ENABLING PROBABILISTIC INFERENCE ON LARGE-SCALE SPIKING NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep spiking neural networks (SNNs) have demonstrated success in many machine learning tasks. However, most previous studies focused on deterministic spiking neurons, neglecting the inherent noisy features of neurons, which have also been shown to improve generalization ability and robustness. In this work, we propose a novel SNN framework called Noisy Spiking Neural Network (NSNN) based on the Noisy LIF neuron. By modeling NSNN as a Bayesian Network, we derive a three-factor learning rule called noise-driven learning (NDL) for NSNN synaptic optimization. The post-synaptic factor in NDL is calculated using the neuronal membrane noise statistics, providing an insightful interpretation for surrogate gradients from the perspective of random noise. Evaluations on CIFAR-10/100, DVS-CIFAR, and DVS-Gesture show that the NSNN framework leads to competitive SNN models. Furthermore, NSNNs exhibit higher robustness against challenging perturbations, including adversarial attacks[1].

1 INTRODUCTION

Spiking Neural Networks (SNNs) (Maass, 1997) have received mounting interest for their high biological plausibility and low power consumption. Recent works introduce deep learning methods to SNNs and use large-scale neural network architectures, which are proven to have superior representation abilities (Simonyan & Zisserman, 2014; Szegedy et al., 2015), and thus achieved success on many tasks (Lee et al., 2016; Wu et al., 2018; Deng et al., 2021; Zhang et al., 2021). Nevertheless, most existing studies consider deterministic SNNs (DSNNs), which ignore the inherent randomness of spiking neurons. Using the neuron model with a noisy dynamic is an effective way to introduce stochasticity into SNNs. This method has two advantages: First, it incurs a potential benefit in generalization performance by encouraging the model to learn a representation space that is more fault-tolerant (Liu et al., 2020b; Camuto et al., 2020; Lim et al., 2021) and preventing overfitting (Bengio et al., 2013; Hinton et al., 2012). Second, spiking neurons with noise-perturbed dynamics are more biologically realistic because ion channel fluctuations and synaptic transmission randomness give rise to noisy sub-threshold membrane voltages (Verveen & DeFelice, 1974; Kempter et al., 1998; Stein et al., 2005; Faisal et al., 2008). Existing related research, however, was limited to small scales (Plesser & Gerstner, 2000; Deneve, 2008; Pecevski et al., 2011); while instructive, they have low scalability and are difficult to scale to larger architectures.

Contributions This work introduces Noisy Spiking Neural Network (NSNN), which enables probabilistic inference on large SNNs and provides a general theoretical framework for investigating spiking neural models from the perspective of random noise. To be specific, we (1) build the NSNN upon the discrete Noisy LIF neuron to form a general framework for SNNs; (2) derive a novel three-factor learning rule called noise-driven learning (NDL) for NSNN synaptic optimization by interpreting NSNN as a Bayesian Network; (3) show a mathematical relationship between surrogate gradient learning and NDL, providing an insightful interpretation for surrogate gradients; (4) show that the NSNN framework leads to competitive SNN models, demonstrated by experiments on CIFAR-10/100, DVS-CIFAR and DVS-Gesture datasets; (5) demonstrate that NSNN framework leads to more robust SNN models when facing challenging perturbations (including adversarial attacks). (6) By NSNN-based neural code analysis, we demonstrate the potential of NSNN as a neural coding framework for computational neuroscience.

[1]Codes are available at `https://cutt.ly/9CxT5jI`

Notations We adopt x, u, o to represent neuron input, membrane potential and neuron output, respectively. Moreover, $x_{l,m}^t, u_{l,m}^t, o_{l,m}^t$ for variables of neuron m in layer l (whose dimension is $\dim(l)$) at time t, where $m \in [1, \dim(l)], l \in [1, L]$ and $t \in [1, T]$. We also use boldface type $\mathbf{x}, \mathbf{u}, \mathbf{o}$ to denote the sets of variables, *e.g.*, variables of layer l at timestep t are marked as $\mathbf{x}_l^t, \mathbf{u}_l^t, \mathbf{o}_l^t$. $\mathbb{E}[\cdot]$, $\mathbb{P}[\cdot]$, $p(\cdot)$ and $F(\cdot)$ are, respectively, expectation, probability, probability distribution and CDF.

2 RELATED WORKS

Surrogate Gradient Learning The main obstacle during the direct-training of prevailing deterministic SNNs is the almost everywhere zero nature of the gradient of the Heaviside firing function. As a remedy, *surrogate gradient function* (SG) (Neftci et al., 2019; Zenke & Vogels, 2021) are adopted, *i.e.* use a smooth function to replace the derivative of the firing function in the backward pass and still use the firing function in the forward passage. Surrogate gradient learning (SGL) refers to synaptic optimization using surrogate gradients.

Noisy spiking neural models Gerstein & Mandelbrot proposed the earliest integrate-and-fire (IF) neuron model with stochastic activity. Following developments (Stein, 1965; Tuckwell, 1989; Plesser & Gerstner, 2000; Di Maio et al., 2004; Burkitt, 2006) have expanded on the diffusion approach by employing stochastic differential equations. Rao demonstrated that recurrent networks of noisy IF neurons could perform approximate Bayesian inference of dynamic graphical models. Patel & Kosko introduced the conditions for the noise benefit (Wiesenfeld & Moss, 1995) of additive white noises. Fiete & Seung proposed an estimator that correlates reinforcement reward signal and synaptic perturbations by introducing white noises and adopting first-order Taylor expansions for noisy neuron learning. In Bengio et al. (2013) a locally-computed gradient estimator for neurons with stochastic decisions is introduced. Skatchkovsky et al. described a Generalized Linear Model variant of the deterministic Spike Response Model.

LIF neuron model The widely-used LIF neuron model includes the following discrete-time dynamics

$$\begin{aligned} \text{sub-threshold dynamic: } & u^t = \tau u^{t-1} + \phi_\theta(x^t), \\ \text{threshold-based firing: } & o^t = \text{spike}(u^t, v_{th}) \triangleq \text{Heaviside}(u^t - v_{th}), \\ \text{resetting: } & u^t = u^t \cdot (1 - o^t) + u_{reset}, \end{aligned} \tag{1}$$

where x^t is the input at time t, τ is the membrane time constant. ϕ_θ denotes a parameterized input transform, and v_{th} is the firing threshold. To introduce a simple model of neuronal spiking and refractoriness, we assume $v_{th} = 1$, $\tau = 0.5$ and $u_{reset} = 0$ throughout this research.

3 NOISY SPIKING NEURAL NETWORKS

We begin by introducing the Noisy LIF model as a fundamental unit, which naturally connects spiking networks to probabilistic graphical models and enables NSNN to function as a general theoretical framework for LIF SNNs.

Noisy LIF model The Noisy LIF presented here is based on previous works that use diffusion approximation (Plesser & Gerstner, 2000; Burkitt, 2006), in which the effective current input to the neuron is described by a deterministic part and a random noise part. As a result, an additive noise term is added to the discrete sub-threshold dynamic:

$$\text{sub-threshold dynamic: } u^t = \tau u^{t-1} + \phi_\theta(x^t) + \epsilon, \tag{2}$$

where ϵ are independently drawn from a known distribution and assumed to satisfy $\mathbb{E}[\epsilon] = 0$ and $\epsilon = -\epsilon$. As an example, we use Gaussian $\epsilon \sim \mathcal{N}$ here. Expression (2) can also be obtained by discretizing an Itô SDE variant of LIF's ODE form (Patel & Kosko, 2005; 2008).

The membrane potentials and spike outputs become random variables as a result of the injection of random noises. Using noise as a medium, we obtain the probability distribution of Noisy LIF firings

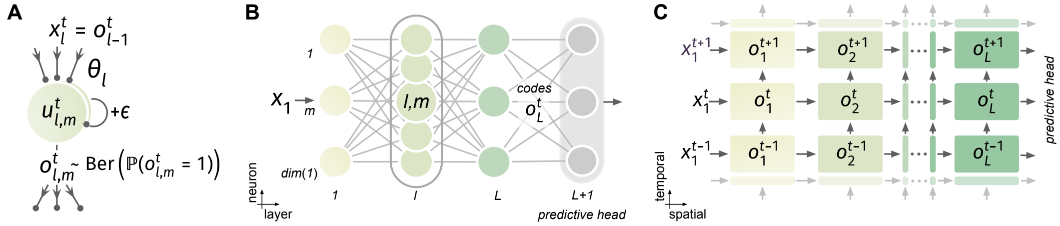


Figure 1: Graphical illustration of the NSNN model. **A**. The diagram of a Noisy LIF neuron. **B**. The sketch of an NSNN example. The final neural codes $\mathbf{o}_L^t$ are drawn from $p_\theta(\mathbf{o}_L^t|x_1^t, \mathbf{o}_{1\ldots L}^t)$, which is obtained through sampling-based probabilistic inference. The predictive head then decodes the codes $\mathbf{o}_L^t$ to produce predictions for specific tasks. **C**. NSNN as a Bayesian Network, which is a representation of the joint probability distribution of a set of random variables with causal relationships.

based on the threshold firing mechanism, as $\epsilon = -\epsilon$, we have

$$o^t = \begin{cases} 1, \text{w.p. } \mathbb{P}[o^t = 1] = \mathbb{P}[\underbrace{u^t + \epsilon > v_{th}}_{\text{threshold-based firing}}] = \mathbb{P}[\epsilon < u^t - v_{th}] = F_\epsilon(u^t - v_{th}), \\ 0, \text{w.p. } 1 - \mathbb{P}[o^t = 1]. \end{cases} \tag{3}$$

The expressions above show how a single neuron encodes for a spike state random variable (Maass, 2014), allowing us to formulate the probabilistic firing of Noisy LIF by

$$\text{probabilistic firing: } o^t \sim \text{Ber}(\mathbb{P}[o^t = 1]), \text{where } \mathbb{P}[o^t = 1] = F_\epsilon(u^t - v_{th}). \tag{4}$$

Specifically, it relates to previous literature in which the difference $u - v_{th}$ governs the neuron firing probabilities (Maass, 1995; Plesser & Gerstner, 2000). In addition, Noisy LIF employs the same resetting mechanism as the LIF model.

Noisy LIF is a general form of spiking neurons, making NSNN a theoretical framework for SNNs. If $\text{Var}[\epsilon] \to 0$, F_ϵ will approach the Heaviside step function; hence, the Noisy LIF model covers the deterministic LIF case. Further, if we consider $\epsilon \sim \text{Logistic}$, the Noisy LIF describes a sigmoidal neuron (Maass, 2014).

Noisy SNN Let x_1^t denote the input at t-th timestep, using the dynamics of Noisy LIF in (2,4), an NSNN with $L+1$ layers is given by

$$\begin{aligned} &\text{layer 1: } \mathbf{x}_1^t = x_1^t, \mathbf{u}_1^t = \tau \mathbf{u}_1^t + \phi_{\theta_1}(\mathbf{x}_1^t) + \vec{\epsilon},\ \mathbf{o}_1^t = \left\{o_{1,m}^t \sim \text{Ber}(\mathbb{P}[o_{1,m}^t = 1])\right\}_{m=1}^{\dim(1)} \\ &\text{layer } 2 \cdots L\text{: } \mathbf{x}_l^t = \mathbf{o}_{l-1}^t, \mathbf{u}_l^t = \tau \mathbf{u}_l^t + \phi_{\theta_l}(\mathbf{x}_l^t) + \vec{\epsilon},\ \mathbf{o}_l^t = \left\{o_{l,m}^t \sim \text{Ber}(\mathbb{P}[o_{l,m}^t = 1])\right\}_{m=1}^{\dim(l)}, \\ &\text{predictive head: } \mathcal{L} = f_{\theta_{L+1}}(\mathbf{o}_L^t) = f(\phi_{\theta_{L+1}}(\mathbf{o}_L^t)). \end{aligned} \tag{5}$$

The spike output $\mathbf{o}_l^t$ of layer l is a representation vector in $\mathbb{S}^{\dim(l)}$, where we denote the spike state space as $\mathbb{S} = \{0, 1\}$. The noise vector $\vec{\epsilon}$ consists of independent random noise with a known distribution (Gaussian here).

The predictive head $f_{\theta_{L+1}}(\mathbf{o}_L^t)$ includes a mapping $\phi_{\theta_{L+1}}(\mathbf{o}_L^t)$ and a loss function f, denoting the part that decodes predictions from the neural representation $\mathbf{o}_L^t$ and compute the loss value. ϕ_{θ_l} represents a map, such as fully-connected or convolution and is thus differentiable w.r.t. parameter θ_l. Also, dividing the synaptic parameters by layers, as mentioned above, results in no loss of generality as they can be defined as any differentiable mapping.

For example, to solve classification problems we shall consider the predictive probability model $p_{\theta_{L+1}}(y|\mathbf{o}_L^t) = \text{softmax}(\phi_{\theta_{L+1}}(\mathbf{o}_L^t))$, where the map $\phi_{\theta_{L+1}}$ computes the predictive scores using the neural representation $\mathbf{o}_L^t$. The function f can be the cross-entropy of the predictive distribution $p_{\theta_{L+1}}(y|\mathbf{o}_L^t)$ and the target distribution $p_{tar}(y|x_1^t)$. Note that $f_{\theta_{L+1}}(\mathbf{o}_L^t)$ here computes the instantaneous loss, different from the $\frac{1}{T}\sum_t f^t$, which is computed over the entire time window and ignores potential online learning (Xiao et al., 2022).

Since each neuron codes for a random variable $o_{l,m}^t$, we can describe the NSNN by the Bayesian Network model and represent the joint distribution of all spike states given the input x_1^t as

$$p_\theta(\mathbf{o}_{1\dots L}^t|x_1^t,\mathbf{o}_{1\dots L}^{t-1}) = p_{\theta_1}(\mathbf{o}_1^t|x_1^t,\mathbf{o}_1^{t-1})\prod_{l=2}^{L} p_{\theta_l}(\mathbf{o}_l^t|\mathbf{o}_{l-1}^t,\mathbf{o}_l^{t-1}), \tag{6}$$

where $p_{\theta_l}(\mathbf{o}_l^t|\mathbf{o}_{l-1}^t,\mathbf{o}_l^{t-1}) = \prod_{m=1}^{\dim(l)} p_{\theta_l}(o_{l,m}^t|\mathbf{o}_{l-1}^t,o_{l,m}^{t-1})$.

3.1 A Noise-driven Learning Rule Induced by Noise Injection

It is suggested that noise supports learning from supervise signals of networks of spiking neurons, rather than being a nuisance (Maass, 2014), and previous ANN literature also suggest similar stances (Liu et al., 2020b; Camuto et al., 2020; Lim et al., 2021). But how exactly is this achieved on spiking neurons? Within the NSNN framework, we derive a novel noise-driven learning rule (Fig. 2.A) induced by membrane noise injection for synaptic optimization.

To perform NSNN synaptic optimization, the central problem is to estimate the gradient of the expected loss:

$$g_l = \nabla_{\theta_l}\sum_{\mathbf{o}_{1\dots L}^t} p_\theta(\mathbf{o}_{1\dots L}^t|x_1^t,\mathbf{o}_{1\dots L}^{t-1}) f_{\theta_{L+1}}(\mathbf{o}_L^t). \tag{7}$$

As (7) is intractable to compute, we expect an estimation so that the parameters can be tuned using gradient-based routines.

The dimensionality of the spike state space is rather limited (either spike or silence). Based on this property, we can derive an estimator by conditioning (local marginalization), which performs exact summation over single random variable to reduce variance (Burt Jr & Garman, 1971; Titsias et al., 2015). We first factorize the joint distribution $p_\theta(\mathbf{o}_{1\dots L}^t|x_1^t,\mathbf{o}_{1\dots L}^{t-1})$ as $\big(\prod_{i\neq l} p_{\theta_i}(\mathbf{o}_i^t|\mathbf{o}_{i-1}^t,\mathbf{o}_i^{t-1})\prod_{k\neq m} p_{\theta_l}(o_{l,k}^t|\mathbf{o}_{l-1}^t,o_{l,k}^{t-1})\big)p_{\theta_l}(o_{l,m}^t|\mathbf{o}_{l-1}^t,o_{l,m}^{t-1})$. Hence, (7) becomes

$$g_l = \sum_{\mathbf{o}_{1\dots L}^t}\sum_m \Big(\prod_{i\neq l} p_{\theta_i}(\mathbf{o}_i^t|\mathbf{o}_{i-1}^t,\mathbf{o}_i^{t-1})\prod_{k\neq m} p_{\theta_l}(o_{l,k}^t|\mathbf{o}_{l-1}^t,o_{l,k}^{t-1})\Big)\nabla_{\theta_l} p_{\theta_l}(o_{l,m}^t|\mathbf{o}_{l-1}^t,o_{l,m}^{t-1}) f_{\theta_{L+1}}(\mathbf{o}_L^t). \tag{8}$$

Since $\mathbb{P}[o_{l,m}^t=0] = 1-\mathbb{P}[o_{l,m}^t=1]$, we have

$$\sum_{o_{l,m}^t}\nabla_{\theta_l} p_{\theta_l}(o_{l,m}^t|\mathbf{o}_{l-1}^t,o_{l,m}^{t-1}) f_{\theta_{L+1}}(\mathbf{o}_L^t) = \nabla_{\theta_l} p_{\theta_l}(o_{l,m}^t|\mathbf{o}_{l-1}^t,o_{l,m}^{t-1})\left(f_{\theta_{L+1}}(\mathbf{o}_L^t) - f_{\theta_{L+1}}(\mathbf{o}_{l,\bar{m}}^t)\right), \tag{9}$$

where we use $\mathbf{o}_{l,\bar{m}}^t$ to denote the new state $\mathbf{o}_L^t$ if $o_{l,m}$ alters. Together with $\sum_{o_{l,m}^t} p_{\theta_l}(o_{l,m}^t) = 1$ and (8, 9), we have

$$g_l = \sum_{\mathbf{o}_{1\dots L}^t}\Big(\prod_{i=1}^{L} p_{\theta_i}(\mathbf{o}_i^t|\mathbf{o}_{i-1}^t,\mathbf{o}_i^{t-1})\Big)\hat{g}_l = \mathbb{E}_{\mathbf{o}_{1\dots L}^t}[\hat{g}_l]\text{, where}$$
$$\hat{g}_l = \sum_m \nabla_{\theta_l} p_{\theta_l}(o_{l,m}^t|\mathbf{o}_{l-1}^t,o_{l,m}^{t-1})\Big(f_{\theta_{L+1}}(\mathbf{o}_L^t) - f_{\theta_{L+1}}(\mathbf{o}_{l,\bar{m}}^t)\Big). \tag{10}$$

To get an estimate of g_l, we can simply sample from $p_\theta(\mathbf{o}^t)$ and calculate using (10). However, it is unwise to compute $f_{\theta_{L+1}}(\mathbf{o}_L^t) - f_{\theta_{L+1}}(\mathbf{o}_{l,\bar{m}}^t)$, as it requires a lot of additional computations, and thus cannot scale to large models. Inspired by Fiete & Seung (2006), we may attribute the change of the loss to the state flip of variable $o_{l,m}^t$. By doing this, we can approximate the change of the loss when the state of $o_{l,m}^t$ alters using a first-order approximation:

$$f_{\theta_{L+1}}(\mathbf{o}_L^t) - f_{\theta_{L+1}}(\mathbf{o}_{l,\bar{m}}^t) \approx \big(o_{l,m}^t - (1-o_{l,m}^t)\big)\frac{\partial f_{\theta_{L+1}}}{\partial o_{l,m}^t} = (2o_{l,m}^t-1)\frac{\partial f_{\theta_{L+1}}}{\partial o_{l,m}^t}. \tag{11}$$

Note that, this approximation introduce bias to the gradient estimator, except when the map f is multilinear (Tokui & Sato, 2017). Substituting (11) into (10), we obtain

$$\hat{g}_l = \sum_m \nabla_{\theta_l} p_{\theta_l}(o_{l,m}^t|\mathbf{o}_{l-1}^t,o_{l,m}^{t-1})(2o_{l,m}^t-1)\frac{\partial f_{\theta_{L+1}}}{\partial o_{l,m}^t}. \tag{12}$$

Proposition 1. *For a Noisy LIF neuron (l,m) in an NSNN, where $l\in[1,L], m\in[1,\dim(l)]$, we have* $\nabla_{\theta_l} p_{\theta_l}(o_{l,m}^t|\mathbf{o}_{l-1}^t,o_{l,m}^{t-1}) = (2o_{l,m}^t-1)F'_\epsilon(u_{l,m}^t - v_{th})\nabla_{\theta_l} u_{l,m}^t$.

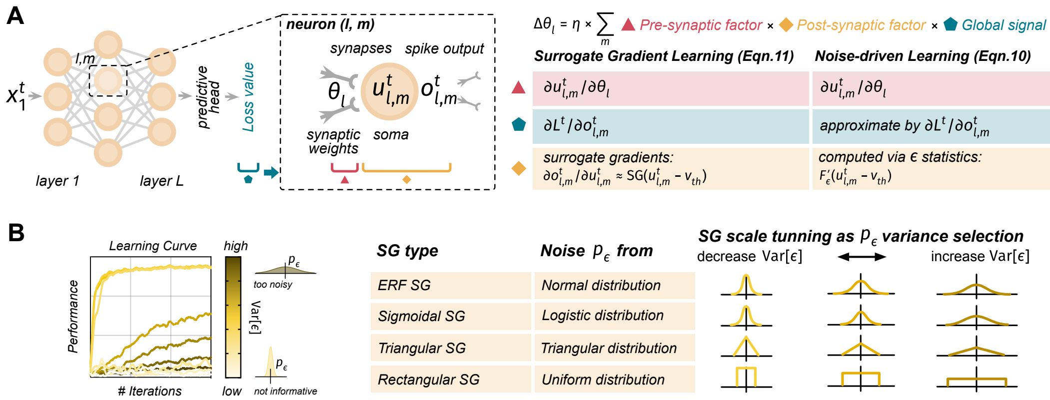


Figure 2: **A**: Illustration of surrogate gradient and noise-driven learning rules. **B**: Right: Relationship between SGL and NDL, where we regard the scale in SGL as the variance of noise in NDL. Left: Learning efficiencies under different $\mathrm{Var}[\epsilon]$ values, results are obtained by training a 64-hidden-unit MLP NSNN on MNIST.

Proof. Proved using (3).

Combining Proposition 1 and (12), we formulate the noise-driven learning rule (Fig. 2.A) as

$$\hat{g}_l^{NDL} = \sum_m \nabla_{\theta_l} u_{l,m}^t F'(u_{l,m}^t - v_{th}) \nabla_{o_{l,m}^t} f_{\theta_{L+1}}. \tag{13}$$

When using (13), there is not need to calculate an additional gradient generator in the forward pass, and $\hat{g}_l$ can be computed layer by layer in a single backward passage. As a result, NDL is easy to implement and can mesh well with modern frameworks of automatic differentiation. For neuron (l, m), the gradient estimation is performed by a backward pass, in which the post-synaptic factor is acquired from the PDF F'_ϵ. Plesser & Gerstner (2000); Shrestha & Orchard (2018) also constructed surrogate gradient function by empirically adding infinitesimal gaussian perturbations to a spiking neuron. However, these works analyze an isolated neuron, whereas results of this work are derived from the network level. Since the estimator in (13) is backpropagation-compatible, we can easily optimize NSNNs of any architecture with the BPTT algorithm (Robinson & Fallside, 1987).

Relationship to the Surrogate Gradient Learning. In the Surrogate Gradient Learning (SGL), the derivative of neuron firing function $\partial o / \partial u$ is replaced by a smooth function SG to mesh with the backpropagation scheme. SGL calculates the gradient g_l by

$$\text{SGL: } \sum_m \frac{\partial u_{l,m}^t}{\partial \theta_l} \underbrace{\mathrm{SG}(u_{l,m}^t - v_{th})}_{\text{approximate } \partial o/\partial u} \frac{\partial f_{\theta_{L+1}}}{\partial o_{l,m}^t}. \tag{14}$$

Eqn. (14) and (13) show a close mathematical relationship between NDL and SGL. The derivative of firing function, provided by surrogate gradient functions in SGL, corresponds to the membrane potential noise's PDF $F'_\epsilon = p_\epsilon$ of the post-synaptic neuron. Indeed, when we extend the Gaussian noise in to general stochastic processes with static increments, commonly-used symmetric (subject to the assumptions we used in the derivation) surrogate gradients can be explained by corresponding PDFs of membrane potential noises (*e.g.*, rectangular SG *v.s.* uniform noise, sigmoidal SG *v.s.* logistic noise).

Biological interpretation: noise as a resource for learning. We re-write the NGL estimator in (13) to frame it as a three-factor learning rule (Frémaux & Gerstner, 2016; Gerstner et al., 2018):

$$\hat{g}_l^{NDL} = \sum_m \underbrace{\partial u_{l,m}^t / \partial \theta_l}_{\text{Pre-synaptic factor}} \underbrace{F'_\epsilon(u_{l,m}^t - v_{th})}_{\text{Post-synaptic factor}} \underbrace{\partial f_{\theta_{L+1}} / \partial o_{l,m}^t}_{\text{Global learning signal}}. \tag{15}$$

The post-synaptic factor in NDL is calculated by the probability density function of the post synaptic neuron's membrane potential noise, which computationally validates the idea of "*noise as a resource*

Table 1: Undisturbed classification task performances in accuracy (%), T for simulation timesteps.

	NSNN	Algorithm	Architecture	Accuracy($T=2$)	Accuracy($T=4$)
CIFAR-10	○	STCA (Gu et al.)	CIFARNet	91.23($T=12$)	
	○	STBP-tdBN (Zheng et al.)	ResNet-19	92.34	92.92
	○	STBP (Wu et al.)	ResNet-18*	93.18±0.07	93.93±0.11
	●	STBP	ResNet-18*	92.87±0.04	93.77±0.12
	○	STBP	CIFARNet	91.88±0.09	92.79±0.14
	●	STBP	CIFARNet	93.90±0.12	94.30±0.08
	○	TET	ResNet-18*	93.62±0.02	94.09±0.20
	●	TET	ResNet-18*	93.12±0.07	94.14±0.05
CIFAR-100	○	TET (Deng et al.)	ResNet-19	72.87±0.10	74.47±0.15
	○	STBP-tdBN (Zheng et al.)	ResNet-19	69.41±0.08	70.86±0.22
	○	STBP	ResNet-18*	70.15±0.14	70.88±0.19
	●	STBP	ResNet-18*	69.57±0.09	71.16±0.40
	○	STBP	CIFARNet	72.25±0.08	72.94±0.21
	●	STBP	CIFARNet	73.36±0.14	74.17±0.28
	○	TET	ResNet-18*	71.72±0.13	74.01±0.43
	●	TET	ResNet-18*	71.34±0.09	73.33±0.03
DVS-CIFAR				Accuracy($T=10$)	
	○	Fang et al.	Wide-7B-Net	74.4($T=16$)	
	○	Wu et al.	LIAFNet	71.70	
	○	STBP	ResNet-19	71.74±0.92	
	●	STBP	ResNet-19	74.30±0.61	
	○	STBP-tdBN	VGGSNN	75.51±0.49	
	●	STBP-tdBN	VGGSNN	76.97±0.10	
	○	TET	VGGSNN	78.26±0.17	
	●	TET	VGGSNN	79.52±0.38	
DVS-GESTURE				Accuracy($T=16$)	
	○	Fang et al.	7B-Net	97.92	
	○	STBP-tdBN(Zheng et al.)	ResNet-17	96.87($T=40$)	
	○	STBP	7B-Net	95.84±0.27	
	●	STBP	7B-Net	96.88±0.28	

*: modified from the original architecture (He et al., 2016), refer to Tab. 5.

for learning" (Maass, 2014). Also, in NDL, adjusting the noise variance causes a change in the shape of its PDF, which corresponds to tuning the scale parameter for the surrogate gradient function in SGL (Fig. 2.B). Thus, the post-synaptic factor of NDL explains the scale tunning (Zenke & Vogels, 2021) in SGL. The scale tunning of SGs can be viewed as variance selection of membrane noise ϵ: a mild noise plays an essential role in learning (Maass, 2014). Proper variance is essential to achieve high performance: small variance noise (low entropy) is not informative enough to learn well, while high variance noise is also harmful (Fig. 2.B) (Yarom & Hounsgaard, 2011). In Sec. 4.3, we investigate the effect of noise level on performance in further detail.

4 Experiments

In this section, we demonstrate that the NSNN framework leads to competitive and more robust SNN models. We focus on the internal randomness in NSNNs and study the effects of membrane noise level on performance. In addition, we offer novel insights on the role of task type in neural coding through NSNN-based neural code analyses, demonstrating how NSNNs can be used as a promising tool for computational neuroscience.

We adopt various network architectures including residual nets and VGG nets. For DSNNs, we use the ERF surrogate gradient $\mathrm{SG}_{\mathrm{ERF}}(x) = \frac{1}{\sqrt{\pi}} \exp(-x^2)$ for SGL. All networks were trained using Adam solvers with the cosine annealing learning rate scheduler.

4.1 Comparison of Recognition Task Performance

In this section, we compare the capabilities of NSNNs and DSNNs on static image benchmarks CIFAR-10/100 (Krizhevsky et al., 2009), dynamic datasets DVS-CIFAR (Li et al., 2017), and DVS-Gesture (Amir et al., 2017). The results are reported as mean±std across three independent runs. More experimental details are provided in Sec. A.2 and more results are presented in Tab. 8. We set

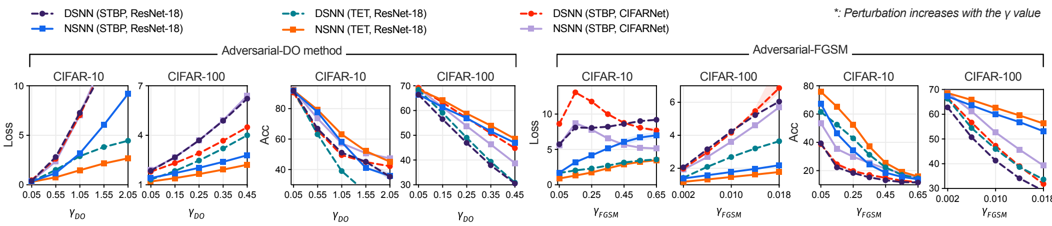


Figure 3: Evaluation results under adversarial attacks on CIFAR datasets. NSNNs exhibit stronger resilience under adversarial attacks.

Table 2: Evaluation results under *EventDrop* perturbations on the DVS-CIFAR. Parameter ρ controls the strength of perturbations (larger for stronger perturbation).

		Loss				**Accuracy**			
Algo. & Arch.	Type \ ρ	0.05	0.25	0.45	0.65	0.05	0.25	0.45	0.65
STBP & ResNet-19	DSNN	$2.27_{\pm 0.16}$	$6.89_{\pm 2.05}$	$8.60_{\pm 1.55}$	$9.06_{\pm 1.08}$	$60.09_{\pm 2.46}$	$17.68_{\pm 5.72}$	$13.21_{\pm 1.31}$	$12.42_{\pm 0.29}$
	NSNN	$1.80_{\pm 0.09}$	$5.84_{\pm 0.76}$	$7.65_{\pm 1.21}$	$8.55_{\pm 1.40}$	$65.66_{\pm 1.80}$	$25.31_{\pm 5.72}$	$16.36_{\pm 3.23}$	$13.32_{\pm 0.61}$
tdBN & VGGSNN	DSNN	$2.24_{\pm 0.14}$	$6.31_{\pm 0.74}$	$8.25_{\pm 1.49}$	$9.49_{\pm 1.61}$	$64.98_{\pm 1.63}$	$26.64_{\pm 3.32}$	$18.41_{\pm 2.43}$	$13.74_{\pm 1.00}$
	NSNN	$1.90_{\pm 0.06}$	$6.91_{\pm 0.16}$	$8.19_{\pm 0.82}$	$8.66_{\pm 1.35}$	$70.28_{\pm 1.36}$	$30.14_{\pm 0.99}$	$22.55_{\pm 2.07}$	$18.78_{\pm 1.97}$
TET & VGGSNN	DSNN	$1.21_{\pm 0.01}$	$2.88_{\pm 0.31}$	$3.44_{\pm 0.31}$	$4.13_{\pm 0.57}$	$67.86_{\pm 0.43}$	$29.26_{\pm 4.34}$	$20.76_{\pm 2.45}$	$15.70_{\pm 2.80}$
	NSNN	$1.03_{\pm 0.05}$	$2.55_{\pm 0.25}$	$4.02_{\pm 0.29}$	$4.15_{\pm 0.18}$	$71.67_{\pm 1.20}$	$29.14_{\pm 1.16}$	$21.34_{\pm 0.73}$	$14.73_{\pm 0.29}$

the standard deviation of membrane noise to 0.3 for CIFAR-10/100, DVS-Gesture experiments and 0.2 for DVS-CIFAR. These configurations offer a fair balance between performance and resilience (refer to Fig. 2 and Sec. 4.3).

According to results presented in Table 1. When compared to their deterministic counterparts, it can be seen that NSNNs with different combinations of training algorithms and network architectures achieve competitive performances. Specifically, our NSNNs show consistent merits for the event-stream classification task on the DVS-CIFAR dataset. We suggest that the intrinsic randomness of the Noisy LIF neurons plays the role of a regularizer (Camuto et al., 2020; Lim et al., 2021), thus alleviating the overfitting to some extent. Evaluations in this part demonstrate the benefit of NSNNs: NSNNs can perform stochastic inference on large-scale architectures while achieving comparable or better performance than those deterministic inference ones.

4.2 Robustness Evaluation

We further evaluate the robustness of DSNNs and NSNNs on CIFAR-10/100 and DVS-CIFAR datasets. The default simulation timestep for static image datasets is $T = 2$. The models we used for evaluation in this section are trained as described in Section 4.1. We consider different perturbations for static, dynamic inputs, respectively. For CIFAR10/100, we consider untargeted adversarial attack to evaluate the model robustness under the "worst case" (Szegedy et al., 2013; Guo et al., 2022). We construct adversarial examples by two methods (details in A.2.1): *(1) Direct Optimization (DO)* method and *(2) Fast Gradient Sign method* (FGSM, Goodfellow et al.). For DVS-CIFAR, we consider the *EventDrop* perturbation (Gu et al., 2021), whose basic idea is to randomly drop a proportion of events, with a probability of $\rho \in [0, 1]$. In addition, the evaluation under hidden state-level (neuronal spike-level) perturbations are presented in Section A.2.2.

Figure 3 shows the performance dynamics on CIFAR-10/100 datasets against DO/FGSM adversarial attacks. Our results indicate that NSNNs are highly resilient to these challenging adversarial perturbations, whereas DSNNs' reliability degrades radically. Table 2 summarizes the losses and accuracies of three groups of models concerning input-level *EventDrop* perturbations on the DVS-CIFAR dataset. In most cases, the proposed NSNNs appear to be less sensitive to perturbations than competitors, demonstrating relatively high robustness to various perturbations and adaptability to multiple training algorithms and network architectures.

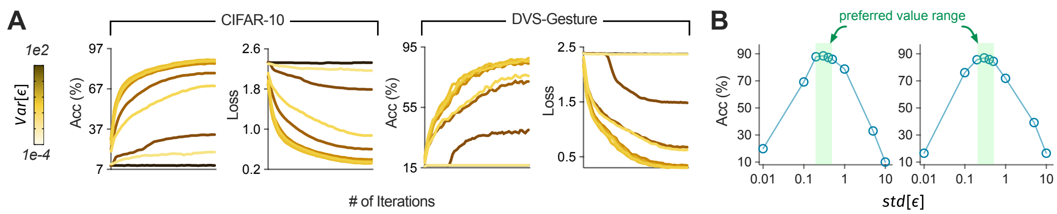


Figure 4: Effect of internal noise level on performance. **A**. Learning curves of NSNNs under different noise levels, we use color to distinguish different noise levels. **B**. The relationship between final test accuracy and the standard deviation of membrane potential noise ϵ. The preferred value range is $[0.2, 0.5]$.

4.3 Effect of Internal Noise Level on Performance

We further explore the effect of the membrane potential noise level in NSNN on the performance as an extension to the related content in Figure 2.B. We run experiments using the CIFAR-10 and DVS-Gesture datasets and train identical networks with different standard deviation settings for 60 epochs. Results are presented by learning curves and the accuracy-standard deviation curves in Fig. 4. As the variance of membrane potential noise ϵ increases, the model performance exhibits a dynamic process of increasing and then declining. In particular, NSNNs achieve high performance near a moderate value (Fig. 4.B), confirming our intuition that moderate noise is essential for high performance. As shown in Fig. 4.B, changes in $\mathrm{std}[\epsilon]$ within a "moderate noise" range (from 0.2 to 0.5) have no significant effect on final performance. This gives us a range of internal noise levels to choose from when using NSNNs in practice.

Biological interpretation of the effect of noise level. As a critical component in NDL, the post-synaptic factor $F'_\epsilon(u - v_{th})$ is calculated by the PDF of membrane noise ϵ during the backward pass. When the noise variance is very small, the noise distribution converges to a Dirac distribution with minimal information (as measured by entropy), and the post-synaptic factor cannot obtain enough information for synaptic optimization. In the case of inference, the noise level directly affects the randomness of the neuron firing distribution. A high variance noise would disrupt the flow of valuable information from the observation in the network, causing NSNN's performance to deteriorate greatly.

4.4 NSNN Neural Code Analysis

The intrinsic randomness of NSNNs results in trial-by-trial variability (Stein et al., 2005), allowing for exploration of neural representations in spiking networks. In this section, we analyze the neural code embedded in the spike trains in NSNNs. Also, we consider an additional sinusoidal series forecasting (SSF) task to investigate possible coding strategies of NSNNs when performing different types of tasks (refer to A.2.3 for details). We estimate the possibility of rate code by measuring the correlation between the neural code (outputs of the penultimate layer) variation and prediction stability. We use the Fano factor (FF) to numerically measure the neural code variation and cosine similarity to assess prediction stability. In addition to the firing rate, it has been suggested that correlations between neurons provide an additional channel of information (Alonso et al., 1996; Hung et al., 2005). In this section, we use simplified network architectures with 16 neurons in the last (L-th) spiking layer for the DVS-Gesture and the SSF experiments to enable pairwise firing correlation analyses. The settings for CIFAR-10/100 and DVS-CIFAR experiments are the same as those in Section 4.1. The simulation timesteps of CIFAR-10/100, DVS-CIFAR, DVS-Gesture, and SSF experiments are 2, 10, 16, and 48, respectively.

The results in Figure 5.B show a decreasing monotonic trend between the prediction similarity and the neural code variation (measured by avg. FF). The average FF and prediction similarity, in particular, on some experiments (*e.g.*, TET+ResNet18), show a strong negative correlation, indicating that these NSNNs are likely to primarily adopt rate code. It makes sense as the membrane noise injection introduces uncertainty into the firing process, lowering the reliability of the precise spiking time-based coding. As the same firing rate (represented as firing count in simulation steps here)

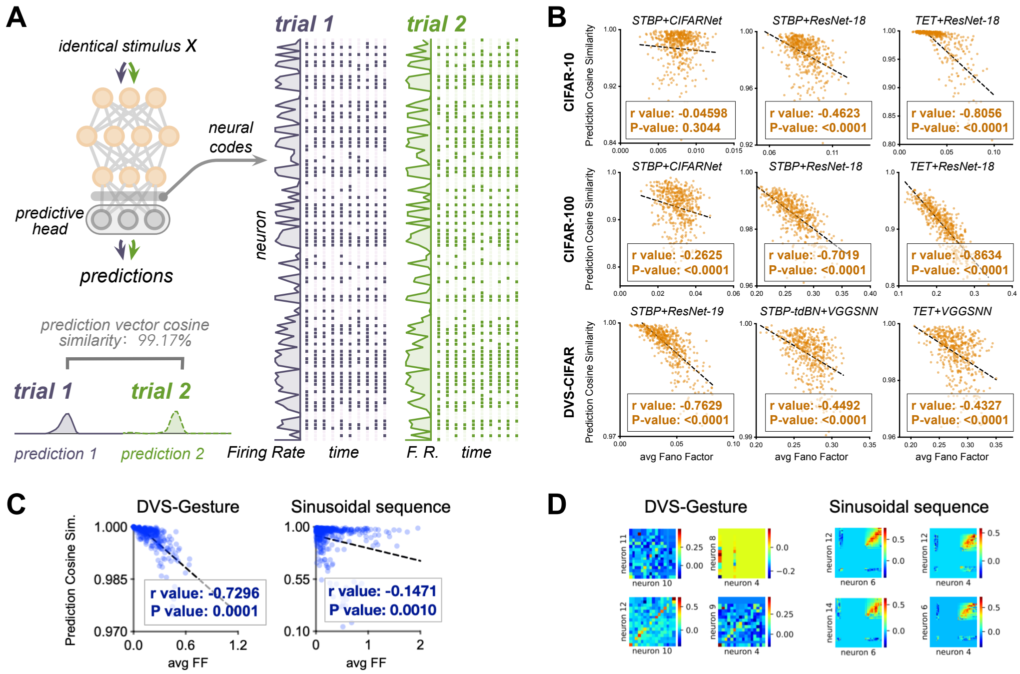


Figure 5: **A**: NSNNs exhibit neural code variability and prediction stability. We display prediction distributions, firing rate and raster plots of the final spiking layer outputs of two repeated trials obtained using an NSNN trained on DVS-CIFAR. **B,C**: The average FF and prediction cosine similarity exhibit a decreasing monotonic trend. The dots represent 500 test samples (200 for DVS-Gesture), the dotted straight line is obtained via linear approximation. The negative correlation coefficient r indicates that one variable tends to decrease when the other one increases. The P value < 0.05 indicates that the result is unlikely to be the outcome of chance. **D**: Part of normalized JPSTH plots generated in the DVS-Gesture recognition and sinusoidal sequence forecasting tasks. The main diagonal of the normalized JPSTH displays for each timestep the Pearson correlation of the two neuron firing simultaneously.

can correspond to different spike trains, the rate-based coding can improve the model's robustness by constructing a representation space with better fault tolerance. Figure 5.C shows that when performing forecasting tasks, NSNN appears to be less dependent on the rate code. By measuring the pairwise firing correlation (Fig. 5.D, full version in Fig. 7,8), we also discover that a neuron population exhibits significant co-activation, which was not observed in the DVS-Gesture experiment. Therefore, NSNN may also utilize the firing correlations to carry important information when performing forecasting tasks, implying that the optimal neural code (neural representation) might be task-dependent (Bredenberg et al., 2020; Xie et al., 2022).

5 Conclusion

We introduce NSNN in this work. Based on its Bayesian Network form, we propose a novel three-factor learning rule called noise-driven learning (NDL), which offers an insightful probabilistic interpretation of the surrogate gradient learning. We demonstrate NSNN's capability through experiments on various recognition tasks. Moreover, we conduct experiments with challenging perturbations (such as adversarial attacks) and demonstrate that NSNNs are more robust than their deterministic counterparts. We investigate the effect of NSNN internal noise level on performance and give a recommended range of standard deviation values (only consider Gaussian noise here). In addition, we demonstrate the potential of NSNN as a neural coding scheme through NSNN-based neural code analysis.

REFERENCES

Jose-Manuel Alonso, W Martin Usrey, and R Clay Reid. Precisely correlated firing in cells of the lateral geniculate nucleus. *Nature*, 383(6603):815–819, 1996.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7243–7252, 2017.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv*, 2013.

Colin Bredenberg, Eero Simoncelli, and Cristina Savin. Learning efficient task-dependent representations with synaptic plasticity. In *Advances in Neural Information Processing Systems*, volume 33, pp. 15714–15724, 2020.

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

John M Burt Jr and Mark B Garman. Conditional monte carlo: A simulation technique for stochastic network analysis. *Management Science*, 18(3):207–217, 1971.

Alexander Camuto, Matthew Willetts, Umut Simsekli, Stephen J Roberts, and Chris C Holmes. Explicit regularisation in gaussian noise injections. In *Advances in Neural Information Processing Systems*, 2020.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

Sophie Deneve. Bayesian spiking neurons i: inference. *Neural Computation*, 20(1):91–117, 2008.

Shikuang Deng, Yuhang Li, Shanghang Zhang, and Shi Gu. Temporal efficient training of spiking neural network via gradient re-weighting. In *International Conference on Learning Representations*, 2021.

V Di Maio, P Lansky, and R Rodriguez. Different types of noise in leaky integrate-and-fire model of neuronal dynamics with discrete periodical input. *General Physiology and Biophysics*, 23:21–38, 2004.

A Aldo Faisal, Luc PJ Selen, and Daniel M Wolpert. Noise in the nervous system. *Nature Reviews Neuroscience*, 9(4):292–303, 2008.

Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep Residual Learning in Spiking Neural Networks. In *Advances in Neural Information Processing Systems*, 2021a.

Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. In *Advances in Neural Information Processing Systems*, 2021b.

Ugo Fano. Ionization yield of radiations. ii. the fluctuations of the number of ions. *Physical Review*, 72(1):26, 1947.

Ila R Fiete and H Sebastian Seung. Gradient learning in spiking neural networks by dynamic perturbation of conductances. *Physical Review Letters*, 97(4):048104, 2006.

Nicolas Frémaux and Wulfram Gerstner. Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Frontiers in Neural Circuits*, 9:85, 2016.

George L Gerstein and Benoit Mandelbrot. Random walk models for the spike activity of a single neuron. *Biophysical Journal*, 4(1):41–68, 1964.

Wulfram Gerstner, Marco Lehmann, Vasiliki Liakoni, Dane Corneil, and Johanni Brea. Eligibility traces and plasticity on behavioral time scales: experimental support of neohebbian three-factor learning rules. *Frontiers in Neural Circuits*, 12:53, 2018.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

Fuqiang Gu, Weicong Sng, Xuke Hu, and Fangwen Yu. Eventdrop: Data augmentation for event-based learning. *arXiv*, 2021.

Pengjie Gu, Rong Xiao, Gang Pan, and Huajin Tang. Stca: Spatio-temporal credit assignment with delayed feedback in deep spiking neural networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019.

Chong Guo, Michael J Lee, Guillaume Leclerc, Joel Dapello, Yug Rao, Aleksander Madry, and James J DiCarlo. Adversarially trained neural representations may already be as robust as corresponding biological neural representations. *arXiv*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv*, 2012.

Yangfan Hu, Huajin Tang, and Gang Pan. Spiking deep residual networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2018.

Chou P Hung, Gabriel Kreiman, Tomaso Poggio, and James J DiCarlo. Fast readout of object identity from macaque inferior temporal cortex. *Science*, 310(5749):863–866, 2005.

Nikola Kasabov. To spike or not to spike: A probabilistic spiking neuron model. *Neural Networks*, 23(1):16–19, 2010.

Richard Kempter, Wulfram Gerstner, J Leo Van Hemmen, and Hermann Wagner. Extracting oscillations: Neuronal coincidence detection with noisy periodic spike input. *Neural computation*, 10 (8):1987–2017, 1998.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv*, 2014.

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

Alexander Kugele, Thomas Pfeil, Michael Pfeiffer, and Elisabetta Chicca. Efficient processing of spatio-temporal data streams with spiking neural networks. *Frontiers in Neuroscience*, 14:439, 2020.

Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 10, 2016.

Mario Lezcano-Casado. Trivializations for gradient-based optimization on manifolds. In *Advances in Neural Information Processing Systems*, pp. 9154–9164, 2019.

Hongmin Li, Hanchao Liu, Xiangyang Ji, Guoqi Li, and Luping Shi. Cifar10-dvs: an event-stream dataset for object classification. *Frontiers in neuroscience*, 11:309, 2017.

Soon Hoe Lim, N Benjamin Erichson, Liam Hodgkinson, and Michael W Mahoney. Noisy recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2021.

Qianhui Liu, Haibo Ruan, Dong Xing, Huajin Tang, and Gang Pan. Effective aer object classification using segmented probability-maximization learning in spiking neural networks. In *AAAI Conference on Artificial Intelligence*, 2020a.

Xuanqing Liu, Tesi Xiao, Si Si, Qin Cao, Sanjiv Kumar, and Cho-Jui Hsieh. How does noise help robustness? explanation and exploration under the neural sde framework. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 282–290, 2020b.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv*, 2016.

Wolfgang Maass. On the computational power of noisy spiking neurons. In *Advances in Neural Information Processing Systems*, 1995.

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

Wolfgang Maass. Noise as a resource for computation and learning in networks of spiking neurons. *Proceedings of the IEEE*, 102(5):860–880, 2014.

Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

Ashok Patel and Bart Kosko. Stochastic resonance in noisy spiking retinal and sensory neuron models. *Neural Networks*, 18(5-6):467–478, 2005.

Ashok Patel and Bart Kosko. Stochastic resonance in continuous and spiking neuron models with levy noise. *IEEE Transactions on Neural Networks*, 19(12):1993–2008, 2008.

Dejan Pecevski, Lars Buesing, and Wolfgang Maass. Probabilistic inference in general graphical models through sampling in stochastic networks of spiking neurons. *PLoS Computational Biology*, 7(12):e1002294, 2011.

Hans E Plesser and Wulfram Gerstner. Noise in integrate-and-fire neurons: from stochastic input to escape rates. *Neural Computation*, 12(2):367–384, 2000.

Bharath Ramesh, Hong Yang, Garrick Orchard, Ngoc Anh Le Thi, Shihao Zhang, and Cheng Xiang. Dart: distribution aware retinal transform for event-based cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(11):2767–2780, 2019.

Rajesh P Rao. Hierarchical bayesian inference in networks of spiking neurons. In *Advances in Neural Information Processing Systems*, volume 17, 2004.

Nitin Rathi and Kaushik Roy. Diet-snn: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. *arXiv*, 2020.

Nitin Rathi, Gopalakrishnan Srinivasan, Priyadarshini Panda, and Kaushik Roy. Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation. In *International Conference on Learning Representations*, 2019.

AJ Robinson and Frank Fallside. *The utility driven dynamic error propagation network*. University of Cambridge Department of Engineering Cambridge, 1987.

Ali Samadzadeh, Fatemeh Sadat Tabatabaei Far, Ali Javadi, Ahmad Nickabadi, and Morteza Haghir Chehreghani. Convolutional spiking neural networks for spatio-temporal feature extraction. *arXiv*, 2020.

Sumit B Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. Hats: Histograms of averaged time surfaces for robust event-based object classification. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1731–1740, 2018.

Nicolas Skatchkovsky, Hyeryung Jang, and Osvaldo Simeone. Spiking neural networks—part ii: Detecting spatio-temporal patterns. *IEEE Communications Letters*, 25(6):1741–1745, 2021.

Richard B Stein. A theoretical analysis of neuronal variability. *Biophysical Journal*, 5(2):173–194, 1965.

Richard B Stein, E Roderich Gossen, and Kelvin E Jones. Neuronal variability: noise or part of the signal? *Nature Reviews Neuroscience*, 6(5):389–397, 2005.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv*, 2013.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

Michalis K. Titsias, Miguel Lázaro-Gredilla, et al. Local expectation gradients for black box variational inference. In *Advances in Neural Information Processing Systems*, 2015.

Seiya Tokui and Issei Sato. Evaluating the variance of likelihood-ratio gradient estimators. In *International Conference on Machine Learning*, pp. 3414–3423. PMLR, 2017.

Henry C Tuckwell. *Stochastic Processes in the Neurosciences*. SIAM, 1989.

Henry C Tuckwell, Jürgen Jost, and Boris S Gutkin. Inhibition and modulation of rhythmic neuronal spiking by noise. *Physical Review E*, 80(3):031907, 2009.

AA Verveen and LJ DeFelice. Membrane noise. *Progress in Biophysics and Molecular Biology*, 28: 189–265, 1974.

Kurt Wiesenfeld and Frank Moss. Stochastic resonance and the benefits of noise: from ice ages to crayfish and squids. *Nature*, 373(6509):33–36, 1995.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 12:1–12, 2018.

Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, Yuan Xie, and Luping Shi. Direct training for spiking neural networks: Faster, larger, better. In *AAAI Conference on Artificial Intelligence*, 2019.

Zhenzhi Wu, Hehui Zhang, Yihan Lin, Guoqi Li, Meng Wang, and Ye Tang. Liaf-net: Leaky integrate and analog fire network for lightweight and efficient spatiotemporal information processing. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Mingqing Xiao, Qingyan Meng, Zongpeng Zhang, Di He, and Zhouchen Lin. Online training through time for spiking neural networks. In *Advances in Neural Information Processing Systems*, 2022.

Marjorie Xie, Samuel Muscinelli, Kameron Decker Harris, and Ashok Litwin-Kumar. Task-dependent optimal representations for cerebellar learning. *bioRxiv*, 2022.

Yosef Yarom and Jorn Hounsgaard. Voltage fluctuations in neurons: signal or noise? *Physiological Reviews*, 91(3):917–929, 2011.

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.

Malu Zhang, Hong Qu, Xiurui Xie, and Jürgen Kurths. Supervised learning in spiking neural networks with noise-threshold. *Neurocomputing*, 219:333–349, 2017.

Shao-Qun Zhang, Zhao-Yu Zhang, and Zhi-Hua Zhou. Bifurcation spiking neural network. *Journal of Maching Learning Research*, 22:253–1, 2021.

Wenrui Zhang and Peng Li. Temporal spike sequence learning via backpropagation for deep spiking neural networks. In *Advances in Neural Information Processing Systems*, 2020.

Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. In *AAAI Conference on Artificial Intelligence*, volume 35, pp. 11062–11070, 2021.

A Appendix

Roadmap For additional experimental details and results, refer to A.2.

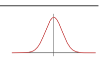
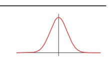
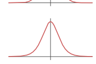
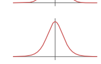
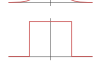
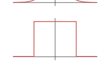
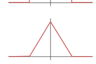
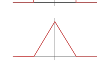
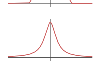
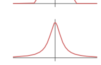
Notations We adopt lower case letters x, u, o to represent neuron input, membrane potential and neuron output respectively. Moreover, $x^t_{l,m}, u^t_{l,m}, o^t_{l,m}$ for variables of neuron m in layer l (whose dimension is $\dim(l)$) at time t, where $m \in [1, \dim(l)], l \in [1, L]$ and $t \in [1, T]$. Similarly, variables of layer l at timestep t are marked as x^t_l, u^t_l, o^t_l. We also use boldface type $\mathbf{x}, \mathbf{u}, \mathbf{o}$ to denote the sets of all variables of the network. $\mathbb{E}[\cdot]$ stands for expectation, $\mathrm{Var}[\cdot]$ for variance, $\mathbb{P}[\cdot]$ for probability, $p(\cdot)$ for probability distribution and $F(\cdot)$ for CDF. Notation $\mathbb{R}$ denotes real number space $\mathbb{R} \triangleq (-\infty, +\infty)$ and $\mathbb{S}$ stands for the spike state space $\mathbb{S} \triangleq \{0, 1\}$.

Fano Factor Fano factor (Fano, 1947) is a measurement of the spike count variability. Let us denote the spike count of neuron (L, m) as $n^{\text{trial ID}}_{L,m}$, the average value as $\mathrm{avg}(n_{L,m}) = \frac{1}{\text{\# trials}} \sum_k n^k_{L,m}$. The deviations from the mean is computed as $\Delta n^{\text{trial ID}}_{L,m} = n^{\text{trial ID}}_{L,m} - \overline{n}_{L,m}$, and the Fano factor is $\mathrm{FF}_{L,m} = \frac{\mathrm{Var}[n_{L,m}]}{\overline{n}_{L,m}}$

A.1 Relating DSNN SGL to NSNN NDL

Surrogate gradient learning is widely-adopted as an empirically solution to overcome the almost-everywhere-zero problem when computing gradients through step spiking functions. In this work, we propose to view surrogate gradient learning as a special form of noise-driven learning, and we reveal the close relationship between surrogate gradient function and voltage-level noise distributions in Table 3.

Table 3: Difference and correlation of inference and learning between DSNNs and NSNNs. The table lists some typical surrogate gradient functions and their corresponding noise distribution p_ϵ. Discarding the biological fact that the ions are subject to Brownian movement that corresponds to the Gaussian case we derived in the main body, we can extend the results in the table to noise to other continuous random distributions with zero-mean and symmetry PDF.

Model / **Phase**	**DSNN** Surrogate Gradient Learning		**NSNN** Noise-driven Learning	
Inference Spiking procedure	deterministic (approximate the stochastic inference) $o^t_{l,m} = \mathbf{1}_{u^t_{l,m} > v_{th}}$		stochastic $o^t_{l,m} \sim \mathrm{Ber}\big(\mathbb{P}[o^t_{l,m} = 1]\big)$	
Learning Post-synaptic factor $\frac{\partial o}{\partial u}$	Approximate by SGs		Acquire from noise statistics	$F'_\epsilon = p_\epsilon$
	ERF		$\mathcal{N}(0, \sigma^2)$	
	Sigmoid		$\mathrm{Logistic}(0, s)$	
	Rectangular		$\mathcal{U}(-a, a)$	
	Triangular		$\mathrm{Triangular}(-a, 0, a)$	
	Arctangent		$\mathrm{Atan}(0, \phi)$	

A.2 Experimental Details and Additional Results

A.2.1 Experimental Details

We conducted most experiments on a workstation with an Intel i5-10400 core, 64 GB memory, and an NVIDIA RTX 3090 card. Experimental results presented in the form of mean±std are acquired across three independent trials. We list experimental details in points as follows.

Training Details We optimize all networks with Adam solvers (Kingma & Ba, 2014) and adopt cosine annealing (decay to zero) learning decay policy (Loshchilov & Hutter, 2016). We list hyper-parameters for our experiments as follows in Table 4, algorithms and network architectures mentioned in Table 5.

Table 4: List of training hyper-parameters of experiments in this work.

	Dataset	Algo.	Arch.	T	Initial LR	mini-batch size	SG	Noise
DSNN	CIFAR-10	STBP	ResNet-18	2/4	0.01	256/256	ERF SG	/
		TET, $\lambda = 0.05$		2/4	0.01	256/256	ERF SG	/
		STBP	CIFARNet	2/4	0.004	256/256	ERF SG	/
	CIFAR-100	STBP	ResNet-18	2/4	0.005	256/256	ERF SG	/
		TET, $\lambda = 0.05$		2/4	0.005	256/256	ERF SG	/
		STBP	CIFARNet	2/4	0.001	256/256	ERF SG	/
	DVS-CIFAR	STBP	ResNet-19	10	0.0005	32	ERF SG	/
		TET, $\lambda = 0.001$	VGGSNN	10	0.0002	64	ERF SG	/
		tdBN		10	0.0002	64	ERF SG	/
NSNN	CIFAR-10	STBP	ResNet-18	2/4	0.002	256/256	/	$\mathcal{N}$
		TET, $\lambda = 0.05$		2/4	0.002	256/256	/	$\mathcal{N}$
		STBP	CIFARNet	2/4	0.003	256/128	/	$\mathcal{N}$
	CIFAR-100	STBP	ResNet-18	2/4	0.001	256/256	/	$\mathcal{N}$
		TET, $\lambda = 0.05$		2/4	0.001	256/256	/	$\mathcal{N}$
		STBP	CIFARNet	2/4	0.002	256/256	/	$\mathcal{N}$
	DVS-CIFAR	STBP	ResNet-19	10	0.0005	20	/	$\mathcal{N}$
		TET, $\lambda = 0.001$	VGGSNN	10	0.0003	32	/	$\mathcal{N}$
		tdBN		10	0.0003	32	/	$\mathcal{N}$

Table 5: List of SNN algorithms and network architectures (functional models) in our experiments.

Type	Name	Description
SNN Algorithm	STBP-tdBN	Zheng et al. (2021)
	STBP	Wu et al. (2018)
	TET	Deng et al. (2021)
Network Architecture	ResNet-19 (Zheng et al.)	128c3-(128c3-128c3)×2-(256c3-256c3)×3-(512c3-512c3)×2-ap-256fc-fc
	ResNet-18[1]	64c3-(64c3-64c3)×2-(128c3-128c3)×2-(256c3-256c3)×2-(512c3-512c3)×2-ap-fc
	VGGSNN (Deng et al.)	64c3-128c3-ap2-256c3-256c3-ap2-512c3-512c3-ap2-512c3-512c3-ap2-fc
	CIFARNet (Wu et al.)	128c3-256c3-ap2-512c3-ap2-1024c3-512c3-1024fc-512fc-fc
	7B-Net (Fang et al.)	Fang et al. (2021a)

1. constructed by modifying the first `conv` layer and removing one `maxpool` layer in the original work (He et al., 2016).

Baselines We summarize the baseline methods (algorithms) mentioned in our comparison experiments below.

- Hybrid conversion and spike timing dependent backpropagation (Rathi et al., 2019).

- Conversion-based spiking residual networks (Hu et al., 2018).
- Backpropagation-based temporal spike sequence learning (TSSL) (Zhang & Li, 2020).
- Spatiotemporal backpropagation (STBP) (Wu et al., 2019).
- DIET-SNN (Rathi & Roy, 2020).
- Spatiotemporal backpropagation with temporal dimension batch norm (STBP-tdBN) (Zheng et al., 2021).
- Histograms of averaged time surfaces (HATS descriptor) (Sironi et al., 2018).
- Distribution-aware retinal transform (DART descriptor) (Ramesh et al., 2019).
- ANN rollout and conversion (Kugele et al., 2020).
- AER object recognition by segmented probability maximisation (SPA) algorithm (Liu et al., 2020a).
- Leaky-integrate and analog fire network (LIAF-Net) (Wu et al., 2021).
- Spike-element-wise (SEW) residual networks (Fang et al., 2021b).
- Temporal efficient training (TET) objective function (Deng et al., 2021).

Datasets and Pre-processings We adopt static datasets including static datasets CIFAR-10&100 (Krizhevsky et al., 2009), dynamic dataset DVS-CIFAR (Li et al., 2017).

- CIFAR dataset includes 50k 32×32 images for training and 10k for evaluation. We adopt random crop, random horizontal flip and AutoAugment (Cubuk et al., 2018) for the training samples. For both training and evaluation phases, the preprocessed samples are normalized using z-score scaling.
- DVS-CIFAR dataset is a challenging neuromorphic benchmark recorded via a DVS camera using CIFAR-10 images. We adopt pre-processing pipeline in Samadzadeh et al. (2020), *i.e.*, divide the original set into a 9k-sample training set and 1k-sample evaluation set and all event stream files are spatially downsampled to 48×48. We augment the training samples following Deng et al. (2021).
- DVS-Gesture dataset (Amir et al., 2017). This datasets is recorded using DVS128, it contains 11 hand gestures from 29 subjects under 3 illumination conditions.

Perturbation Details We list the details about the perturbations in our experiments as following:

- In the Direct Optimization (DO) method, we construct the adversarial samples by directly solving the constrained optimization problem $\arg\max_{||\Delta x||_2=\gamma} \ell(f(x+\Delta x), y)$, where Δx for the adversarial perturbation and $x + \Delta x$ is the adversarial example. It is implemented using PyTorch and GeoTorch (Lezcano-Casado, 2019) toolkits. The L-2 norm bounded additive disturbance tensor are first zero-initialized and then optimized by an Adam solver with learning rate 0.002 for 30 iterations. After that, the additive perturbations are used to produce *adversarial samples* and then fed into the testees to evaluate (attack) the target models (either DSNNs or NSNNs in this work).
- The implementation of FGSM method follows Goodfellow et al. (2015), the adversarial example is constructed as $\tilde{x}^{adv} = x + \gamma_{FGSM} \times \mathrm{sign}[\nabla_x \ell\,(\mathrm{NN}(x), y))]$.
- The input-level *EventDrop* perturbation for dynamic inputs are constructed by randomly dropping spikes in the raw input spike trains. The dropping probability is set by a parameter ρ, the strategy of dropping we consider is Random Drop (Gu et al., 2021), which combines spatial and temporal-wise event dropping strategies. During the evaluation, we first individually perform EventDrop over every samples from the test set, and then fed our testees with the disturbed inputs.
- The hidden state-level noise includes two types of disturbances, the emission state from 1 to 0 (spike to silence) and emission state from 0 to 1 (silence to spike). To simplify the settings, we use one parameter β to control the probability of both kinds of changes. Let variable o denotes a spike state, if $o = 1$, we have $\mathbb{P}[o_{\text{new}} = 0] = \beta$, else, if $o = 0$, $\mathbb{P}[o_{\text{new}} = 1] = \beta$.

Table 6: List of notations of noise or perturbation parameters, grouped as perturbation-related ones and noises in Noisy LIF neurons by a thick horizontal line.

Parameter	**Description**
α	Membrane noise parameter: standard deviation of the normal distribution $\mathcal{N}_{\Delta u}$
β	Hidden state (spike train) noise parameter: adding or dropping probability for a spike.
γ_{DO}	Static input adversarial perturbation-DO method parameter: L-2 norm constraint of the vector Δx.
γ_{FGSM}	Static input adversarial perturbation-FGSM parameter: Δx updating step size.
ρ	Dynamic input EventDrop perturbation parameter: event dropping probability for events (spikes).
σ	Used in gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$.
s	Used in logistic noise $\epsilon \sim \text{Logistic}(0, s)$.
a	Used in rectangular $\epsilon \sim \mathcal{U}(-a, a)$ or triangular $\epsilon \sim \text{Triang}(-a, 0, a)$ noises.
ϕ	Used in arctangent noise $\epsilon \sim \text{Atan}(0, \phi)$.

Table 7: Evaluation results under hidden-state (spike) perturbation on the DVS-CIFAR, NSNNs achieve lower loss and higher accuracy under almost all conditions. Parameter β controls the strength of perturbations (larger for stronger perturbation).

		Loss				**Accuracy**			
	Type \ β	0.01	0.02	0.03	0.04	0.01	0.02	0.03	0.04
STBP & ResNet-19	DSNN	$1.43_{\pm 0.04}$	$1.72_{\pm 0.06}$	$2.44_{\pm 0.03}$	$3.43_{\pm 0.25}$	$69.73_{\pm 0.88}$	$63.91_{\pm 1.37}$	$53.60_{\pm 1.39}$	$40.32_{\pm 1.91}$
	NSNN	$1.23_{\pm 0.04}$	$1.30_{\pm 0.03}$	$1.41_{\pm 0.13}$	$1.74_{\pm 0.32}$	$72.88_{\pm 0.69}$	$70.44_{\pm 0.42}$	$67.27_{\pm 2.66}$	$58.99_{\pm 6.80}$
tdBN & VGGSNN	DSNN	$1.29_{\pm 0.05}$	$1.30_{\pm 0.08}$	$1.49_{\pm 0.18}$	$1.88_{\pm 0.29}$	$74.09_{\pm 0.66}$	$70.61_{\pm 1.37}$	$63.19_{\pm 2.89}$	$50.74_{\pm 3.32}$
	NSNN	$1.25_{\pm 0.01}$	$1.19_{\pm 0.02}$	$1.22_{\pm 0.06}$	$1.50_{\pm 0.13}$	$76.16_{\pm 0.13}$	$73.38_{\pm 0.55}$	$68.77_{\pm 0.78}$	$55.64_{\pm 1.87}$
TET & VGGSNN	DSNN	$0.82_{\pm 0.03}$	$0.91_{\pm 0.06}$	$1.08_{\pm 0.08}$	$1.37_{\pm 0.06}$	$76.41_{\pm 0.92}$	$72.60_{\pm 1.11}$	$66.46_{\pm 1.93}$	$56.06_{\pm 1.05}$
	NSNN	$0.75_{\pm 0.01}$	$0.80_{\pm 0.01}$	$0.94_{\pm 0.06}$	$1.25_{\pm 0.14}$	$78.28_{\pm 0.27}$	$76.32_{\pm 1.01}$	$71.54_{\pm 1.07}$	$62.48_{\pm 0.52}$

A.2.2 Evaluation results under hidden state-level perturbation

We further consider hidden state-level perturbations to directly mimic the spike train variability (Tuckwell et al., 2009; Yarom & Hounsgaard, 2011) aside the spike variance caused by membrane voltage fluctuations. The hidden state-level perturbation is directly put on the emitted spikes of all spiking neurons (Kasabov, 2010; Zhang et al., 2017), implemented by randomly flipping the state $o^t_{l,m}$ of all hidden neurons, the probability of flipping is controlled by a parameter β as $\mathbb{P}[1 \text{ to } 0] = \mathbb{P}[0 \text{ to } 1] = \beta$. Results are presented in Tab. 7 and Fig. 6.

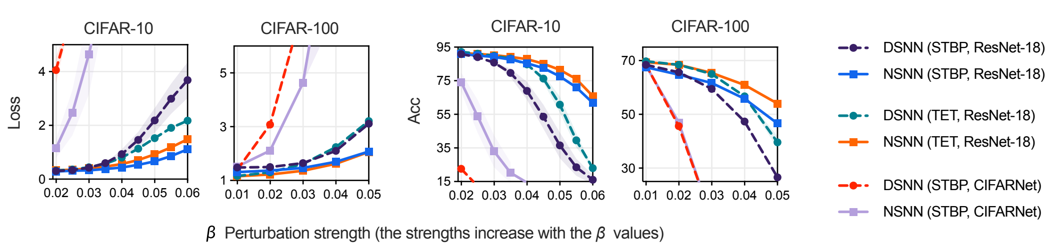


Figure 6: Evaluation results under hidden state perturbation on CIFAR datasets. NSNNs exhibit stronger resilience under perturbations, report lower loss and higher accuracy in most cases compared to the deterministic counterparts.

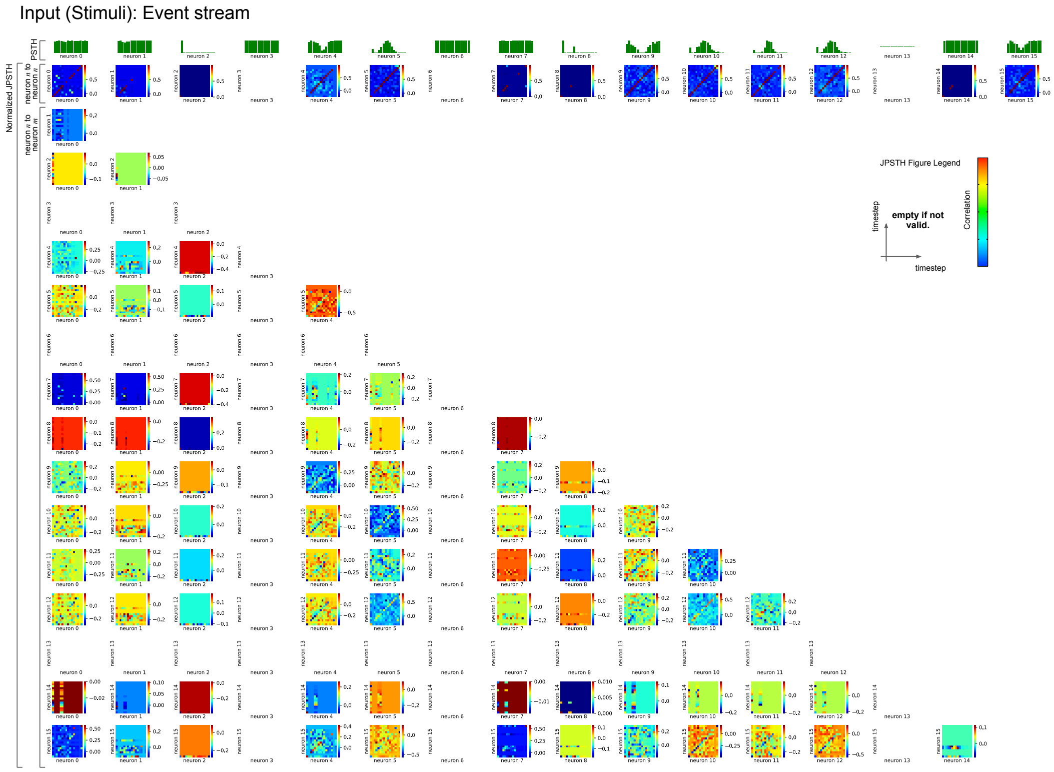


Figure 7: Normalized JPSTH plots of all neuron pairs of one observation in the DVS-Gesture event sequence classification task. The main diagonal of the JPSTH displays for each timestep the Pearson correlation of the two neuron firing simultaneously.

A.2.3 Neural Code Analysis Details

Experimental Details The sine series forecasting (SSF) uses a NSNN MLP with 32`fc`-16`fc` hidden units, simulation timestep is set to 48. The sinusoidal series is generated using $sin(x)$, where the step of x is 0.1. The training loss function is MSE loss. For the DVS-Gesture experiment in Section 4.4, we use a 7B-Net (in Tab. 5) and reduce the last layer's dimension to 16.

The PSTH, normalized JPSTH plots are presented in Fig. 7,8. The normalized JPSTH is a two-dim gram whose value at position u, v represents the Pearson correlation of one neuron firing at time u and the other one firing at time v.

A.2.4 All comparison results on recognition benchmarks.

We list the results of the full comparison experiment in Tab. 8 due to space constraints.

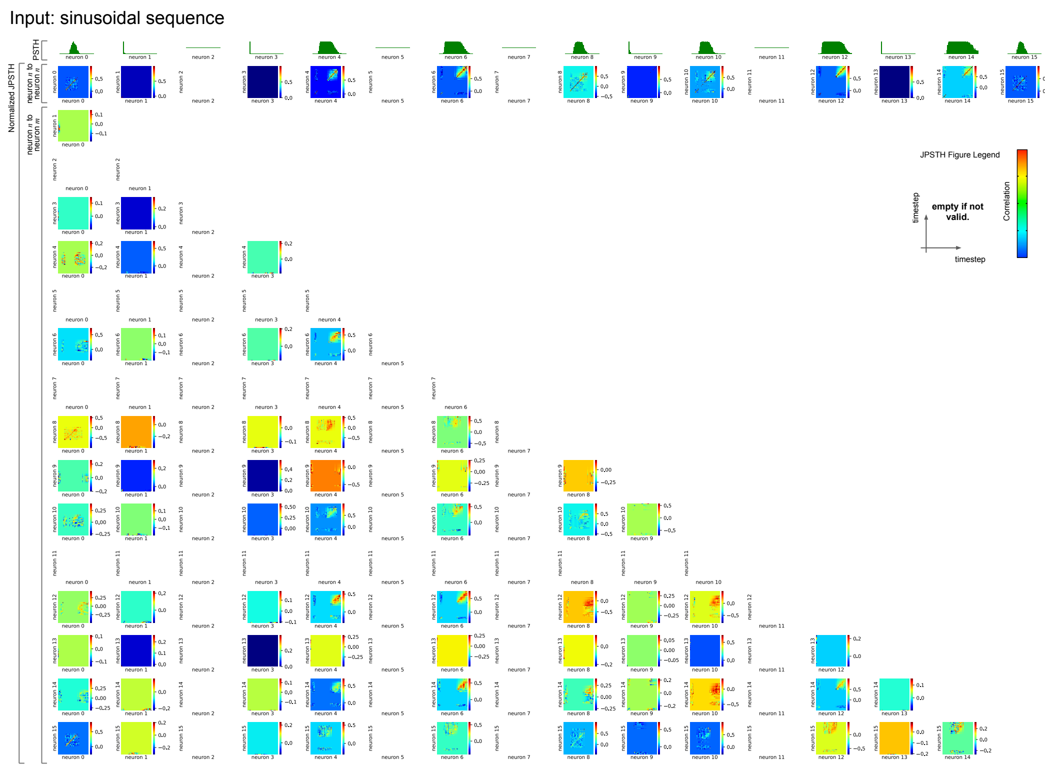


Figure 8: Normalized JPSTH plots of all neuron pairs in forecasting one sinusoidal sequence.

Table 8: Undisturbed evaluation results in accuracy (%), T for simulation timesteps.

	NSNN	Algorithm	Architecture	**Accuracy** $\text{avg} \pm \text{sd}(T)$		
CIFAR-10	○	STCA (Gu et al., 2019)	CIFARNet	91.23(12)		
	○	Rathi et al.	VGG-16	92.02(200)		
	○	Hu et al.	ResNet-44	92.37(350)		
	○	Zhang & Li	CIFARNet	91.41(5)		
	○	Wu et al.	CIFARNet	90.53(12)		
	○	STBP-tdBN (Zheng et al.)	ResNet-19	92.34(2)	92.92(4)	93.16(6)
	○	STBP† (Wu et al.)	ResNet-18*	93.18±0.07(2)	93.93±0.11(4)	
	●	STBP†	ResNet-18*	92.87±0.04(2)	93.77±0.12(4)	
	○	STBP†	CIFARNet	91.88±0.09(2)	92.79±0.14(4)	
	●	STBP	CIFARNet	93.90±0.12(2)	94.30±0.08(4)	
	○	TET (Deng et al.)	ResNet-19	94.16±0.03(2)	94.44±0.08(4)	94.50±0.07(6)
	○	TET† (Deng et al.)	ResNet-18*	93.62±0.02(2)	94.09±0.20(4)	
	●	TET	ResNet-18*	93.12±0.07(2)	94.14±0.05(4)	
CIFAR-100	○	Rathi & Roy	ResNet-20	64.07(5)		
	○	STBP-tdBN	ResNet-19	69.41±0.08(2)	70.86±0.22(4)	71.12±0.57(6)
	○	STBP†	ResNet-18*	70.15±0.14(2)	70.88±0.19(4)	
	○	STBP-tdBN (Zheng et al.)	ResNet-19	72.22±0.03(2)	73.41(4)	
	●	STBP	ResNet-18*	69.57±0.09(2)	71.16±0.40(4)	
	○	STBP†	CIFARNet	72.25±0.08(2)	72.94±0.21(4)	
	●	STBP	CIFARNet	73.36±0.14(2)	74.17±0.28(4)	
	○	TET	ResNet-19	72.87±0.10(2)	74.47±0.15(4)	74.72±0.28(6)
	○	TET†	ResNet-18*	71.72±0.13(2)	74.01±0.43(4)	
	●	TET	ResNet-18*	71.34±0.09(2)	73.33±0.03(4)	
DVS-CIFAR	○	Sironi et al.	N/A	52.40		
	○	Ramesh et al.	N/A	65.78		
	○	Kugele et al.	3B-DenseNet	66.75		
	○	Liu et al.	1-layer SNN	32.20		
	○	Wu et al.	LIAFNet	71.70		
	○	Wu et al.	6-layer SNN	60.50		
	○	Fang et al.	Wide-7B-Net	74.4(16)		
	○	STBP†	ResNet-19	71.74±0.92(10)		
	●	STBP	ResNet-19	74.30±0.61(10)		
	○	STBP-tdBN†	VGGSNN	75.51±0.49(10)		
	●	STBP-tdBN	VGGSNN	76.97±0.10(10)		
	○	TET†	VGGSNN	78.26±0.17(10)		
	●	TET	VGGSNN	79.52±0.38(10)		
DVS-GESTURE				**Accuracy**($T = 16$)		
	○	Fang et al.	7B-Net(Fang et al.)	97.92		
	○	Zheng et al.	ResNet-17	96.87($T = 40$)		
	○	STBP	7B-Net	95.84±0.27		
	●	STBP	7B-Net	96.88±0.28		

*: modified from the original implementation (He et al., 2016), refer to Tab. 5.
†: Re-produced results.