Efficient Training of Deep Spiking Neural Networks with Input-Driven Derivative-Free Updates

Yongbo Zhang[1,*] **Xinzhe Li**[1] **Katsuma Inoue**[1] **Mitsumasa Nakajima**[2]
Toshikazu Hashimoto[2] **Yasuo Kuniyoshi**[1] **Kohei Nakajima**[1,*]
[1]The University of Tokyo [2]NTT Device Technology Labs.
zhang@isi.imi.i.u-tokyo.ac.jp

https://github.com/RCsniper/IDDFT-SNN

Abstract

Spiking neural networks (SNNs) are a promising paradigm for efficient neuromorphic computing, yet their training remains challenging due to the non-differentiability nature of spike firing. Backpropagation through time (BPTT) with surrogate gradients (SG) has become a mainstream approach due to its superior performance. However, this method incurs high computational and memory demands during training, while its update rule depends on specific neuronal dynamics. Furthermore, when targeting neuromorphic hardware implementation, the temporal error-propagation process and the full membrane-potential access required by SG are difficult to realize; these can introduce additional readout overhead, measurement noise, or perturbations to the system dynamics. To address these issues, we propose the input-driven derivative-free training (IDDFT) method. Rather than relying on membrane-potential-based surrogate derivatives, IDDFT constructs derivative-free error signals by applying a relaxed nonlinearity to the inputs, thereby avoiding the temporal error-propagation process and reducing the dependence of the update rule on specific neuronal dynamics. By eliminating access to internal membrane-potential states, our method reduces computational and memory costs, enhances compatibility with neuromorphic hardware, and enables training of black-box SNN models. We validate the IDDFT method through theoretical analysis and systematic experiments, demonstrating its effectiveness as an input-driven, derivative-free mechanism for constructing update signals. When integrated with a biologically inspired training strategy, IDDFT maintains comparable performance while substantially enhancing robustness to the choice of relaxed nonlinearities. Based on the resulting bio-inspired combined framework, we conduct evaluations on CIFAR-10, CIFAR-100, CIFAR10-DVS, and Tiny-ImageNet. Experimental results demonstrate that our method achieves performance comparable to BPTT and state-of-the-art methods, while exhibiting remarkable robustness under adversarial attacks.

1 Introduction

Spiking neural networks (SNNs), considered as the third generation of neural networks [1], are biologically inspired by brain mechanism where information is transmitted and processed via discrete spikes [2, 3]. This event-driven processing mechanism enables efficient and low-energy computation, positioning SNNs as a promising paradigm for neuromorphic computing [4–6]. However, the non-differentiable dynamics of neurons and their discrete spike generation make direct training challenging.

*Corresponding authors

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

To address this training challenge, backpropagation through time (BPTT) [7] combined with surrogate gradients (SG) [8] has become a mainstream strategy for training deep SNNs. This approach preserves the discrete spike-firing process in the forward pass, while using continuous surrogate gradients in the backward pass to approximate the derivatives of non-differentiable spike functions. By unfolding the iterative states of spiking neurons along the temporal dimension, this method propagates error signals backward through time, facilitating effective gradient propagation across both network layers and temporal steps. This enables deep SNNs to be trained through gradient-based optimization and to achieve competitive performance even under low-latency conditions (e.g., 4-6 time steps) [9–14]. However, the temporal unfolding underlying BPTT requires storing neuronal states across all time steps for backward propagation across both network layers and temporal steps, resulting in substantial computational and memory costs, both of which grow in proportion to the number of time steps [10, 15, 16]. Moreover, its temporal error-propagation mechanism is challenging to implement on neuromorphic devices. Beyond the issues associated with BPTT, SG also imposes membrane-potential-dependent constraints on gradient computation. Since surrogate derivatives require precise tuning based on the underlying neuronal dynamics, the training process becomes highly model-specific and sensitive to the parameters and scale of these derivatives. [17]. More importantly, SG-based gradient computation requires continuous access to membrane potentials across all layers and time steps during training. In software implementations, this requires storing a large number of internal states, further increasing the memory burden. Conversely, in neuromorphic hardware, these membrane potentials may be difficult to access directly, and the readout process can impose additional I/O overhead and may even introduce measurement noise or perturb the system dynamics [18]. Recent research has attempted to alleviate the memory and computational overhead of BPTT by truncating or partially decoupling temporal gradient propagation [19–22]. However, these approaches still rely on local gradients derived from membrane potential, leaving the issues of dependency on neuronal dynamics and internal membrane-potential access unresolved.

Gradient-free training methods [23, 24] suggest that effective update directions can be obtained without exact activation derivatives or precise backward weights. Motivated by this observation, we explore how learning signals are constructed in deep SNN training arguing that update signals need not be derived from membrane-potential-based surrogate derivatives. In this work, we propose the input-driven derivative-free training (IDDFT) method for deep SNNs. Starting from the standard BPTT-SG formulation, we identify membrane-potential-based error signal construction as a primary factor coupling the update rule to temporal error-propagation, specific neuronal dynamics, and internal membrane-state access. Instead of relying on membrane-potential-based surrogate derivatives, IDDFT decouples these elements by constructing derivative-free error signals via a relaxed nonlinearity applied to the inputs. In this formulation, the backward nonlinearity is no longer intended to approximate the derivative of the spike function; instead, it serves as an independent input-response transformation for generating update signals. By bypassing membrane potentials, IDDFT disrupts temporal error-propagation through membrane-potential states and eliminates the need to read internal membrane dynamics during training. As a result, IDDFT mitigates computational and memory burdens while enhancing compatibility with neuromorphic hardware, and enables training of black-box SNN models. Our theoretical analysis (see Appendix A, with supporting statistical evidence in Appendix B) and experiments confirm the effectiveness of IDDFT as a versatile input-driven mechanism for constructing derivative-free update signals. To further improve its compatibility with neuromorphic learning principles, we then integrate IDDFT with a biologically inspired training strategy: uniform sign-concordant feedbacks (uSF) [25]. This hybrid training paradigm sustains competitive performance while broadening the range of suitable relaxed-nonlinearity choices, offering greater flexibility for practical neuromorphic implementation. Extensive experiments further demonstrate that uSF-IDDFT achieves performance comparable to state-of-the-art SNN training methods across various benchmarks. Furthermore, it demonstrates remarkable robustness in the face of adversarial attacks. Our main contributions are summarized as follows:

1. We propose Input-driven Derivative-free Training (IDDFT) for SNNs. By constructing derivative-free update signals from relaxed nonlinear input responses, IDDFT bypasses temporal error propagation through membrane-potential states while minimizing reliance on specific neuronal dynamics, enhancing neuromorphic compatibility, and substantially lowering computational and memory overheads during training. Under the same task setting, IDDFT reduces peak training memory by 67% compared with BPTT-SG; details are provided in Appendix D.1.

2. We provide theoretical analyses characterizing the approximation relationship between the update signals produced by IDDFT and BPTT, together with empirical evidence demonstrating the effectiveness of our method.
3. We integrate IDDFT with uSF, a biologically inspired training strategy, resulting in a combined paradigm that is closely aligned with neuromorphic learning principles and offers greater implementation flexibility.
4. We systematically evaluate the combined paradigm on CIFAR-10, CIFAR-100, CIFAR10-DVS, and Tiny-ImageNet, demonstrating performance comparable to BPTT and state-of-the-art SNN training methods, as well as significant robustness under adversarial attacks.

2 Related work

Methods for Training Deep SNNs. Supervised training of deep SNNs primarily follows two approaches: artificial neural network (ANN)-to-SNN conversion and direct training. ANN-to-SNN conversion typically first trains an ANN and then maps its continuous activations to spike-based representations through analytical formulations, such as firing rates, spike counts, or temporal encoding [26–29]. By leveraging mature ANN pipelines, this approach has achieved strong performance on large-scale tasks [30–34]. However, such methods rely on an approximate equivalence between ANN activations and SNN spike representations, typically requiring extensive time steps to accumulate spikes. Under low-latency conditions, insufficient time steps can introduce conversion errors and degrade performance [35–38]. In contrast, direct training methods optimize SNNs from scratch using BPTT combined with SG, enabling strong performance with extremely low time steps while supporting flexible neuron and network architecture designs [8–10, 12, 39–48]. However, BPTT requires unfolding the network along the temporal dimension and maintaining membrane potentials, spiking states, and intermediate variables during training, resulting in substantial computational and memory overhead [10, 15, 20, 49, 50]. Meanwhile, SG-based methods still rely on membrane-potential states and their approximate derivatives, making the training rules tightly coupled to specific neuronal dynamics [17, 18, 51]. To mitigate these costs, efficient training methods such as OTTT , SLTT, and RateBP break the reliance on full temporal backpropagation through online updates, attenuating temporal error terms, or rate-based backpropagation, and have demonstrated effectiveness in large-scale tasks [19–22, 50, 52]. However, although these methods effectively reduce the temporal error-propagation costs of BPTT, they still rely on designed surrogate derivatives, access to internal membrane-potential states, assumptions of deterministic and explicitly modeled neuronal dynamics, or rate-coding assumptions. Further decoupling training from these dependencies is critical for efficient SNN training and neuromorphic hardware implementation.

Bio-Inspired Feedback-Based Credit Assignment. In neuromorphic computing research, various bio-inspired alternatives to BP-based training have been explored, among which feedback-based credit assignment methods are particularly relevant, including feedback alignment (FA), direct feedback alignment (DFA), and their variants [23, 25, 53–56]. These methods are primarily designed to alleviate the reliance of BP on exact backward paths and the transposition of forward weights, known as the weight transport problem. FA [53] replaces the exact transposition of forward weights with fixed random feedback weights, while DFA [54] projects the global output error directly to each hidden layer, thereby avoiding layer-wise error backpropagation. Building on this, augmented DFA (aDFA) [23] replaces exact activation-derivative terms in the backward process with arbitrary nonlinear functions, thereby introducing a gradient-free training scheme; uSF [25] improves feedback alignment by maintaining sign consistency between feedback weights and forward weights, enabling effective training in modern deep architectures. These ideas have also been incorporated into SNN training to construct learning rules that are more biologically plausible and hardware-friendly. However, when applied to SNNs, existing bio-inspired feedback-based methods still face important limitations in handling temporal error-propagation, dependence on surrogate derivatives, and scalability to complex architectures and tasks. FA/uSF-style methods primarily improve spatial feedback pathways, but when applied to SNNs, they are often still embedded within through-time training, requiring membrane potentials and spike states to be unfolded along the temporal dimension. Therefore, they retain key limitations of standard BPTT-SG training, including temporal error propagation and reliance on surrogate derivatives or neuronal-dynamics approximations [8, 57–59]. DFA-style methods reduce the burden of layer-wise and temporal error backpropagation by using global output feedback. However, in SNNs, their feedback signals are usually derived from

the final readout or the whole spike train, thereby compressing time-step-wise credit into a single global output-level signal rather than providing separate feedback at each time step. Moreover, they typically still rely on surrogate derivatives or neuronal-dynamics approximations for weight updates [60, 61], while such global random feedback mapping can also lead to unstable alignment in deep architectures, limiting their scalability to large-scale networks and complex tasks [62, 63]. aDFA reduces the dependence on surrogate derivatives through relaxed nonlinear feedback, taking a step toward derivative-free learning. However, it still follows DFA-style update rule and therefore inherits the limitations of DFA-style SNN training, including the lack of time-local feedback signals and limited scalability to deep architectures and complex large-scale tasks [24]. In contrast, our method constructs time-local update signals at each time step, avoiding error backpropagation through time and membrane-potential derivative approximations. It can also be combined with uSF to form a unified bio-inspired training paradigm for large-scale networks and complex tasks.

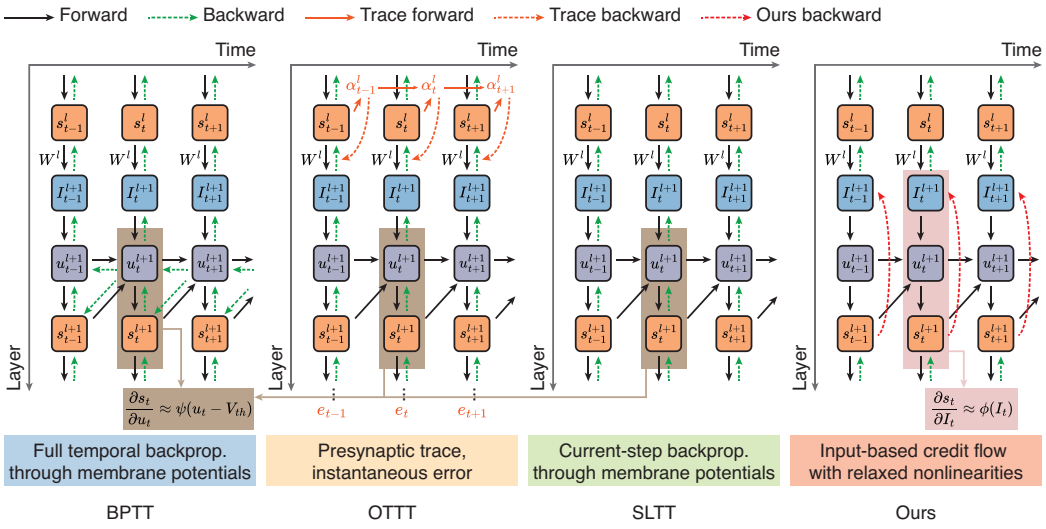


Figure 1: Illustration of unfolded computational graphs for BPTT, OTTT, SLTT, and the proposed method, together with a comparison between SG-based and IDDF update signals.

3 Preliminaries

3.1 Spiking Neural Networks

SNNs process and transmit information through discrete spikes. The leaky integrate-and-fire (LIF) model is one of the most commonly used neuron models in SNNs due to its simplicity and computational efficiency [64]. An LIF neuron accumulates input current into a membrane potential, leaks over time, and emits a spike once the membrane potential exceeds a firing threshold. In practice, we typically use its discrete-time formulation. For the l-th layer at time step t, the LIF dynamics are updated as:

$$\boldsymbol{u}_t^l = \beta \boldsymbol{u}_{t-1}^l + \boldsymbol{I}_t^l - V_{\text{th}} \boldsymbol{r}_t^l, \quad \boldsymbol{I}_t^l = \boldsymbol{W}^{l-1} \boldsymbol{s}_t^{l-1} + \boldsymbol{b}^l. \tag{1}$$

Here, $\boldsymbol{u}_t^l$ is the membrane potential, $\boldsymbol{I}_t^l$ the input drive, β the decay factor, and V_{th} the firing threshold. $\boldsymbol{s}_t^{l-1}$ denotes the spike output from the previous layer, and $\boldsymbol{r}_t^l = H(\boldsymbol{u}_{t-1}^l - V_{\text{th}})$ is the reset signal. $\boldsymbol{W}^{l-1}$ and $\boldsymbol{b}^l$ are the weights and bias. The spike output is generated by $\boldsymbol{s}_t^l = H(\boldsymbol{u}_t^l - V_{\text{th}})$, where $H(\cdot)$ is the Heaviside step function. Due to its non-differentiability, direct training methods typically use surrogate derivatives to construct pseudo gradients of the spike function for backpropagation. Throughout our experiments, we set $\beta = 0.5, V_{\text{th}} = 1.0$.

3.2 Training SNNs with BPTT and SG

Considering a deep SNN, the network output at time step t is given by $\boldsymbol{o}_t = \boldsymbol{W}^o \boldsymbol{s}_t^L$, where $\boldsymbol{W}^o$ denotes the readout weights. The final prediction is obtained by averaging the outputs over all time

steps as $\bar{\boldsymbol{o}} = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{o}_t$. Accordingly, the loss function $\mathcal{L}$ is defined as $\mathcal{L} = \ell(\bar{\boldsymbol{o}}, \boldsymbol{y})$, where $\boldsymbol{y}$ denotes the ground-truth label and $\ell(\cdot)$ can be instantiated as the cross-entropy loss, a choice widely adopted in prior SNN studies [10, 19, 21, 48]. BPTT unfolds the iterative LIF dynamics in Eq. (1) over T time steps and backpropagates errors through the resulting spatio-temporal computational graph, as illustrated in Fig. 1. For the connection weights $\boldsymbol{W}^l$ between layer l and layer $l+1$, the gradient is calculated by accumulating contributions over all time steps:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^l} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial \boldsymbol{s}_t^{l+1}} \frac{\partial \boldsymbol{s}_t^{l+1}}{\partial \boldsymbol{u}_t^{l+1}} \left(\frac{\partial \boldsymbol{u}_t^{l+1}}{\partial \boldsymbol{W}^l} + \sum_{\tau<t} \prod_{i=t-1}^{\tau} \left(\frac{\partial \boldsymbol{u}_{i+1}^{l+1}}{\partial \boldsymbol{u}_i^{l+1}} + \frac{\partial \boldsymbol{u}_{i+1}^{l+1}}{\partial \boldsymbol{s}_i^{l+1}} \frac{\partial \boldsymbol{s}_i^{l+1}}{\partial \boldsymbol{u}_i^{l+1}} \right) \frac{\partial \boldsymbol{u}_\tau^{l+1}}{\partial \boldsymbol{W}^l} \right). \tag{2}$$

where the inner summation and product terms capture the propagation of errors across previous time steps, requiring maintaining the temporal computational graph. Since the spike function $H(\cdot)$ is non-differentiable, BPTT-based direct training is typically combined with SG by replacing $\partial \boldsymbol{s}_t^{l+1}/\partial \boldsymbol{u}_t^{l+1}$ with a well-behaved surrogate derivative $\psi(\cdot)$, such as the arctangent function [44] as $\partial \boldsymbol{s}_t^{l+1}/\partial \boldsymbol{u}_t^{l+1} = H'(\boldsymbol{u}_t^{l+1} - V_{\text{th}}) \approx \psi(\boldsymbol{u}_t^{l+1} - V_{\text{th}}), \quad \psi(x) = (\alpha/2)[1 + (\pi\alpha x/2)^2]^{-1}$. where α controls the slope of the surrogate derivative.

3.3 Adversarial Attack

To assess the robustness of the proposed training method, we further evaluate its performance under adversarial attacks. Since SG-based training can be particularly sensitive to minor disturbances due to its strong dependency on specific patterns or features [65–67], adversarial attacks provide a useful setting for examining the robustness of different training methods. Here, we introduce the basic formulation of adversarial attacks, which construct small, carefully designed input perturbations to induce incorrect model predictions [68]. Given a clean input $\boldsymbol{x}$ with label $\boldsymbol{y}$ and a trained model $f(\cdot)$, an adversarial example is defined as $\boldsymbol{x}_{\text{adv}} = \boldsymbol{x} + \boldsymbol{\delta}$, where $\boldsymbol{\delta}$ denotes the adversarial perturbation. The attack seeks the worst-case perturbation by maximizing $\mathcal{L}(f(\boldsymbol{x} + \boldsymbol{\delta}), \boldsymbol{y})$ under the constraint $\|\boldsymbol{\delta}\|_p \leq \epsilon$, where ϵ controls the perturbation magnitude and $\mathcal{L}(\cdot)$ is the classification loss. Gradient-based attacks approximate this objective by perturbing the input along the direction that increases the loss. The fast gradient sign method (FGSM) generates adversarial inputs with a single-step perturbation [69] as $\boldsymbol{x}_{\text{adv}} = \boldsymbol{x} + \epsilon \cdot \text{sign}\left(\nabla_{\boldsymbol{x}} \mathcal{L}(f(\boldsymbol{x}), \boldsymbol{y})\right)$. Projected gradient descent (PGD) extends FGSM by iteratively applying gradient-ascent updates to maximize the loss, followed by projection onto the allowed perturbation set [70] as $\boldsymbol{x}_{\text{adv}}^{k+1} = \text{Clip}_{\boldsymbol{x},\epsilon}\left(\boldsymbol{x}_{\text{adv}}^{k} + \gamma \cdot \text{sign}\left(\nabla_{\boldsymbol{x}} \mathcal{L}(f(\boldsymbol{x}_{\text{adv}}^{k}), \boldsymbol{y})\right)\right)$, where k denotes the attack iteration, $\boldsymbol{x}_{\text{adv}}^{k}$ is the adversarial example at iteration k, γ is the step size, and $\text{Clip}_{\boldsymbol{x},\epsilon}(\cdot)$ ensures that the perturbation remains within the ϵ-ball centered at the clean input $\boldsymbol{x}$.

4 Input-Driven Derivative-Free Training for SNNs

IDDFT treats spiking at each time step as a local response to the current inputs. Accordingly, the backward error signal is redirected from the membrane potential used for thresholding to these inputs, producing a time-local update signal without temporal error propagation and decoupling the update rule from specific neuronal dynamics.

4.1 Derivation of Input-Driven Derivative-Free Training

Single-step spike response reconstruction. We first revisit the spike generation process at a single time step. According to Eq. (1), the membrane potential of the l-th layer at time step t can be decomposed into the current input drive and a state-dependent term as $\boldsymbol{u}_t^l = \boldsymbol{I}_t^l + \boldsymbol{c}_t^l$, where $\boldsymbol{c}_t^l = \beta \boldsymbol{u}_{t-1}^l - V_{\text{th}} \boldsymbol{r}_t^l$ collects the contribution from the previous membrane potential and the reset signal. Therefore, the spiking function can be written as $\boldsymbol{s}_t^l = H(\boldsymbol{u}_t^l - V_{\text{th}}) = H(\boldsymbol{I}_t^l + \boldsymbol{c}_t^l - V_{\text{th}})$. This formulation views spike generation as a nonlinear response to the current input drive $\boldsymbol{I}_t^l$ conditioned on the current neuronal state $\boldsymbol{c}_t^l$. This perspective preserves the temporal dynamics of LIF neurons in the forward pass, while providing a different entry point for the backward pass: time-local update signals can be constructed from the current input drive $\boldsymbol{I}_t^l$.

Decoupling temporal error propagation. Although the spike generation can be viewed as input-driven at each time step, the state term $\boldsymbol{c}_t^l$ still carries information about past membrane potentials.

Therefore, in exact BPTT, the derivative of the spike response with respect to the trainable weights contains both a direct contribution from the current input drive and a temporal contribution through the historical state as $\partial \boldsymbol{s}_t^l/\partial \boldsymbol{W}^{l-1} = \partial \boldsymbol{s}_t^l/\partial \boldsymbol{I}_t^l \cdot \partial \boldsymbol{I}_t^l/\partial \boldsymbol{W}^{l-1} + \partial \boldsymbol{s}_t^l/\partial \boldsymbol{c}_t^l \cdot \partial \boldsymbol{c}_t^l/\partial \boldsymbol{W}^{l-1}$, where the first term corresponds to the input-driven contribution at the current time step, while the second term represents the temporal contribution induced by historical membrane states, which is associated with the inner summation and product terms in BPTT as shown in Eq. 2. Inspired by the efficient SNN training method SLTT, which suggests that the temporal-domain backpropagation term often contributes weakly to the final update [21], IDDFT treats $\boldsymbol{c}_t^l$ as a state condition for forward spike generation, rather than as a backward error-propagation path. Thus, IDDFT keeps the input-driven contribution and decouples the temporal term as

$$\frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{W}^{l-1}} \approx \frac{\partial \boldsymbol{s}_t^l}{\partial \boldsymbol{I}_t^l} \frac{\partial \boldsymbol{I}_t^l}{\partial \boldsymbol{W}^{l-1}}, \tag{3}$$

where the error signal is no longer propagated through previous membrane-potential states, but is instead applied directly to the current input drive $\boldsymbol{I}_t^l$. This approximation preserves the forward temporal dynamics of LIF neurons, while treating $\boldsymbol{c}_t^l$ as a fixed state condition during the backward pass. Thus, IDDFT does not backpropagate errors through the temporal path.

Input-driven derivative-free update signal. Based on Eq. (3), rewriting the error signal through the membrane-potential entry $\boldsymbol{u}_t^l$ gives the standard SG form for LIF neurons $\partial \boldsymbol{s}_t^l/\partial \boldsymbol{W}^{l-1} \approx \psi(\boldsymbol{u}_t^l - V_{\mathrm{th}})(\boldsymbol{s}_t^{l-1})^\top$. The corresponding SG update is $\Delta \boldsymbol{W}_{\mathrm{SG}}^{l-1} = -\eta \sum_{t=1}^{T} \left[\boldsymbol{e}_t^l \odot \psi(\boldsymbol{u}_t^l - V_{\mathrm{th}})\right](\boldsymbol{s}_t^{l-1})^\top$, where η is the learning rate, and $\boldsymbol{e}_t^l$ denotes the error signal propagated to layer l from the upper layer at time step t. IDDFT instead keeps the input entry in Eq. (3) and does not access or exploit the membrane potential in the backward path. Consequently, the time-local error signal is no longer obtained by approximating the spike-function derivative with a membrane-potential-based surrogate function, but is constructed directly from the current input drive through a relaxed signal $\phi(\boldsymbol{I}_t^l)$. The resulting IDDFT update is

$$\Delta \boldsymbol{W}_{\mathrm{IDDFT}}^{l-1} = -\eta \sum_{t=1}^{T} \left[\boldsymbol{e}_t^l \odot \phi(\boldsymbol{I}_t^l)\right](\boldsymbol{s}_t^{l-1})^\top. \tag{4}$$

We next clarify the relaxed requirement imposed on $\phi(\cdot)$. If the angle between two update directions is smaller than 90°, their inner product is positive, indicating that they are positively aligned. Therefore, when the reference direction is descent-compatible, a positively aligned direction can still provide a useful update signal [53, 54]. After the temporal path is decoupled, the SG-based simplified BPTT update, which retains only the spatial term as in SLTT [21], and the proposed IDDFT update share the same outer-product structure. The only difference lies in the local nonlinear factor: $\psi(\boldsymbol{u}_t^l - V_{\mathrm{th}})$ for the SG-based simplified BPTT update and $\phi(\boldsymbol{I}_t^l)$ for IDDFT. Therefore, the directional alignment between these two updates can be characterized by the compatibility between their local nonlinear factors. Specifically, a sufficient condition for positive directional alignment is that $\phi(\boldsymbol{I}_t^l)$ is positively correlated with $\psi(\boldsymbol{u}_t^l - V_{\mathrm{th}})$ as $\mathrm{Corr}\left(\phi(\boldsymbol{I}_t^l), \psi(\boldsymbol{u}_t^l - V_{\mathrm{th}})\right) > 0$. When this condition holds, IDDFT provides an effective update direction by preserving positive alignment with the SG-based simplified BPTT update. Therefore, $\phi(\cdot)$ does not need to provide a pointwise approximation to the membrane-potential-based surrogate derivative, nor is it restricted to a specific functional form. It only needs to be positively correlated with the surrogate derivative in a broad statistical sense, making the choice of effective nonlinear functions in IDDFT highly flexible. A formal proof that positive alignment supports effective learning is provided in Appendix A.

4.2 Combination with Bio-Inspired Training

To further improve the compatibility of our training method with neuromorphic learning principles, we integrate IDDFT with the uSF-style bio-inspired training method [25]. In contrast to the BP-style training, which propagates error signals using the exact transposed weight matrix $\boldsymbol{W}^{l,\top}$ during the backward process, uSF propagates the upper-layer error signal using only the sign information of the forward weights $\mathrm{sign}(\boldsymbol{W}^{l,\top})$. Specifically, at time step t, the error signal propagated to layer l from the upper layer is given by

$$\boldsymbol{e}_t^l = \mathrm{sign}(\boldsymbol{W}^{l,\top})\boldsymbol{\delta}_t^{l+1}, \qquad \boldsymbol{\delta}_t^{l+1} = \boldsymbol{e}_t^{l+1} \odot \phi(\boldsymbol{I}_t^{l+1}). \tag{5}$$

The above combined formulation highlights the complementary roles of IDDFT and uSF, yielding a training rule that is time-local, derivative-free, and compatible with bio-inspired training principles, forming a unified training paradigm for deep SNNs and complex tasks.

5 Experiments

We conduct experiments on CIFAR-10 [71], CIFAR-100 [71], CIFAR10-DVS [72], and Tiny-ImageNet [73] to evaluate the proposed training method. The SNN models are implemented using PyTorch [74] and snnTorch [75], while the uSF training framework is implemented based on BioTorch [63]. We set $\beta = 0.5, V_{\text{th}} = 1.0$ for all SNN models, and adopt the arctangent surrogate function for SG-based baselines [44]. Detailed experimental settings are provided in Appendix F.

5.1 Effectiveness of IDDFT

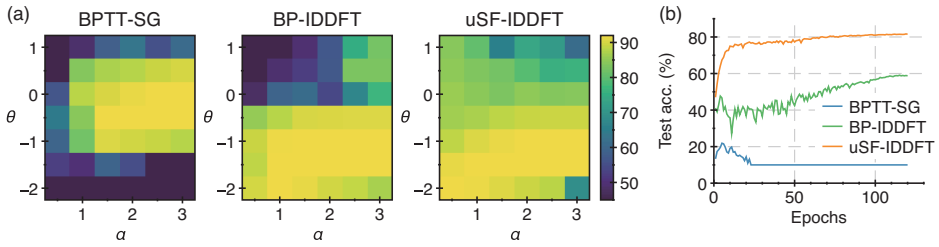


Figure 2: Performance of BPTT-SG, BP-IDDFT, and uSF-IDDFT under relaxed nonlinear functions. (a) Accuracy under different Gaussian settings. (b) Training curves under a cosine nonlinearity.

This section first verifies the effectiveness of the proposed IDDFT training mechanism. To investigate whether our method can support effective training without relying on membrane-potential-based surrogate derivatives and with relaxed nonlinear functions, we compare the performance of BPTT with SG (BPTT-SG), IDDFT with BP-style training (BP-IDDFT), and IDDFT with uSF-style training (uSF-IDDFT) under various Gaussian function settings. Experiments are conducted on CIFAR-10 using a VGG9-based SNN architecture. The detailed settings are summarized in Appendix F. The Gaussian function used in these settings is defined as $G(x;\alpha,\theta) = \exp[-\alpha(x+\theta)^2]$, where α controls the sharpness of the function and θ denotes the horizontal shift. Specifically, for BPTT-SG, this function is applied to the membrane-potential threshold variable as $\psi(\boldsymbol{u}_t^l - V_{\text{th}}) = G(\boldsymbol{u}_t^l - V_{\text{th}};\alpha,\theta)$. For BP-IDDFT and uSF-IDDFT, the same function is applied to the current input drive as $\phi(\boldsymbol{I}_t^l) = G(\boldsymbol{I}_t^l;\alpha,\theta)$. We scan α from 0.5 to 3.0 with an interval of 0.5, and θ from -2.0 to 1.0 with an interval of 0.5. As shown in Fig. 2a, BPTT-SG is effective only within a limited range of settings, where the surrogate function is sufficiently localized around the firing threshold. Otherwise, the surrogate gradient is assigned to less relevant membrane-potential states, leading to ineffective training. In contrast, our IDDFT shows a much broader effective region. BP-IDDFT remains trainable for most tested Gaussian functions, and uSF-IDDFT further expands the effective region, working for almost all tested settings. Moreover, compared with the baseline BPTT-SG using the arctangent surrogate, which achieves an accuracy of 92.38%, BP-IDDFT and uSF-IDDFT reach optimal accuracies of 93.02% and 92.55% under the tested Gaussian settings, respectively. We further examine an extreme relaxed setting using a cosine function. As shown in Fig. 2b, BPTT-SG completely fails when cosine is used as the surrogate function. By contrast, BP-IDDFT remains partially trainable and reaches around 60% test accuracy, while uSF-IDDFT successfully learns and achieves over 80% accuracy. Overall, these results verify the effectiveness of the proposed IDDFT mechanism: IDDFT supports effective training while maintaining strong performance, and its combination with uSF substantially relaxes the requirements on the choice of nonlinear functions.

5.2 Comparison with the State-of-the-Art

As shown in Table 1, we compare the proposed combined uSF-IDDFT training paradigm with advanced direct-training methods for SNNs on CIFAR-10, CIFAR-100, CIFAR10-DVS, and Tiny-ImageNet. The results show that uSF-IDDFT achieves competitive performance across these bench-

marks while relying on a substantially simplified training rule that avoids temporal error propagation, employs relaxed input-driven nonlinear signals instead of membrane-potential surrogate derivatives, and propagates spatial errors through sign-concordant pathways. In these experiments, the relaxed nonlinear signal is instantiated by the Gaussian function defined above, with the specific parameter settings summarized in Appendix F. On CIFAR-10, uSF-IDDFT achieves 96.71% accuracy with only 4 time steps, outperforming the compared direct-training methods listed in the table. On CIFAR10-DVS, uSF-IDDFT reaches 83.88% accuracy, showing strong performance on event-based inputs and outperforming several methods based on temporal losses with SG, as well as the adaptive-gradient-learning method without TET loss. On CIFAR-100, uSF-IDDFT achieves 78.88% accuracy, remaining competitive with advanced direct-training methods. On Tiny-ImageNet, uSF-IDDFT obtains 62.70% accuracy with 4 time steps, approaching the performance of CLIF while outperforming the other listed SG-based methods. It is worth noting that these results are obtained using standard LIF neurons, cross-entropy loss, and the proposed simplified training mechanism, without specially designed neuron models, temporal losses, or adaptive surrogate-gradient learning mechanisms. This suggests that our uSF-IDDFT can serve as a practical and competitive training paradigm for deep SNNs under neuromorphic-compatible learning settings.

Table 1: Performance comparison with advanced direct-training methods for SNNs on CIFAR-10, CIFAR-100, CIFAR10-DVS, and Tiny-ImageNet.

Dataset	Methods	Training	Architecture	Time steps	Accuracy (%)
CIFAR-10	OS [22]	online+SG	ResNet-19	4	95.20†
	RateBP [19]	one-step+SG	ResNet-18	4	95.61
	MPD-AGL [76]	SG+AGL	ResNet-19	4 / 6	96.35 / 96.54
	PseudoSN [77]	Noise-based proxy	ResNet-19	3.46 $\pm$ 0.05*	96.62
	Ours	IDDFT+uSF	ResNet-19	4	96.71 $\pm$ 0.09
CIFAR-100	OS [22]	online+SG	ResNet-19	4	77.86†
	RateBP [19]	one-step+SG	ResNet-18	4	78.26
	MPD-AGL [76]	SG+AGL	ResNet-19	4	79.72
	PseudoSN [77]	Noise-based proxy	ResNet-19	3.01 $\pm$ 0.03*	81.29
	Ours	IDDFT+uSF	ResNet-19	4	78.88 $\pm$ 0.13
CIFAR10-DVS	TET [11]	temporal loss+SG	VGGSNN	10	83.17†
	SLTT [21]	SLTT+SG	VGG11	10	82.20†
	OS [22]	online+SG	VGGSNN	10	82.40†
	RateBP [19]	one-step+SG	VGG-11	10	76.96
	MPD-AGL [76]	SG+AGL	VGGSNN	10	82.50 / 84.10†
	IM-LIF [78]	SG	VGGSNN	10	80.50
	PseudoSN [77]	Noise-based proxy	VGG-11	4.50 $\pm$ 0.14*	79.82
	Ours	IDDFT+uSF	VGGSNN	10	83.88 $\pm$ 0.60
Tiny-ImageNet	MPD-AGL [76]	SG+AGL	VGG-13	4	58.14
	IM-LIF [78]	SG	ResNet-19	3 / 6	54.82 / 55.37
	CLIF 3 [79]	SG	VGG-13	4	63.16†
	Ours	IDDFT+uSF	ResNet-19	4	62.70 $\pm$ 0.39

†: Results reported with TET loss. *: Reported time steps are learned average inference latencies (not fixed integer simulation steps). AGL: adaptive gradient learning.

5.3 Robustness under Adversarial Attacks

We further evaluate the robustness of the uSF-IDDFT method under gradient-based adversarial attacks [69, 70, 80] and compare it with the BPTT-SG baseline. To clarify the robustness gain comes from the IDDFT mechanism rather than uSF alone, we also include uSF-SG as an ablation control. Since different training methods follow different gradient-computation rules, the generated adversarial perturbations also differ. Therefore, relying only on self-attacks is insufficient, as relaxed gradients may generate weaker perturbations and thus overestimate robustness. To address this issue, we adopt a transfer-attack setting, where BPTT-SG, uSF-SG, and uSF-IDDFT are each used as attack sources, and the resulting perturbations are applied to target models trained by the three methods. All target models are the same as those used in Section 5.2. The detailed attack settings are summarized in Appendix F. The results are shown in Table 2. Under the clean setting, our uSF-IDDFT achieves performance comparable to or better than BPTT-SG, and consistently outperforms uSF-SG, indicating that replacing SG with IDDFT under the same uSF framework can

Table 2: Accuracy under adversarial attacks on CIFAR-10, CIFAR-100, and CIFAR10-DVS. Columns report the remaining accuracy after attack of each target model trained with the indicated method, and rows indicate the source training method used to generate gradient-based adversarial perturbations. BPTT, uSF, and IDDFT denote BPTT-SG, uSF-SG, and uSF-IDDFT, respectively. The Clean row reports the standard accuracy without adversarial attack. The Avg. row reports the mean accuracy of each target model across different attack sources. Gray, green, and blue cells indicate self-attacks, clean accuracies, and summary rows, respectively; orange cells with bold numbers indicate the best-performing target model under each task and attack source.

		CIFAR-10			CIFAR-100			CIFAR10-DVS		
Attack	Src./Tgt.	BPTT	uSF	IDDFT	BPTT	uSF	IDDFT	BPTT	uSF	IDDFT
Clean	Acc.	96.75	96.03	96.71	80.44	78.09	78.88	82.18	83.42	83.88
FGSM	BPTT	62.60	68.79	**70.96**	18.40	28.10	**29.46**	15.46	32.00	**56.22**
	uSF	78.10	74.41	**79.60**	38.03	25.38	**39.28**	36.00	30.54	**59.34**
	IDDFT	74.41	**75.36**	69.99	41.63	**41.91**	32.63	38.92	41.10	**54.06**
	Avg.	71.70	72.85	**73.52**	32.69	31.79	**33.79**	30.13	34.55	**56.54**
PGD-7	BPTT	4.80	43.53	**47.67**	0.01	26.47	**29.99**	0.14	24.84	**57.90**
	uSF	56.34	44.41	**63.30**	16.72	2.81	**23.66**	22.18	2.46	**58.56**
	IDDFT	40.91	**48.29**	25.82	22.32	**26.02**	7.34	35.58	**35.96**	28.48
	Avg.	34.01	45.41	**45.60**	13.02	18.43	**20.33**	19.30	21.09	**48.31**
PGD-10	BPTT	2.63	39.52	**44.16**	0.00	25.67	**29.48**	0.14	25.76	**57.64**
	uSF	52.26	38.27	**60.07**	13.98	1.72	**21.54**	22.28	1.98	**58.82**
	IDDFT	35.06	**43.71**	19.67	18.80	**22.59**	5.02	35.30	**36.04**	27.08
	Avg.	29.98	40.50	**41.30**	10.93	16.66	**18.68**	19.24	21.26	**47.85**
PGD-20	BPTT	0.84	36.82	**42.21**	0.00	25.68	**29.58**	0.04	26.76	**58.14**
	uSF	44.27	27.63	**55.12**	11.82	0.75	**19.94**	22.50	1.38	**58.82**
	IDDFT	27.78	**38.31**	11.49	14.24	**18.44**	2.68	35.02	**35.78**	25.34
	Avg.	24.30	34.25	**36.27**	8.69	14.96	**17.40**	19.19	21.31	**47.43**
PGD-40	BPTT	0.36	37.61	**42.87**	0.00	26.75	**30.74**	0.02	27.32	**58.36**
	uSF	38.41	19.79	**51.74**	11.21	0.39	**19.55**	22.40	1.22	**59.16**
	IDDFT	23.52	**35.85**	7.23	12.21	**16.59**	1.57	34.82	**36.18**	24.34
	Avg.	20.76	31.08	**33.95**	7.80	14.58	**17.29**	19.08	21.57	**47.29**

further improve standard classification performance. Under attacks, the diagonal elements correspond to self-attacks, where the source and target models are trained by the same method. The off-diagonal elements evaluate how adversarial perturbations generated from one training method affect models trained by another method. For perturbations generated by BPTT-SG or uSF-SG, uSF-IDDFT consistently achieves the highest remaining accuracy, showing that its robustness is not merely due to weak self-attacks but also generalizes to transfer attacks from other SG-based methods. The Avg. row further summarizes the overall robustness across all attack sources. For all datasets and attacks, uSF-IDDFT-trained model achieves the highest average remaining accuracy, followed by uSF-SG- and BPTT-SG-trained models. This indicates that IDDFT is the main source of the robustness improvement, rather than the effect of uSF alone. The advantage becomes more pronounced under stronger PGD attacks. For example, under PGD-40 on CIFAR10-DVS, the uSF-IDDFT-trained model maintains an average accuracy of 47.29%, compared with 21.57% and 19.08% for the uSF-SG- and BPTT-SG-trained models, respectively. One possible explanation is that small perturbations in membrane potentials can substantially change spike patterns [67], making conventional SG-based training sensitive to local membrane-potential variations. In contrast, uSF-IDDFT constructs input-driven derivative-free update signals, alleviating the dependence of error-signal computation on membrane potentials and thereby improving robustness. Additional results and a robustness evaluation of BP-IDDFT are provided in Appendices D.2 and D.3.

6 Conclusion

We propose input-driven derivative-free training (IDDFT), which constructs update signals from input-driven nonlinear responses rather than membrane-potential-based surrogate derivatives. By removing the dependence on membrane-potential states during training, IDDFT decouples error

computation from temporal state propagation, reducing reliance on specific neuronal dynamics and substantially lowering computational and memory costs during training. Theoretical analysis and experiments show that IDDFT supports effective training with relaxed nonlinear functions. Combined with uSF, it forms a neuromorphic-friendly training paradigm with greater implementation flexibility. Benchmark results demonstrate competitive performance with BPTT and state-of-the-art SNN training methods, together with significantly improved robustness under adversarial attacks.

7 Discussion and Limitations

Neuromorphic implementation potential. Although this study evaluates IDDFT in simulation, the following properties highlight its potential for neuromorphic implementation:

- IDDFT does not require access to internal membrane potentials during training. Therefore, it has a natural mapping advantage for neuromorphic systems where membrane potentials or internal neuronal states are difficult or costly to access, or only synaptic input interfaces are available.
- IDDFT avoids retaining the BPTT computation graph and historical neuronal states across timesteps, reducing the training-memory complexity with respect to the number of timesteps from $O(T)$ to $O(1)$, thus substantially decreasing the capacity required for temporal-state buffers and the associated data-access demands. Since memory capacity and bandwidth are often limited on neuromorphic hardware, this property is particularly beneficial for hardware implementation, and its advantage further increases with the number of timesteps.
- When IDDFT is combined with uSF, if the hardware explicitly stores a feedback matrix with P elements, its FP32 representation requires $32P$ bits, whereas uSF requires only P sign bits and a small number of layer-wise scaling factors, reducing the amount of backward-weight data to store or transmit by up to a factor of 32. Meanwhile, element-wise full-precision multiplications can be replaced by sign-controlled additions or subtractions and layer-wise scaling, thereby reducing the hardware overhead of backward computation and data transmission.

When mapping IDDFT onto existing neuromorphic devices, whether hardware modifications are required depends on whether the platform supports programmable learning.

- For platforms supporting programmable learning (e.g., Loihi 2 [81]), the input drive and presynaptic spikes can be maintained as local quantities, while the error signal can be transmitted as a modulatory signal through the feedback pathway. Therefore, IDDFT can in principle be mapped by configuring the learning rule, signal routing, and execution schedule, without modifying the neuronal cores or physical architecture of the hardware.
- For platforms primarily designed for deployment and inference without programmable learning, IDDFT training may require off-chip assistance or additional learning modules. However, such extensions do not need to support BPTT-style temporal backpropagation and therefore require a much simpler training datapath than BPTT-SG.

Beyond digital neuromorphic chips, IDDFT could also be realized on photonic and memristive platforms, where neuronal dynamics are implemented through physical processes and internal states can be difficult to read out or digitize accurately. The signs of the uSF backward weights could be encoded through signal polarity, with a shared gain per layer. This can reduce high-precision programming requirements and the associated signal-conversion, control, and wiring overhead. Implementing IDDFT on both digital and physical neuromorphic platforms remains future work.

Limitations. Appendix C provides practical guidance for choosing the relaxed nonlinearity, but systematic optimization for specific tasks or architectures has not been explored. Our theoretical analysis explains why an input-driven nonlinear factor can yield effective updates in place of the membrane-potential-based surrogate derivative under specific conditions, but does not establish convergence guarantees for arbitrary networks, datasets, or nonlinear factors, leaving stronger theoretical guarantees for future work. The neuromorphic mappings discussed above have not yet been tested on hardware. Finally, although our experiments cover both static images and event-based data, they are limited to visual classification. Evaluating IDDFT on tasks with stronger temporal dependencies, such as speech recognition and sequence prediction, and testing alternative input encoding schemes would help clarify its applicability beyond the settings studied here.

Acknowledgments

This work was supported by a project (Grant Number JPNP14004) commissioned by the New Energy and Industrial Technology Development Organization (NEDO), by JST Moonshot R&D Program Japan Grant Number JPMJMS263O, and by JSPS KAKENHI Grant Number 25H01134. The authors declare no competing interests.

References

[1] Wolfgang Maass. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9):1659–1671, 1997.

[2] Samanwoy Ghosh-Dastidar and Hojjat Adeli. Spiking neural networks. *International journal of neural systems*, 19(04):295–308, 2009.

[3] Kashu Yamazaki, Viet-Khoa Vo-Ho, Darshan Bulsara, and Ngan Le. Spiking neural networks and their applications: A review. *Brain sciences*, 12(7):863, 2022.

[4] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. Towards spike-based machine intelligence with neuromorphic computing. *Nature*, 575(7784):607–617, 2019.

[5] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *Ieee micro*, 38(1):82–99, 2018.

[6] Jing Pei, Lei Deng, Sen Song, Mingguo Zhao, Youhui Zhang, Shuang Wu, Guanrui Wang, Zhe Zou, Zhenzhi Wu, Wei He, et al. Towards artificial general intelligence with hybrid tianjic chip architecture. *Nature*, 572(7767):106–111, 2019.

[7] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 2002.

[8] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[9] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11062–11070, 2021.

[10] Yuhang Li, Yufei Guo, Shanghang Zhang, Shikuang Deng, Yongqing Hai, and Shi Gu. Differentiable spike: Rethinking gradient-descent for training spiking neural networks. *Advances in neural information processing systems*, 34:23426–23439, 2021.

[11] Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. *arXiv preprint arXiv:2202.11946*, 2022.

[12] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in neuroscience*, 12:331, 2018.

[13] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 1311–1318, 2019.

[14] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. *Advances in neural information processing systems*, 31, 2018.

[15] Wenrui Zhang and Peng Li. Temporal spike sequence learning via backpropagation for deep spiking neural networks. *Advances in neural information processing systems*, 33:12022–12033, 2020.

[16] Jinseok Kim, Kyungsu Kim, and Jae-Joon Kim. Unifying activation-and timing-based learning rules for spiking neural networks. *Advances in neural information processing systems*, 33: 19534–19544, 2020.

[17] Friedemann Zenke and Tim P Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural computation*, 33(4):899–925, 2021.

[18] Julian Göltz, Jimmy Weber, Laura Kriener, Sebastian Billaudelle, Peter Lake, Johannes Schemmel, Melika Payvand, and Mihai A Petrovici. Delgrad: exact event-based gradients for training delays and weights on spiking neuromorphic hardware. *Nature Communications*, 16(1):8245, 2025.

[19] Chengting Yu, Lei Liu, Gaoang Wang, Erping Li, and Aili Wang. Advancing training efficiency of deep spiking neural networks through rate-based backpropagation. *Advances in Neural Information Processing Systems*, 37:115786–115815, 2024.

[20] Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. Online training through time for spiking neural networks. *Advances in neural information processing systems*, 35:20717–20730, 2022.

[21] Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Towards memory-and time-efficient backpropagation for training spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6166–6176, 2023.

[22] Yaoyu Zhu, Jianhao Ding, Tiejun Huang, Xiaodong Xie, and Zhaofei Yu. Online stabilization of spiking neural networks. In *The Twelfth International Conference on Learning Representations*, 2024.

[23] Mitsumasa Nakajima, Katsuma Inoue, Kenji Tanaka, Yasuo Kuniyoshi, Toshikazu Hashimoto, and Kohei Nakajima. Physical deep learning with biologically inspired training method: gradient-free approach for physical hardware. *Nature communications*, 13(1):7847, 2022.

[24] Yongbo Zhang, Katsuma Inoue, Mitsumasa Nakajima, Toshikazu Hashimoto, Yasuo Kuniyoshi, and Kohei Nakajima. Training spiking neural networks via augmented direct feedback alignment. *arXiv preprint arXiv:2409.07776*, 2024.

[25] Qianli Liao, Joel Leibo, and Tomaso Poggio. How important is weight symmetry in backpropagation? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[26] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-yolo: spiking neural network for energy-efficient object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11270–11277, 2020.

[27] Hao Wu, Yueyi Zhang, Wenming Weng, Yongting Zhang, Zhiwei Xiong, Zheng-Jun Zha, Xiaoyan Sun, and Feng Wu. Training spiking neural networks with accumulated spiking flow. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10320–10328, 2021.

[28] Shibo Zhou, Xiaohua Li, Ying Chen, Sanjeev T Chandrasekaran, and Arindam Sanyal. Temporal-coded deep spiking neural network with easy training and robust performance. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11143–11151, 2021.

[29] Jibin Wu, Yansong Chua, Malu Zhang, Guoqi Li, Haizhou Li, and Kay Chen Tan. A tandem learning rule for effective training and rapid inference of deep spiking neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 34(1):446–460, 2021.

[30] Yongqiang Cao, Yang Chen, and Deepak Khosla. Spiking deep convolutional neural networks for energy-efficient object recognition. *International Journal of Computer Vision*, 113(1):54–66, 2015.

[31] Jianhao Ding, Zhaofei Yu, Yonghong Tian, and Tiejun Huang. Optimal ann-snn conversion for fast and accurate inference in deep spiking neural networks. *arXiv preprint arXiv:2105.11654*, 2021.

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

[33] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in neuroscience*, 11:682, 2017.

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

[35] Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. *arXiv preprint arXiv:2303.04347*, 2023.

[36] Zecheng Hao, Jianhao Ding, Tong Bu, Tiejun Huang, and Zhaofei Yu. Bridging the gap between anns and snns by calibrating offset spikes. *arXiv preprint arXiv:2302.10685*, 2023.

[37] Haiyan Jiang, Srinivas Anumasa, Giulia De Masi, Huan Xiong, and Bin Gu. A unified optimization framework of ann-snn conversion: towards optimal mapping from activation values to firing rates. In *International Conference on Machine Learning*, pages 14945–14974. PMLR, 2023.

[38] Zecheng Hao, Tong Bu, Jianhao Ding, Tiejun Huang, and Zhaofei Yu. Reducing ann-snn conversion error through residual membrane potential. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11–21, 2023.

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

[40] Pengjie Gu, Rong Xiao, Gang Pan, and Huajin Tang. Stca: Spatio-temporal credit assignment with delayed feedback in deep spiking neural networks. In *Ijcai*, volume 15, pages 1366–1372, 2019.

[41] Kazuma Suetake, Shin-ichi Ikegawa, Ryuji Saiin, and Yoshihide Sawada. S3nn: Time step reduction of spiking surrogate gradients for training energy efficient single-step spiking neural networks. *Neural Networks*, 159:208–219, 2023.

[42] Ziming Wang, Runhao Jiang, Shuang Lian, Rui Yan, and Huajin Tang. Adaptive smoothing gradient learning for spiking neural networks. In *International conference on machine learning*, pages 35798–35816. PMLR, 2023.

[43] Chaoteng Duan, Jianhao Ding, Shiyan Chen, Zhaofei Yu, and Tiejun Huang. Temporal effective batch normalization in spiking neural networks. *Advances in Neural Information Processing Systems*, 35:34377–34390, 2022.

[44] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothée Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2661–2671, 2021.

[45] Yufei Guo, Xuhui Huang, and Zhe Ma. Direct learning-based deep spiking neural networks: a review. *Frontiers in Neuroscience*, 17:1209795, 2023.

[46] Xingting Yao, Fanrong Li, Zitao Mo, and Jian Cheng. Glif: A unified gated leaky integrate-and-fire neuron for spiking neural networks. *Advances in Neural Information Processing Systems*, 35:32160–32171, 2022.

[47] Zeyu Huang, Wei Meng, Quan Liu, Kun Chen, and Li Ma. Ar-lif: Adaptive reset leaky integrate-and-fire neuron for spiking neural networks. In *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 20322–20326. IEEE, 2026.

[48] Jiaqiang Jiang, Wenfeng Xu, Jing Fan, and Rui Yan. Ds-atgo: Dual-stage synergistic learning via forward adaptive threshold and backward gradient optimization for spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 1855–1863, 2026.

[49] Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Yisen Wang, and Zhouchen Lin. Training feedback spiking neural networks by implicit differentiation on the equilibrium state. *Advances in neural information processing systems*, 34:14516–14528, 2021.

[50] Qingyan Meng, Mingqing Xiao, Shen Yan, Yisen Wang, Zhouchen Lin, and Zhi-Quan Luo. Training high-performance low-latency spiking neural networks by differentiation on spike representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12444–12453, 2022.

[51] Shuang Lian, Jiangrong Shen, Qianhui Liu, Ziming Wang, Rui Yan, and Huajin Tang. Learnable surrogate gradient for direct training spiking neural networks. In Edith Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 3002–3010. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/335. URL https://doi.org/10.24963/ijcai.2023/335. Main Track.

[52] Hesham Mostafa. Supervised learning based on temporal coding in spiking neural networks. *IEEE transactions on neural networks and learning systems*, 29(7):3227–3235, 2017.

[53] Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature communications*, 7(1):13276, 2016.

[54] Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. *Advances in neural information processing systems*, 29, 2016.

[55] Mitsumasa Nakajima, Yongbo Zhang, Katsuma Inoue, Yasuo Kuniyoshi, Toshikazu Hashimoto, and Kohei Nakajima. Reservoir direct feedback alignment: deep learning by physical dynamics. *Communications Physics*, 7(1):411, 2024.

[56] Yongbo Zhang, Mitsumasa Nakajima, Katsuma Inoue, Toshikazu Hashimoto, Yasuo Kuniyoshi, and Kohei Nakajima. Scalable physical deep learning using optical dynamics with state-skipping training and single-map architecture. *Nature Computational Science*, 2026. doi: 10.1038/s43588-026-01075-w.

[57] Friedemann Zenke and Surya Ganguli. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6):1514–1541, 2018.

[58] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. Synaptic plasticity dynamics for deep continuous local learning (decolle). *Frontiers in Neuroscience*, 14:424, 2020.

[59] Jiaqi Lin and Abhronil Sengupta. Toward spiking neural network local learning modules resistant to adversarial attacks. *arXiv e-prints*, pages arXiv–2504, 2025.

[60] Jeongjun Lee, Renqian Zhang, Wenrui Zhang, Yu Liu, and Peng Li. Spike-train level direct feedback alignment: Sidestepping backpropagation for on-chip training of spiking neural nets. *Frontiers in neuroscience*, 14:143, 2020.

[61] Arash Samadi, Timothy P Lillicrap, and Douglas B Tweed. Deep learning with dynamic spiking neurons and fixed feedback weights. *Neural computation*, 29(3):578–602, 2017.

[62] Julien Launay, Iacopo Poli, François Boniface, and Florent Krzakala. Direct feedback alignment scales to modern deep learning tasks and architectures. *Advances in neural information processing systems*, 33:9346–9360, 2020.

[63] Albert JimÃŠnez Sanfiz and Mohamed Akrout. Benchmarking the accuracy and robustness of feedback alignment algorithms. *arXiv preprint arXiv:2108.13446*, 2021.

[64] Anthony N Burkitt. A review of the integrate-and-fire neuron model: I. homogeneous synaptic input. *Biological cybernetics*, 95(1):1–19, 2006.

[65] Jianhao Ding, Tong Bu, Zhaofei Yu, Tiejun Huang, and Jian Liu. Snn-rat: Robustness-enhanced spiking neural network through regularized adversarial training. *Advances in Neural Information Processing Systems*, 35:24780–24793, 2022.

[66] Bhaskar Mukhoty, Hilal AlQuabeh, Giulia De Masi, Huan Xiong, and Bin Gu. Certified adversarial robustness for rate encoded spiking neural networks. In *The Twelfth International Conference on Learning Representations*, 2024.

[67] Shuai Wang, Malu Zhang, Yulin Jiang, Dehao Zhang, Ammar Belatreche, Yu Liang, Yimeng Shan, Zijian Zhou, Yang Yang, and Haizhou Li. Robust spiking neural networks against adversarial attacks. *arXiv preprint arXiv:2602.20548*, 2026.

[68] Pavel Laskov and Richard Lippmann. Machine learning in adversarial environments. *Machine learning*, 81(2):115–119, 2010.

[69] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[70] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[71] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[72] Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:244131, 2017.

[73] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[74] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[75] Jason K Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D Lu. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, 111(9):1016–1054, 2023.

[76] Jiaqiang Jiang, Lei Wang, Runhao Jiang, Jing Fan, and Rui Yan. Adaptive gradient learning for spiking neural networks by exploiting membrane potential dynamics. *arXiv preprint arXiv:2505.11863*, 2025.

[77] Yuxuan Zhang, Yuhang Sun, Hongjue Li, Yue Deng, and Wen Yao. Pseudo-spiking neurons: A noise-based training framework for heterogeneous-latency spiking neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 28555–28563, 2026.

[78] Shuang Lian, Jiangrong Shen, Ziming Wang, and Huajin Tang. Im-lif: Improved neuronal dynamics with attention mechanism for direct training deep spiking neural network. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 8(2):2075–2085, 2024.

[79] Yulong Huang, Xiaopeng Lin, Hongwei Ren, Haotian Fu, Yue Zhou, Zunchang Liu, Biao Pan, and Bojun Cheng. Clif: Complementary leaky integrate-and-fire neuron for spiking neural networks. *arXiv preprint arXiv:2402.04663*, 2024.

[80] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

[81] Intel Labs. Taking neuromorphic computing to the next level with Loihi 2. Technology brief, Intel Corporation, 2021. URL https://download.intel.com/newsroom/2021/new-technologies/neuromorphic-computing-loihi-2-brief.pdf.

[82] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International conference on machine learning*, pages 1568–1577. PMLR, 2018.

[83] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[84] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[85] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 113–123, 2019.

[86] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. arxiv 2017. *arXiv preprint arXiv:1708.04552*, 10, 2017.

[87] Yuhang Li, Youngeun Kim, Hyoungseob Park, Tamar Geller, and Priyadarshini Panda. Neuromorphic data augmentation for training spiking neural networks. In *European Conference on Computer Vision*, pages 631–649. Springer, 2022.

[88] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pages 7388–7397. IEEE, 2017.

Appendix Contents

A Theoretical Analysis **18**

A.1 Relation between IDDFT and BPTT 18

A.2 Bound on the Temporal Residual 19

A.3 Positive Alignment Supports Effective Learning 20

B Statistical Alignment between IDDFT and Surrogate Gradient **22**

C Practical Selection of the Relaxed Nonlinearity **23**

D More Results **24**

D.1 Training Memory Cost and Runtime 24

D.2 Robustness under the RFGSM Adversarial Attack 25

D.3 Further Results on Robustness under Adversarial Attacks 25

D.4 Sensitivity to Neuronal Parameters 26

D.5 Evaluation under Different Neuronal Dynamics 27

D.6 Evaluation on Tasks with Richer Temporal Structure 27

E Pseudo code of the IDDFT **27**

F Experiments Details **28**

F.1 Environment and Parameter Settings 28

F.2 Datasets and Preprocessing 28

F.3 Adversarial Attack Settings 29

F.4 Network Architectures 29

A Theoretical Analysis

A.1 Relation between IDDFT and BPTT

Proposition 1 (Decomposition of BPTT-SG with respect to IDDFT)**.** *Starting from the BPTT-SG gradient in Eq. (2), the full BPTT-SG direction for $\boldsymbol{W}^l$ can be decomposed as*

$$\boldsymbol{G}^l_{\mathrm{BPTT}} = \boldsymbol{G}^l_{\mathrm{IDDFT}} + \boldsymbol{R}^l_{\phi} + \boldsymbol{R}^l_{\mathrm{temp}}, \tag{6}$$

where $\boldsymbol{G}^l_{\mathrm{IDDFT}}$ denotes the IDDFT update direction, $\boldsymbol{R}^l_{\mathrm{temp}}$ denotes the temporal residual induced by temporal error propagation, and $\boldsymbol{R}^l_{\phi}$ denotes the residual induced by replacing the membrane-potential-based surrogate term with the input-driven relaxed term.

Proof. Starting from Eq. (2), define the one-step temporal term as

$$\boldsymbol{J}^{l+1}_i = \frac{\partial \boldsymbol{u}^{l+1}_{i+1}}{\partial \boldsymbol{u}^{l+1}_i} + \frac{\partial \boldsymbol{u}^{l+1}_{i+1}}{\partial \boldsymbol{s}^{l+1}_i} \frac{\partial \boldsymbol{s}^{l+1}_i}{\partial \boldsymbol{u}^{l+1}_i}, \tag{7}$$

where the second term accounts for the dependence of the next membrane state on the previous spike output. The accumulated temporal term is then written as

$$\boldsymbol{J}^{l+1}_{t,\tau} = \prod_{i=t-1}^{\tau} \boldsymbol{J}^{l+1}_i. \tag{8}$$

Using the membran-potential-based SG approximation

$$\frac{\partial \boldsymbol{s}^{l+1}_t}{\partial \boldsymbol{u}^{l+1}_t} \approx \psi(\boldsymbol{u}^{l+1}_t - V_{\mathrm{th}}), \tag{9}$$

we define the SG-induced time-local error signal as

$$\boldsymbol{\delta}^{l+1}_{\psi,t} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{s}^{l+1}_t} \odot \psi(\boldsymbol{u}^{l+1}_t - V_{\mathrm{th}}). \tag{10}$$

For LIF neurons, the direct dependence of the membrane potential on the l-th layer's weight is

$$\frac{\partial \boldsymbol{u}^{l+1}_t}{\partial \boldsymbol{W}^l} = (\boldsymbol{s}^l_t)^\top. \tag{11}$$

Substituting these definitions into Eq. (2) gives the decomposition

$$\boldsymbol{G}^l_{\mathrm{BPTT}} = \boldsymbol{G}^l_{\mathrm{sp}} + \boldsymbol{R}^l_{\mathrm{temp}}, \tag{12}$$

where

$$\boldsymbol{G}^l_{\mathrm{sp}} = \sum_{t=1}^{T} \boldsymbol{\delta}^{l+1}_{\psi,t} (\boldsymbol{s}^l_t)^\top \tag{13}$$

is the SG-based spatial term, and

$$\boldsymbol{R}^l_{\mathrm{temp}} = \sum_{t=1}^{T} \boldsymbol{\delta}^{l+1}_{\psi,t} \sum_{\tau<t} \boldsymbol{J}^{l+1}_{t,\tau} (\boldsymbol{s}^l_\tau)^\top \tag{14}$$

is the temporal residual induced by temporal error propagation.

IDDFT further removes the membrane-potential entry from the local derivative construction and keeps only the current input-drive entry after temporal decoupling. Specifically, it uses $\boldsymbol{I}^{l+1}_t = \boldsymbol{W}^l \boldsymbol{s}^l_t + \boldsymbol{b}^{l+1}$, whose dependence on the trainable weight is $\partial \boldsymbol{I}^{l+1}_t / \partial \boldsymbol{W}^l = (\boldsymbol{s}^l_t)^\top$. Therefore, the resulting update contains only the time-local spatial contribution and does not include the temporal backpropagation term. Instead of approximating the spike derivative through the membrane potential $\boldsymbol{u}^{l+1}_t$, IDDFT introduces a relaxed nonlinear factor $\phi(\cdot)$ on the current input drive. This factor

is not obtained as a derivative of the spike function, but is used to construct an input-driven local learning signal defined as:

$$\boldsymbol{\delta}_{\phi,t}^{l+1} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{s}_t^{l+1}} \odot \phi(\boldsymbol{I}_t^{l+1}). \tag{15}$$

Accordingly, the IDDFT update direction is

$$\boldsymbol{G}_{\mathrm{IDDFT}}^{l} = \sum_{t=1}^{T} \boldsymbol{\delta}_{\phi,t}^{l+1} (\boldsymbol{s}_t^{l})^{\top}. \tag{16}$$

The difference between the SG-based spatial term and IDDFT is defined as

$$\boldsymbol{R}_{\phi}^{l} = \boldsymbol{G}_{\mathrm{sp}}^{l} - \boldsymbol{G}_{\mathrm{IDDFT}}^{l}. \tag{17}$$

explicitly shown as

$$\boldsymbol{R}_{\phi}^{l} = \sum_{t=1}^{T} \left[\frac{\partial \mathcal{L}}{\partial \boldsymbol{s}_t^{l+1}} \odot \Big(\psi(\boldsymbol{u}_t^{l+1} - V_{\mathrm{th}}) - \phi(\boldsymbol{I}_t^{l+1}) \Big) \right] (\boldsymbol{s}_t^{l})^{\top}. \tag{18}$$

Since

$$\boldsymbol{G}_{\mathrm{sp}}^{l} = \boldsymbol{G}_{\mathrm{IDDFT}}^{l} + \boldsymbol{R}_{\phi}^{l}, \tag{19}$$

and

$$\boldsymbol{G}_{\mathrm{BPTT}}^{l} = \boldsymbol{G}_{\mathrm{sp}}^{l} + \boldsymbol{R}_{\mathrm{temp}}^{l}, \tag{20}$$

we obtain

$$\boldsymbol{G}_{\mathrm{BPTT}}^{l} = \boldsymbol{G}_{\mathrm{IDDFT}}^{l} + \boldsymbol{R}_{\phi}^{l} + \boldsymbol{R}_{\mathrm{temp}}^{l}. \tag{21}$$

This proves the decomposition. □

This proposition shows that IDDFT can be viewed as a relaxed variant of the spatial term in baseline BPTT-SG: it removes the temporal error-propagation term and replaces the membrane-potential-based surrogate term with an input-driven relaxed term.

A.2 Bound on the Temporal Residual

We analyze the temporal residual from two complementary perspectives. The first is a gradient-decomposition perspective, where the temporal term is bounded through the product of temporal Jacobians in BPTT. The second is a neuron-dynamics perspective, where the contraction condition is interpreted from the step-wise leaky and reset dynamics of LIF neurons.

Lemma 1 (Bounded temporal residual)**.** *Assume that the SG-induced time-local error signal and the presynaptic spike activity are bounded:*

$$\|\boldsymbol{\delta}_{\psi,t}^{l+1}\|_2 \le A, \qquad \|\boldsymbol{s}_t^{l}\|_2 \le S, \tag{22}$$

for all time steps t. Assume further that the one-step temporal term is contractive:

$$\|\boldsymbol{J}_i^{l+1}\|_2 \le \kappa, \qquad 0 \le \kappa < 1. \tag{23}$$

Then the temporal residual satisfies

$$\|\boldsymbol{R}_{\mathrm{temp}}^{l}\|_F \le AST \frac{\kappa}{1-\kappa}. \tag{24}$$

Proof. By definition,

$$\boldsymbol{R}_{\mathrm{temp}}^{l} = \sum_{t=1}^{T} \boldsymbol{\delta}_{\psi,t}^{l+1} \sum_{\tau<t} \boldsymbol{J}_{t,\tau}^{l+1} (\boldsymbol{s}_{\tau}^{l})^{\top}, \tag{25}$$

where

$$\boldsymbol{J}_{t,\tau}^{l+1} = \prod_{i=t-1}^{\tau} \boldsymbol{J}_i^{l+1}. \tag{26}$$

Using submultiplicativity of the spectral norm, we have

$$\|\boldsymbol{J}_{t,\tau}^{l+1}\|_2 = \left\| \prod_{i=t-1}^{\tau} \boldsymbol{J}_i^{l+1} \right\|_2 \leq \prod_{i=t-1}^{\tau} \|\boldsymbol{J}_i^{l+1}\|_2 \leq \kappa^{t-\tau}. \tag{27}$$

Therefore,

$$\begin{aligned} \|\boldsymbol{R}_{\text{temp}}^l\|_F &\leq \sum_{t=1}^{T} \|\boldsymbol{\delta}_{\psi,t}^{l+1}\|_2 \sum_{\tau<t} \|\boldsymbol{J}_{t,\tau}^{l+1}\|_2 \|\boldsymbol{s}_\tau^l\|_2 \\ &\leq AS \sum_{t=1}^{T} \sum_{\tau<t} \kappa^{t-\tau}. \end{aligned} \tag{28}$$

For each t,

$$\sum_{\tau<t} \kappa^{t-\tau} = \sum_{k=1}^{t-1} \kappa^k \leq \frac{\kappa}{1-\kappa}. \tag{29}$$

Thus,

$$\|\boldsymbol{R}_{\text{temp}}^l\|_F \leq AST \frac{\kappa}{1-\kappa}. \tag{30}$$

This proves the bound. □

This lemma shows that the temporal residual is suppressed when the temporal term is contractive. For LIF neurons, this contraction has a direct interpretation from the leaky membrane dynamics. Ignoring the layer superscript for clarity, the membrane-potential evolves as

$$\boldsymbol{u}_{i+1} = \beta \boldsymbol{u}_i + \boldsymbol{I}_{i+1} - V_{\text{th}} \boldsymbol{r}_{i+1}, \tag{31}$$

where $0 < \beta < 1$ is the membrane decay factor. Thus, in the absence of spike-reset effects, the temporal dependence is directly scaled by β, which already attenuates historical membrane states. With the SG approximation, the one-step temporal term can be written as

$$\boldsymbol{J}_i = \frac{\partial \boldsymbol{u}_{i+1}}{\partial \boldsymbol{u}_i} + \frac{\partial \boldsymbol{u}_{i+1}}{\partial \boldsymbol{s}_i} \frac{\partial \boldsymbol{s}_i}{\partial \boldsymbol{u}_i} \approx \beta \boldsymbol{I} - V_{\text{th}} \operatorname{diag}\left[\psi(\boldsymbol{u}_i - V_{\text{th}})\right], \tag{32}$$

where the second term comes from the reset path. Since the surrogate derivative is bounded and active mainly around the firing threshold, and the reset term suppresses the carry-over of membrane states after spikes, the temporal dependence is prevented from accumulating indefinitely. A sufficient contraction condition is

$$\left\|\beta \boldsymbol{I} - V_{\text{th}} \operatorname{diag}\left[\psi(\boldsymbol{u}_i - V_{\text{th}})\right]\right\|_2 \leq \kappa < 1. \tag{33}$$

Under this condition, the contribution from a past membrane-potential state at time τ to a later time t is attenuated as

$$\|\boldsymbol{J}_{t,\tau}\|_2 \leq \kappa^{t-\tau}. \tag{34}$$

Therefore, the temporal residual is suppressed by the leaky-reset dynamics of LIF neurons. This provides a sufficient condition under which removing the temporal error-propagation path has limited influence, consistent with the theoretical and experimental analysis in SLTT [21].

A.3 Positive Alignment Supports Effective Learning

After bounding the temporal residual, we analyze the remaining relaxed-factor replacement. According to Eq. (6), the residual $\boldsymbol{R}_\phi^l$ measures the difference caused by replacing the membrane-potential surrogate element $\psi(\boldsymbol{u}_t^{l+1} - V_{\text{th}})$ with the input-driven relaxed element $\phi(\boldsymbol{I}_t^{l+1})$. This replacement does not need to be pointwise accurate or small in norm. After the temporal residual is controlled, we treat the SG-based spatial term direction $\boldsymbol{G}_{\text{sp}}^l$ as the descent-compatible reference direction. A sufficient condition for the relaxed replacement to preserve the effective update is that the resulting IDDFT direction remains positively aligned with this reference direction:

$$\left\langle \boldsymbol{G}_{\text{sp}}^l, \boldsymbol{G}_{\text{IDDFT}}^l \right\rangle_F > 0. \tag{35}$$

Since the two directions differ only in their local nonlinear factors, such positive alignment can be supported by a positive statistical correlation between $\phi(\boldsymbol{I}_t^{l+1})$ and $\psi(\boldsymbol{u}_t^{l+1} - V_{\mathrm{th}})$. This correlation condition is not an additional computational step, but an analytical explanation for why IDDFT can use relaxed input-driven nonlinear functions without pointwise approximation of surrogate derivatives. We next show that positive alignment at the update-direction level can provide an effective learning direction.

Proposition 2 (Loss decrease under positive alignment)**.** *Let the loss $\mathcal{L}(\boldsymbol{W}^l)$ be differentiable at $\boldsymbol{W}^l$. Define*

$$\boldsymbol{g}_{\mathrm{true}}^l := \nabla_{\boldsymbol{W}^l} \mathcal{L}(\boldsymbol{W}^l)$$

as the exact spatial term gradient with respect to $\boldsymbol{W}^l$, and let $\boldsymbol{G}_{\mathrm{IDDFT}}^l \neq \boldsymbol{0}$ denote the IDDFT update direction. Consider the IDDFT update

$$\boldsymbol{W}^{l,+} = \boldsymbol{W}^l - \eta \boldsymbol{G}_{\mathrm{IDDFT}}^l, \qquad \eta > 0.$$

If

$$\left\langle \boldsymbol{g}_{\mathrm{true}}^l, \boldsymbol{G}_{\mathrm{IDDFT}}^l \right\rangle_F > 0,$$

equivalently,

$$\angle \left(\boldsymbol{g}_{\mathrm{true}}^l, \boldsymbol{G}_{\mathrm{IDDFT}}^l \right) < 90^\circ,$$

then there exists $\bar{\eta} > 0$ such that for all $0 < \eta < \bar{\eta}$,

$$\mathcal{L}\left(\boldsymbol{W}^l - \eta \boldsymbol{G}_{\mathrm{IDDFT}}^l \right) < \mathcal{L}(\boldsymbol{W}^l).$$

That is, when the reference direction is descent-compatible and the IDDFT direction preserves positive alignment with it, IDDFT can provide an effective learning direction.

Proof. Define the scalar function

$$h(\eta) := \mathcal{L}\left(\boldsymbol{W}^l - \eta \boldsymbol{G}_{\mathrm{IDDFT}}^l \right).$$

Since $\mathcal{L}$ is differentiable at $\boldsymbol{W}^l$, $h(\eta)$ is differentiable at $\eta = 0$. By the first-order Taylor expansion around $\eta = 0$, we have

$$h(\eta) = h(0) + \eta h'(0) + O(\eta^2), \qquad \eta \to 0.$$

Since

$$h(0) = \mathcal{L}(\boldsymbol{W}^l).$$

By the chain rule,

$$h'(0) = -\left\langle \nabla_{\boldsymbol{W}^l} \mathcal{L}(\boldsymbol{W}^l), \boldsymbol{G}_{\mathrm{IDDFT}}^l \right\rangle_F = -\left\langle \boldsymbol{g}_{\mathrm{true}}^l, \boldsymbol{G}_{\mathrm{IDDFT}}^l \right\rangle_F.$$

where $\langle \boldsymbol{A}, \boldsymbol{B} \rangle_F = \mathrm{Tr}(\boldsymbol{A}^\top \boldsymbol{B})$ denotes the Frobenius inner product. Therefore,

$$\mathcal{L}\left(\boldsymbol{W}^l - \eta \boldsymbol{G}_{\mathrm{IDDFT}}^l \right) = \mathcal{L}(\boldsymbol{W}^l) - \eta \left\langle \boldsymbol{g}_{\mathrm{true}}^l, \boldsymbol{G}_{\mathrm{IDDFT}}^l \right\rangle_F + O(\eta^2).$$

If

$$\left\langle \boldsymbol{g}_{\mathrm{true}}^l, \boldsymbol{G}_{\mathrm{IDDFT}}^l \right\rangle_F > 0,$$

then the first-order term is strictly negative. Since $O(\eta^2)/\eta \to 0$ as $\eta \to 0$, the higher-order term is dominated by the negative first-order term for sufficiently small $\eta > 0$. Hence, there exists $\bar{\eta} > 0$ such that for all $0 < \eta < \bar{\eta}$,

$$\mathcal{L}\left(\boldsymbol{W}^l - \eta \boldsymbol{G}_{\mathrm{IDDFT}}^l \right) < \mathcal{L}(\boldsymbol{W}^l).$$

Thus, under the positive-alignment condition with a descent-compatible direction, the IDDFT update can provide an effective learning direction for a sufficiently small learning rate. □

B Statistical Alignment between IDDFT and Surrogate Gradient

This experiment provides quantitative support for the statistical-alignment argument in Section 4 and Appendix A.3. As shown in Eqs. (13) and (16), the SG-based spatial update and the IDDFT update share the same outer-product structure but use different local nonlinear factors. We therefore examine the statistical relationship between the input-driven relaxed factor $\phi(I_t^l)$ and the membrane-potential-based surrogate factor $\psi(u_t^l - V_{\text{th}})$ to assess whether the proposed positive-correlation condition holds in practice. Specifically, we measure their empirical Pearson correlation coefficient:

$$r_{l,t} = \text{Corr}\left[\phi(I_t^l), \psi(u_t^l - V_{\text{th}})\right]. \tag{36}$$

In this experiment, we feed five mini-batches into each network and record the input drive I_t^l and membrane potential u_t^l at every spiking layer and timestep during forward propagation. We then compute the corresponding $\phi(I_t^l)$ and $\psi(u_t^l - V_{\text{th}})$ and calculate their correlation. Measurements are performed both at model initialization and after training across CIFAR-10/ResNet-19, CIFAR-100/ResNet-19, and CIFAR10-DVS/VGGSNN. For each experimental setting, we aggregate the correlation coefficients computed across all spiking layers and timesteps (20 spiking layers with $T = 4$ for ResNet-19; 10 spiking layers with $T = 10$ for VGGSNN) and compute summary statistics. The overall results are summarized in Table 3.

Table 3: Correlations between the input-driven relaxed factor and the membrane-potential-based surrogate factor. Each row reports statistics across all spiking layers and timesteps. Positive denotes the percentage of coefficients satisfying $r_{l,t} > 0$.

Dataset	Stage	Mean $\pm$ SD	Median	Min.	Max.	Positive (%)
CIFAR-10	Initial	0.57 ± 0.02	0.57	0.45	0.62	100
	Trained	0.55 ± 0.05	0.53	0.46	0.69	100
CIFAR-100	Initial	0.56 ± 0.02	0.57	0.44	0.61	100
	Trained	0.55 ± 0.05	0.53	0.47	0.72	100
CIFAR10-DVS	Initial	0.79 ± 0.10	0.74	0.55	1.00	100
	Trained	0.78 ± 0.11	0.81	0.47	1.00	100

To examine the behavior across timesteps, we additionally average $r_{l,t}$ over all spiking layers at each timestep. Table 4 reports the resulting layer-averaged correlations.

Table 4: Layer-averaged correlations at each timestep. Dashes indicate timesteps not used in the corresponding experiment.

Dataset	Stage	Timestep t									
		1	2	3	4	5	6	7	8	9	10
CIFAR-10	Initial	0.57	0.56	0.57	0.57	–	–	–	–	–	–
	Trained	0.62	0.52	0.52	0.53	–	–	–	–	–	–
CIFAR-100	Initial	0.57	0.56	0.56	0.57	–	–	–	–	–	–
	Trained	0.62	0.53	0.52	0.53	–	–	–	–	–	–
CIFAR10-DVS	Initial	0.97	0.80	0.77	0.78	0.78	0.78	0.73	0.78	0.77	0.77
	Trained	0.97	0.81	0.77	0.79	0.76	0.75	0.75	0.76	0.74	0.75

Across all evaluated settings, the correlation coefficients at every spiking layer and timestep remain positive both at initialization and after training, satisfying the proposed positive-correlation condition. These models are also successfully trained and achieve strong performance, consistent with the expectation that positive alignment supports effective learning. The consistently positive correlations across the evaluated datasets, architectures, training stages, and timestep counts provide direct numerical evidence supporting the statistical justification of the relaxed nonlinearity.

C Practical Selection of the Relaxed Nonlinearity

IDDFT reduces the dependence of effective training on the precise form and parameterization of the backward nonlinearity, but it does not imply that every nonlinear function or parameter setting is equally effective. Rather, it provides a broader range of viable choices, while the performance of individual settings may still vary across tasks and network architectures. We therefore provide two practical guidelines for selecting the relaxed nonlinearity.

Threshold-centered default. A simple and conservative default is to select a nonlinearity that peaks around V_{th} and decays gradually on both sides. Near-threshold perturbations are most likely to change the spike state and should therefore receive the strongest local update signal, whereas this sensitivity decreases farther away from the threshold. Since IDDFT uses the input drive, we conservatively center the function at $I = V_{\mathrm{th}}$ with a moderate width, allowing potentially relevant off-center inputs to receive weaker but nonzero signals while suppressing excessive irrelevant updates.

Distribution-based selection. A more adaptive strategy is to use a small amount of unlabeled data to collect the input drive I and output spike s from each spiking layer and estimate the empirical firing probability $P(s = 1 \mid I)$, i.e., the proportion of samples that generate a spike when the input drive falls within a given interval. Regions where this probability changes rapidly indicate that small variations in the input drive can alter the spike state and should therefore receive larger backward-nonlinearity values. We can then select a function whose effective range covers these sensitive regions across all spiking layers while maintaining large values within them.

For quantitative validation of this strategy, we use only five mini-batches under the VGG9/CIFAR-10 setting described in Section 5.1. For this analysis, a sensitive input location is considered covered by the backward nonlinear function when $\phi(I) \geq 0.1$. Robust coverage denotes the sensitive-region coverage achieved by at least 80% of the spiking layers. Among candidates with robust coverage $\geq 85\%$, we prefer narrower functions to prevent trivially high coverage from excessive width. Table 5 reports the five highest-ranked parameter settings for the examined Gaussian relaxed nonlinearity $\phi(I) = \exp[-\alpha(I + \theta)^2]$, together with their robust coverage and effective response widths. Here, the effective response width denotes the length of the input interval over which $\phi(I) \geq 0.1$.

Table 5: Gaussian parameter settings selected by the distribution-based strategy. Width denotes the length of the input interval satisfying $\phi(I) \geq 0.1$.

(α, θ)	Robust coverage	Width
$(3.0, -1.0)$	0.99	1.75
$(3.0, -0.5)$	0.90	1.75
$(3.0, -1.5)$	0.88	1.75
$(2.5, -1.0)$	1.00	1.92
$(2.5, -1.5)$	0.94	1.92

These settings, selected solely from input-drive and spike statistics obtained using limited unlabeled data, closely overlap with the high-performing region in the accuracy heatmap in Section 5.1 (Fig. 2). This agreement provides empirical support for the distribution-based selection strategy.

Overall, the optimal nonlinearity and parameter settings may vary across tasks and architectures. IDDFT does not eliminate this dependence but broadens the high-performing region. When task-specific optimality is unnecessary, the threshold-centered default or distribution-based selection can provide a reliable configuration. This flexibility also allows neuromorphic implementations to prioritize nonlinearities and parameters that are easier to realize while retaining strong performance. When the highest task-specific accuracy is required, the parameters can still be optimized through grid search, genetic algorithms, or, in principle, bilevel hyperparameter optimization [82]. IDDFT is therefore not tuning-free, but such optimization does not change its core advantages. It requires neither membrane-potential access, membrane-potential-based surrogate derivatives, nor temporal error backpropagation. The broader high-performing region also offers more viable configurations and thus reduces the practical difficulty of task-specific parameter optimization.

D More Results

D.1 Training Memory Cost and Runtime

We further evaluate the training memory and runtime efficiency of IDDFT. BPTT-SG requires storing the through-time computation graph and its associated activations, including membrane potentials and spike activity, for temporal error propagation. In contrast, IDDFT removes the temporal error-propagation path and constructs the time-local update signal directly from the current input drive, allowing each timestep's graph to be released after computing and accumulating the spatial weight gradients. This substantially reduces peak training memory.

IDDFT supports two memory-efficient implementations. Recomputation first performs a complete sequence-level forward pass without gradient tracking to obtain the output error signal. It then repeats forward computation timestep by timestep, using current input drives, presynaptic spikes, and that error signal to compute and accumulate weight gradients, releasing each graph immediately afterward. Alternatively, step-wise training follows the execution schedule of SLTT [21]. At each timestep, it obtains the output error signal directly from the per-step loss, computes and accumulates the corresponding spatial weight gradients, and releases the current computation graph. Parameters are updated once after all timesteps in both implementations. Both avoid retaining historical membrane potentials and the complete temporal computation graph, giving comparable peak memory usage in our measurements. However, step-wise training avoids the additional forward pass required by recomputation and is therefore more time-efficient.

Table 6 reports peak allocated CUDA memory and wall-clock time per epoch on CIFAR-10 and CIFAR-100 using ResNet-19 on a single GPU. Standard batch normalization replaces tdBN in these measurements to avoid coupling normalization across timesteps.

Table 6: Training memory and runtime on CIFAR-10 and CIFAR-100 with standard batch normalization. Memory denotes peak allocated CUDA memory in GiB, and Time denotes wall-clock time per epoch in seconds, reported as mean $\pm$ standard deviation.

		CIFAR-10		CIFAR-100	
Method	Implementation	Memory	Time	Memory	Time
BPTT-SG	—	10.30	79.57 ± 0.46	10.30	79.33 ± 0.23
BP-IDDFT	Recomputation	**3.39**	108.13 ± 0.15	**3.39**	108.70 ± 0.70
	Step-wise	**3.39**	**76.80** ± 0.44	**3.39**	**76.90** ± 0.52
uSF-SG	Full-sequence	10.50	80.20 ± 0.70	10.50	**79.63** ± 0.40
uSF-IDDFT	Recomputation	**3.44**	111.20 ± 0.70	**3.44**	111.43 ± 0.76
	Step-wise	**3.44**	**80.17** ± 0.64	**3.44**	80.03 ± 0.49

The results show that both BP-IDDFT and uSF-IDDFT reduce peak memory by approximately 67% relative to their respective SG baselines. Recomputation introduces additional runtime because it repeats the full forward computation. Step-wise training preserves the same memory reduction while achieving comparable or better runtime than SG. In a separate 100-epoch accuracy comparison with standard batch normalization, BP-IDDFT achieves 93.47% and 64.15% on CIFAR-10 and CIFAR-100, respectively, compared with 92.42% and 60.62% for BPTT-SG. These results demonstrate that IDDFT can substantially reduce peak training memory without sacrificing performance or incurring a substantial runtime penalty with the step-wise implementation, making it more suitable for training deep SNNs under limited hardware resources. The slightly higher memory usage of uSF arises from our PyTorch implementation, which explicitly stores a per-layer FP32 feedback matrix $c \cdot \mathrm{sign}(W)$. BP instead reuses the forward weights. This implementation-specific overhead could be avoided by reading $\mathrm{sign}(W)$ on demand rather than storing additional feedback tensors.

We also evaluate the training memory and runtime of IDDFT with tdBN [9]. Since tdBN computes normalization statistics jointly across the batch, temporal, and spatial dimensions, the normalized

activations at each timestep depend on the entire sequence. The step-wise scheme therefore cannot be applied directly, but recomputation still allows IDDFT to avoid retaining the full temporal computation graph. In our implementation, tdBN increases memory use and runtime by aggregating activations across all timesteps and introducing tensor permutations, memory copies, and additional normalization operations. Recomputation further increases runtime by adding a full forward pass. Table 7 reports the results on CIFAR-10 with all other experimental settings unchanged.

Table 7: Peak allocated CUDA memory (GiB) and wall-clock time per epoch (mean $\pm$ standard deviation, in seconds) for training ResNet-19 on CIFAR-10 with tdBN. Both IDDFT variants use recomputation.

Method	Memory	Time
BPTT-SG	12.21	95.17 ± 0.47
BP-IDDFT	**4.25**	122.93 ± 0.42
uSF-SG	12.26	95.37 ± 0.81
uSF-IDDFT	**4.29**	125.23 ± 0.35

As shown in Table 7, IDDFT still reduces peak memory by approximately 65% with tdBN, although recomputation introduces a runtime trade-off. Thus, substantial memory savings also hold under temporal normalization. Moreover, the memory required for backward computation in IDDFT scales as $O(1)$ with respect to the number of timesteps T, compared with $O(T)$ for BPTT-SG. The resulting memory advantage becomes more pronounced as T increases. This graph-storage advantage is not specific to a particular dataset or architecture, although the exact peak-memory reduction depends on the model, batch size, and normalization scheme.

D.2 Robustness under the RFGSM Adversarial Attack

RFGSM is a randomized variant of FGSM that first applies a small random perturbation to the input and then performs a one-step gradient-sign update within the remaining perturbation budget [80]. Compared with standard FGSM, the random initialization makes the generated adversarial examples less tied to a single deterministic gradient direction, providing an additional robustness evaluation under one-step attacks.

Table 8 reports the transfer-attack results under RFGSM. Overall, the observations are consistent with the main results under other adversarial attacks. uSF-IDDFT achieves better remaining accuracy across CIFAR-10, CIFAR-100, and CIFAR10-DVS, especially when evaluated as the target model under attacks generated from different source models. The results further support that IDDFT can improve robustness while maintaining strong clean accuracy.

Table 8: Accuracy under the RFGSM adversarial attack on CIFAR-10, CIFAR-100, CIFAR10-DVS.

		CIFAR-10			CIFAR-100			CIFAR10-DVS		
Attack	Src./Tgt.	BPTT	uSF	IDDFT	BPTT	uSF	IDDFT	BPTT	uSF	IDDFT
Clean	Acc.	96.75	96.03	96.71	80.44	78.09	78.88	82.18	83.42	83.88
RFGSM	BPTT	65.01	74.31	**76.27**	19.84	35.59	**37.93**	3.28	28.16	**57.88**
	uSF	80.27	75.41	**81.89**	44.99	28.62	**47.45**	28.42	12.38	**58.72**
	IDDFT	78.87	**79.64**	72.86	49.52	**49.88**	37.42	37.98	39.52	**42.10**
	Avg.	74.72	76.45	**77.01**	38.12	38.03	**40.93**	23.23	26.69	**52.90**

D.3 Further Results on Robustness under Adversarial Attacks

The comparisons between uSF-SG and uSF-IDDFT in Tables 2 and 8 show that IDDFT improves the average remaining accuracy under the same uSF feedback mechanism. To further examine the contribution of IDDFT independently of uSF, we additionally evaluate BP-IDDFT, which retains the input-driven learning rule but uses standard BP for spatial error propagation. We evaluate BP-IDDFT on CIFAR-10, CIFAR-100, and CIFAR10-DVS under FGSM, RFGSM, and PGD-10 attacks, following the same attack settings and preprocessing described in Appendix F. For transfer attacks,

adversarial perturbations are computed using source models trained with BPTT-SG, uSF-SG, and uSF-IDDFT and then added to the input samples. The resulting perturbed samples are fed into the BP-IDDFT-trained target model to measure its remaining accuracy. For self-attacks, the perturbations are computed using the BP-IDDFT-trained model itself. Table 9 reports the remaining test accuracy under the different attacks.

Table 9: Remaining accuracy (%) of BP-IDDFT-trained models under transfer and self-attacks. Each transfer-attack column identifies the training method of the source model used to generate the attack perturbations. Green, gray, and blue cells denote clean accuracies, self-attacks, and averages, respectively. Avg. reports the mean accuracy across all four attack sources, including self-attacks. Orange cells with bold numbers highlight the highest remaining accuracy among the four attack-source columns in each row.

Dataset	Attack	Clean	Transfer attacks			Self-attack	Avg.
			BPTT-SG	uSF-SG	uSF-IDDFT	BP-IDDFT	
CIFAR-10	FGSM	97.00	72.33	**80.04**	76.82	55.01	71.05
	RFGSM		77.06	**82.61**	80.50	58.58	74.69
	PGD-10		43.14	**61.56**	40.64	4.27	37.40
CIFAR-100	FGSM	79.78	30.90	40.23	**43.59**	23.49	34.55
	RFGSM		38.61	48.52	**51.52**	26.33	41.24
	PGD-10		**29.51**	22.97	23.18	0.73	19.10
CIFAR10-DVS	FGSM	84.38	55.10	**59.88**	57.34	41.92	53.56
	RFGSM		57.04	**58.86**	53.74	30.90	50.13
	PGD-10		57.04	**58.44**	47.36	11.94	43.70

The results show that BP-IDDFT-trained models retain good accuracy under various attacks across all three examined datasets. Compared with BPTT-SG-trained models, BP-IDDFT achieve higher remaining accuracy under the same adversarial attacks. For example, on CIFAR-10, under PGD-10 attacks generated using source models trained with uSF-SG and uSF-IDDFT, BP-IDDFT-trained target models retain accuracies of 61.56% and 40.64%, respectively. While under the corresponding attacks, BPTT-SG-trained target models retain 52.26% and 35.06%, respectively (Table 2). These results provide further evidence that IDDFT itself contributes to improved robustness under adversarial attacks.

D.4 Sensitivity to Neuronal Parameters

In the main experiments, all methods use $(\beta, V_{\text{th}}) = (0.5, 1.0)$ to ensure identical membrane-potential decay and firing conditions, allowing performance comparisons to primarily reflect differences between the learning algorithms rather than differences in neuronal dynamics. Here, to evaluate sensitivity to these parameters, we additionally test all methods on CIFAR-10 with ResNet-19 under three representative (β, V_{th}) combinations, while keeping all other experimental settings unchanged. The results are reported in Table 10.

Table 10: Test accuracy (%) on CIFAR-10 under different membrane-potential decay factors β and firing thresholds V_{th}.

	(β, V_{th})		
Method	$(0.25, 1.00)$	$(0.50, 0.75)$	$(0.50, 1.25)$
BPTT-SG	96.81	96.75	96.75
uSF-SG	96.30	96.33	96.12
BP-IDDFT	96.60	95.88	97.14
uSF-IDDFT	96.17	96.00	96.55

The results show that IDDFT works effectively and maintains competitive performance across various settings of the decay factor and firing threshold, indicating that its effectiveness is not sensitive to changes in these neuronal parameters. It should be noted that the parameter settings used in the main experiments and this sensitivity analysis are intended for controlled comparison and parameter-

robustness evaluation rather than method-specific optimization. These settings are not necessarily optimal for each algorithm, and further method-specific tuning may improve absolute accuracy.

D.5 Evaluation under Different Neuronal Dynamics

To further examine the applicability of IDDFT to SNNs with different neuronal dynamics, we extend the evaluation beyond the LIF neuron model. Specifically, we replace the LIF neurons in ResNet-19 with integrate-and-fire (IF) neurons and evaluate the network on CIFAR-10. IF neuronal dynamics are obtained by setting the membrane-potential decay factor to $\beta = 1$, thereby eliminating leakage. The firing threshold remains unchanged at $V_{\mathrm{th}} = 1$. The network uses $T = 4$ timesteps and is trained for 400 epochs with a batch size of 128. We use SGD with an initial learning rate of 0.1, momentum of 0.9, weight decay of 10^{-4}, and cosine learning-rate decay.

With IF neurons, BP-IDDFT and uSF-IDDFT achieve test accuracies of 94.47% and 95.34%, respectively. These results show that IDDFT remains effective under different neuronal dynamics, indicating that our method is not confined to a specific neuronal model.

D.6 Evaluation on Tasks with Richer Temporal Structure

To further examine the applicability of IDDFT to tasks with stronger temporal structure, we consider event-based gesture recognition on DVS128 Gesture. Specifically, we use VGGSNN with $T = 20$ timesteps and train it using uSF-IDDFT. We use the dataset-provided training and test splits. Each event stream is divided into 20 consecutive segments containing approximately equal numbers of events, which are accumulated into two-polarity frames and resized to 48×48. The LIF neurons use $\beta = 0.5$ and $V_{\mathrm{th}} = 1$. Training runs for 300 epochs with a batch size of 16, using SGD with an initial learning rate of 0.05, momentum of 0.9, weight decay of 5×10^{-4}, and cosine learning-rate decay. The results show that uSF-IDDFT achieves a test accuracy of 96.18%, demonstrating effective learning on this event-based task with strong temporal structure.

E Pseudo code of the IDDFT

Algorithm 1 summarizes the training procedure of IDDFT. The forward pass follows the standard LIF-based SNN dynamics, while the backward pass replaces the membrane-potential-based surrogate derivative with an input-driven nonlinear factor to construct time-local update signals.

Algorithm 1 One Training Update with IDDFT

Require: Mini-batch $(\boldsymbol{x}, \boldsymbol{y})$, SNN model $f_{\boldsymbol{\Theta}}$, timesteps T, learning rate η
Require: Relaxed IDDFT nonlinear factor $\phi(\cdot)$
Ensure: Updated parameters $\boldsymbol{\Theta}$

Directly encode the input as $\boldsymbol{x}_t = \boldsymbol{x}$ for $t = 1, \ldots, T$
Initialize membrane states and spikes for all layers
for $t = 1$ to T **do**
    Perform one-step LIF forward propagation
    Store input drives $\{\boldsymbol{I}_t^l\}$ and presynaptic spikes $\{\boldsymbol{s}_t^{l-1}\}$ for all layers
    Obtain output logits $\boldsymbol{o}_t$
end for
Compute time-averaged logits $\boldsymbol{o} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{o}_t$
Compute loss $\mathcal{L} = \mathrm{CE}(\boldsymbol{o}, \boldsymbol{y})$
for each layer l **do**
    Obtain the spatial error signals $\{\boldsymbol{e}_t^l\}_{t=1}^{T}$, where $\boldsymbol{e}_t^l = \partial\mathcal{L}/\partial\boldsymbol{s}_t^l$
    Generate the IDDFT update signal (can be computed in parallel)

$$\boldsymbol{G}_{\mathrm{IDDFT}}^{l-1} = \sum_{t=1}^{T} \left(\boldsymbol{e}_t^l \odot \phi(\boldsymbol{I}_t^l)\right) (\boldsymbol{s}_t^{l-1})^{\top}$$

    Update $\boldsymbol{W}^{l-1} \leftarrow \boldsymbol{W}^{l-1} - \eta \boldsymbol{G}_{\mathrm{IDDFT}}^{l-1}$
end for

Because IDDFT removes the temporal error-propagation path, the update signal at each timestep does not require recursive backward propagation through neighboring timesteps. After the forward pass, the input drives $\{\boldsymbol{I}_t^l\}$, presynaptic spikes $\{\boldsymbol{s}_t^{l-1}\}$, and nonlinear factors $\{\phi(\boldsymbol{I}_t^l)\}$ are already available. Once the spatial error signals $\{\boldsymbol{e}_t^l\}$ are obtained, the timestep-wise IDDFT update contributions can, in principle, be computed independently and summed along the temporal dimension. This makes it possible to implement the temporal part of the backward update as a parallel tensor operation, offering the potential to further accelerate training.

F Experiments Details

F.1 Environment and Parameter Settings

All experiments are performed on an HPC server equipped with Ubuntu 20.04.6 LTS, two Intel Xeon Platinum 8360Y CPUs running at 2.40 GHz, 2.0 TiB RAM, and 8 NVIDIA A100 80GB PCIe GPUs. The code is implemented in Python 3.11.13 and PyTorch 2.9.1 with CUDA 12.8 and cuDNN 9.10.2. Model weights are randomly initialized using the default initialization method in PyTorch.

Table 11 lists the hyperparameters used in our experiments. SGD with an initial learning rate of 0.1, momentum of 0.9, and weight decay of 10^{-4} is used for all experiments [83]. All experiments use the CosineAnnealingLR scheduler to adjust the learning rate, which decays to 10^{-6} over epochs [84]. Cross-entropy loss with label smoothing of 0.1 is adopted. For all spiking neurons, the membrane decay factor and firing threshold are fixed to $\beta = 0.5$ and $V_{\text{th}} = 1.0$, respectively.

Table 11: Hyperparameter settings for the mechanism validation and benchmark experiments. The validation column reports the settings used to verify the effectiveness of the proposed IDDFT mechanism. In IDDFT, α and θ denote the sharpness and horizontal-shift parameters of the Gaussian nonlinear function used in the backward process, respectively. In SG-based baselines, the arctangent surrogate function with slope factor 2.0 is used.

Hyperparameter	Validation	CIFAR-10	CIFAR-100	CIFAR10-DVS	Tiny-ImageNet
Architecture	VGG9	ResNet-19	ResNet-19	VGGSNN	ResNet-19
Epochs	120	400	400	400	300
Batch size	128	128	128	50	112
α	varied	3.0	3.0	2.5	3.0
θ	varied	-1.5	-1.5	-1.0	-1.5

For the effectiveness validation experiments, each setting is repeated over 3 independent trials. For all other benchmark experiments, each setting is repeated over 5 independent trials. Tiny-ImageNet experiments are conducted with multi-GPU training using `torchrun`. Specifically, we use 8 GPUs on a single node, with a per-GPU batch size of 14, corresponding to a global batch size of 112. Gradients are manually averaged across GPUs after backpropagation.

F.2 Datasets and Preprocessing

CIFAR-10 and CIFAR-100. The CIFAR-10 and CIFAR-100 datasets [71] contain RGB static images with a resolution of 32×32 pixels. CIFAR-10 consists of 60,000 images across 10 classes, with 50,000 images for training and 10,000 images for testing. CIFAR-100 has the same number of images but is distributed over 100 classes. In data preprocessing, we normalize the dataset by using dataset-specific statistics. Random cropping and random horizontal flipping are applied during training. We further use AutoAugment [85] and Cutout [86] for data augmentation. The pixel values are directly fed into the input layer at each timestep as direct encoding.

CIFAR10-DVS. The CIFAR10-DVS dataset [72] is an event-based version of CIFAR-10, containing neuromorphic event streams from 10 object classes. Since the dataset does not provide an official training/test split, we follow standard method [76, 87] that use 90% of the samples in each class for training and the remaining 10% for testing. In data preprocessing, each event stream is divided into 10 temporal blocks, and spike events within each block are accumulated into two-polarity frames with a spatial resolution of 128×128. The resulting frame sequences are then resized to 48×48.

During training, random cropping and random horizontal flipping are applied for data augmentation [87].

Tiny-ImageNet. The Tiny-ImageNet dataset is a modified subset of ImageNet [73]. It contains RGB static images with a resolution of 64×64 pixels across 200 classes, including 100,000 training images and 10,000 validation images. In data preprocessing, images are normalized using ImageNet statistics. During training, random cropping and random horizontal flipping are applied, and ImageNet-style AutoAugment [85] is used for data augmentation. Similar to CIFAR datasets, the pixel values are directly fed into the input layer at each timestep as direct encoding.

DVS128 Gesture. The DVS128 Gesture dataset [88] contains 11 hand and arm gesture categories performed by 29 subjects under three illumination conditions, recorded using a DVS sensor with a spatial resolution of 128×128. We follow the provided training/test split, using 1,176 training samples and 288 test samples. In data preprocessing, each event stream is divided into 20 consecutive segments containing approximately equal numbers of events. Events within each segment are accumulated into two-polarity frames with a spatial resolution of 128×128. The resulting frame sequences are then resized to 48×48 using bilinear interpolation. No additional data augmentation is applied during training.

F.3 Adversarial Attack Settings

To evaluate adversarial robustness, we conduct transfer-based attacks among models trained with different methods. Specifically, adversarial examples are generated by one source model and then evaluated on all target models, forming a source-to-target transfer matrix. We consider three gradient-based attacks, namely FGSM [69], RFGSM [80], and PGD [70]. For all attacks, the perturbation budget is set to $\epsilon = 8/255$ under the ℓ_∞ constraint. For RFGSM, we use a random initialization scale of $\gamma = 0.01$ followed by one gradient-sign update within the remaining perturbation budget. For PGD, we use random initialization within the ℓ_∞ ball and set the step size to $\gamma = 0.01$. To examine the effect of attack strength, PGD is evaluated with 7, 10, 20, and 40 iterations. After each update, adversarial examples are projected back to the ℓ_∞ ball around the original input and clipped to the valid input range.

For CIFAR-10 and CIFAR-100, attacks are applied to the raw image tensors before normalization, while normalization is performed inside the model forward pass. For CIFAR10-DVS, since the input consists of accumulated event frames rather than RGB images, attacks are directly applied to the event-frame tensors, and identity normalization is used during adversarial evaluation. The attack loss is cross-entropy.

F.4 Network Architectures

We use three network architectures in our experiments. Specifically, VGG9 [43] is used for the mechanism validation experiments, ResNet-19 backbone [9] is used for CIFAR-10, CIFAR-100, and Tiny-ImageNet, and VGGSNN [11] is used for CIFAR10-DVS. The detailed structures are summarized in Tables 12–14. In the tables, nC3 denotes a convolutional layer with n output channels and a 3×3 kernel, nC1 denotes a convolutional layer with n output channels and a 1×1 kernel, AP2 denotes average pooling with a 2×2 kernel, AAP1 and AAP3 denote adaptive average pooling to 1×1 and 3×3, respectively, nFC denotes a fully connected layer with n output units, DO denotes dropout, and C denotes the number of output classes. Time-dependent batch normalization [9] is applied unless otherwise specified.

Table 12: VGG9 architecture.

Model	Structure
VGG9	64C3-LIF-64C3-LIF-AP2 128C3-LIF-128C3-LIF-AP2 256C3-LIF-256C3-LIF-256C3-LIF-AP2 1024FC-LIF-DO CFC

Table 13: ResNet-19 architecture.

Layer	ResNet-19
Input	128C3-LIF
Group 1	$(128\text{C}3\text{-LIF-}128\text{C}3\text{-LIF}) \times 3$
Group 2	$(256\text{C}3\text{-LIF-}256\text{C}3\text{-LIF}) \times 3$
Group 3	$(512\text{C}3\text{-LIF-}512\text{C}3\text{-LIF}) \times 2$
Classifier	AAP1-256FC-LIF-CFC

Table 14: VGGSNN architecture.

Model	Structure
VGGSNN	64C3-LIF 128C3-LIF-AP2 256C3-LIF-256C3-LIF-AP2 512C3-LIF-512C3-LIF-AP2 512C3-LIF-512C3-LIF AAP3-1024FC-LIF-DO-1024FC-LIF-DO-CFC

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction are supported by the derivation of the proposed method in Section 4, the theoretical analysis in Appendix A, and the empirical results in Section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section 7

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: See Section 4 and Appendix A

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Section 5 and Appendices E and F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: See Supplementary Materials.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: See Appendix F.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: See Table 1 and Appendix F.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: See Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: This research conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss potential positive societal impacts and negative societal impacts of the research.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: No such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: See Appendix F.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not use new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We use the existing common datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We use the existing common datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: We do not use LLM.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.