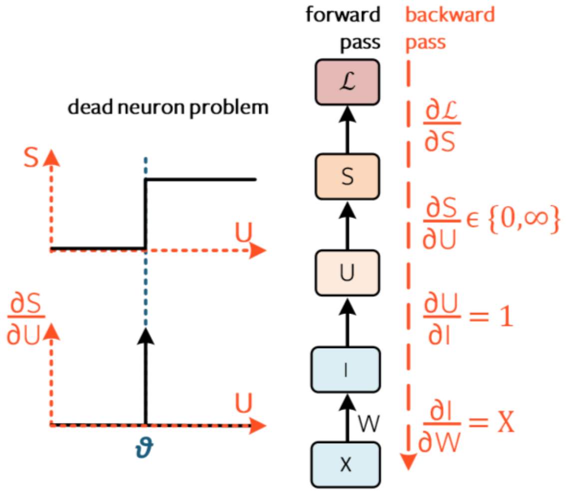
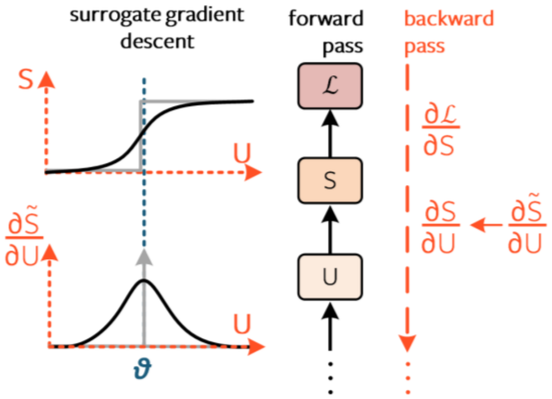
Efficient Training of Spiking Neural Networks with Multi-Parallel Implicit Stream Architecture

1 The Dilemma of Training SNNs

① Non-differentiability



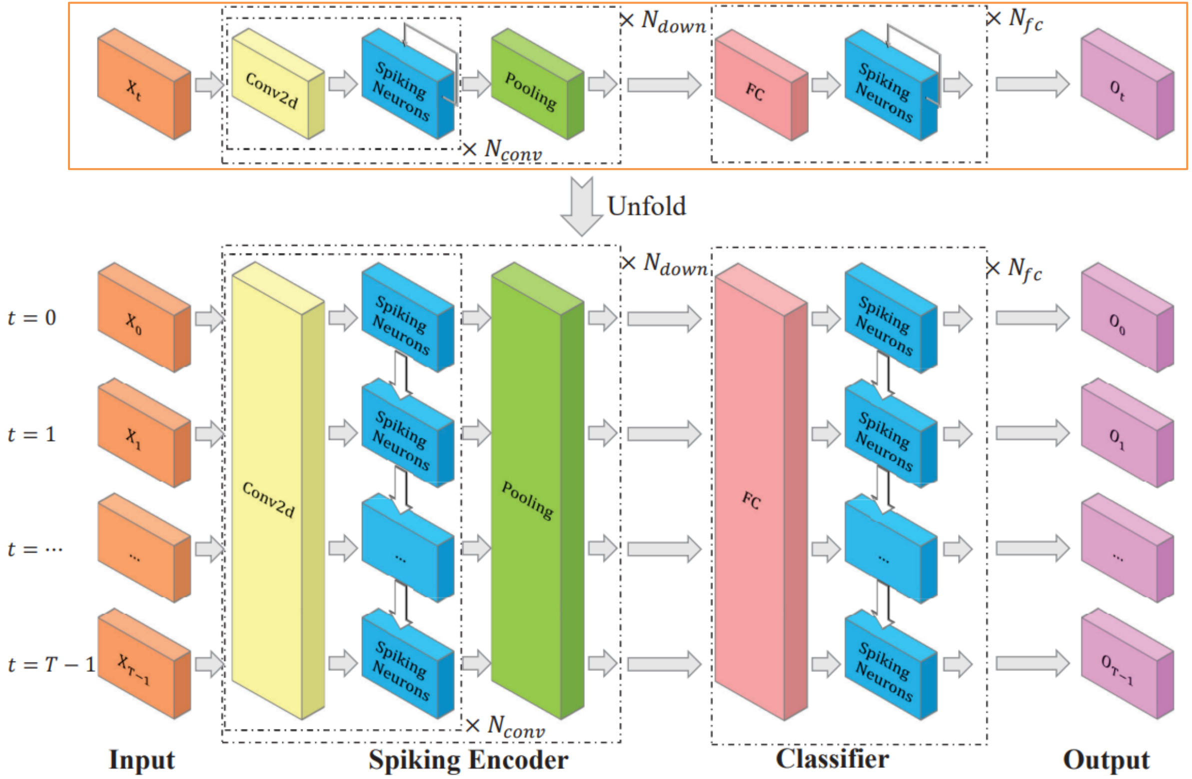
Surrogate Gradient (SG)



Due to the spiking process of neurons being a step function, the derivative at the spike is infinite, which prevents the direct use of backpropagation for training SNNs.

The SG method uses a smooth function, similar to the step function, to replace the step function for differential calculations during the backward process. However, this approach introduces surrogate errors, which accumulate over time steps.

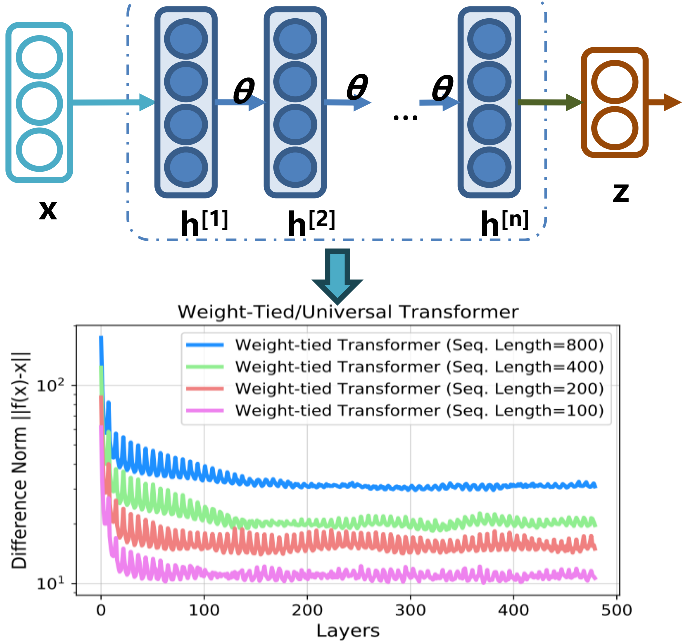
② Memory Overhead



As the number of simulation time steps increases during the training of SNNs, the memory consumption also increases. Specifically, each simulation time step requires storing all the activation values of the current model, leading to a linear increase in memory consumption as the number of simulation time steps rises. However, it is essential to explore different simulation time step lengths.

2 Deep Equilibrium Model

① Only One Layer of Activation Values Needs to Be Stored



As the number of layers increases, the outputs of the weight-tied model gradually converge.

Forward Process:

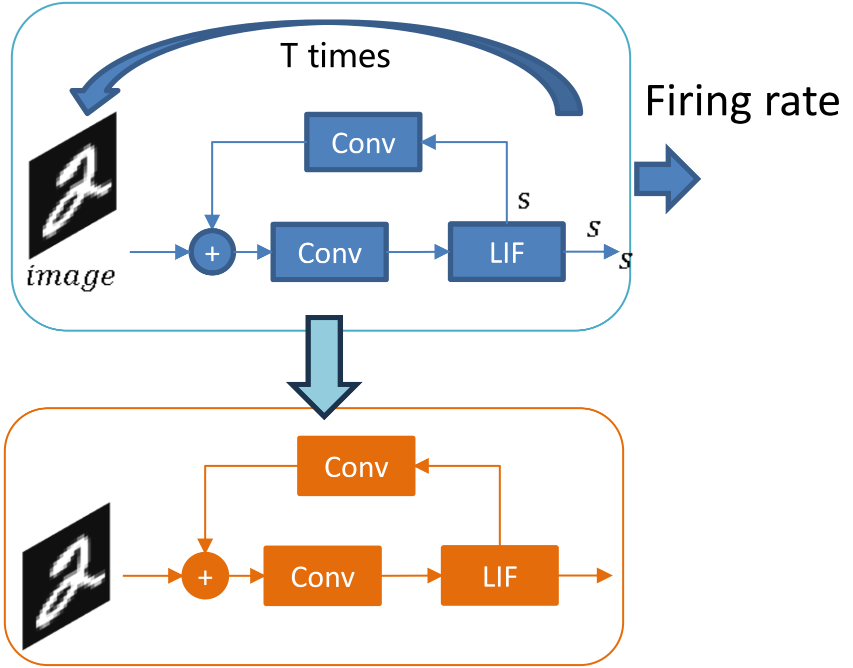
Since the output of the weight-tied network ultimately converges to a fixed point, the forward process of the network can be transformed into the process of solving a fixed point equation. Define a single-layer network f_θ with the network output h^*. By solving the equation $f_\theta(h^*;x) = h^*$, we can obtain the output of f_θ stacked with infinite layers.

Backward Process:

Since the network's forward process is solved through root-finding methods for the network output, there is no explicit path in the forward process. We utilize the implicit function theorem on the fixed point equation $f_\theta(h^*;x) = h^*$ to replace the backpropagation calculation of the derivative of the network output h^* with respect to any parameter $(\cdot)$, denoted as $\frac{\partial h^*(\cdot)}{\partial(\cdot)}$.

2 Deep Equilibrium Model

② Integrating Deep Equilibrium Theory with SNNs



A Single Time Step Simulation Is Equivalent to an Arbitrarily Long Simulation Time.

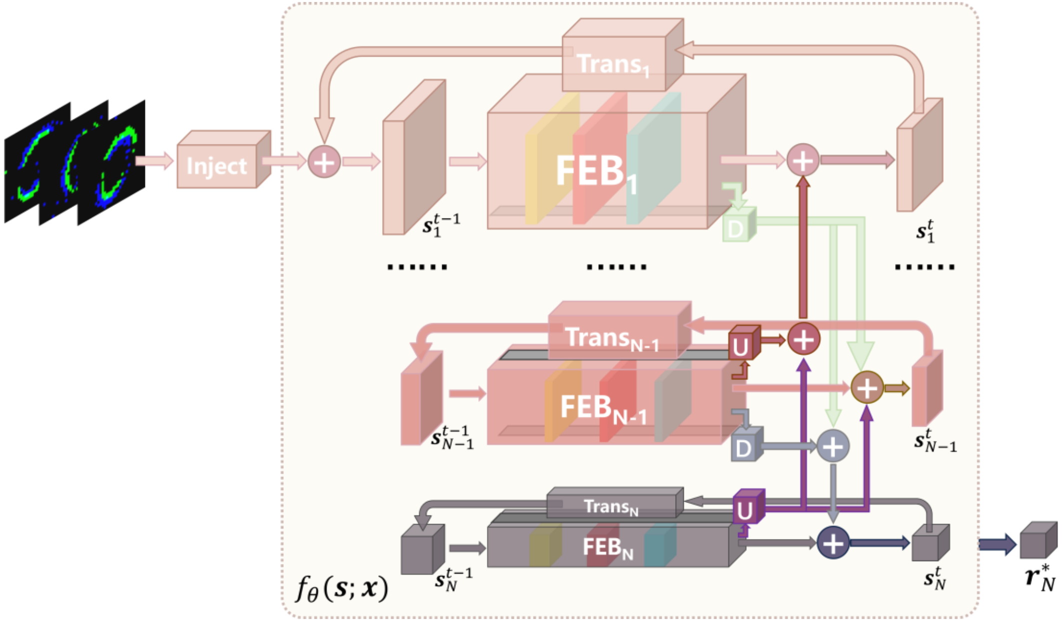
By treating SNNs as a weight-tied block and applying the equilibrium model theory, we can separate the forward and backward processes of SNNs.

This allows for error propagation without explicit backpropagation over time, facilitating SNN training with constant memory overhead.

However, **both SNNs and the equilibrium model encounter time delay issues**, including simulation time and fixed point solving time, which we address through a shallower parallel structure.

3 Methods and Model

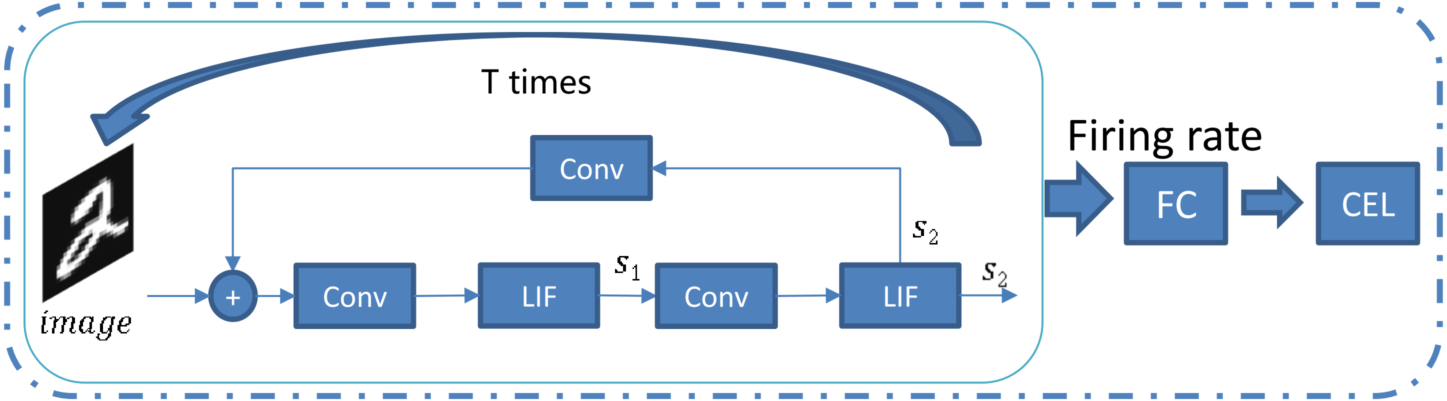
① MPIS-SNNs



MPIS-SNNs

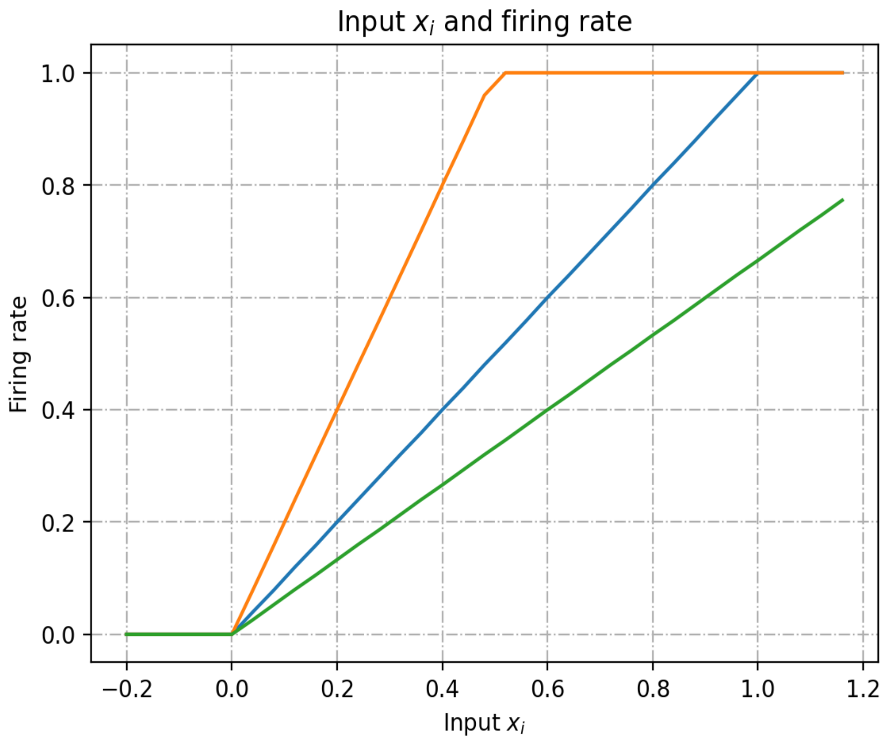
The main idea of MPIS is to **reduce the simulation time of a single time step** in SNNs by **decomposing their vertical complexity**. Additionally, it **accelerates model convergence** by **merging feature maps** from various implicit streams (IS), thus reducing the number of iterations required for fixed point solving and shortening the forward process time. Although shallower IS can lower the time cost of the forward process, a clear issue arises from the reduced model complexity. To address this, we parallelly **increase model parameters and inject input only into the top layer of the IS** to ensure the model's capacity.

② Double-Bounded Rectified Linear Unit (DBReLU)



We observe that when SNNs have a multi-layer structure, even when using a single time step equivalent to T time steps for gradient computation, the presence of the step function (neuron spikes) within SNNs prevents direct calculation of derivatives. Inspired by the conversion of ANNs to SNNs and the implicit differentiation in equilibrium SNNs, we derive the Double-Bounded Rectified Linear Unit (DBReLU) as a firing rate calculation function for SNNs when reaching equilibrium states.

② Double-Bounded Rectified Linear Unit (DBReLU)



Firing Rate Curves of Neurons with Different Thresholds

DBReLU:

$$r_i^l = Min\left(Max\left(0, \frac{\left(\sum_{j=1}^{M^{l-1}} W_{ij}^l r_j^{l-1}\right)}{V_{th}}\right), 1\right)$$

In ANNs-SNNs, neurons simulate the ReLU function, with performance positively correlated to the number of simulation time steps. However, hardware limitations prevent indefinite increases in simulation steps. In MPIS-SNNs, the final output represents the model's fixed point, equivalent to the firing rate after infinite time steps. At this stage, Integrate-and-Fire (IF) neurons serve as unbiased estimators of the linear rectifier over time, but since firing rates cannot exceed 1, 1 is set as the upper bound.

4 Results and Discussion

① Comparison with the BPTT Training Method

	Method	Size	Architecture	T	Acc	Time	Memory
Fashion-MNIST	BPTT	133K	16C3-32C3-48C3-FC10	30	89.60%	31s	2.1G
				100	89.70%	1min32s	4.8G
	MPIS	133K	16C3-32C3-48C3-FC10	**30**	**93.14%**	**30s**	**1.2G**
				100	93.23%	1min27s	1.2G
N-MNIST	BPTT	213K	32C3-32C3-64C3-FC10	30	98.21%	1min24s	12.8G
				100	-	-	Out of memory
	MPIS	213K	32C3-32C3-64C3-FC10	**30**	**99.31%**	**1min35s**	**3.3G**
				100	99.27%	5min5s	3.3G

	Firing Rate		
	Layer1	Layer2	Layer3
BPTT	5.06e-2	7.24e-2	8.98e-2
MPIS	**8.0e-4**	**7.0e-4**	**7.3814e-5**

MPIS-SNNs have a constant memory cost independent of simulation duration, while BPTT's memory cost increases with simulation length. MPIS-SNNs also have a lower firing rate, leading to reduced energy consumption.

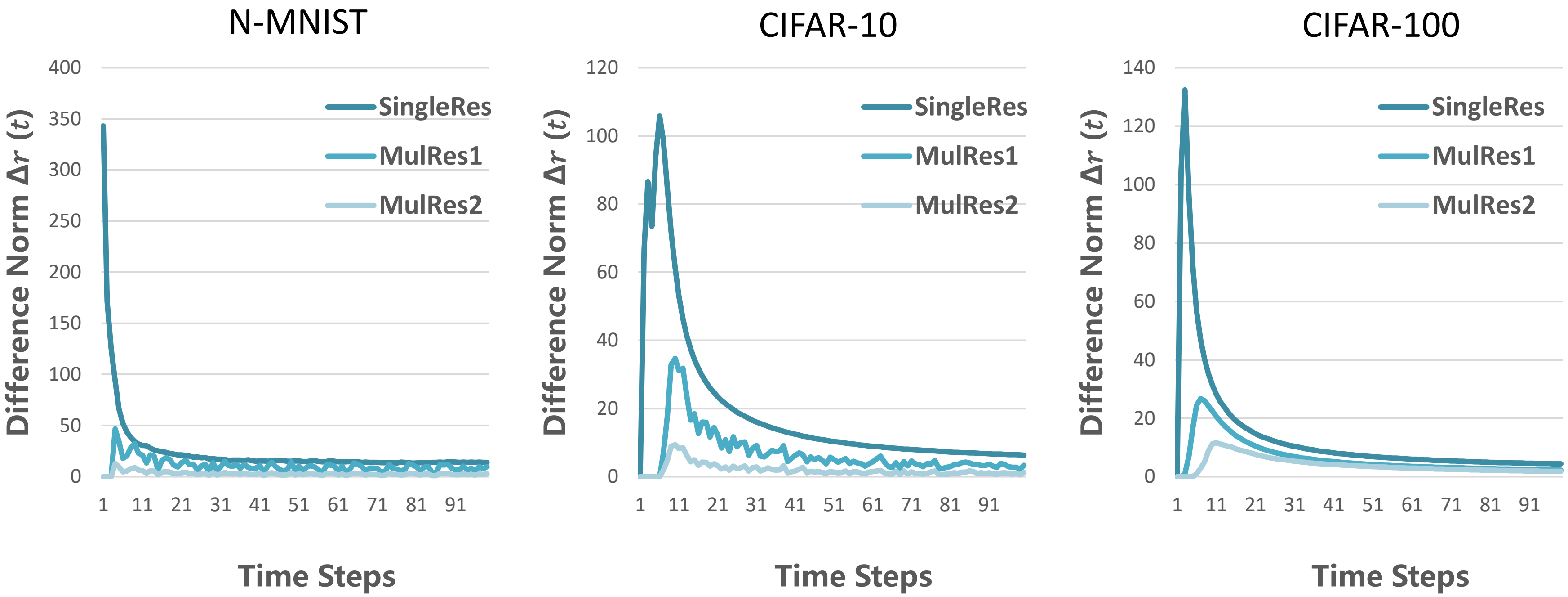
4 Results and Discussion

② Comparison with Conventional Equilibrium SNNs

	Method	Size	T	Acc	Time
CIFAR-10	IDE-Nets	11.8M	30	90.37%	12min10s
			100	90.57%	22min34s
	MPIS-SNNs	11.8M	30	92.79%	2min34s
		28.5M	**30**	**93.27%**	**3min48s**
CIFAR-100	IDE-Nets	14.8M	30	70.26%	12min33s
			100	71.12%	21min50s
	MPIS-SNNs	14.8M	30	73.19%	5min33s
		30.0M	**30**	**74.40%**	**8min38s**

In terms of accuracy, MPIS-SNNs achieve higher accuracy with fewer simulation time steps. Regarding speed, MPIS-SNNs, despite having more parameters, are still faster than IDE-Net.

③ Convergence Speed of MPIS-SNNs



SingleRes represents the convergence curve of conventional equilibrium SNNs, while MulRes*n* denotes the convergence curve of the n-th implicit stream branch of MPIS-SNNs. The convergence rate and final stability of each branch in MPIS-SNNs are superior to those of conventional equilibrium SNNs.

4 Results and Discussion

④ Comparing with the Latest Efficient Training Methods for SNNs

	Method	T	Accuracy
N-MNIST	IDE-Net(2021)[14]	30	99.47%
	HS-IF(2023)[43]	15	99.44%
	MPIS	30	99.51%
Fashion-MNIST	IDE-Net(2021)[14]	5	90.25%
	LTC-SNNs(2023)[27]	784	93.58%
	MPIS	1	93.83%
CIFAR-10			
	Hybrid SL(2021)[44]	100	91.29%
	Temporal pruning (2022) [40]	1	93.05%
	OTTT(2022)[28]	6	93.73%
	AC2AS (2023)[8]	5	92.88%
	MPIS	10	93.27%

CIFAR-100			
	IDE-Net(2021)[14]	100	73.43%
	Hybrid SL(2021)[44]	120	64.98%
	Temporal pruning (2022)[40]	1	70.15%
	OTTT(2022)[28]	6	71.11%
	AC2AS (2023)[8]	5	73.61%
	MPIS	5	74.93%

MPIS-SNNs are highly competitive in various tasks, particularly, MPIS-SNNs have achieved optimal performance on the N-MNIST, Fashion-MNIST, and CIFAR-100 datasets.

Thanks

西安交通大學
XI'AN JIAOTONG UNIVERSITY