

# Benchmarking Spiking Neural Network Learning Methods with Varying Locality

Jiaqi Lin, Sen Lu, Malyaban Bal, Abhranil Sengupta

**Abstract**—Spiking Neural Networks (SNNs), providing more realistic neuronal dynamics, have been shown to achieve performance comparable to Artificial Neural Networks (ANNs) in several machine learning tasks. Information is processed as spikes within SNNs in an event-based mechanism that significantly reduces energy consumption. However, training SNNs is challenging due to the non-differentiable nature of the spiking mechanism. Traditional approaches, such as Backpropagation Through Time (BPTT), have shown effectiveness but come with additional computational and memory costs and are biologically implausible. In contrast, recent works propose alternative learning methods with varying degrees of locality, demonstrating success in classification tasks. In this work, we show that these methods share similarities during the training process, while they present a trade-off between biological plausibility and performance. Further, given the implicitly recurrent nature of SNNs, this research investigates the influence of the addition of explicit recurrence to SNNs. We experimentally prove that the addition of explicit recurrent weights enhances the robustness of SNNs. We also investigate the performance of local learning methods under gradient and non-gradient-based adversarial attacks.

**Index Terms**—Spiking Neural Networks, Local Learning, Training Methods, Feedback Alignment, Direct Feedback Alignment, Adversarial Attack, Backdoor Attack.

## I. INTRODUCTION

Spiking Neural Networks (SNNs) are a type of Artificial Neural Networks (ANNs) that are inspired by biological neurons in the brain [1], [2]. In contrast to traditional ANNs, SNNs use a more biologically realistic model of neuron behavior, where information is processed through the propagation of spikes between neurons in the network [3]. The timing and rate of spikes are influenced by the strength of synaptic connections between neurons, which can be modified through a process called synaptic plasticity [4], [5]. In SNNs, neuronal activities are based on discrete spikes, which occur only when the membrane potential of a neuron exceeds a certain threshold. This event-driven processing operates at significantly lower energy levels than traditional computing architectures [6]–[8], as the neuronal and synaptic state updates consume power only when they are actively spiking [9]. Neuromorphic hardware implementations, like Intel’s Loihi [10] and the SpiNNaker project [11], are specifically designed to take advantage of SNNs. The energy efficiency offered by SNNs integrated with neuromorphic hardware provides efficient possibilities for deploying machine learning applications in power-constrained environments, such as mobile devices, wearable technology, and remote sensing systems, among others.

J. Lin, S. Lu, M. Bal and A. Sengupta are with the School of Electrical Engineering and Computer Science, The Pennsylvania State University, University Park, PA 16802, USA. E-mail: jkl6467@psu.edu, sengupta@psu.edu.

Additionally, since membrane potentials are accumulated over time retaining information from previous time steps, SNNs are inherently recurrent [12]. This characteristic of SNNs allows them to naturally process time-dependent data, rendering them suitable for tasks such as speech recognition or time-series prediction [12]–[16]. SNNs with explicit recurrent connections have been shown to further augment their performance in speech processing tasks [13].

Although SNNs are demonstrated to provide multiple benefits from algorithms to hardware by leveraging bio-realistic computations, Backpropagation Through Time (BPTT), which is widely used to train SNNs, is not biologically plausible. Firstly, BPTT assumes weight symmetry for the forward and backward passes, which is undesirable in biological neural networks [17]. Secondly, BPTT requires global error propagation, which is inconsistent with the local learning mechanisms observed in biological neural networks [18], [19]. Moreover, the explicit calculation and back-propagation of gradients and error signals in BPTT are not directly observed in the brain [20], [21]. Lastly, BPTT maintains a history of the membrane potential and spike history of each neuron over time. This process can be memory-intensive and computationally expensive [15], [19], [22]–[24]. To solve this problem, local learning alternatives to BPTT have been proposed for training SNNs [13], [19], [25]–[27]. These methods aim to address the challenges associated with training SNNs while still maintaining biologically plausible learning rules and enabling more efficient and accurate learning in spiking networks.

Based on the locality of training methods [26], we revisit these alternatives to BPTT. Figure 1 distinguishes SNN training methods based on varying levels of locality. Beginning with BPTT, which is a global learning method, it unrolls neural dynamics over time, and it involves backpropagating errors and learning signals through layers. The connection used for backpropagation is the same weight used for forward propagation. An alternative, named feedback alignment, replaces the symmetric weight connections with random matrices during backward passes. This method removes the need for symmetric weights during learning. E-prop introduces a type of direct feedback alignment that provides higher biological plausibility [15]. In this method, the error calculated at the output layer is propagated directly to each layer without propagating errors layer by layer. To obtain better performance, e-prop employs symmetric weights in the process of direct propagation. We next consider a more localized method, DECOLLE, in the regime of local error propagation [19]. In this learning rule, errors are propagated at each layer. A cost function is assigned

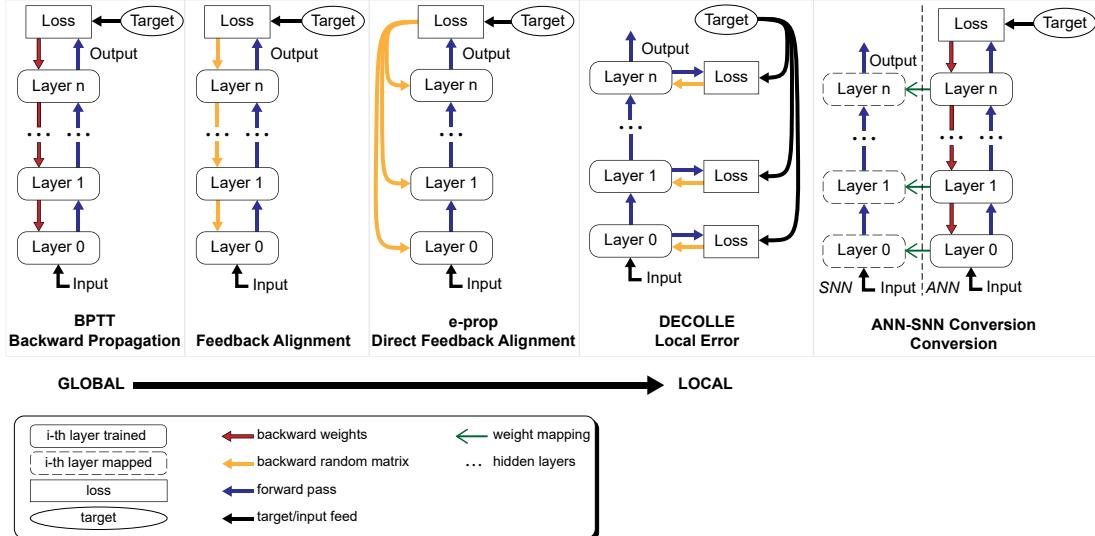


Fig. 1: SNN training methods considered in this study from global learning to local learning: BPTT, Feedback Alignment, e-prop, DECOLLE. ANN-SNN conversion is also depicted on the extreme right.

at each layer by a random matrix mapping the output from each layer to the pseudo-targets. In general, all these learning rules provide bio-plausible alternatives to global learning rules.

## II. RELATED WORKS

### A. Performance Evaluation

Recently, the development of novel learning algorithms to train SNNs has gained traction, propelled by SNNs' unique spiking nature and energy advantage over ANNs. Whether energy efficiency is a compromise of performance remains a question. Ref. [28] explored such a co-design evaluation with respect to ANN performance. Their results indicated that SNNs maintained acceptable accuracy drops on simpler visual tasks like MNIST classification with much better energy efficiency, while in relatively more complicated visual tasks like CIFAR10 classification, the accuracy of SNNs degraded further. Despite this, SNNs evaluated on neuromorphic datasets were illustrated to outperform ANNs. Subsequent progress in SNN training has produced state-of-the-art performance of SNNs in several image recognition benchmarks, with recent works indicating highly similar representations learned by residual SNNs and ANNs using representational similarity measurements [29].

Moreover, the inherently recurrent nature of SNNs resembles the operation of recurrent neural networks (RNNs) in terms of temporal and spatial neural dynamics. Inspired by such characteristics, prior work has compared SNNs and RNNs, highlighting the superior performance of SNNs over RNNs on neuromorphic datasets [30]. Additionally, the work investigated the influence of temporal resolution, leakage, and reset mechanisms on the performance of SNNs, illustrating that the exclusion of leakage and reset mechanisms negatively impacted the performance, and the increment of temporal resolution was not necessarily useful. In these benchmarking

works, SNNs were trained with the BPTT training method, where surrogate functions were used to overcome the non-differentiability of spiking neurons. To quantitatively analyze the impact of surrogate functions on the performance of SNNs, experimental results from previous work have demonstrated that surrogate functions help recurrent SNNs outperform conventional RNNs, and the choice of surrogate functions exhibits negligible impact on performance [14]. Additionally, this work examined the advantage of adaptive neurons [13], where thresholds are updated based on firing rates, and illustrated that recurrent SNNs with adaptive neurons gain performance advantages over SNNs with fixed thresholds. Mixed usage of both types of neurons was shown to improve the performance of SNNs over LSTM architectures [13].

In conclusion, prior works have mainly targeted benchmarking comparable performance of SNNs and ANNs. Some works have investigated the influence of various SNN-specific control knobs like leakage, temporal dynamics, among others, on SNN performance. However, the majority of benchmarking efforts optimize SNNs via the BPTT algorithm, thereby not providing insights related to local learning methods. In contrast to prior works, we perform an extensive benchmarking analysis with a complementary perspective where we investigate the performance of SNNs with regard to local learning methods in terms of accuracy and robustness with both implicit and explicit recurrent architectures.

### B. Adversarial Attacks

The adversarial robustness of SNNs has been proven by various researchers. Sharmin et al. [31] explored gradient-based attacks and their variants on both the ANN-converted SNN and SNN trained from scratch and found SNN's robustness against adversarial attacks. Similarly, Liang et al. [32] conducted untargeted gradient attacks and found that SNNs require more perturbations for successful attacks. Marchisio

et al. [33] considered a deep Spiking Deep Belief Network (SDBN) using noise attacks with the same motive of backdoor trigger attacks [34] and reached similar conclusions. However, their studies were either focused on a single training method or relied entirely on the gradient. While at its exploration stage, many recent SNN training methods look for alternatives to using local training methods [24], [35]. In this work, we aim to bridge this gap by revisiting adversarial attacks on SNNs trained by local learning methods and provide a benchmarking analysis for non-gradient poisoning attacks as well. Although adversarial training strategies strengthen machine learning models against adversarial inputs in the domain of SNNs [36]–[38], these mechanisms are out of the scope of this study.

The primary contributions of this work are the following:

- We extensively benchmark SNNs' performance, concentrating on learning methods such as BPTT, e-prop, and DECOLLE applied to both implicit and explicit recurrent architectures. We highlight the trade-off between biological plausibility and performance, as well as the enhancement of accuracy achieved by incorporating additional weights in explicit recurrent SNNs.
- We perform an in-depth analysis to reveal that training methods with similar locality share representational similarities. Additionally, we find recurrent and linear weights have comparable importance, with each linear layer's significance growing from input to output.
- This study offers insights into the robustness of local learning methods by revisiting adversarial attacks for both gradient-based and non-gradient poisoning attacks. SNNs trained with local learning methods are more robust than those using global learning methods for gradient-based attacks. Analysis shows that explicit recurrent SNNs have greater robustness, supported by the Centered Kernel Alignment (CKA) metric.

### III. METHODS

Sections A-B discuss SNN computational models and learning algorithms considered in this work. Section C analyzes the computational complexity of learning methods. Sections D-E introduce empirical tools used to understand the learning dynamics of various SNN topologies and learning methods.

#### A. Spiking Neurons

In this study, we use SNNs with Leaky-Integrate-and-Fire (LIF) neurons [14], [26]. The membrane potential  $u^{l,t}$  and synaptic currents  $\iota^{l,t}$  for a layer  $l$  at time step  $t$  in a discrete time setting are expressed as:

$$\begin{aligned} \iota^{l,t} &= \exp\left(-\frac{1}{\tau_{\text{syn}}}\right)\iota^{l,t-1} + w^l s^{l-1,t} + v^l s^{l,t-1} \\ u^{l,t} &= \exp\left(-\frac{1}{\tau_{\text{mem}}}\right)u^{l,t-1} + \iota^{l,t} - R\mathcal{V}_{\text{th}} \end{aligned} \quad (1)$$

Here,  $w^l$  represents the weight connection between the current layer  $l$  and the previous layer  $l-1$ , and  $v^l$  represents the recurrent weight connection of the current layer.  $s^{l-1,t}$  denotes the activation of the previous layer  $l-1$  at time step  $t$ .

The parameters  $\exp\left(-\frac{1}{\tau_{\text{syn}}}\right)$  and  $\exp\left(-\frac{1}{\tau_{\text{mem}}}\right)$  correspond to the synaptic and membrane decay rates respectively. Each neuron maintains a membrane potential  $u^{l,t}$ , and if it surpasses the threshold  $\mathcal{V}_{\text{th}}$  of the spiking neuron, a spike will be fired. The membrane potential  $u^{l,t}$  is subtracted by the threshold  $\mathcal{V}_{\text{th}}$  after each firing event, which is captured by the refractory period  $R$ .

#### B. Training Methods

Training a neural network consists of steps to optimize the loss function  $L$  with respect to the parameter set  $w$ . In this section, we will introduce typical training algorithms to train an SNN.

1) *Backpropagation Through Time (BPTT)*: Backpropagation Through Time (BPTT) is an optimization algorithm specifically designed for training Recurrent Neural Networks (RNNs) as an extension of the standard backpropagation algorithm to handle time-dependent sequences and recurrent connections [22], [39]. As SNNs are inherently recurrent in time, training SNNs through BPTT is similar to training RNNs with BPTT. The algorithm functions by unfolding the network through time to capture the temporal dependencies in the input data and applying the backpropagation algorithm over time. The weight update formula for BPTT can be written as follows:

$$\begin{aligned} \Delta w_{ij} &= -\eta \sum_{t=1}^T \frac{\partial L}{\partial s_{ij}^t} \frac{\partial s_{ij}^t}{\partial w_{ij}} = -\eta \sum_{t=1}^T \frac{\partial L}{\partial s_i^t} \frac{\partial s_i^t}{\partial u_i^t} \frac{\partial u_i^t}{\partial w_{ij}} \\ &= -\eta \sum_{t=1}^T \frac{\partial L}{\partial s_i^t} \sigma'(u_i^t) \frac{\partial u_i^t}{\partial w_{ij}} \end{aligned} \quad (2)$$

where,  $T$  is the total number of time steps,  $\eta$  is the learning rate, and  $L$  is the loss function. At time step  $t$ ,  $w_{ij}$  represents the weight connections between neurons  $i$  and  $j$ ,  $s_i^t$  is the activation of neuron  $i$ , and  $\sigma'(u_i^t)$  approximates the derivative of the activation function with respect to the pre-activation value  $u_i^t$  for neuron  $i$  with a surrogate function.

2) *E-prop*: Recent studies of error propagation methods in SNNs propose a new method called e-prop [15]. Compared with BPTT, e-prop offers higher biological plausibility, as it is inspired by the concept of synaptic eligibility traces observed in neuroscience [40], which weigh the temporal differences of presynaptic and postsynaptic neuronal activations. While BPTT relies on a global learning rule that requires error information to be propagated backward through the network between layers, e-prop aims to derive a local learning rule, in which the error signals are propagated directly from the output layer to hidden units. The local learning rule in e-prop makes it more suitable for neuromorphic hardware implementations.

In the e-prop method, the weight updates are based on the product of eligibility traces and learning signals. The eligibility

trace  $e_{ij}^t$  for a synapse connecting neuron  $i$  to neuron  $j$  at time step  $t$  can be computed recursively by eligibility vector  $v$ :

$$\begin{aligned} e_{ij}^t &= \frac{\partial s_i^t}{\partial w_{ij}} = \frac{\partial s_i^t}{\partial u_i^t} v_{ij}^t \\ v_{ij}^t &= \frac{\partial u_i^t}{\partial u_i^{t-1}} v_{ij}^{t-1} + \frac{\partial u_i^t}{\partial w_{ij}} \end{aligned} \quad (3)$$

The eligibility trace captures the recent history of a synapse's contribution to the post-synaptic neuron's activation. The learning signal  $L_i^t$  for neuron  $i$  at time step  $t$ , following direct feedback alignment, routes the error calculated at the output layer to the current neuron:

$$L_i^t = \sum_k g_{ik} \frac{\partial L_k}{\partial s_k^t} \quad (4)$$

where  $g_{ik}$  is a fixed random matrix and  $\partial L_k / \partial s_k^t$  captures the loss calculated at the output layer. The weight update can be expressed as:

$$\Delta w_{ij} = -\eta \sum_{t=1}^T L_i^t \cdot e_{ij}^t \quad (5)$$

3) *DECOLLE*: Deep Continuous Local Learning (DECOLLE) is a biologically plausible online learning algorithm for training SNNs [19]. DECOLLE combines the principles of local learning with deep architectures, enabling the training of SNNs in a layer-by-layer manner without the need for backpropagation or weight transport. In DECOLLE, each layer in the network learns features using local information, which allows the network to adapt online without relying on global error signals. This is achieved by attaching an auxiliary cost function to the random readouts at each layer. The layer-specific random readouts  $y_i^t$  at time  $t$  are calculated as:

$$\begin{aligned} \frac{\partial L_i^t}{\partial s_i^t} &= \frac{\partial L(y_i^t, \hat{y}_i)}{\partial s_i^t} \\ y_i^t &= g_i s_i^t \end{aligned} \quad (6)$$

where  $g_i$  is a fixed random matrix for layer  $i$ . Then, the loss  $L$  is calculated as the sum of layerwise differences between readouts  $y_i^t$  and pseudo-target  $\hat{y}_i$ . Based on this, weight updates at each layer for a specific time step  $t$  are determined using a local learning rule:

$$\begin{aligned} \Delta w_{ij}^t &= -\eta \frac{\partial L_i^t}{\partial w_{ij}} = -\eta \frac{\partial L_i^t}{\partial s_i^t} \frac{\partial s_i^t}{\partial w_{ij}} \\ \frac{\partial s_i^t}{\partial w_{ij}} &= \sigma'(u_i^t) (p_i^t - \rho \frac{\partial r_i^t}{\partial w_{ij}}) \end{aligned} \quad (7)$$

where  $p_i^t$  refers to the traces of the membrane potential of neuron  $i$  driven solely by incoming spikes,  $r_i^t$  with a constant factor  $\rho$  that captures the refractory dynamics depending on the spiking history of the neuron  $i$ . DECOLLE ignores the spiking history dependencies, considering  $r_i^t$  to have a negligible

impact on the membrane potential. Consequently, the synaptic weight updates become:

$$\Delta w_{ij} = -\eta \sum_{t=1}^T \frac{\partial L_i^t}{\partial s_i^t} \sigma'(u_i^t) p_i^t \quad (8)$$

Ignoring refractory dynamics ensures that the learning process relies exclusively on local information, adhering to the principles of biologically plausible learning rules.

### C. Computational Advantages of Local Learning Methods

TABLE I: Complexity analysis of gradient computation for different training methods at a specific layer [19]. Here, we assume neurons between layers are fully connected.  $N_{in}$  is the input size,  $N_{neu}$  represents the number of neurons in current layer,  $T$  is time steps,  $N_{ro}$  indicates the number of readout neurons in DECOLLE. The calculations below does not account for the gradient accumulation overhead in epoch-wise learning.

Method	Space	Time
BPTT	$O(N_{in}T)$	$O(N_{in}N_{neu}T)$
e-prop	$O(N_{in}N_{neu})$	$O(N_{in}N_{neu})$
DECOLLE	$O(1)$	$O(N_{neu}N_{ro} + N_{in}N_{neu})$

Table I provides the computational complexity for the aforementioned training methods. BPTT performs backpropagation on unrolled  $N_{in}$  neuron states across time  $T$ , resulting in a space complexity of  $O(N_{in}T)$  and a time complexity of  $O(N_{in}N_{neu}T)$ , as it multiplies error signals with gradients computed in the current layer with  $N_{neu}$  neurons over  $T$ . The e-prop learning rule maintains  $N_{in}N_{neu}$  number of eligibility traces (Equation 3) in forward manner for each synaptic connection, leading to a space complexity of  $O(N_{in}N_{neu})$  [15]. During the gradient computation, the error signals of size  $N_{neu}$  are multiplied by the eligibility traces (Equation 5) with a time complexity of  $O(N_{in}N_{neu})$ . DECOLLE performs online weight updates at each time step [19], using available neuron states without storing, resulting in a space complexity of  $O(1)$ . During the weight gradient computation, local errors of size  $N_{neu}$  are computed (Equation 6) using  $N_{ro}N_{neu}$  multiplication operations for  $N_{neu}$  neurons. Then these local errors are multiplied by the number of inputs  $N_{in}$ , yielding a total time complexity of  $O(N_{neu}N_{ro} + N_{in}N_{neu})$ .

### D. Fisher information

In statistical modeling, Fisher information measures the quantity of information that a given data sample holds about an unknown parameter that the data sample depends on [41]. Previous work has applied Fisher information to analyze the learning dynamics of SNNs [42].

As SNNs rely on accumulating input data over multiple time steps to predict class probabilities, the amount of Fisher information Matrix (FIM)  $M_t$  accumulated in SNNs at a given

time step is the sum of information across all previous time steps, from 1 to  $t$ .

$$M_t = \mathbb{E}_{x \sim D} \mathbb{E}_{y \sim f_w(y|x_{i \leq t})} [\nabla_w \log f_w(y|x_{i \leq t}) \nabla_w \log f_w(y|x_{i \leq t})^\top] \quad (9)$$

Here,  $x$  is input image sampled from data distribution  $D$ ,  $y$  is the output variable, and  $i \in \{1, \dots, t\}$  represents the index of the time step. Direct calculation on the Fisher information Matrix (FIM) is computationally expensive, due to the large number of parameters in an SNN. Consequently, computing the trace of the FIM is a more efficient alternative [42]. Given a dataset of  $N$  training samples, we can compute the Fisher information at time  $t$  as follows:

$$F_t = \frac{1}{N} \sum_{n=1}^N \|\nabla_w \log f_w(y|x_{i \leq t}^n)\|^2 \quad (10)$$

We will use this metric to quantify the relative importance of recurrent weights in later sections.

#### E. Centered Kernel Alignment (CKA)

CKA is used to quantify the similarity between representations in two arbitrary layers through the normalized Hilbert-Schmidt Independence Criterion (HSIC) [29], [43]. In this work, the CKA calculation is adopted from Li et al. [29].

Let  $\Gamma_\alpha \in \mathbb{R}^{b \times Tp_\alpha}$  and  $\Gamma_\beta \in \mathbb{R}^{b \times Tp_\beta}$  be the representations of layer  $\alpha$  of one SNN with  $p_\alpha$  hidden neurons and layer  $\beta$  of another SNN with  $p_\beta$  hidden neurons over  $T$  time steps, where  $b$  is the batch size. The representations across all time steps are concatenated in SNNs. Then CKA is defined as:

$$CKA(\mathbf{K}, \mathbf{L}) = \frac{HSIC(\mathbf{K}, \mathbf{L})}{\sqrt{HSIC(\mathbf{K}, \mathbf{K})HSIC(\mathbf{L}, \mathbf{L})}}. \quad (11)$$

where,  $\mathbf{K} = \Gamma_\alpha \Gamma_\alpha^\top$  and  $\mathbf{L} = \Gamma_\beta \Gamma_\beta^\top$  are the Gram matrices with shape  $b \times b$  representing the similarity between examples. The HSIC is a statistical measure that assesses the independence of two sets of variables [44]. HSIC is defined as following:

$$HSIC(\mathbf{K}, \mathbf{L}) = \frac{1}{(b-1)^2} \text{tr}(\mathbf{K} \mathbf{C} \mathbf{L} \mathbf{C}^\top) \quad (12)$$

$$\mathbf{C} = \mathbf{J} - \frac{1}{b} \mathbf{O}$$

where,  $\mathbf{J}$  is the identity matrix and  $\mathbf{O}$  is a matrix consisting of all 1's. HSIC score of 0 means independence between two variables. CKA normalizes this into a similarity index from 0 to 1, where higher values imply greater similarity. We adhere to the use of an unbiased estimator for calculating HSIC across mini-batches [29], [45], [46]. CKA metric will also be used in later sections to empirically explain the performance differences of various SNN learning methods in terms of accuracy and robustness.

## IV. RESULTS

### A. Experimental Setup

Experiments conducted in this study include Neuromorphic MNIST (N-MNIST), DVS Gesture, and TIMIT for the investigation of the performance of various training methods.

**N-MNIST:** The N-MNIST dataset is a neuromorphic adaptation of the handwritten digits classification task [47]. In contrast to the MNIST dataset [48], which consists of static images, N-MNIST encapsulates changes in pixel values as events unfold over time. N-MNIST is specifically designed for the evaluation of neuromorphic algorithms and models [19], [49].

**DVS Gesture:** The DVS Gesture dataset [50] contains data from 29 subjects under three lighting conditions. A time step of 60 is used to evaluate performance [30].

**TIMIT:** The TIMIT dataset is a speech and audio processing sequence-to-sequence task [13], [15], [51]. Although N-MNIST, DVS Gesture, and TIMIT datasets all involve the temporal dimension, data in TIMIT exhibit a higher degree of dependency between time steps.

It is worth mentioning here that the e-prop method is currently limited to linear layers [15]. To ensure an equitable comparison between training methods without performance degradation, we therefore restrict our main experiments to fully connected architectures. To assess the generalizability of our conclusions, we further evaluate convolutional architectures trained with BPTT and DECOLLE in Appendix A. In this work, performance and robustness measurements are conducted for two types of SNN architectural designs. One is a feed-forward SNN (FF) and another is a recurrent SNN (REC) architecture, which introduces additional weights in the linear layers in FF. To classify the N-MNIST dataset, both FF and REC SNNs consisting of two hidden layers of 120 and 84 neurons are used. The DVS Gesture dataset is classified by FF and REC SNNs with one hidden layer of 512 neurons. The TIMIT classification task utilizes FF and REC SNNs with one hidden layer of 400 neurons. Notably, due to the configuration of e-prop, the output layer does not contain an explicit recurrent weight connection.

To ensure a fair comparison between the training methods, the hyperparameters of the SNN models trained with different paradigms were optimized for the best performance (Appendix B). The experiments were implemented using the snnTorch and PyTorch libraries in Python and conducted on an Nvidia RTX 2080 Ti GPU with 11GB of memory.

### B. Performance Evaluation

Table II presents the outcomes of previous studies that have benchmarked the performance of SNNs. These results show promising performance of SNNs in several classification tasks in comparison to state-of-the-art ANNs. However, due to the lack of consistency in neural network architecture designs and variations in training sets, direct comparison between different training methods is infeasible. To address this problem, we have conducted a series of benchmark tests with consistent neural network architectures and training datasets. This work aims to offer a more comprehensive and insightful assessment of the performance of local learning methods and the impact of explicit recurrence on SNNs.

In our investigations, we have observed that the introduction of recurrent weights in the REC architecture yields significant

TABLE II: SNN accuracy (%) reported in prior works on the N-MNIST, DVS Gesture, and TIMIT datasets. Diverse architectures and datasets hinder direct comparison of each training method.

Training Methods	N-MNIST		DVS Gesture		TIMIT	
	FF	REC	FF	REC	FF	REC
BPTT	98.3 <sup>a</sup>		87.5 <sup>a</sup>		57.7 <sup>b</sup>	67.1 <sup>c</sup>
e-prop					55.2 <sup>d</sup>	65.4 <sup>e</sup>
DECOLLE	99.0 <sup>f</sup>		95.5 <sup>f</sup>			

<sup>a</sup>This architecture uses one hidden layer with 512 neurons [30].

<sup>b</sup>This architecture uses 3 hidden layers each with 900 neurons in an LSNN architecture [13], which is an LSTM-equivalent architecture in the spiking domain. In general, LSNN has better performance compared to recurrent SNN. The performance is obtained from [15].

<sup>c</sup>This architecture uses one hidden layer with 400 neurons using an LSNN architecture [15].

<sup>d</sup>With 1 hidden layer of 1200 neurons, a feed-forward LSNN trained with e-prop achieves 49.94% accuracy; With 3 hidden layers, each of which consists of 900 neurons, a feed-forward LSNN trained with e-prop achieves 55.2% accuracy [15].

<sup>e</sup>This architecture uses one hidden layer with 400 neurons using an LSNN architecture [15].

<sup>f</sup>This architecture consists of 3 convolutional layers with 64, 128, and 128 channels respectively [19].

TABLE III: SNN accuracy (%) is benchmarked on the N-MNIST, DVS Gesture, and TIMIT datasets, with results averaged across five independent runs. Performance degradation in local learning methods highlights a trade-off between biological plausibility and performance.

Training Methods	N-MNIST		DVS Gesture		TIMIT	
	FF	REC	FF	REC	FF	REC
BPTT	97.12	97.77	87.73	89.23	52.31	56.93
e-prop	96.44	96.80	86.15	87.20	51.12	54.99
DECOLLE	93.61	90.52	84.16	85.17	42.30	45.70

performance improvements when classifying the DVS Gesture and TIMIT datasets, in contrast to the FF architecture. Table III shows SNN performance across various training methods for N-MNIST, DVS Gesture, and TIMIT classification. When trained with BPTT, the REC architecture exhibits a notable enhancement in accuracy when compared to the FF architecture. Similar accuracy improvements are also observed in the case of e-prop and DECOLLE. Benefits of recurrent weights are not evident in the N-MNIST classification task. Here, the performance of SNNs trained with BPTT and e-prop displays only slight differences between the REC module and the FF module. Interestingly, the explicit inclusion of recurrent weights leads to a performance degradation in SNNs trained with DECOLLE for the N-MNIST dataset. Further investigation is required to understand the underlying reason for this degradation.

**Additionally, a more local learning method slightly degrades the performance of SNNs, which indicates a trade-off between biological plausibility and performance.** In this study, the locality of the learning method is considered to increase from BPTT (global learning method) to e-prop and DECOLLE successively. In N-MNIST classification, a

slight accuracy difference is observed between BPTT and e-prop in both REC and FF modules. DECOLLE exhibits an accuracy reduction in both architectures compared with e-prop. A similar trend is observed for DVS Gesture and TIMIT classification tasks. In subsequent sections, we will focus on analyzing the performance differences and robustness between these methods. To provide a case study, we have selected the N-MNIST dataset.

### C. Representational Differences

In the previous section, we observed a trade-off between biological plausibility and performance among various learning methods. Here, we investigate whether this performance discrepancy can be attributed to variations in the learned representations at corresponding layers across different training methods.

CKA is a technique used to measure the similarity between two sets of data, and it has been increasingly applied to understand the internal representation in neural networks. Pioneering work by Cortes et al. introduced CKA as an alignment measure for kernels in the context of support vector machines [52]. Later, Kornblith et al. adapted CKA for deep learning, using it to compare layers across different neural network models, which provided insights into which layers of networks learn similar features [43]. Williams et al. expanded this analysis to study the similarity between representations learned by convolutional neural networks [53]. Li et al. investigated how SNNs differ from traditional ANNs in learning representations, using CKA to analyze both spatial and temporal dimensions [29]. In this study, CKA is employed as a metric to assess the similarity of representations across layers that are learned using various methods with differing degrees of local learning. Consistent with prior works, the CKA is averaged over 4096 instances from the testing set of the N-MNIST dataset [29], [45]. We also compare the similarity of learned representations of the various training methods with ANN-SNN conversion, where the SNN is trained using conventional ANN training methods [54], [55].

**Firstly, significant difference is observed between the representations learned from ANN-SNN conversion and those obtained through other training methods.** Figure 2 plots the normalized HSIC between representations learned at each layer of SNN converted from ANN and that of SNNs trained directly. All the layer-wise similarities between the ANN-SNN conversion method and other direct SNN training methods are low, underscoring the fact that SNNs and ANNs learn very different representations.

**Secondly, aside from ANN-SNN conversion, we observe that methods with a closer level of locality exhibit a substantial degree of representational similarity between corresponding layers.** Figure 2 plots the similarity between the representations of a layer trained by pairs of training methods. In particular, BPTT and e-prop, which have a closer level of locality, show a strong correlation in the representation of their first two layers based on CKA scores. E-prop and DECOLLE pair also exhibit a similar trend. However, the

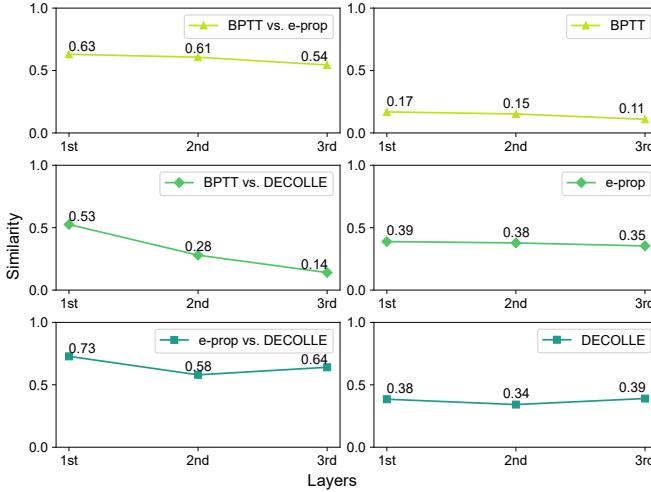


Fig. 2: (Left) Layer-wise similarity among training methods; (Right) Similarity between ANN-SNN conversion method and different direct SNN training methods. The experiments are conducted on the N-MNIST dataset averaged over 4096 images from the test dataset. Significant differences exist between representations from ANN-SNN conversion and other methods. Additionally, training methods with closer locality typically show representational similarity between layers.

TABLE IV: NN accuracy (%) under FGSM attack is evaluated on the N-MNIST dataset. SNNs are trained by methods with varying localities. Model robustness improves with explicit recurrent weights as well as increasing locality of training methods.

Epsilon	0.001	0.005	0.01	0.02	0.05
<i>Training Methods</i>					
<i>FF</i>					
BPTT	83.82	16.78	4.34	1.22	0.73
e-prop	89.90	41.27	12.33	3.27	2.87
DECOLLE	83.48	45.11	28.57	21.29	17.79
<i>Training Methods</i>					
<i>REC</i>					
BPTT	94.66	80.76	58.73	27.69	18.55
e-prop	93.04	78.28	60.57	35.07	18.25
DECOLLE	86.76	84.14	72.81	47.56	23.75

small representational similarity between corresponding layers trained by BPTT and DECOLLE is substantiated by a low CKA score, which addresses the performance gap between the two methods.

#### D. Robustness under Attack

In the previous section, we observed accuracy improvements after the addition of recurrence in spiking architectures, especially for sequential data processing. However, their impact on other aspects of SNN performance, like robustness to adversarial attacks, remains unclear. In this section, we explore this aspect along with empirical tools to explain our observations.

Specifically, the Fast Gradient Sign Method (FGSM) attack leverages the sign of gradients to create adversarial instances that maximize the loss of a neural network, misleading it to make incorrect classifications [56]. However, due to the

non-differentiable nature of SNNs, the direct calculation of gradients is not applicable. To solve this problem, previous work has proposed an effective framework which involves crafting an adversarial example for an SNN by creating its ANN counterpart and then calculating gradients for the corresponding ANN [57]. These gradients are subsequently utilized to construct an adversarial example for the SNN. We adopt this variation of the FGSM attack to compare the robustness of various training methods and to investigate the impact of recurrent weights on robustness.

Then, we employ a targeted backdoor attack, which is a special type of poisoning attack. Such an attack involves the process of injecting a malicious imperceptible trigger during the training phase. Following the trigger mechanism established in previous work [34], we initiate the poisoning of images by modifying 4 pixel locations at all time steps in the input data. For each poison ratio, five random sources and targets are selected. To provide a more robust and unbiased assessment, we calculate the average results from 5 runs with randomly assigned sources and targets, thus mitigating the potential bias introduced by specific poisoning choices. To assess the impact of these backdoor attacks, we examine changes in accuracy and attack success rate (ASR), which is the proportion of labels for the source class that are predicted as the target class. A lower ASR signifies greater robustness.

Table IV reports the performance of different training methods when subjected to FGSM attacks, where the perturbation level, denoted as “Epsilon”, varies across a range from 0.001 to 0.05. The results of our benchmarking analysis for backdoor attacks are illustrated in Table V. We mainly focus on poison rates in realistic settings below 50% as shown in prior works [34], [58]–[60]. Both attacks are evaluated on the N-MNIST dataset.

**Enhancement of the model’s robustness is observed with the inclusion of explicit recurrent weights.** Under FGSM attacks, compared to FF models trained by the same methods and subjected to the same “Epsilon” values, the REC model retains more accuracy than the FF model. Besides, the robustness advantage of explicit recurrence is also observed under backdoor attack. Specifically, under more realistic settings with poison rates at 10% and 30%, we observe that the REC model consistently exhibits a lower ASR compared to its corresponding FF model in all three training methods.

**Local learning methods showed improved robustness under FGSM attacks compared to global learning methods.** Specifically, the local learning method DECOLLE demonstrates enhanced robustness against attacks with high perturbation values. In contrast, the global training methods experience a significant drop in accuracy under high “Epsilon” values, which is consistent in both FF and REC architectures. This advantage can be attributed to the local training methods’ use of individual loss functions for each layer, which enhances robustness in comparison to global loss functions that are vulnerable to globally calculated gradients. **Under backdoor attacks, where global gradients are not used, such robustness gains of local learning methods are not observed.** In

TABLE V: Attack success rate (ASR) of targeted backdoor attack on SNNs along with their accuracy under attack is measured on the N-MNIST dataset with 5 random selections of target and source classes. Local training methods do not demonstrate robustness against backdoor attacks.

Poison Rate	0.1		0.3		0.5		0.7		0.9	
<i>FF</i>										
Training Methods	Accuracy	ASR	Accuracy	ASR	Accuracy	ASR	Accuracy	ASR	Accuracy	ASR
BPTT	96.23%	4.80%	93.34%	33.54%	91.62%	43.76%	89.64%	66.06%	87.47%	78.19%
e-prop	96.85%	1.93%	93.46%	33.35%	92.30%	46.94%	89.38%	75.09%	87.32%	96.95%
DECOLLE	92.74%	4.66%	89.88%	30.04%	88.14%	48.31%	85.12%	76.77%	83.62%	90.80%
<i>REC</i>										
Training Methods	Accuracy	ASR	Accuracy	ASR	Accuracy	ASR	Accuracy	ASR	Accuracy	ASR
BPTT	96.43%	2.64%	94.19%	22.03%	92.24%	43.78%	91.22%	62.24%	87.38%	90.73%
e-prop	96.69%	0.45%	94.81%	17.65%	91.92%	44.56%	88.51%	74.15%	87.23%	93.15%
DECOLLE	90.21%	0.86%	87.46%	23.75%	81.55%	74.10%	81.29%	88.96%	80.81%	91.02%

the previous section, we show that there is a trade-off between biological plausibility and performance. Such a trade-off is not demonstrated for adversarial attacks. Local learning methods retain robustness at par with global learning methods with respect to non-gradient-based attacks and exhibit enhanced robustness for gradient-based attacks.

#### E. CKA Measurements to Explain Robustness

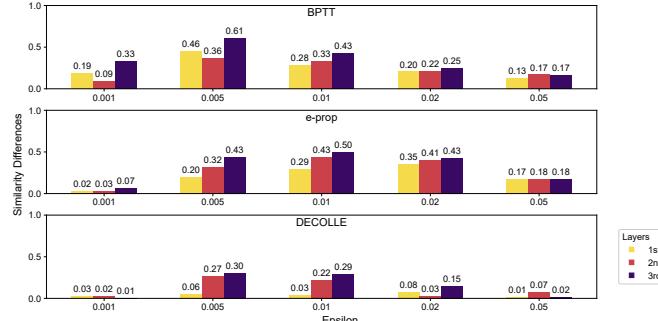


Fig. 3: Layerwise representational similarity between clean and adversarial input data of FF architecture subtracted from the REC architecture for FGSM adversarial attacks. The experiments are conducted on the N-MNIST dataset. Adding recurrent weights improves the robustness leading to higher similarity between clean and adversarial inputs. Robustness improvements due to recurrent weights are more pronounced in global methods.

In this section, we further assess the robustness of SNNs from the perspective of CKA. Prior research has quantified the robustness of SNNs directly trained under adversarial attacks [29] and established a correlation between robustness against adversarial attacks and the degree of representational similarity of a particular network between clean and perturbed input images [61]. Here, we employ the CKA metric to evaluate the robustness of SNNs trained using each training method under FGSM attacks with the same set of “Epsilon” values used in the previous section. Figure 3 measures the layerwise representational similarity between clean and adversarial inputs of REC architecture subtracted by the similarity of FF

architecture under FGSM attacks. Each similarity is computed by contrasting representations of a layer by passing clean data and a layer at the same architectural level by passing adversarial data. Each color corresponds to one specific layer.

**It is observed that the addition of recurrent weights contributes to better preservation of the representations learned at each layer across these training methods.** In these experiments, positive similarity differences are obtained at all perturbation levels, revealing that REC SNNs maintain larger representational similarity (implying more robust representations learned in the presence of attacks) compared to FF SNNs between clean and adversarial data. **The results also reveal that global learning methods exhibit a greater enhancement in robustness with recurrent connections**, as the similarity differences are larger in BPTT than in e-prop and DECOLLE. The trend matches the data provided in Table IV. Through this analysis, we provide evidence that the inclusion of recurrent weights enhances the robustness of neuromorphic architectures.

#### F. Relative Importance of Recurrent Weights

In the previous section, we demonstrated that there is a robustness advantage for explicit recurrent SNNs. Here, we examine the importance of additional recurrence in SNNs from the perspective of Fisher information. A high Fisher information exhibited in weight connections is shown to play a substantial role in making predictions for the given data [62], [63]. Specifically, the Fisher information in SNNs evaluates the accumulation of temporal information during the training process. In previous work [42], normalized layer-wise Fisher information was calculated across time steps, providing an assessment of the relative importance of each layer. We follow the same procedure to calculate Fisher information during the training process by capturing the gradient of weights.

In Figure 4, we present the Fisher information of linear weights in the FF architecture, denoted as LW-FF in the figure. We then compare this Fisher information with both linear and recurrent weights in the REC architecture, denoted as LW-REC and RW-REC in the graph. It is important to note that due to the nature of e-prop, the additional recurrent weights are not added to the last layer in the REC architecture.

**Recurrent weights are demonstrated to have comparable importance with respect to linear weights.** After the addition of explicit recurrent weights, the importance of linear weights (LW-REC) degrades in comparison to linear weights in the FF model (LW-FF). On the other hand, the importance of recurrent weights (RW-REC) shows comparable Fisher information values with regards to linear weights (LW-REC). Based on these two observations, we can conclude that the degradation of importance in linear weights from the values in the FF model (LW-FF) is due to reallocation of importance to linear weights (LW-REC) and recurrent weights (RW-REC) in the REC model since Fisher information is normalized such that the summation of values of all layers is one.

**Additionally, it is found that layerwise increments in the relative importance of linear weights are independent of recurrent weights.** Specifically, an increase of relative importance from input layer to output layer is observed among all training methods. Although the magnitude of this trend may vary across different methods, this trend remains consistent independent of the addition of extra recurrent weights (LW-REC). The magnitude variations mainly come from the emphasis on input layers. In particular, as the locality of learning methods shifts from global to local, the importance of the input layer drops significantly.

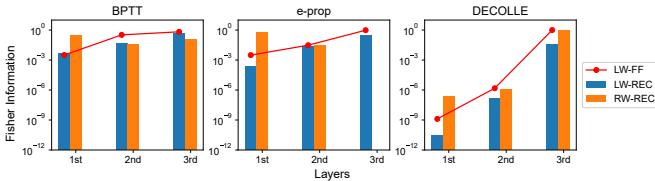


Fig. 4: Layerwise Fisher information measured in SNNs trained by learning methods with varying degrees of locality on the N-MNIST dataset averaged over 5 independent runs. Recurrent weights are as important as linear weights, and layerwise increments in linear weights' importance is independent of recurrent weights.

## V. CONCLUSIONS

Edge devices like smartphones, Internet of things (IoT) gadgets, and embedded systems require improved energy efficiency due to strict memory and computational constraints [64], [65]. Recent advancements in local learning methods enable on-chip learning [66], [67], allowing models to be trained and updated directly on edge devices with significantly reduced energy consumption compared to global learning methods [15], [19], [68]. In this work, these local learning methods are compared in terms of performance and robustness. To understand the performance disparity of these methods, representational differences are examined through the CKA metric. SNN learning methods with a higher degree of locality are shown to have performance degradation, but they show better robustness under FGSM adversarial attacks. However, the robustness advantages of local learning methods are not observed for non-gradient-based attacks like backdoor attacks.

Additionally, we investigate the contribution of additional recurrent weights on the performance of SNNs. Measurements on the relative importance of recurrent weights are performed using Fisher information. We illustrate that recurrent weights not only have positive influences on the performance of SNNs but also augment the robustness of SNNs. These observations are further substantiated by the fact that recurrent weights and linear weights exhibit comparable informative significance. Overall, this work provides valuable insights into the selection of learning methods for SNNs and offers a benchmarking framework for future SNN designers. We acknowledge that our conclusions may currently be limited to feed-forward architectures. Our observations could potentially be generalized to more complex network topologies. Additionally, the integration of more advanced analytical tools could further enhance these insights. Future work could also focus on extending our analysis to convolutional layers [69], [70] and transformer-based SNN models [71]–[73].

## ACKNOWLEDGMENTS

This material is based upon work supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under Award Number #DE-SC0021562, the U.S. National Science Foundation under award No. CCF #1955815, CAREER # 2337646 and EFRI BRAID #2318101 and by Oracle Cloud credits and related resources provided by the Oracle for Research program.

## REFERENCES

- [1] W. Maass, “Networks of spiking neurons: the third generation of neural network models,” *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.
- [2] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.
- [3] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, “Deep learning in spiking neural networks,” *Neural networks*, vol. 111, pp. 47–63, 2019.
- [4] P. U. Diehl and M. Cook, “Unsupervised learning of digit recognition using spike-timing-dependent plasticity,” *Frontiers in computational neuroscience*, vol. 9, p. 99, 2015.
- [5] G.-q. Bi and M.-m. Poo, “Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type,” *Journal of neuroscience*, vol. 18, no. 24, pp. 10 464–10 472, 1998.
- [6] C. Mead, “Neuromorphic electronic systems,” *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.
- [7] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, “A million spiking-neuron integrated circuit with a scalable communication network and interface,” *Science*, vol. 345, no. 6197, pp. 668–673, 2014.
- [8] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, “Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations,” *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, 2014.
- [9] K. Boahen, “Neuromorphic microchips,” *Scientific American*, vol. 16, no. 3, pp. 20–27, 2006.
- [10] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, “Loihi: A neuromorphic manycore processor with on-chip learning,” *Ieee Micro*, vol. 38, no. 1, pp. 82–99, 2018.
- [11] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, “The spinnaker project,” *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.

[12] W. Ponghiran and K. Roy, "Spiking neural networks with improved inherent recurrence dynamics for sequential learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 7, 2022, pp. 8001–8008.

[13] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," *Advances in neural information processing systems*, vol. 31, 2018.

[14] B. Yin, F. Corradi, and S. M. Bohté, "Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks," *Nature Machine Intelligence*, vol. 3, no. 10, pp. 905–913, 2021.

[15] G. Bellec, F. Scherr, A. Subramoney, E. Hajek, D. Salaj, R. Legenstein, and W. Maass, "A solution to the learning dilemma for recurrent networks of spiking neurons," *Nature communications*, vol. 11, no. 1, p. 3625, 2020.

[16] M. Bal and A. Sengupta, "Spikingbert: Distilling bert to train spiking language models using implicit differentiation," *arXiv preprint arXiv:2308.10873*, 2023.

[17] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature communications*, vol. 7, no. 1, p. 13276, 2016.

[18] P. Baldi, P. Sadowski, and Z. Lu, "Learning in the machine: the symmetries of the deep learning channel," *Neural Networks*, vol. 95, pp. 110–133, 2017.

[19] J. Kaiser, H. Mostafa, and E. Neftci, "Synaptic plasticity dynamics for deep continuous local learning (decolle)," *Frontiers in Neuroscience*, vol. 14, p. 424, 2020.

[20] F. Crick, "The recent excitement about neural networks," *Nature*, vol. 337, pp. 129–132, 1989.

[21] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, "Backpropagation and the brain," *Nature Reviews Neuroscience*, vol. 21, no. 6, pp. 335–346, 2020.

[22] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[23] R. J. Williams and D. Zipser, *Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity*. USA: L. Erlbaum Associates Inc., 1995, p. 433–486.

[24] S. Lu and A. Sengupta, "Neuroevolution guided hybrid spiking neural network training," *Frontiers in neuroscience*, vol. 16, p. 838523, 2022.

[25] F. Zenke and S. Ganguli, "Superspike: Supervised learning in multilayer spiking neural networks," *Neural computation*, vol. 30, no. 6, pp. 1514–1541, 2018.

[26] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Processing Magazine*, vol. 36, no. 6, pp. 51–63, 2019.

[27] M. Bal and A. Sengupta, "Sequence learning using equilibrium propagation," *arXiv preprint arXiv:2209.09626*, 2022.

[28] L. Deng, Y. Wu, X. Hu, L. Liang, Y. Ding, G. Li, G. Zhao, P. Li, and Y. Xie, "Rethinking the performance comparison between snns and anns," *Neural networks*, vol. 121, pp. 294–307, 2020.

[29] Y. Li, Y. Kim, H. Park, and P. Panda, "Uncovering the representation of spiking neural networks trained with surrogate gradient," *arXiv preprint arXiv:2304.13098*, 2023.

[30] W. He, Y. Wu, L. Deng, G. Li, H. Wang, Y. Tian, W. Ding, W. Wang, and Y. Xie, "Comparing snns and rnns on neuromorphic vision datasets: Similarities and differences," *Neural Networks*, vol. 132, pp. 108–120, 2020.

[31] S. Sharmin, P. Panda, S. S. Sarwar, C. Lee, W. Ponghiran, and K. Roy, "A comprehensive analysis on adversarial robustness of spiking neural networks," in *2019 IJCNN*. IEEE, 2019, pp. 1–8.

[32] L. Liang, X. Hu, L. Deng, Y. Wu, G. Li, Y. Ding, P. Li, and Y. Xie, "Exploring adversarial attack in spiking neural networks with spike-compatible gradient," *IEEE transactions on neural networks and learning systems*, 2021.

[33] A. Marchisio, G. Nanfa, F. Khalid, M. A. Hanif, M. Martina, and M. Shafique, "Is spiking secure? a comparative study on the security vulnerabilities of spiking and deep neural networks," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[34] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain.(2017)," *arXiv preprint arXiv:1708.06733*, 2019.

[35] A. Bhargava, M. R. Rezaei, and M. Lankarany, "Gradient-free neural network training via synaptic-level reinforcement learning," *Applied Math*, vol. 2, no. 2, pp. 185–195, 2022.

[36] S. Kundu, M. Pedram, and P. A. Beerel, "Hire-snn: Harnessing the inherent robustness of energy-efficient deep spiking neural networks by training with crafted input noise," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5209–5218.

[37] J. Ding, T. Bu, Z. Yu, T. Huang, and J. Liu, "Snn-rat: Robustness-enhanced spiking neural network through regularized adversarial training," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24780–24793, 2022.

[38] O. Özdenizci and R. Legenstein, "Adversarially robust spiking neural networks through conversion," *arXiv preprint arXiv:2311.09266*, 2023.

[39] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[40] W. Gerstner, M. Lehmann, V. Liakoni, D. Corneil, and J. Brea, "Eligibility traces and plasticity on behavioral time scales: Experimental support of neohibbian three-factor learning rules," *Frontiers in Neural Circuits*, vol. 12, 2018. [Online]. Available: <https://www.frontiersin.org/articles/10.3389/fncir.2018.00053>

[41] R. A. Fisher, "Theory of statistical estimation," in *Mathematical proceedings of the Cambridge philosophical society*, vol. 22, no. 5. Cambridge University Press, 1925, pp. 700–725.

[42] Y. Kim, Y. Li, H. Park, Y. Venkatesha, A. Hambitzer, and P. Panda, "Exploring temporal information dynamics in spiking neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 7, 2023, pp. 8308–8316.

[43] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton, "Similarity of neural network representations revisited," in *International conference on machine learning*. PMLR, 2019, pp. 3519–3529.

[44] D. Greenfeld and U. Shalit, "Robust learning with the hilbert-schmidt independence criterion," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3759–3768.

[45] T. Nguyen, M. Raghu, and S. Kornblith, "Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth," *arXiv preprint arXiv:2010.15327*, 2020.

[46] L. Song, A. Smola, A. Gretton, J. Bedo, and K. Borgwardt, "Feature selection via dependence maximization," *Journal of Machine Learning Research*, vol. 13, no. 5, 2012.

[47] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in neuroscience*, vol. 9, p. 437, 2015.

[48] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[49] J. H. Lee, T. Delbrück, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience*, vol. 10, p. 508, 2016.

[50] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza *et al.*, "A low power, fully event-based gesture recognition system," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7243–7252.

[51] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continuous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, p. 27403, 1993.

[52] C. Cortes, M. Mohri, and A. Rostamizadeh, "Algorithms for learning kernels based on centered alignment," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 795–828, 2012.

[53] A. H. Williams, E. Kunz, S. Kornblith, and S. Linderman, "Generalized shape metrics on neural representations," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4738–4750, 2021.

[54] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: Vgg and residual architectures," *Frontiers in neuroscience*, vol. 13, p. 95, 2019.

[55] S. Lu and A. Sengupta, "Exploring the connection between binary and spiking neural networks," *Frontiers in neuroscience*, vol. 14, 2020.

[56] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[57] S. Sharmin, P. Panda, S. S. Sarwar, C. Lee, W. Ponghiran, and K. Roy, "A comprehensive analysis on adversarial robustness of spiking neural

networks,” in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[58] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, “Targeted backdoor attacks on deep learning systems using data poisoning,” *arXiv preprint arXiv:1712.05526*, 2017.

[59] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, “Turning your weakness into a strength: Watermarking deep neural networks by backdooring,” in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 1615–1631.

[60] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, “Strip: A defence against trojan attacks on deep neural networks,” in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 113–125.

[61] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards deep learning models resistant to adversarial attacks,” *arXiv preprint arXiv:1706.06083*, 2017.

[62] A. Achille, M. Rovere, and S. Soatto, “Critical learning periods in deep networks,” in *International Conference on Learning Representations*, 2018.

[63] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[64] E. Covi, E. Donati, X. Liang, D. Kappel, H. Heidari, M. Payvand, and W. Wang, “Adaptive extreme edge computing for wearable devices,” *Frontiers in Neuroscience*, vol. 15, p. 611300, 2021.

[65] T. Gao, B. Deng, J. Wang, and G. Yi, “Presynaptic spike-driven plasticity based on eligibility trace for on-chip learning system,” *Frontiers in Neuroscience*, vol. 17, p. 1107089, 2023.

[66] C. Pehle, S. Billaudelle, B. Cramer, J. Kaiser, K. Schreiber, Y. Stradmann, J. Weis, A. Leibfried, E. Müller, and J. Schemmel, “The brainscales-2 accelerated neuromorphic system with hybrid plasticity,” *Frontiers in Neuroscience*, vol. 16, p. 795876, 2022.

[67] C. Frenkel and G. Indiveri, “Reckon: A 28nm sub-mm2 task-agnostic spiking recurrent neural network processor enabling on-chip learning over second-long timescales,” in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65. IEEE, 2022, pp. 1–3.

[68] C. Liu, G. Bellec, B. Vogginger, D. Kappel, J. Partzsch, F. Neumärker, S. Höppner, W. Maass, S. B. Furber, R. Legenstein *et al.*, “Memory-efficient deep learning on a spinnaker 2 prototype,” *Frontiers in neuroscience*, vol. 12, p. 840, 2018.

[69] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian, “Deep residual learning in spiking neural networks,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 21056–21069, 2021.

[70] Y. Hu, H. Tang, and G. Pan, “Spiking deep residual networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 8, pp. 5200–5205, 2021.

[71] Z. Zhou, Y. Zhu, C. He, Y. Wang, S. Yan, Y. Tian, and L. Yuan, “Spikformer: When spiking neural network meets transformer,” *arXiv preprint arXiv:2209.15425*, 2022.

[72] J. Zhang, B. Dong, H. Zhang, J. Ding, F. Heide, B. Yin, and X. Yang, “Spiking transformers for event-based single object tracking,” in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2022, pp. 8801–8810.

[73] M. Bal and A. Sengupta, “Spikingbert: Distilling bert to train spiking language models using implicit differentiation,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 38, no. 10, 2024, pp. 10998–11006.

## APPENDIX A SNN ACCURACY IN CONVOLUTIONAL ARCHITECTURES

In this section, the DVS Gesture dataset is classified by FF and REC SNNs consisting of 3 convolutional layers with 64, 128, and 128 channels respectively (each with a kernel size of 7, a stride of 1, and a padding of 1). The observations (Table VI) are consistent with those in Section IV-B.

TABLE VI: SNN accuracy (%) is benchmarked on the DVS Gesture dataset, with results averaged across five independent runs. The observations align with prior findings.

Training Methods	DVS Gesture	
	FF	REC
BPTT	91.74	93.04
DECOLLE	90.15	91.28

## APPENDIX B HYPERPARAMETER CONFIGURATION

Table VII summarizes the hyper-parameters for optimal performance achieved by each method, where LR represents learning rate and BS represents batch size.

TABLE VII: Hyper-parameters for optimal performance on N-MNIST, DVS Gesture, and TIMIT datasets.

Model	$e^{\frac{-1}{\tau_{\text{syn}}}}$	$e^{\frac{-1}{\tau_{\text{mem}}}}$	$\mathcal{V}_{\text{th}}$	LR	BS	Epochs
<b>N-MNIST</b>						
BP FF	0.9	0.5	0.9	1e-4	16	100
e-prop FF	0.99	0.95	0.2	5e-3	5	100
DECOLLE FF	0.97	0.92	1.0	1e-3	72	100
BP REC	0.9	0.5	0.9	1e-2	256	100
e-prop REC	0.99	0.95	0.2	5e-3	4	100
DECOLLE REC	0.97	0.92	1.0	1e-5	72	100
<b>DVS-Gesture</b>						
BP FF	0.9	0.5	1.0	1e-3	16	100
e-prop FF	0.95	0.65	0.3	1e-3	15	100
DECOLLE FF	0.9	0.65	0.9	2e-4	72	100
BP REC	0.95	0.5	0.9	1e-3	32	100
e-prop REC	0.95	0.6	0.7	3e-3	15	100
DECOLLE REC	0.05	0.2	1.0	3e-5	72	100
<b>TIMIT</b>						
BP FF	0.9	0.5	1.0	1e-3	8	100
e-prop FF	0.99	0.3	1.0	5e-4	4	100
DECOLLE FF	0.97	0.9	0.9	2e-4	128	100
BP REC	0.9	0.7	1.0	1e-3	8	100
e-prop REC	0.99	0.3	1.0	5e-4	4	100
DECOLLE REC	0.97	0.3	1.0	1e-5	32	100