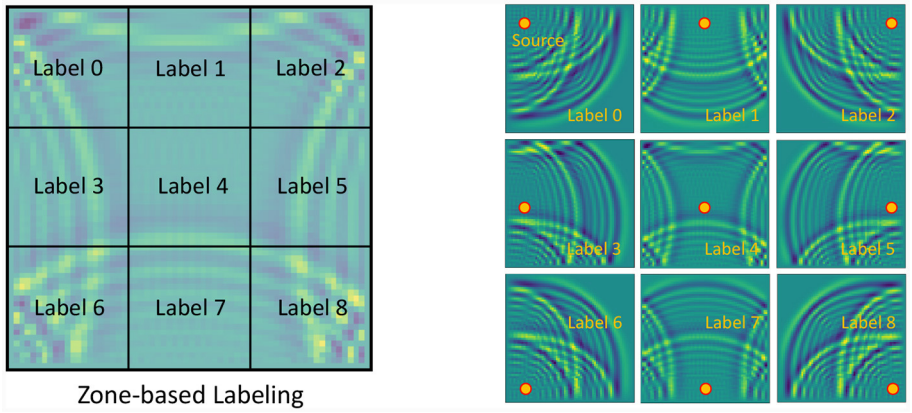

FIGURE 9
The figure on the **left** illustrates an example of zone-based labeling for the source localization problem for the wave equation. The image is divided into 3 × 3 zones, with each zone assigned a distinct label. On the **right**, nine different wave images corresponding to each of the nine labels are displayed. Each image represents the wave shape at the 100-th time step, originating from a wave source located at the center of the respective zone.

TABLE 7 **Classification accuracy (%) and latency on time-reversal source localization problem for the wave equation.**

Method	Labeling strategy	Accuracy (%)	Latency
Baseline (Convolution layers without skip connection)	3 × 3	99.48	4.13
Addition-based skip connection	3 × 3	99.50	4.94
Concatenation-based skip connection	3 × 3	96.36	1.06
Concatenation-based skip connection + Delay Block	3 × 3	99.74	2.83
Baseline (Convolution layers without skip connection)	6 × 6	97.93	4.57
Addition-based skip connection	6 × 6	97.94	4.87
Concatenation-based skip connection	6 × 6	97.68	2.42
Concatenation-based skip connection + Delay Block	6 × 6	98.19	3.45

Labeling strategy means we use $M \times M$ zones for assigning a label to each source location. Figure 9 shows an example of 3 × 3 zone-based labeling.

the wave pressure at the 100-th time step. To create the labels, we segment the domain into $M \times M$ zones and assign each source location a label from $0 - (M^2 - 1)$, as shown in Figure 9 for a 3 × 3 zone-based labeling. The labeling process is the basic quantization of the domain into smaller segments, and the assigning label for each source according to the region it belongs to. For the 3 × 3 labeling, we have nine zones, meaning a total of nine classes. To get a more precise localization one can use 6 × 6 zones (as shown later), thus having 36 classes for the classification mechanism. Finally, we partition these data samples randomly into training and testing sets at an 80:20 ratio, respectively.

In Table 7, we report the accuracy and latency of the wave equation problem. We use the architecture shown in Figure 4. Note that 6 × 6 zone-based labeling is a more difficult task than 3 × 3 zone-based labeling as the model requires classifying a larger number of zones. We make the following observations: (1) The general performance trend aligns with what we observed for image recognition tasks. Concatenation-based skip connection architectures, especially when paired with a delay block, show superior performance in terms of balancing high accuracy with lower latency. This supports the effectiveness of our proposed architecture for not only static image datasets but also for SciML tasks such as solving time-reversal problems for wave equations. (2) The table also reveals that as we change the labeling strategy from 3 × 3 zones to 6 × 6 zones, we see an increase in latency across all models. This is intuitively right as a higher number of zones (classes) involves more computations and hence results in a longer latency. In addition to the increased latency, a larger zone number also results in slightly reduced accuracy for all methods. This could be due to the increased complexity of the task with more zones, potentially requiring a more sophisticated model or additional training to achieve similar levels of accuracy as those observed for the smaller number of zones. In Table 8, we compare the energy efficiency of SNN with ANN. Similar to image classification, our SNN consumes only 14% of ANN energy, while sacrificing $\leq$ 1% accuracy.

5 Conclusion

In conclusion, this study has made significant strides in the exploration of TTFS coding and the optimization of skip connection architectures for improving the efficiency and accuracy of SNNs. We discovered that while addition-based skip connections

TABLE 8 Energy-efficiency comparison between ANN and SNN (with concatenation-based skip + delay) at inference on the wave equation problem.

Method	Relative energy cost	Accuracy (%)
ANN	1	98.92
Concat-based skip + Delay	0.14	98.19

We use the 6 × 6 zone-based labeling for the experiments.

introduce temporal delays, concatenation-based skip connections tend to miss crucial information from the non-linear operation branch. To address these challenges, we proposed a novel approach that introduces a learnable delay for skip connections, bridging the gap between the spike timing discrepancies of the convolution and skip branches. We demonstrated that this method not only accelerates the first spike's timing but also maintains accuracy, offering an effective solution for faster prediction in TTFS coding. We also extended our exploration to SciML tasks, unveiling the potential of TTFS coding beyond image recognition applications. Our findings suggest that there is room for further research in optimizing the network architecture of temporal SNNs, and we hope that our work will inspire new approaches and applications in this exciting field. In the future, we aim to further improve the effectiveness of our proposed method and explore its applicability to even larger and more complex tasks. We believe that the continuing evolution of SNN architectures will significantly contribute to the advancement of low-power, efficient, and highly accurate artificial intelligence systems.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

YK: Conceptualization, Investigation, Methodology, Software, Visualization, Writing - original draft, Writing - review & editing. AK: Data curation, Methodology, Project administration, Supervision, Validation, Writing - review & editing. RY: Conceptualization, Investigation, Methodology, Software, Writing - review & editing. YL: Investigation, Software, Writing - review & editing. PS: Formal analysis, Funding acquisition, Project administration, Supervision, Validation, Writing - review & editing. GK: Funding acquisition, Project administration, Resources, Supervision, Validation, Visualization, Writing - review & editing. PP: Conceptualization, Funding acquisition, Investigation, Project administration, Resources, Supervision, Validation, Writing - original draft, Writing - review & editing.

Funding

The author(s) declare financial support was received for the research, authorship, and/or publication of this article. This work was supported in part by CoCoSys, a JUMP2.0 center sponsored by DARPA and SRC, the National Science Foundation (CAREER Award, Grant #2312366, Grant #2318152), TII (Abu Dhabi), and the U.S. Department of Energy, Advanced Scientific Computing Research program, under the Scalable, Efficient and Accelerated Causal Reasoning Operators, Graphs and Spikes for Earth and Embedded Systems (SEA-CROGS) project (Project No. 80278). Pacific Northwest National Laboratory (PNNL) is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL01830.

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.

The author(s) declared that they were an editorial board member of Frontiers, at the time of submission. This had no impact on the peer review process and the final decision.

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

Bardos, C., and Fink, M. (2002). Mathematical foundations of the time reversal mirror. *Asympt. Anal.* 29, 157–182.

Benmeziane, H., Ounnoughene, A. Z., Hamzaoui, I., and Bouhadjar, Y. (2023). Skip connections in spiking neural networks: an analysis of their effect on network training. *arXiv preprint arXiv:2303.13563*.

Bohte, S. M., Kok, J. N., and La Poutré, J. A. (2000). "Spikeprop: backpropagation for networks of spiking neurons," in *The European Symposium on Artificial Neural Networks*, Vol. 48 (Bruges), 419–424.

Cao, Y., Chen, Y., and Khosla, D. (2015). Spiking deep convolutional neural networks for energy-efficient object recognition. *Int. J. Comput. Vis.* 113, 54–66. doi: 10.1007/s11263-014-0788-3

Christensen, D. V., Dittmann, R., Linares-Barranco, B., Sebastian, A., Le Gallo, M., Redaelli, A., et al. (2022). 2022 roadmap on neuromorphic computing and engineering. *Neuromorph. Comput. Eng.* 2, 022501. doi: 10.1088/2634-4386/ac4a83

Comsa, I. M., Fischbacher, T., Potempa, K., Gesmundo, A., Versari, L., and Alakuijala, J. (2020). "Temporal coding in spiking neural networks with alpha synaptic

function," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE), 8529–8533.

Diehl, P. U., and Cook, M. (2015). Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front. Comput. Neurosci.* 9, 99. doi: 10.3389/fncom.2015.00099

Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S.-C., and Pfeiffer, M. (2015). "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)* (IEEE), 1–8.

Fang, W., Yu, Z., Chen, Y., Masquelier, T., Huang, T., and Tian, Y. (2020). Incorporating learnable membrane time constant to enhance learning of spiking neural networks. *arXiv preprint arXiv:2007.05785*.

Fang, W., Yu, Z., Chen, Y., Huang, T., Masquelier, T., and Tian, Y. (2021). Deep residual learning in spiking neural networks. *arXiv preprint arXiv:2102.04159*.

Givoli, D., and Turkel, E. (2012). Time reversal with partial information for wave refocusing and scatterer identification. *Comput. Methods Appl. Mech. Eng.* 213, 223–242. doi: 10.1016/j.cma.2011.12.005

Grumiaux, P.-A., Kitić, S., Girin, L., and Guérin, A. (2022). A survey of sound source localization with deep learning methods. *J. Acoust. Soc. Am.* 152, 107–151. doi: 10.1121/10.0011809

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

Han, B., Srinivasan, G., and Roy, K. (2020). "RMP-SNN: residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13558–13567.

Hao, Y., Huang, X., Dong, M., and Xu, B. (2020). A biologically plausible supervised learning method for spiking neural networks using the symmetric STDP rule. *Neural Netw.* 121, 387–395. doi: 10.1016/j.neunet.2019.09.007

He, K., Zhang, X., Ren, S., and Sun, J. (2016). "Deep residual learning for image recognition," in *CVPR*, 770–778.

Horowitz, M. (2014). "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* (IEEE), 10–14.

Hu, Y., Tang, H., and Pan, G. (2021). Spiking deep residual networks. *IEEE Trans. Neural Netw. Learn. Syst.* 34, 5200–5205. doi: 10.1109/TNNLS.2021.3119238

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4700–4708.

Huang, S., Lu, Z., Deb, K., and Boddeti, V. N. (2023). "Revisiting residual networks for adversarial robustness," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8202–8211.

Ikegawa, S.-I., Saiin, R., Sawada, Y., and Natori, N. (2022). Rethinking the role of normalization and residual blocks for spiking neural networks. *Sensors* 22, 2876. doi: 10.3390/s22082876

Kahana, A., Turkel, E., Dekel, S., and Givoli, D. (2022). A physically-informed deep-learning model using time-reversal for locating a source from sparse and highly noisy sensors data. *J. Comput. Phys.* 470, 111592. doi: 10.1016/j.jcp.2022.111592

Kheradpisheh, S. R., and Masquelier, T. (2020). Temporal backpropagation for spiking neural networks with one spike per neuron. *Int. J. Neural Syst.* 30, 2050027. doi: 10.1142/S0129065720500276

Kheradpisheh, S. R., Mirsadeghi, M., and Masquelier, T. (2022). BS4NN: binarized spiking neural networks with temporal coding and learning. *Neural Process. Lett.* 54, 1255–1273. doi: 10.1007/s11063-021-10680-x

Kim, Y., Li, Y., Park, H., Venkatesha, Y., and Panda, P. (2022a). "Neural architecture search for spiking neural networks," in *Computer Vision–ECCV 2022: 17th European Conference* (Tel Aviv: Springer), 36–56.

Kim, Y., Park, H., Moitra, A., Bhattacharjee, A., Venkatesha, Y., and Panda, P. (2022b). "Rate coding or direct coding: which one is better for accurate, robust, and energy-efficient spiking neural networks?" in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE), 71–75.

LeCun, Y. (1998). *The Mnist Database of Handwritten Digits*. Available online at: http://yann.lecun.com/exdb/mnist/

Lee, C., Sarwar, S. S., Panda, P., Srinivasan, G., and Roy, K. (2020). Enabling spike-based backpropagation for training deep neural network architectures. *Front. Neurosci.* 14, 119. doi: 10.3389/fnins.2020.00119

Lee, J. H., Delbruck, T., and Pfeiffer, M. (2016). Training deep spiking neural networks using backpropagation. *Front. Neurosci.* 10, 508. doi: 10.3389/fnins.2016.00508

Li, Y., Deng, S., Dong, X., Gong, R., and Gu, S. (2021a). A free lunch from ANN: towards efficient, accurate spiking neural networks calibration. *arXiv preprint arXiv:2106.06984*.

Li, Y., Guo, Y., Zhang, S., Deng, S., Hai, Y., and Gu, S. (2021b). "Differentiable spike: rethinking gradient-descent for training spiking neural networks," in *Advances in Neural Information Processing Systems 34*, 23426–23439.

Loshchilov, I., and Hutter, F. (2016). SGDR: stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

Ma, N., Zhang, X., Zheng, H.-T., and Sun, J. (2018). "ShuffleNet V2: practical guidelines for efficient CNN architecture design," in *Proceedings of the European conference on computer vision (ECCV)*, 116–131.

Montemurro, M. A., Rasch, M. J., Murayama, Y., Logothetis, N. K., and Panzeri, S. (2008). Phase-of-firing coding of natural visual stimuli in primary visual cortex. *Curr. Biol.* 18, 375–380. doi: 10.1016/j.cub.2008.02.023

Mostafa, H. (2017). 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. *IEEE Signal Process. Mag.* 36, 61–63. doi: 10.1109/MSP.2019.2931595

Ovadia, O., Kahana, A., Stinis, P., Turkel, E., and Karniadakis, G. E. (2023). ViTO: vision transformer-operator. *arXiv preprint arXiv:2303.08891*.

Panda, P., Aketi, S. A., and Roy, K. (2020). Toward scalable, efficient, and accurate deep spiking neural networks with backward residual connections, stochastic softmax, and hybridization. *Front. Neurosci.* 14, 653. doi: 10.3389/fnins.2020.00653

Park, S., Kim, S., Choe, H., and Yoon, S. (2019). "Fast and efficient information transmission with burst spikes in deep spiking neural networks," in *2019 56th ACM/IEEE Design Automation Conference (DAC)* (IEEE), 1–6.

Park, S., Kim, S., Na, B., and Yoon, S. (2020). T2FSNN: deep spiking neural networks with time-to-first-spike coding. *arXiv preprint arXiv:2003.11741*.

Ranjan, J. A. K., Sigamani, T., and Barnabas, J. (2020). A novel and efficient classifier using spiking neural network. *J. Supercomput.* 76, 6545–6560. doi: 10.1007/s11227-019-02881-y

Roy, K., Jaiswal, A., and Panda, P. (2019). Towards spike-based machine intelligence with neuromorphic computing. *Nature* 575, 607–617. doi: 10.1038/s41586-019-1677-2

Rueckauer, B., and Liu, S.-C. (2018). "Conversion of analog to spiking neural networks using sparse temporal coding," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE), 1–5.

Rueckauer, B., Lungu, I.-A., Hu, Y., Pfeiffer, M., and Liu, S.-C. (2017). Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Front. Neurosci.* 11, 682. doi: 10.3389/fnins.2017.00682

Sakemi, Y., Morino, K., Morie, T., and Aihara, K. (2021). 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

Sengupta, A., Ye, Y., Wang, R., Liu, C., and Roy, K. (2019). Going deeper in spiking neural networks: VGG and residual architectures. *Front. Neurosci.* 13, 95. doi: 10.3389/fnins.2019.00095

Shrestha, S. B., and Orchard, G. (2018). Slayer: spike layer error reassignment in time. *arXiv preprint arXiv:1810.08646*.

Shrestha, S. B., and Song, Q. (2017). Robustness to training disturbances in spikeprop learning. *IEEE Trans. Neural Netw. Learn. Syst.* 29, 3126–3139. doi: 10.1109/TNNLS.2017.2713125

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). "Training very deep networks," in *Advances in Neural Information Processing Systems*, 28.

Weisstein, E. W. (2002). *Z-transform*. Available online at: https://mathworld.wolfram.com/

Wu, H., Zhang, Y., Weng, W., Zhang, Y., Xiong, Z., Zha, Z.-J., et al. (2021). Training spiking neural networks with accumulated spiking flow. *ijo* 1. doi: 10.1609/aaai.v35i12.17236

Wu, J., Xu, C., Zhou, D., Li, H., and Tan, K. C. (2020). Progressive tandem learning for pattern recognition with deep spiking neural networks. *arXiv preprint arXiv:2007.01204*.

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

Wu, Y., Deng, L., Li, G., Zhu, J., Xie, Y., and Shi, L. (2019). "Direct training for spiking neural networks: faster, larger, better," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, 1311–1318.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

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

Yang, S., and Chen, B. (2023a). Effective surrogate gradient learning with high-order information bottleneck for spike-based machine intelligence. *IEEE Trans. Neural Netw. Learn. Syst.* doi: 10.1109/TNNLS.2023.3329525

Yang, S., and Chen, B. (2023b). SNIB: improving spike-based machine learning using nonlinear information bottleneck. *IEEE Trans. Syst. Man Cybernet. Syst.* 1–12. doi: 10.1109/TSMC.2023.3300318

Yang, S., Wang, H., and Chen, B. (2023). Sibols: robust and energy-efficient learning for spike-based machine intelligence in information bottleneck framework. *IEEE Trans. Cogn. Dev. Syst.* doi: 10.1109/TCDS.2023.3329532

Zhang, A., Wu, J., Li, X., Li, H. C., Gao, Y., and Pun, S. H. (2023). Highway connection for low-latency and high-accuracy spiking neural networks. *IEEE Trans. Circ. Syst. II Exp. Briefs*. doi: 10.1109/TCSII.2023.3294418

Zhang, L., Zhou, S., Zhi, T., Du, Z., and Chen, Y. (2019). "TDSNN: from deep neural networks to deep spike neural networks with temporal-coding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, 1319–1326.

Zhang, M., Wang, J., Wu, J., Belatreche, A., Amornpaisannon, B., Zhang, Z., et al. (2021). 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. (2020). Temporal spike sequence learning via backpropagation for deep spiking neural networks. *arXiv preprint arXiv:2002.10085*.

Zheng, H., Wu, Y., Deng, L., Hu, Y., and Li, G. (2020). Going deeper with directly-trained larger spiking neural networks. *arXiv preprint arXiv:2011.05280*.