Supplementary Information

Efficient and robust temporal processing with neural oscillations modulated spiking neural networks

Yinsong Yan[1†], Qu Yang[2†], Yujie Wu[3], Hanwen Liu[4], Malu Zhang[4], Haizhou Li[2, 5], Kay Chen Tan[1, 6], Jibin Wu[1, 3, 6*]

Contents

List of Figures

[1] Department of Data Science and Artificial Intelligence, The Hong Kong Polytechnic University, Hong Kong SAR, China.
[2] Department of Electrical and Computer Engineering, National University of Singapore, Singapore.
[3] Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR, China.
[4] Department of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China
[5] School of Data Science, The Chinese University of Hong Kong, Shenzhen, China.
[6] Research Center on Data Sciences & Artificial Intelligence, The Hong Kong Polytechnic University, Hong Kong SAR, China
[†] These authors contributed equally to this work.
[*] Corresponding author. E-mail: jibin.wu@polyu.edu.hk.

List of Tables

S1 Details of experimental settings for temporal processing tasks

Here, we detail how to integrate the proposed rhythmic modulation mechanism into different neuron models used in temporal processing tasks.

Rhythm-FFSNN is developed by integrating the oscillation mechanism into the SNN with a feedforward structure. The mathematical formulations of Rhythm-FFSNN can be formalized as follows:

$$I_i^l[t] = \sum_j w_{ij}^l S_j^{l-1}[t] + b_i^l, \tag{1}$$

$$U_i^l[t] = \begin{cases} \alpha U_i^l[t-1] + I_i^l[t] - \vartheta S_i^l[t-1], & \text{if} \quad m_i^l[t] = 1 \\ U_i^l[t-1], & \text{if} \quad m_i^l[t] = 0 \end{cases}, \tag{2}$$

$$S_i^l[t] = m_i^l[t]\,\Theta\left(U_i^l[t] - \vartheta\right), \tag{3}$$

with

$$m_i^l[t] = \begin{cases} 1, & \text{if} \quad 0 \le (t - \lfloor \varphi_i^l c_i^l \rfloor) \bmod c_i^l < \lfloor d_i^l c_i^l \rfloor \\ 0, & \text{otherwise} \end{cases}, \tag{4}$$

where $I_i^l[t]$ denotes the input current to neuron i in layer l, w_{ij}^l and b_i^l represent the connection weight from neuron j of the preceding layer $l-1$ and the constant injecting current to neuron i, respectively, $U_i^l[t]$ represents the membrane potential of the neuron i in layer l. The constant $\alpha \equiv exp(-dt/\tau_m)$ captures the membrane potential decay with τ_m as the membrane time constant and dt as the simulation time step. The term ϑ stands for the neuronal firing threshold. $S_i^l[t]$ is the spike emitted from neuron i at time step t. $m_i^l[t]$ is the oscillatory modulating signal corresponding to neuron i at time step t. φ_i^l signifies the phase of the modulating signal. c_i^l and d_i^l denote the rhythm period and duty cycle of the modulating signal, respectively. $\lfloor \cdot \rfloor$ is the floor function. $\Theta(\cdot)$ represents the Heaviside step function which is defined as $\Theta(x) = 1$ for $x \ge 0$ and $\Theta(x) = 0$ for $x < 0$.

Rhythm-SRNN incorporates additional recurrent connections compared to the above Rhythm-FFSNN. Consequently, the updating formula for the input current is augmented with a recurrent term. The Rhythm-SRNN model is defined as follows:

$$I_i^l[t] = \sum_j w_{ij}^{l-1} S_j^{l-1}[t] + \sum_j w_{ij}^l S_j^l[t-1] + b_i^l, \tag{5}$$

$$U_i^l[t] = \begin{cases} \alpha U_i^l[t-1] + I_i^l[t] - \vartheta S_i^l[t-1], & \text{if} \quad m_i^l[t] = 1 \\ U_i^l[t-1], & \text{if} \quad m_i^l[t] = 0 \end{cases}, \tag{6}$$

$$S_i^l[t] = m_i^l[t]\,\Theta\left(U_i^l[t] - \vartheta\right), \tag{7}$$

with

$$m_i^l[t] = \begin{cases} 1, & \text{if} \quad 0 \le (t - \lfloor \varphi_i^l c_i^l \rfloor) \bmod c_i^l < \lfloor d_i^l c_i^l \rfloor \\ 0, & \text{otherwise} \end{cases}, \tag{8}$$

where w_{ij}^l denotes the recurrent connection weight in layer l. H_i^l, U_i^l, S_i^l, and m_i^l remain the same as those defined in the Rhythm-FFSNN.

Rhythm-LSNN integrates our rhythmic neural modulation mechanisms into the advanced spiking neuron model, LSNN[1], which incorporates an adaptive firing threshold. This enhancement improves the SRNN's ability to preserve information over an extended period. The adaptive threshold $B_i^l[t]$, adjusts by a fixed quantity $\frac{\beta}{\tau_{a,i}}$ upon each firing event, and subsequently decays exponentially towards a baseline b_i^0, with a time constant $\tau_{a,i}$. This dynamic threshold can be defined as:

$$\eta_i^l[t] = \rho_i \eta_i^l[t-1] + (1-\rho_i) S_i^l[t-1], \tag{9}$$

$$B_i^l[t] = b_i^0 + \beta \eta_i^l[t], \tag{10}$$

where $\rho_i = exp(-1/\tau_{a,i})$. The parameter β, denoting the magnitude of threshold adaptation, is set to 1.8 in this work. Consequently, the Rhythm-LSNN can be defined formally as follows:

$$I_i^l[t] = \sum_j w_{ij}^{l-1} S_j^{l-1}[t] + \sum_j w_{ij}^l S_j^l[t-1] + b_i^l, \tag{11}$$

$$U_i^l[t] = \begin{cases} \alpha U_i^l[t-1] + I_i^l[t] - B_i^l[t] S_i^l[t-1], & \text{if} \quad m_i^l[t] = 1 \\ U_i^l[t-1], & \text{if} \quad m_i^l[t] = 0 \end{cases}, \tag{12}$$

$$S_i^l[t] = m_i^l[t]\,\Theta\left(U_i^l[t] - B_i^l[t]\right), \tag{13}$$

with

$$m_i^l[t] = \begin{cases} 1, & \text{if} \quad 0 \le (t - \lfloor \varphi_i^l c_i^l \rfloor) \bmod c_i^l < \lfloor d_i^l c_i^l \rfloor \\ 0, & \text{otherwise} \end{cases}. \tag{14}$$

Rhythm-ASRNN integrates the proposed rhythmic neural modulation mechanism into the advanced spiking neuron model[2], which incorporates two learnable parameters to enhance the temporal processing capacity of LIF neurons: the membrane potential time constant $\tau_{m,i}$ and the adaptive threshold time constant $\tau_{a,i}$. The Rhythm-ASRNN is formulated as follows:

$$I_i^l[t] = \sum_j w_{ij}^{l-1} S_j^{l-1}[t] + \sum_j w_{ij}^l S_j^l[t-1] + b_i^l, \tag{15}$$

$$\alpha_i = exp(-1/\tau_{m,i}), \tag{16}$$

$$\rho_i = exp(-1/\tau_{a,i}), \tag{17}$$

$$\eta_i^l[t] = \rho_i \eta_i^l[t-1] + (1-\rho_i) S_i^l[t-1], \tag{18}$$

$$B_i^l[t] = b_i^0 + \beta \eta_i^l[t], \tag{19}$$

$$U_i^l[t] = \begin{cases} \alpha_i U_i^l[t-1] + I_i^l[t] - B_i^l[t] S_i^l[t-1], & \text{if} \quad m_i^l[t] = 1 \\ U_i^l[t-1], & \text{if} \quad m_i^l[t] = 0 \end{cases}, \tag{20}$$

$$S_i^l[t] = m_i^l[t]\,\Theta\left(U_i^l[t] - B_i^l[t]\right), \tag{21}$$

with

$$m_i^l[t] = \begin{cases} 1, & \text{if} \quad 0 \le (t - \lfloor \varphi_i^l c_i^l \rfloor) \bmod c_i^l < \lfloor d_i^l c_i^l \rfloor \\ 0, & \text{otherwise} \end{cases}, \tag{22}$$

where B_i^l denotes the adaptive threshold of neuron i in layer l, $\tau_{m,i}$ and $\tau_{a,i}$ represents the learnable parameters that will be jointly trained with the network parameters in an end-to-end manner.

Rhythm-DHSNN integrates the proposed rhythmic neural modulation mechanism into the advanced spiking neuron model, DH-SNN[3], which incorporates temporal dendritic heterogeneity into spiking neurons to enhance their temporal processing capablity. The dynamics of Rhythm-DHSNN can be modeled as follows:

$$I_d^l[t] = \alpha_d^l I_d^l[t-1] + \left(1-\alpha_d^l\right) I_d^l[t], \tag{23}$$

$$U_i^l[t] = \begin{cases} \beta_i^l U_i^l[t-1] + \left(1-\beta_i^l\right)\sum_d I_d^l[t] - \vartheta S_i^l[t-1], & \text{if} \quad m_i^l[t]=1 \\ U_i^l[t-1], & \text{if} \quad m_i^l[t]=0 \end{cases}, \tag{24}$$

$$S_i^l[t] = m_i^l[t]\,\Theta\left(U_i^l[t]-\vartheta\right), \tag{25}$$

with

$$m_i^l[t] = \begin{cases} 1, & \text{if} \quad 0 \le (t-\lfloor \varphi_i^l c_i^l \rfloor) \bmod c_i^l < \lfloor d_i^l c_i^l \rfloor \\ 0, & \text{otherwise} \end{cases}, \tag{26}$$

where α and β are learnable time constants of dendritic branches and the membrane potential, respectively. d is the index of dendritic branches in DHSNN. Depending on whether there are recurrent connections in the network, it can be further categorized into Rhythm-DH-SRNN and Rhythm-DH-SFNN.

To ensure a fair comparison, we fine-tuned model hyperparameters and training configurations across different datasets for optimal performance. An exhaustive list of these configurations are summarized in Table S1.

S2 Details of experimental settings for working memory assessment tasks

We assess the working memory processing capabilities of Rhythm-SNN on the STORE-RECALL task by using two different neuron models, Adaptive-Leaky Integrate and Fire (ALIF) and Double EXponential Adaptive Threshold (DEXAT) neurons. The details of these neuron models are summarized as follows.

Rhythm-ALIF, whose conventional counterpart is originally detailed in[1], enhances temporal computational capabilities of SNNs. This enhancement stems from its ability to adaptively modify a neuron's firing threshold based on spike frequencies. The formulation of the ALIF model is described in equations (9) and (10). Consistent with the experimental setup in[4], we employed a fixed time constant $\tau_{a,i}$. The mathematical formulation of Rhythm-ALIF is described as follows:

$$\eta_i^l[t] = \rho_i \eta_i^l[t-1] + (1-\rho_i) S_i^l[t-1], \tag{27}$$

$$B_i^l[t] = b_i^0 + \beta \eta_i^l[t], \tag{28}$$

$$I_i^l[t] = \sum_j w_{ij}^{l-1} S_j^{l-1}[t] + \sum_j w_{ij}^l S_j^l[t-1] + b_i^l, \tag{29}$$

$$U_i^l[t] = \begin{cases} \alpha U_i^l[t-1] + I_i^l[t] - B_i^l[t] S_i^l[t-1], & \text{if} \quad m_i^l[t]=1 \\ U_i^l[t-1], & \text{if} \quad m_i^l[t]=0 \end{cases}, \tag{30}$$

$$S_i^l[t] = m_i^l[t]\,\Theta\left(U_i^l[t]-B_i^l[t]\right), \tag{31}$$

Table S1. Training configurations and hyperparameter settings for temporal processing tasks.

Task	Network	Epoch	Batch Size	Learning Rate	Architecture[1,2]	Rhythm Period	Duty Cycle	Phase
PS-MNIST	FFSNN	150	256	$5e^{-3}$	1-F64-F256-F256-10	-	-	-
	Rhythm-FFSNN	150	256	$5e^{-3}$	1-F64-F256-F256-10	[10, 50]	[0.02, 0.1]	[0, 0.5]
	SRNN	150	256	$1e^{-3}$	1-R64-R256-R256-10	-	-	-
	Rhythm-SRNN	150	256	$1e^{-3}$	1-R64-R256-R256-10	[10, 50]	[0.02, 0.1]	[0, 0.5]
	LSNN	150	256	$5e^{-3}$	1-R64-R256-R256-10	-	-	-
	Rhythm-LSNN	150	256	$5e^{-3}$	1-R64-R256-R256-10	[10, 50]	[0.02, 0.1]	[0, 0.5]
	ASRNN	150	256	$1e^{-2}$	1-R64-R256-R256-10	-	-	-
	Rhythm-ASRNN	150	256	$1e^{-2}$	1-R64-R256-R256-10	[10, 50]	[0.02, 0.1]	[0, 0.5]
	DH-SRNN	150	128	$1e^{-2}$	1-R64-R256-10	-	-	-
	Rhythm-DH-SRNN	150	128	$1e^{-2}$	1-R64-R256-10	[2, 10]	[0.05, 0.5]	[0, 0.5]
ECG	FFSNN	250	64	$1e^{-2}$	4-F36-6	-	-	-
	Rhythm-FFSNN	250	64	$1e^{-2}$	4-F36-6	[1, 3]	[0.1, 0.5]	[0, 0.5]
	SRNN	250	64	$1e^{-3}$	4-R36-6	-	-	-
	Rhythm-SRNN	250	64	$1e^{-3}$	4-R36-6	[1, 3]	[0.1, 0.5]	[0, 0.5]
	LSNN	250	64	$1e^{-2}$	4-R36-6	-	-	-
	Rhythm-LSNN	250	64	$1e^{-2}$	4-R36-6	[1, 3]	[0.1, 0.5]	[0, 0.5]
	ASRNN	250	64	$1e^{-2}$	4-R36-6	-	-	-
	Rhythm-ASRNN	250	64	$4e^{-2}$	4-R36-6	[1, 3]	[0.1, 0.5]	[0, 0.5]
	DH-SRNN	250	64	$1e^{-2}$	4-R36-6	-	-	-
	Rhythm-DH-SRNN	250	64	$1e^{-2}$	4-R36-6	[10, 50]	[0.95, 1.0]	[0, 0.5]
DVS-Gesture	SRNN	30	64	$3e^{-3}$	2048-MPK4S4-F1024-R512-11	-	-	-
	Rhythm-SRNN	30	64	$3e^{-3}$	2048-MPK4S4-F1024-R512-11	[1, 50]	[0.1, 0.5]	[0, 0.5]
	ASRNN	30	64	$3e^{-2}$	2048-MPK4S4-F1024-R512-11	-	-	-
	Rhythm-ASRNN	30	64	$3e^{-2}$	2048-MPK4S4-F1024-R512-11	[1, 50]	[0.1, 0.5]	[0, 0.5]
S-MNIST	DH-SRNN	150	128	$1e^{-2}$	1-R64-R256-10	-	-	-
	Rhythm-DH-SRNN	150	128	$1e^{-2}$	1-R64-R256-10	[1, 50]	[0.15, 0.95]	[0, 0.3]
SHD	ASRNN	100	100	$5e^{-3}$	700-R128-R128-20	-	-	-
	Rhythm-ASRNN	100	100	$5e^{-3}$	700-R128-R128-20	[10, 50]	[0.15, 0.95]	[0, 0.5]
	DH-SFNN	100	100	$1e^{-2}$	700-F64-F64-20	-	-	-
	Rhythm-DH-SFNN	100	100	$1e^{-2}$	700-F64-F64-20	[10, 60]	[0.15, 0.99]	[0, 0.3]
GSC	DH-SFNN	150	200	$1e^{-2}$	120-F200-F200-F200-15	-	-	-
	Rhythm-DH-SFNN	150	200	$1e^{-2}$	120-F200-F200-F200-15	[10, 50]	[0.95, 1.0]	[0, 0.5]
VoxCeleb1	FFSNN	200	512	$1e^{-3}$	40-F512-F512-1251	-	-	-
	Rhythm-FFSNN	200	512	$1e^{-3}$	40-F512-F512-1251	[10, 50]	[0.95, 1.0]	[0, 0.3]
	PLIF	200	512	$1e^{-3}$	40-F512-F512-1251	-	-	-
	Rhythm-PLIF	200	512	$1e^{-3}$	40-F512-F512-1251	[10, 50]	[0.95, 1.0]	[0, 0.3]
PTB	SRNN	100	25	$5e^{-1}$	650-R650-R650-10000	-	-	-
	Rhythm-SRNN	100	25	$5e^{-1}$	650-R650-R650-10000	[10, 50]	[0.95, 0.99]	[0, 0.5]
	PLIF	100	25	$5e^{-1}$	650-R650-R650-10000	-	-	-
	Rhythm-PLIF	100	25	$5e^{-1}$	650-R650-R650-10000	[10, 50]	[0.95, 0.99]	[0, 0.5]
	ASRNN	100	25	$5e^{-1}$	650-R650-R650-10000	-	-	-
	Rhythm-ASRNN	100	25	$5e^{-1}$	650-R650-R650-10000	[10, 50]	[0.95, 0.99]	[0, 0.5]

[1] The prefixes 'F' and 'R' in network architecture denote the feedforward and recurrent connections, respectively.
[2] MPK4S4 refers to the max-pooling layer with a kernel size of 4 and a stride of 4.

with

$$m_i^l[t] = \begin{cases} 1, & \text{if} \quad 0 \le (t - \lfloor \varphi_i^l c_i^l \rfloor) \bmod c_i^l < \lfloor d_i^l c_i^l \rfloor \\ 0, & \text{otherwise} \end{cases}, \tag{32}$$

where $\rho_i = exp(-1/\tau_{a,i})$. The parameter β, dictating the magnitude of threshold adaptation, is set to 1.8 by default in this work.

Rhythm-DEXAT, DEXAT[4] extends the ALIF model by incorporating two auxiliary variables into the threshold voltage. Specifically, the firing threshold undergoes a rapid initial reduction, followed by a slower decay over an extended period. The Rhythm-DEXAT model can be formalized as:

$$b_{i1}^l[t] = \rho_{i1} b_{i1}^l[t-1] + (1-\rho_{i1}) S_i^l[t-1], \tag{33}$$

$$b_{i2}^{l}[t] = \rho_{i2} b_{i2}^{l}[t-1] + (1-\rho_{i2}) S_i^{l}[t-1], \tag{34}$$

$$B_i^{l}[t] = b_i^{0} + \beta_1 b_{i1}^{l}[t] + \beta_2 b_{i2}^{l}[t], \tag{35}$$

$$I_i^{l}[t] = \sum_j w_{ij}^{l-1} S_j^{l-1}[t] + \sum_j w_{ij}^{l} S_j^{l}[t-1] + b_i^{l}, \tag{36}$$

$$U_i^{l}[t] = \begin{cases} \alpha U_i^{l}[t-1] + I_i^{l}[t] - B_i^{l}[t] S_i^{l}[t-1], & \text{if} \quad m_i^{l}[t] = 1 \\ U_i^{l}[t-1], & \text{if} \quad m_i^{l}[t] = 0 \end{cases}, \tag{37}$$

$$S_i^{l}[t] = m_i^{l}[t]\,\Theta\left(U_i^{l}[t] - B_i^{l}[t]\right), \tag{38}$$

with

$$m_i^{l}[t] = \begin{cases} 1, & \text{if} \quad 0 \le (t - \lfloor \varphi_i^{l} c_i^{l} \rfloor) \bmod c_i^{l} < \lfloor d_i^{l} c_i^{l} \rfloor \\ 0, & \text{otherwise} \end{cases}, \tag{39}$$

where $\rho_{i1} = exp(-1/\tau_{a1,i})$, $\rho_{i2} = exp(-1/\tau_{a2,i})$. b_{i1}^{l} and b_{i2}^{l} update independently upon the output spike S_i^{l}, and governed by two decaying factors ρ_{i1}, ρ_{i2}. The hyperparameters β_1 and β_2 in equation (35) represent the magnitude of threshold adaptation. In our STORE-RECALL task, β_1 and β_2 are set to 1.8 and 9, respectively.

For ALIF and Rhythm-ALIF models, the membrane potential decay time constant and adaptive threshold time constant are set to 20 ms and 600 ms, respectively. For DEXAT and Rhythm-DEXAT, the membrane potential decay time constant and the two adaptive threshold time constants are set to 20 ms, 30 ms, and 600 ms, respectively. Details about the utilized model setups and the training configurations can be found in Table S2. The comparison of ALIF and Rhythm-ALIF employed in this task is displayed in Fig. S5.

Table S2. Training configurations and hyperparameter settings for the STORE-RECALL task.

Model	Epoch	Batch Size	Learning Rate	Architecture*	Rhythm Period	Duty Cycle	Phase
ALIF	200	64	$5e-2$	100-R20-R20-R20-2	-	-	-
Rhythm-ALIF	200	64	$5e-2$	100-R20-R20-R20-2	[1, 5]	[0.1, 0.8]	[0, 0.5]
DEXAT	200	64	$5e-2$	100-R20-R20-R20-2	-	-	-
Rhythm-DEXAT	200	64	$5e-2$	100-R20-R20-R20-2	[1, 5]	[0.1, 0.8]	[0, 0.5]

* The prefix 'R' in the network architecture represents the recurrent connection.

We designed a more challenging delayed recall task to evaluate the memory capacity of Rhythm-SNNs and their non-Rhythm counterparts. In this task, the model receives a temporally encoded spike pattern and must recall the input pattern after a specified delay. The input patterns are generated by temporally concatenating 4-dimensional column vectors, each consisting of random values of 0 (white) and 1 (black), presented for a duration of 1 time step to signify the presence of spikes. By concatenating these column vectors over time, we create more complex spatiotemporal input patterns, thereby increasing the difficulty of memory recall. The delay period, defined as the interval between the last input vector and the first recall cue provided to the network, is set to 10 in our experiments.

Fig. S7 illustrates an example of this delayed recall task, comparing ALIF[5] and Rhythm-ALIF models. It can be observed that, after the delay period, the ALIF model fails to recall the original input patterns accurately, whereas Rhythm-ALIF successfully recalls them without errors.

Fig. S8 shows the recall accuracy of ALIF[5] and Rhythm-ALIF as the number of input patterns increases. The shaded areas indicate standard deviation across three runs with different random seeds. As shown in Fig. S8, Rhythm-ALIF maintains a recall accuracy exceeding 70%, even with a significantly large number of input patterns (2^{24}), whereas the recall accuracy of the ALIF model drops below 40%. These findings demonstrate that incorporating the proposed rhythmic modulation mechanism significantly improves the memory capacity of the model. The membrane potential decay time constant and adaptive threshold time constant are set to 20 ms and 700 ms for both the ALIF and Rhythm-ALIF models. The details of the model setups and training configurations used for the delayed recall task are provided in Table S3.

Table S3. Training configurations and hyperparameter settings for the delayed recall task.

Model	Epoch	Batch Size	Learning Rate	Architecture*	Rhythm Period	Duty Cycle	Phase
ALIF	100	50000	$1e-4$	5-R512-R512-5	-	-	-
Rhythm-ALIF	100	50000	$1e-4$	5-R512-R512-5	[1, 10]	[0.8, 0.9]	[0, 0.5]

* The prefix 'R' in the network architecture represents the recurrent connection.

S3 Details of experimental settings for robustness evaluation tasks

We assess the robustness of the Rhythm-SNN using input Gaussian noise, network-related noise, and adversarial attacks (i.e., FGSM and PGD). Following the temporal processing experiments, we evaluated the robustness of the ASRNN[5] and Rhythm-ASRNN (as described in equations (15)–(22)) model on the PS-MNIST dataset. The models undergo testing under different levels of perturbation.

Gaussian Noise: We introduced Gaussian noise to perturb the testing data. In our experiments, we generated the noise from the Gaussian distribution with zero mean and variance ranging from $(2/255)^2$ to $(8/255)^2$. Specifically, the perturbed input is expressed as:

$$\mathbf{x}_{noise} = \mathbf{x}_{original} + \mathcal{N}(0, \sigma^2), \tag{40}$$

where $\mathbf{x}_{noise}$ and $\mathbf{x}_{original}$ denotes the noisy data and original data, respectively, $\mathcal{N}$ is the Gaussian distribution function, and $\sigma^2 = (\varepsilon/255)^2$ represents the variance of the Gaussian noise. In this work, we sampled ε at equal intervals from 2 to 8 (i.e., 2, 4, 6, and 8), corresponding to noise level 1 to 4, respectively.

For network-related noises, we have considered thermal noise, silence noise, and quantization noise that are commonly available in analog neuromorphic systems. Specifically, we follow the noise model used in a previous work[6].

Thermal Noise: Thermal noise is an intrinsic characteristic of analog neuromorphic networks and can be modeled as Gaussian noise affecting the input currents $I_0 \sim \mathcal{N}(I, \sigma I)$. In this experiment, we simulate varying levels of thermal noise by adjusting σ at equal intervals from 0.05 to 0.2 (i.e., 0.05, 0.1, 0.15, and 0.2), corresponding to noise level 1 to 4.

Silence Noise: Neuron silence noise occurs when a subset of spiking neurons fails to respond, disrupting the network's dynamics. We simulate this by randomly masking neuron outputs with failure rates ranging from 5% to 20%.

Quantization Noise: Analog neuromorphic networks often have limited analog states, restricting the bit resolution of network parameters. To simulate quantization noise, we perform post-training quantization, reducing the bit number progressively from 8 down to 2.

The adversarial attacks consider a perturbation, δ, around original data $\mathbf{x}_{oroginal}$. This can be formulated as an optimization problem:

$$\arg\max_{\mathbf{x}} \mathcal{L}((f(\mathbf{x}_{original} + \delta); \mathbf{W}), \hat{\mathbf{y}}) \quad \text{s.t.} ||\delta||_p \leq \varepsilon, \tag{41}$$

where $\hat{\mathbf{y}}$ is the target, $\mathcal{L}$ is the loss function, f denotes the network model with $\mathbf{W}$ as parameters, and ε controls the magnitude of perturbation, which guarantees the perturbation is imperceptible.

Fast Gradient Sign Method (FGSM): It perturbs the input data in the direction of the gradient of the loss function with respect to the input data[7]. The adversarially perturbed input can be described as:

$$\mathbf{x}_{noise} = \mathbf{x}_{original} + \varepsilon \times sign(\Delta_{\mathbf{x}} \mathcal{L}(\mathbf{x}_{original}, \hat{\mathbf{y}})), \tag{42}$$

where *sign* denotes the sign function, and $\Delta_{\mathbf{x}}\mathcal{L}$ is the derivative of the loss $\mathcal{L}$ with respect to input $\mathbf{x}$.

Projected Gradient Descent (PGD):

PGD introduces perturbations across multiple time steps[8]. In PGD, the FGSM operation is applied with a specified step size, followed by projecting the perturbed input back into a permissible set. The adversarial example generation is defined as:

$$\mathbf{x}_{t+1} = \Pi_{\varepsilon}(\mathbf{x}_t + \gamma \times sign(\Delta_{\mathbf{x}}\mathcal{L}(\mathbf{x}_{original}, \hat{\mathbf{y}})), \tag{43}$$

where γ is the step size for every iteration, which is set as 2.55 in this work. Π_{ε} denotes the projection operation of the data in each iteration onto the space of the l_p ball centered at $\mathbf{x}$ with radius ε, and t represents the iteration instance.

The robustness of the models is quantified by their classification accuracy in response to each type of perturbation. It is important to note that while these experiments provide valuable insights into the model's robustness, no defense mechanisms were integrated into the Rhythm-SNN during this evaluation. Therefore, the results presented reflect the inherent robustness of the model without the influence of any tailored defenses.

S4 Details of experimental settings for the speech enhancement task

To demonstrate the practical benefits of our proposed method, we utilize the speech enhancement task to validate the model's energy efficiency and performance in real-world scenarios that demand superior temporal processing capabilities, low latency, and high power efficiency.

Dataset: The adopted Microsoft DNS Challenge dataset is a corpus of human speech audio samples, which includes English, German, French, Spanish, and Russian, along with various noise types. The training dataset includes 500 hours in total (60,000 audio samples), with a synthesized signal-to-noise ratio (SNR) ranging from 20 dB to -5 dB, recorded at 16 kHz and a 16-bit depth.

Speech enhancement pipeline: The overall network architecture of Rhythm-GSNN is illustrated in Fig. S9. The Rhythm-GSNN consists of one Fullband SNN module and three Subband SNN modules, each composed of two layers of Rhythm-GSN and one linear layer. A key innovation of this design is the incorporation of our proposed rhythmic gating signal, which modulates the membrane potential updates and firing activity of the Rhythm-GSN, as indicated by the red lines in Fig. S9. This rhythmic gating mechanism enhances the model's capacity to process multiscale patterns in noisy speech signals effectively by introducing temporal adaptability. Regarding the denoising pipeline, the Rhythm-GSNN employs a multi-stage processing approach. First, the noisy audio waveform is encoded by performing a Short-Time Fourier Transform (STFT), configured with a window length of 32 ms, a hop length of 8 ms, and a Hanning window function. This step transforms the time-domain waveform into a time-frequency representation. The encoded signals are then processed sequentially through one Fullband SNN module and three Subband SNN modules. The Fullband SNN module operates on the global spectral features of the noisy speech, performing a coarse denoising step that captures the overall structure and energy distribution of the signal. Subsequently, the Subband SNN modules perform more fine-grained processing by focusing on specific subband frequency ranges. These modules are designed to exploit the rhythmic and temporal dynamics present within different frequency bands, thereby improving the separation of speech from noise and enhancing the auditory quality of the output. Finally, the denoised signals generated by the Subband SNN modules are decoded back into audio signals using an inverse STFT (iSTFT) decoder. This decoding step reconstructs both the temporal and spectral information of the enhanced speech signal, ensuring that the output waveform maintains high fidelity to the original clean speech signal. The modular design of Rhythm-GSNN enables it to efficiently handle complex noise environments and adapt to variations in speech characteristics.

Metrics: we employ the Scale-Invariant Source-to-Noise Ratio (SI-SNR) as the audio quality metric, which is a widely used metric in the audio processing domain[9]. In terms of perceptual audio quality assessment, we adopt the DNSMOS[10] metric to validate the performance of the speech enhancement solution. DNSMOS includes three values: Overall Audio Quality (OVR)[10], Speech Signal Quality (SIG)[10], and Background Noise Quality (BAK)[10]. The pairing of SI-SNR and DNSMOS provides an objective and perceptual audio quality evaluation. In addition to the SI-SNR and DNSMOS metrics, the energy cost is taken into consideration for this task to compare different speech enhancement approaches.

Training configurations: The detailed training configurations for the speech enhancement task are provided as follows:
Denoising Samples: More listening samples of Intel DNS Challenge are provided in this Google Drive folder (est.wav is our model's denoising result, raw.wav and ref.wav are raw and clean audios, respectively).

Table S4. Training configurations and hyperparameter settings for the speech enhancement task.

Model	Epoch	Batchsize	Learning Rate	Optimizer	Rhythm Period	Duty Cycle	Phase
GSNN	200	64	$1e-3$	AdamW	-	-	-
Rhythm-GSNN	200	64	$1e-3$	AdamW	[10, 50]	[0.05, 0.10]	[0, 0.5]

S5 Details of derivations regarding perturbation robustness

In this section, we first recall the spiking neuron model for our proposed Rhythm-SNN. Then, we theoretically analyze the robustness of Rhythm-SNN against perturbations by comparing the representation distance between the output spike trains in response to the original patterns and the corresponding perturbed patterns.

Firstly, the computational model of our proposed Rhythm-SNN with LIF neurons can be reformulated as follows:

$$\text{Input current updating} : \mathbf{I}^l[t] = \mathbf{W}^l \mathbf{s}^{l-1}[t], \tag{44}$$

where $\mathbf{I}^l[t]$ denotes the input current in layer l at time step t. $\mathbf{W}^l$ represents the connection weight in layer l, and $\mathbf{s}^{l-1}[t]$ indicates the emitted spike from the preceding layer $l-1$ at time step t. The membrane potential dynamics can be formalized as:

$$\text{Potential updating} : \mathbf{U}^l[t] = \mathbf{V}^l[t-1] + \mathbf{m}^l[t]\mathbf{I}^l[t], \tag{45}$$

where $\mathbf{V}^l[t-1]$ and $\mathbf{U}^l[t]$ represent the membrane potential in layer l before modulation at time step $t-1$ and after modulation at time step t, respectively. Then, our proposed oscillatory signal $\mathbf{m}$ will modulate the neuron's firing activities as follows:

$$\text{Spike generation} : \mathbf{s}^l[t] = \mathbf{m}^l[t]\Theta\left(\mathbf{U}^l[t] - \vartheta\right), \tag{46}$$

with

$$\text{Oscillatory signal} : \mathbf{m}^l[t] = \begin{cases} 1, & \text{if} \quad 0 \le (t - \lfloor \varphi^l c^l \rfloor) \bmod c^l < \lfloor d^l c^l \rfloor \\ 0, & \text{otherwise} \end{cases}, \tag{47}$$

where ϑ stands for the neuronal firing threshold, and $\mathbf{m}^l[t]$ denotes the oscillatory signal in layer l at time step t. φ^l signifies the phase of the modulating signal, c^l and d^l denotes the rhythm period and duty cycle of the modulating signal, respectively. $\lfloor \cdot \rfloor$ denotes the floor function. $\Theta(\cdot)$ is the Heaviside step function which is defined as $\Theta(x) = 1$ for $x \ge 0$ and $\Theta(x) = 0$ for $x < 0$.

$$\text{Potential reset} : \mathbf{V}^l[t] = \left((\alpha - 1)\mathbf{m}^l[t] + 1\right)\mathbf{U}^l[t] - \vartheta \mathbf{m}^l[t]\mathbf{s}^l[t], \tag{48}$$

where α represents the decay factor of the membrane potential. For notational simplicity, we introduce the variable $\boldsymbol{\alpha}_m^l[t]$ defined as follows:

$$\boldsymbol{\alpha}_m^l[t] = (\alpha - 1)\mathbf{m}^l[t] + 1 = \begin{cases} \alpha, & \text{if} \quad \mathbf{m}^l[t] = 1 \\ 1, & \text{if} \quad \mathbf{m}^l[t] = 0 \end{cases} \tag{49}$$

Therefore, equation (48) can be reformulated as:

$$\mathbf{V}^l[t] = \boldsymbol{\alpha}_m^l[t]\mathbf{U}^l[t] - \vartheta \mathbf{m}^l[t]\mathbf{s}^l[t]. \tag{50}$$

We now introduce the upper and lower bounds of the input current, followed by the derivation of the spiking Lipschitz constant for our proposed Rhythm-SNN. This constant provides an effective upper bound for perturbation estimation.

From equations (44), (45) and (50), equation (50) can be rewritten as:

$$\mathbf{V}^l[t] = \boldsymbol{\alpha}_m^l[t]\mathbf{V}^l[t-1] + \boldsymbol{\alpha}_m^l[t]\mathbf{m}^l[t]\mathbf{W}^l\mathbf{s}^{l-1}[t] - \vartheta \mathbf{m}^l[t]\mathbf{s}^l[t]. \tag{51}$$

By shifting the term $\boldsymbol{\alpha}_m^l[t]\mathbf{V}^l[t-1]$ in equation (51) to the left, we have:

$$\mathbf{V}^l[t] - \boldsymbol{\alpha}_m^l[t]\mathbf{V}^l[t-1] = \boldsymbol{\alpha}_m^l[t]\mathbf{m}^l[t]\mathbf{W}^l\mathbf{s}^{l-1}[t] - \vartheta \mathbf{m}^l[t]\mathbf{s}^l[t], \tag{52}$$

Accumulating equation (52) from $t = 0$ to T, yields:

$$\begin{aligned}\mathbf{V}^l[t] &= \mathbf{W}^l \sum_{i=1}^{t} \left(\prod_{j=i}^{t} \boldsymbol{\alpha}_m^l[j] \right) \mathbf{m}^l[i]\mathbf{s}^{l-1}[i] - \vartheta \sum_{i=1}^{t} \left(\prod_{j=i+1}^{t} \boldsymbol{\alpha}_m^l[j] \right) \mathbf{m}^l[i]\mathbf{s}^l[i] \\ &= \mathbf{W}^l \sum_{i=1}^{t} \left(\prod_{j=i}^{t} \boldsymbol{\alpha}_m^l[j] \right) \mathbf{m}^l[i]\mathbf{s}^{l-1}[i] - \vartheta \sum_{i=1}^{t} \left(\prod_{j=i}^{t} \boldsymbol{\alpha}_m^l[j] \right) \frac{1}{\boldsymbol{\alpha}_m^l[i]} \mathbf{m}^l[i]\mathbf{s}^l[i] \\ &= \mathbf{W}^l \sum_{i=1}^{t} \left(\prod_{j=i}^{t} \boldsymbol{\alpha}_m^l[j] \right) \mathbf{m}^l[i]\mathbf{s}^{l-1}[i] - \frac{\vartheta}{\alpha} \sum_{i=1}^{t} \left(\prod_{j=i}^{t} \boldsymbol{\alpha}_m^l[j] \right) \mathbf{m}^l[i]\mathbf{s}^l[i].\end{aligned} \tag{53}$$

where the initial membrane potential $\mathbf{V}^l[0]$ is set as zero.

To simplify further derivations, we introduce the notation as follows:

$$h^l[t] = \sum_{i=1}^{t} \left(\prod_{j=i}^{t} \boldsymbol{\alpha}_m^l[j] \right) \mathbf{m}^l[i]\mathbf{s}^l[i]. \tag{54}$$

Thus, the spike train $\mathbf{s}^l$ can be formulated as:

$$\mathbf{s}^l[t] = \frac{\mathbf{m}^l[t]}{\alpha} \left(h^l[t] - h^l[t-1] \right). \tag{55}$$

The upper bound of the input current in layer l can be expressed by:

$$\eta^l = \sup_{\mathbf{s}\neq 0, \mathbf{s}\in\Omega^{N_{l-1}}} \left\| \mathbf{W}^l\mathbf{s} \right\|_\infty, \tag{56}$$

where $\Omega = \{0,1\}$ and N_{l-1} denotes the number of hidden neurons in layer $l-1$.

Similarly, the lower bound of the input current in layer l is set by:

$$\mu^l = \begin{cases} \inf\limits_{\mathbf{s}\neq 0, \mathbf{s}\in\Omega^{N_{l-1}}} \left\| \mathbf{W}^l\mathbf{s} \right\|_\infty, & \text{if} \quad \mathbf{W}^l \text{ does not include negative items} \\ -\sup\limits_{\mathbf{s}\neq 0, \mathbf{s}\in\Omega^{N_{l-1}}} \left\| -\mathbf{W}^l\mathbf{s} \right\|_\infty, & \text{otherwise} \end{cases}. \tag{57}$$

According to the proposed rhythm mechanism, spiking neurons in Rhythm-SNN neither accept the input current from other pre-neurons nor fire spikes to post-neurons during the phase when the oscillatory signal $\mathbf{m}^l[t]$ equals zero. Hence, the accumulated membrane potential $\mathbf{V}^l[t]$ is constrained to be less than $\eta^l d^l t$. Consider the case where the neuron receives negative input currents all the time, $\mathbf{V}^l[t]$ is supposed to be no less than $\mu^l d^l t$. Thus, according to equation (53) and (54), we have:

$$\mu^l d^l t \leq \mathbf{W}^l h^{l-1}[t] - \frac{\vartheta}{\alpha} h^l[t] \leq \eta^l d^l t. \tag{58}$$

Deriving from equation (58), we obtain:

$$\frac{\alpha}{\vartheta} \left(\mathbf{W}^l h^{l-1}[t] - \eta^l d^l t \right) \leq h^l[t] \leq \frac{\alpha}{\vartheta} \left(\mathbf{W}^l h^{l-1}[t] - \mu^l d^l t \right). \tag{59}$$

Hence, according to equation (55) and (59), we deduce the distance between the original spike $\mathbf{s}^l[t]$ and the perturbed spike $\hat{\mathbf{s}}^l[t]$ as:

$$\begin{aligned}\left\| \mathbf{s}^l[t] - \hat{\mathbf{s}}^l[t] \right\|_2 &= \frac{\mathbf{m}^l[t]}{\alpha} \left\| h^l[t] - h^l[t-1] - \hat{h}^l[t] + \hat{h}^l[t-1] \right\|_2 \\ &\leq 2\mathbf{m}^l[t] \frac{(\eta^l - \mu^l) d^l t}{\vartheta} + \frac{\mathbf{m}^l[t]}{\alpha} \left\| \frac{\alpha}{\vartheta} \left(\mathbf{W}^l h^{l-1}[t] - \mathbf{W}^l h^{l-1}[t-1] - \mathbf{W}^l \hat{h}^{l-1}[t] + \mathbf{W}^l \hat{h}^{l-1}[t-1] \right) \right\|_2 \\ &\leq 2\mathbf{m}^l[t] \frac{(\eta^l - \mu^l) d^l t}{\vartheta} + \frac{\alpha}{\vartheta} \left\| \frac{\mathbf{m}^l[t]}{\alpha} \mathbf{W}^l \left(h^{l-1}[t] - h^{l-1}[t-1] \right) + \frac{\mathbf{m}^l[t]}{\alpha} \mathbf{W}^l \left(\hat{h}^{l-1}[t] - \hat{h}^{l-1}[t-1] \right) \right\|_2 \\ &\leq 2\mathbf{m}^l[t] \frac{(\eta^l - \mu^l) d^l t}{\vartheta} + \frac{\alpha}{\vartheta} \left\| \mathbf{W}^l \left(\mathbf{s}^{l-1}[t] - \hat{\mathbf{s}}^{l-1}[t] \right) \right\|_2.\end{aligned} \tag{60}$$

Meanwhile, we have:

$$\left\|\mathbf{s}^l[t]-\hat{\mathbf{s}}^l[t]\right\|_2 \leq 1, \tag{61}$$

and

$$\frac{\alpha}{\vartheta}\left\|\mathbf{W}^l\left(\mathbf{s}^{l-1}[t]-\hat{\mathbf{s}}^{l-1}[t]\right)\right\|_2 \leq \frac{\alpha\left(\eta^l-\mu^l\right)}{\vartheta}. \tag{62}$$

Combining equations (61) and (62), we have:

$$\left\|\mathbf{s}^l[t]-\hat{\mathbf{s}}^l[t]\right\|_2+\frac{\alpha}{\vartheta}\left\|\mathbf{W}^l\left(\mathbf{s}^{l-1}[t]-\hat{\mathbf{s}}^{l-1}[t]\right)\right\|_2 \leq 1+\frac{\alpha\left(\eta^l-\mu^l\right)}{\vartheta}. \tag{63}$$

Shifting the term $\frac{\alpha}{\vartheta}\left\|\mathbf{W}^l\left(\mathbf{s}^{l-1}[t]-\hat{\mathbf{s}}^{l-1}[t]\right)\right\|_2$ in equation (60) to the left and then multiplying it with equation (63) yields:

$$\left\|\mathbf{s}^l[t]-\hat{\mathbf{s}}^l[t]\right\|_2^2-\frac{\alpha^2}{\vartheta^2}\left\|\mathbf{W}^l\left(\mathbf{s}^{l-1}[t]-\hat{\mathbf{s}}^{l-1}[t]\right)\right\|_2^2 \leq \frac{2\mathbf{m}^l[t]t}{\alpha}\left[\frac{\alpha\left(\eta^l-\mu^l\right)d^l}{\vartheta}+\frac{\alpha^2\left(\eta^l-\mu^l\right)^2d^l}{\vartheta^2}\right]. \tag{64}$$

For layer l, the calculation of $\left\|\mathbf{s}^l[t]-\hat{\mathbf{s}}^l[t]\right\|_2^2$ should add the N_l nodes up, which yields:

$$\begin{aligned}\left\|\mathbf{s}^l[t]-\hat{\mathbf{s}}^l[t]\right\|_2^2 &= \sum_{i=1}^{N_l}\left|s_i^l[t]-\hat{s}_i^l[t]\right|^2 \\ &\leq \frac{\alpha^2}{\vartheta^2}\left\|\mathbf{W}^l\left(\mathbf{s}^{l-1}[t]-\hat{\mathbf{s}}^{l-1}[t]\right)\right\|_2^2+\frac{2N_l\mathbf{m}^l[t]t}{\alpha}\left[\frac{\alpha\left(\eta^l-\mu^l\right)d^l}{\vartheta}+\frac{\alpha^2\left(\eta^l-\mu^l\right)^2d^l}{\vartheta^2}\right].\end{aligned} \tag{65}$$

The l_2 norm of input currents from the preceding layer $l-1$ is bounded by its matrix norm induced by l_2 norm:

$$\left\|\mathbf{W}^l\left(\mathbf{s}^{l-1}[t]-\hat{\mathbf{s}}^{l-1}[t]\right)\right\|_2^2 \leq \sup_{\|\mathbf{s}\|_2\leq 1,\mathbf{s}\in\Phi^{N_{l-1}}}\left\|\mathbf{W}^l\mathbf{s}\right\|_2^2\left\|\mathbf{s}^{l-1}[t]-\hat{\mathbf{s}}^{l-1}[t]\right\|_2^2, \tag{66}$$

where $\Phi=\{-1,0,1\}$. Hence, the l_2 distance between the output spike trains in response to the original patterns and the perturbed patterns can be described as the following form:

$$\begin{aligned}D_2\left(\mathbf{S}^l,\hat{\mathbf{S}}^l\right)^2 &= \left\|\mathbf{S}^l-\hat{\mathbf{S}}^l\right\|_2^2 \\ &= \sum_{t=1}^{T}\left\|\mathbf{s}^l[t]-\hat{\mathbf{s}}^l[t]\right\|_2^2 \\ &\leq \frac{1}{\vartheta^2}(\tilde{\Lambda}^l)^2D_2\left(\mathbf{S}^{l-1},\hat{\mathbf{S}}^{l-1}\right)^2+\tilde{\Gamma}^l,\end{aligned} \tag{67}$$

with

$$\tilde{\Lambda}^l=\alpha\sup_{\|\mathbf{s}\|_2\leq 1,\mathbf{s}\in\Phi^{N_{l-1}}}\left\|\mathbf{W}^l\mathbf{s}\right\|_2, \tag{68}$$

$$\tilde{\Gamma}^l=\sum_{t=1}^{T}\frac{2N_l\mathbf{m}^l[t]t}{\alpha}\left[\frac{\alpha\left(\eta^l-\mu^l\right)d^l}{\vartheta}+\frac{\alpha^2\left(\eta^l-\mu^l\right)^2d^l}{\vartheta^2}\right]. \tag{69}$$

Similarly, by following the derivation in prior work[11], we derive the l_2 distance between the original and perturbed spike trains for the conventional SNN without rhythm mechanisms as

$$D_2\left(\mathbf{S}^l,\hat{\mathbf{S}}^l\right)^2 \leq \frac{1}{\vartheta^2}(\Lambda^l)^2D_2\left(\mathbf{S}^{l-1},\hat{\mathbf{S}}^{l-1}\right)^2+\Gamma^l, \tag{70}$$

with

$$\Lambda^l=\sup_{\|\mathbf{s}\|_2\leq 1,\mathbf{s}\in\Phi^{N_{l-1}}}\|\mathbf{W}^l\mathbf{s}\|_2, \tag{71}$$

$$\Gamma^l = \frac{N_l T\,(T+1)}{\alpha}\left[\frac{\eta^l-\mu^l}{\vartheta}+\left(\frac{\eta^l-\mu^l}{\vartheta}\right)^2\right]. \tag{72}$$

Given that $0<\alpha<1$, we have $\tilde{\Lambda}^l<\Lambda^l$. This implies that our proposed Rhythm-SNN possesses a reduced spiking Lipschitz constant compared to a conventional SNN without rhythm mechanisms. According to prior works[11–14], a smaller spiking Lipschitz constant implies a decreased magnitude of network output perturbations. Therefore, it demonstrates the robustness enhancement provided by the proposed rhythm mechanism.

S6 Impact of rhythmic hyperparameters on Rhythm-SNNs

We evaluate the impact of the rhythmic hyperparameter of Rhythm-SNN with both feedforward and recurrent architectures on the PS-MNIST dataset. As shown in Fig. S13a, for Rhythm-FFSNN, a smaller rhythm period results in decreased performance, possibly due to insufficiently rich oscillation patterns. Fig. S13b reveals that Rhythm-FFSNN's performance declines with an increased duty cycle, indicating that sparser neuronal activities help enhance the model's sequence modeling capability. Similarly, for Rhythm-SRNN, a smaller rhythm period and a larger duty cycle lead to performance degradation. Compared to Rhythm-FFSNN, Rhythm-SRNN exhibits a more stable performance and is less sensitive to the hyperparameter distribution. Fig. S13c shows that the phase parameter has a lesser impact on the performance of both Rhythm-FFSNN and Rhythm-SRNN compared to the other two hyperparameters.

To comprehensively evaluate the impact of oscillatory signal heterogeneity on the performance of Rhythm-SNNs, we conducted a series of experiments by varying the range of the Rhythm Period, Duty Cycle, and Phase based on a vision dataset (PS-MNIST) and a text dataset (PTB). The detailed results are summarized as follows:

- **Rhythm Period**: As illustrated in Figure S14a and d, the performance of Rhythm-SNNs improves as the heterogeneity level increases, peaking when the periods are randomly drawn from the "10-30" range, after which it saturates. This performance saturation is likely due to the sufficient temporal span provided by this period range, considering that the input sequence lengths are only 784 and 200 for the PS-MNIST and PTB datasets, respectively.
- **Duty Cycle**: The duty cycle controls the duration for which neurons remain active, with higher values indicating a longer "On" period. As illustrated in Figure S14b and e, the performance of Rhythm-SNNs improves as the level of heterogeneity of duty cycle increases, as this facilitates information of different temporal scale to be propagated through the network.
- **Phase**: To elucidate how the heterogeneity level of the Phase affects model performance at different Rhythm Periods, we employed different settings of the rhythm period for the PS-MNIST and PTB datasets. For the PS-MNIST dataset, we set the rhythm period uniformly at a value of 10 for all neurons. As illustrated in Figure S14c, when the heterogeneity level of the Rhythm Period is low, the model's performance improves as the heterogeneity level of the Phase increases. In contrast, when the heterogeneity level of the Rhythm Period is sufficiently high (i.e., Rhythm Periods spanning from 10 to 50 on the PTB dataset), the model's performance shows only subtle variations with respect to the heterogeneity level of the Phase, as demonstrated in Figure S14f.

S7 Energy cost comparison

We provide a detailed comparison of energy cost between LSTM, conventional SNNs, and Rhythm-SNNs based on Synaptic Operations (SynOps) and Neuron Operations (NeuOps) incurred during data processing and neuron updates. The SynOps consists of Accumulate (AC) and Multiply-And-Accumulate (MAC) operations. As indicated in Table S5, the LSTM model predominantly utilizes MAC operations for computation. Conversely, SNNs mainly utilize AC operations. Compared with conventional SNNs, the NeuOps of Rhythm-SNNs can be considerably decreased by tuning down the duty cycle of modulating signals. By leveraging oscillatory modulating signals, this periodic deactivation of spiking neurons in Rhythm-SNNs significantly reduces the required operations, enhancing models' energy efficiency without sacrificing performance.

Table S5. Comparison of the energy cost estimation of LSTM, FFSNN, SRNN, LSNN, ASRNN, and their rhythm counterparts. N_{in} and N_{out} denote the numbers of input and output neurons. Fr_{in} and Fr_{out} are the firing rates of input and output neurons, respectively. E_{AC}, E_{MAC}, and E_{NeuOp} represent the energy costs of MAC and AC synaptic operations and neuron operations. $\overline{a}$ is the average firing rate of spiking neurons and $0 < \overline{a} < 1$ for Rhythm-SNNs. Compared with conventional SNNs, Rhythm-SNN possess lower firing rates (i.e., Fr_{in}, and Fr_{out}) and NeuOps (corresponding to $\overline{a}$), thus drastically reducing the computational energy cost.

Model	SynOps Energy	NeuOps Energy
LSTM	$4(N_{in}N_{out}+N_{out}N_{out})E_{MAC}+17N_{out}E_{MAC}$	-
FFSNN	$N_{in}N_{out}Fr_{in}E_{AC}+N_{out}Fr_{out}E_{AC}$	$N_{out}E_{NeuOp}$
Rhythm-FFSNN	$N_{in}N_{out}Fr_{in}E_{AC}+N_{out}Fr_{out}E_{AC}$	$\overline{a}N_{out}E_{NeuOp}$
SRNN	$N_{in}N_{out}Fr_{in}E_{AC}+(N_{out}N_{out}+N_{out})Fr_{out}E_{AC}$	$N_{out}E_{NeuOp}$
Rhythm-SRNN	$N_{in}N_{out}Fr_{in}E_{AC}+(N_{out}N_{out}+N_{out})Fr_{out}E_{AC}$	$\overline{a}N_{out}E_{NeuOp}$
LSNN	$N_{in}N_{out}Fr_{in}E_{AC}+(N_{out}N_{out}+2N_{out})Fr_{out}E_{AC}+3N_{out}E_{MAC}$	$N_{out}E_{NeuOp}$
Rhythm-LSNN	$N_{in}N_{out}Fr_{in}E_{AC}+(N_{out}N_{out}+2N_{out})Fr_{out}E_{AC}+3N_{out}E_{MAC}$	$\overline{a}N_{out}E_{NeuOp}$
ASRNN	$N_{in}N_{out}Fr_{in}E_{AC}+(N_{out}N_{out}+2N_{out})Fr_{out}E_{AC}+3N_{out}E_{MAC}$	$N_{out}E_{NeuOp}$
Rhythm-ASRNN	$N_{in}N_{out}Fr_{in}E_{AC}+(N_{out}N_{out}+2N_{out})Fr_{out}E_{AC}+3N_{out}E_{MAC}$	$\overline{a}N_{out}E_{NeuOp}$

Table S6. Detailed energy cost comparison between conventional FFSNN, SRNN, LSNN, ASRNN, and their rhythm counterparts on the PS-MNIST dataset. The activating rate denotes the proportion of time that neurons remain in the 'ON' state over the entire time span, as determined by our duty cycle parameter. For the computational costs associated with addition (AC) and multiply-accumulate (MAC) operations used in SynOps and NeuOps, we refer to[15,16]. These references demonstrate that, using 45 nm CMOS technology, a single 32-bit floating-point addition operation consumes 0.9 pJ, while a single multiply-accumulate operation consumes 4.6 pJ.

Model	Architecture[1]	Accuracy	Firing Rate			Activating Rate			SynOps Energy (pJ)			NeuOps Energy (pJ)			Total Energy Cost (pJ)
			Layer 1	Layer 2	Layer 3	Layer 1	Layer 2	Layer 3	Layer 1	Layer 2	Layer 3	Layer 1	Layer 2	Layer 3	
FFSNN	F64-F256-F256	42.96	0.2495	0.0882	0.1100	1.0000	1.0000	1.0000	71.973	3,699.35	5,227.59	294.40	1,177.60	1,177.60	11,648.51
Rhythm-FFSNN	F64-F256-F256	94.90	0.0215	0.0089	0.0132	0.0697	0.0653	0.0635	58.84	319.08	527.98	20.51	76.85	74.74	1,078.00
SRNN	R64-R256-R256	67.71	0.2580	0.1491	0.2822	1.0000	1.0000	1.0000	1,023.55	12,632.99	25,504.13	294.40	1,177.60	1,177.60	41,810.27
Rhythm-SRNN	R64-R256-R256	95.07	0.0279	0.0326	0.0225	0.0697	0.0653	0.0635	162.06	2,341.74	3,255.11	20.51	76.85	74.74	5,931.01
LSNN	R64-R256-R256	80.20	0.1487	0.1507	0.3288	1.0000	1.0000	1.0000	1,514.66	14,718.28	32,042.13	294.40	1,177.60	1,177.60	50,924.67
Rhythm-LSNN	R64-R256-R256	96.42	0.0240	0.0233	0.0336	0.0697	0.0653	0.0635	1,033.42	5,277.09	6,912.12	20.51	76.85	74.74	13,394.73
ASRNN	R64-R256-R256	88.90	0.1558	0.0859	0.0613	1.0000	1.0000	1.0000	1,542.06	10,956.13	12,257.38	294.40	1,177.60	1,177.60	27,405.17
Rhythm-ASRNN	R64-R256-R256	97.13	0.0193	0.0167	0.0284	0.0697	0.0653	0.0635	1,015.28	4,813.94	6,212.54	20.51	76.85	74.74	12,213.86

[1] The prefixes 'F' and 'R' in network architecture denote the feedforward and recurrent connections, respectively.

Table S7. Detailed comparison between GSNN and Rhythm-GSNN on Intel N-DNS Challenge.

Model	Params	SI-SNR	DNSMOS			Latency (ms)	Power Proxy (Ops/s)	PDP Proxy (Ops)	Total Energy Cost (J)
			OVR	SIG	BAK				
DCCRN[17]	1.25M	12.46	2.85	3.11	3.77	20.02	5.07G	0.10G	$0.46m$
FullSubNet[18]	1.14M	13.55	2.93	3.22	3.84	32.03	3.65G	0.12G	$0.55m$
Microsoft NsNet2[19]	2.68M	11.63	2.95	3.26	3.93	20.02	136.13M	2.72M	12.51μ
SDNN[20]	0.53M	12.26	2.70	3.20	3.45	32.02	14.52M	0.46M	0.41μ
PSNN[20]	0.72M	12.32	2.68	2.91	3.96	32.00	57.24M	1.83M	1.65μ
GSNN[21]	0.95M	14.71	3.05	3.35	3.97	32.02	80.94M	2.59M	2.33μ
Rhythm-GSNN	0.95M	15.43	3.02	3.37	3.89	32.02	39.61M	1.27M	1.14μ

Table S8. The execution performance comparison between FFSNNs and Rhythm-FFSNNs deployed on the FPGA-based neuromorphic system. The FFANN refers to the feedforward ANN using the ReLU activation function. The experiments on the CPU and GPU were conducted using an Intel(R) Xeon(R) Platinum 8350C CPU @ 2.60GHz and an NVIDIA GeForce RTX 3090 GPU card, respectively.

Dataset	Model	Architecture	Accuracy (%)	Batch Size	Duty Cycle	Latency (ms)	Avg. Event	Throughput (Sample/s)	Power (W)	Energy Efficiency (Sample/s/W)
PS-MNIST	FFANN (CPU)	64-256-256-10	-	32	-	35.30	-	28.33	83.87	0.34
PS-MNIST	FFANN (CPU)	64-256-256-10	-	64	-	18.09	-	55.28	88.26	0.63
PS-MNIST	FFANN (GPU)	64-256-256-10	-	32	-	21.99	-	45.48	20.20	2.25
PS-MNIST	FFANN (GPU)	64-256-256-10	-	64	-	10.54	-	94.90	20.32	4.67
PS-MNIST	FFSNN	64-256-256-10	41.47	1	-	5.22	182,293.66	191.6	0.703	272.55
PS-MNIST	Rhythm-FFSNN	64-256-256-10	95.01	1	[0.02, 0.10]	0.70	4,457.11	1,418.9	0.703	2,018.35
PS-MNIST	Rhythm-FFSNN	64-256-256-10	95.83	1	[0.22, 0.30]	2.38	30,549.79	419.9	0.703	597.30
PS-MNIST	Rhythm-FFSNN	64-256-256-10	94.79	1	[0.42, 0.50]	3.06	62,415.62	327.2	0.703	465.43
PS-MNIST	Rhythm-FFSNN	64-256-256-10	92.63	1	[0.62, 0.70]	3.73	97,755.64	268.4	0.703	381.79
PS-MNIST	Rhythm-FFSNN	64-256-256-10	88.98	1	[0.82, 0.90]	4.48	139,877.81	223.3	0.703	317.64
S-MNIST	FFANN (CPU)	64-256-256-10	-	32	-	35.41	-	28.24	85.43	0.33
S-MNIST	FFANN (CPU)	64-256-256-10	-	64	-	18.02	-	55.50	86.91	0.64
S-MNIST	FFANN (GPU)	64-256-256-10	-	32	-	22.91	-	43.64	20.19	2.16
S-MNIST	FFANN (GPU)	64-256-256-10	-	64	-	11.22	-	89.09	20.28	4.39
S-MNIST	FFSNN	64-256-256-10	59.24	1	-	5.10	180,103.11	196.0	0.703	278.80
S-MNIST	Rhythm-FFSNN	64-256-256-10	96.36	1	[0.02, 0.10]	1.37	8,887.16	730.3	0.703	1,038.83
S-MNIST	Rhythm-FFSNN	64-256-256-10	96.34	1	[0.22, 0.30]	2.60	42,822.87	384.4	0.703	546.80
S-MNIST	Rhythm-FFSNN	64-256-256-10	96.22	1	[0.42, 0.50]	3.27	73,926.29	305.7	0.703	434.85
S-MNIST	Rhythm-FFSNN	64-256-256-10	95.77	1	[0.62, 0.70]	4.03	114,071.43	248.3	0.703	353.20
S-MNIST	Rhythm-FFSNN	64-256-256-10	93.41	1	[0.82, 0.90]	4.76	155,695.02	209.9	0.703	298.58

S8 Hardware implementation scheme for Rhythm-SNNs

In this section, we provide a low-latency, energy-efficient FPGA-based neuromorphic hardware implementation scheme for Rhythm-SNNs. The overall architecture and main functional modules are illustrated in Fig. S15. The processing system (PS) serves as the central control unit, managing the system state and facilitating data access to external memory. The programmable logic (PL) is dedicated to accelerating the inference process of Rhythm-SNNs. The weights and event data are exchanged between the PS and PL via the AXI-Stream interface, while the PS configures the PL through the AXI-Lite interface. Within the PL, the Neuron Processing Unit (NPU) is responsible for event-driven data stream processing. The NPU module integrates weight random access memory (RAM), an event encoder, an event first in-first out (FIFO), and a rhythm generator. Specifically, the weight RAM stores the synaptic weight parameters for the corresponding layer of the Rhythm-SNN; the event FIFO buffers and receives event streams from the preceding layer, ensuring orderly data transmission; the event encoder converts the generated spike into Address-Event Representation (AER) packets, which contain a unique neuron ID and the precise timestamp at which the spike occurred; and the rhythm generator produces rhythmic gating signals for each neuron based on the timestamp information extracted from the AER packets.

The proposed hardware architecture is deployed on the AX7350B FPGA development board, which features the ZYNQ XC7Z035FFG676 SoC, integrating a dual-core ARM Cortex-A9 CPU and 1 GB DDR3 SDRAM. The board provides ample programmable resources, including 171,900 LUTs, 343,800 FFs, 500 BRAMs (36 Kb RAM), and 900 DSPs. Both the synthesis and implementation of the proposed architecture are performed using the Xilinx Vivado toolchain. The hardware operates at a typical frequency of 100 MHz, with timing optimizations employed to ensure sufficient setup and hold time margins.

Supplementary Figures

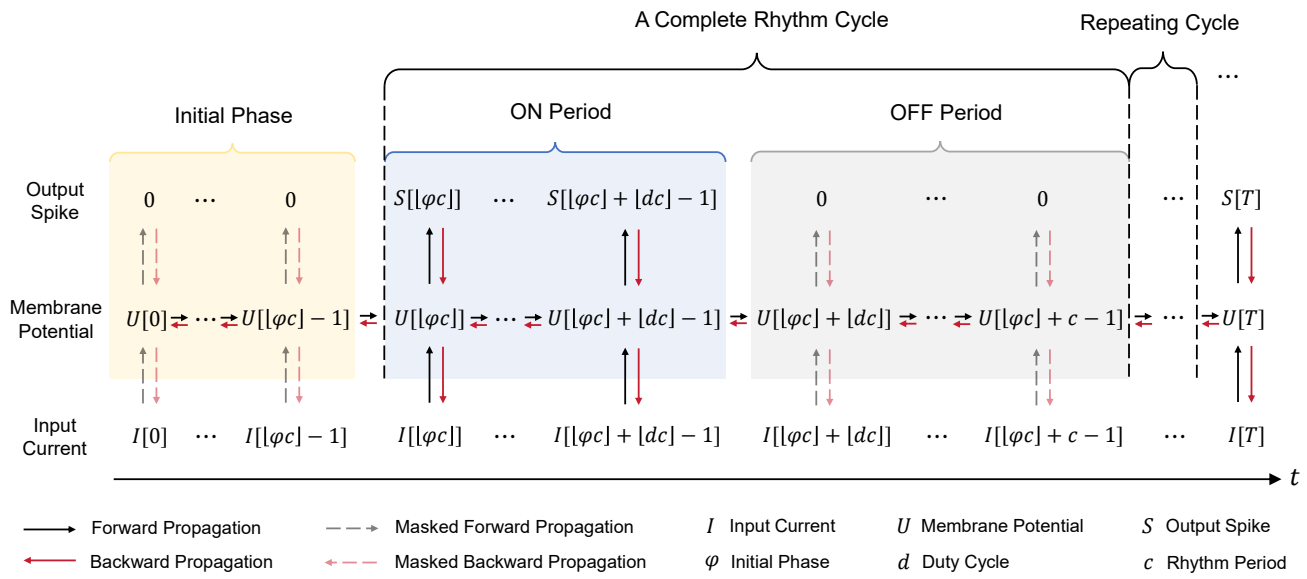

Figure S1. Illustration of the proposed rhythmic modulation mechanism in Rhythm-SNN over the temporal dimension. A complete rhythm cycle consists of an 'ON Period' and an 'OFF Period', which repeats over time (denoted as the 'Repeating Cycle'). Specifically, the rhythmic neurons initially experience a silent period (denoted as 'Initial Phase'), governed by the phase parameter φ. During this period, neurons neither emit spikes nor receive input currents until time $t = \lfloor \varphi c \rfloor$. Following this phase, neurons transition into the 'ON Period', whose duration is dictated by the duty cycle parameter d. During the 'ON Period', neurons are active, allowing them to receive input currents from pre-synaptic neurons and emit spikes when their membrane potential surpasses the threshold. After the 'ON Period', neurons enter the 'OFF Period', during which they are inactive and neither receive input currents nor emit spikes.

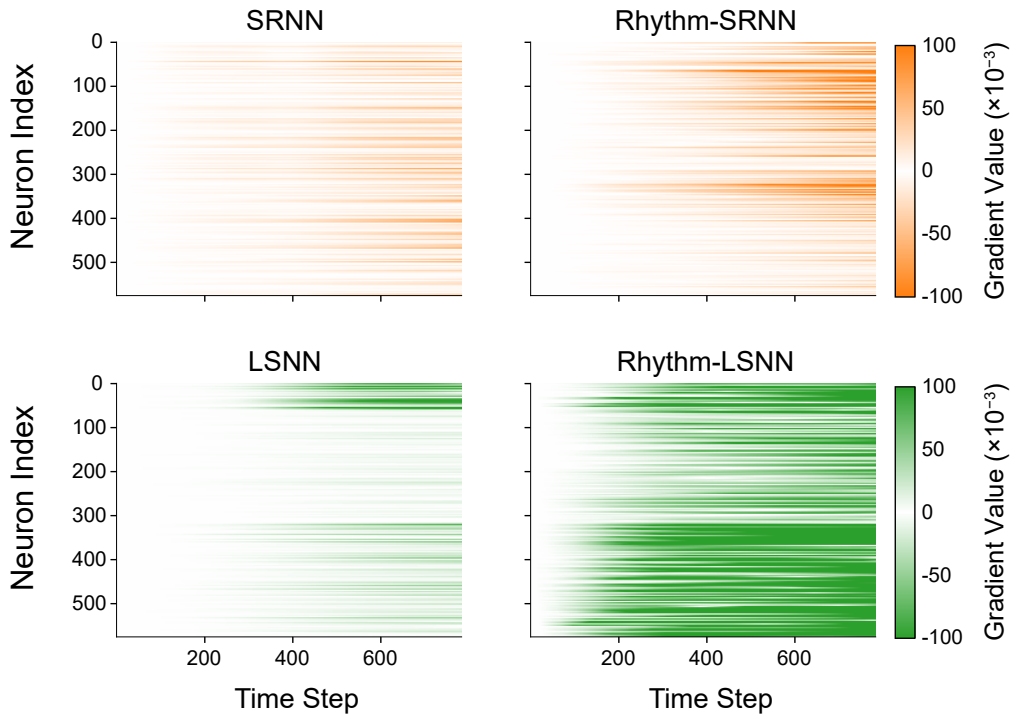

Figure S2. Visualization of normalized temporal gradients over time for SRNN and LSNN models on the PS-MNIST dataset. It can be observed that Rhythm-SRNN and Rhythm-LSNN could convey more gradients to earlier time steps compared to conventional SRNN and LSNN. It demonstrates the effectiveness of the proposed method for supporting more direct pathways for gradient flow, thereby enhancing learning long-range dependencies.

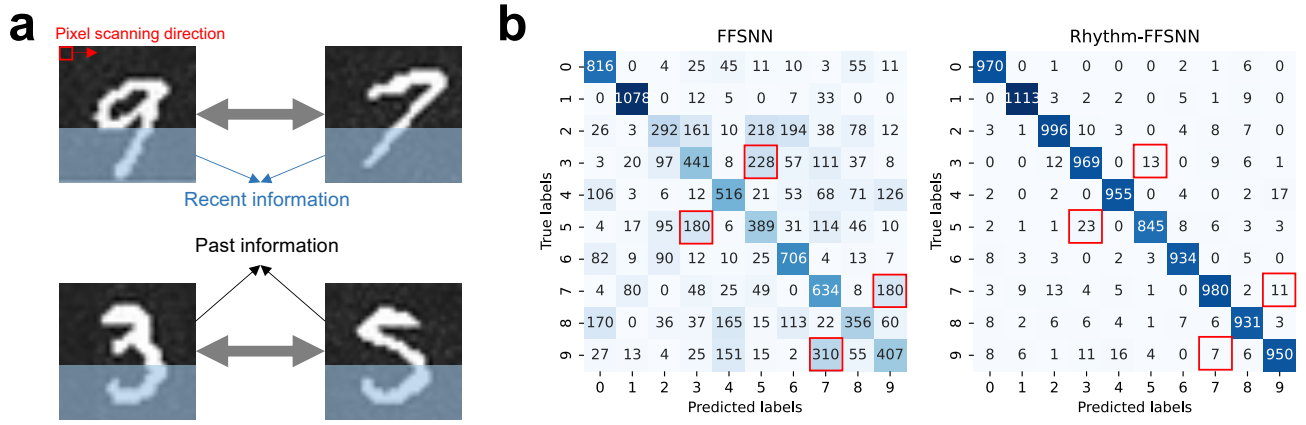

Figure S3. Illustration of how Rhythm-SNNs improve performance in sequential tasks necessitating long-range dependency. a, Illustration of two misclassification examples from the S-MNIST dataset. Note that the information is scanned into the network pixel by pixel, starting from the top-left corner and proceeding to the bottom-right corner. Conventional FFSNNs are only able to manage the short-range temporal dependencies of the most recent information (indicated in grey areas), which leads to confusion between digits such as "9" and "7", as well as "3" and "5". **b**, Comparison of the confusion matrices for FFSNN and Rhythm-FFSNN. The results indicate that Rhythm-FFSNN exhibits fewer misclassifications for these two pairs of digits, suggesting that Rhythm-FFSNN possesses enhanced modeling capacity for long-range temporal dependencies.

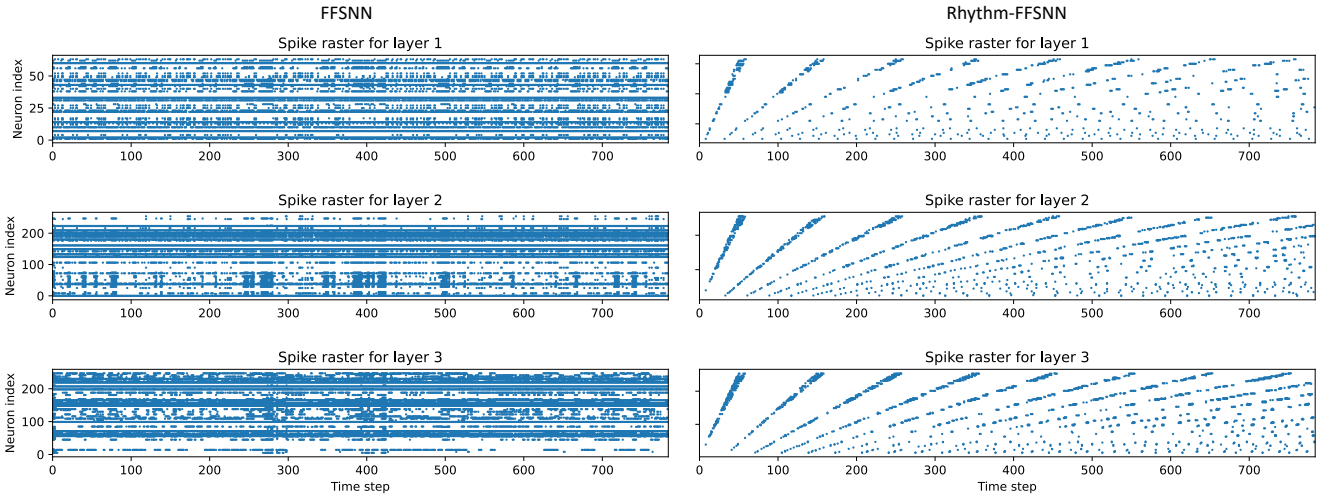

Figure S4. Visualization of spike raster of FFSNN and Rhythm-FFSNN on the PS-MNIST dataset. It is worth noting that Rhythm-FFSNN demonstrates periodic and sparser firing activities compared to FFSNN.

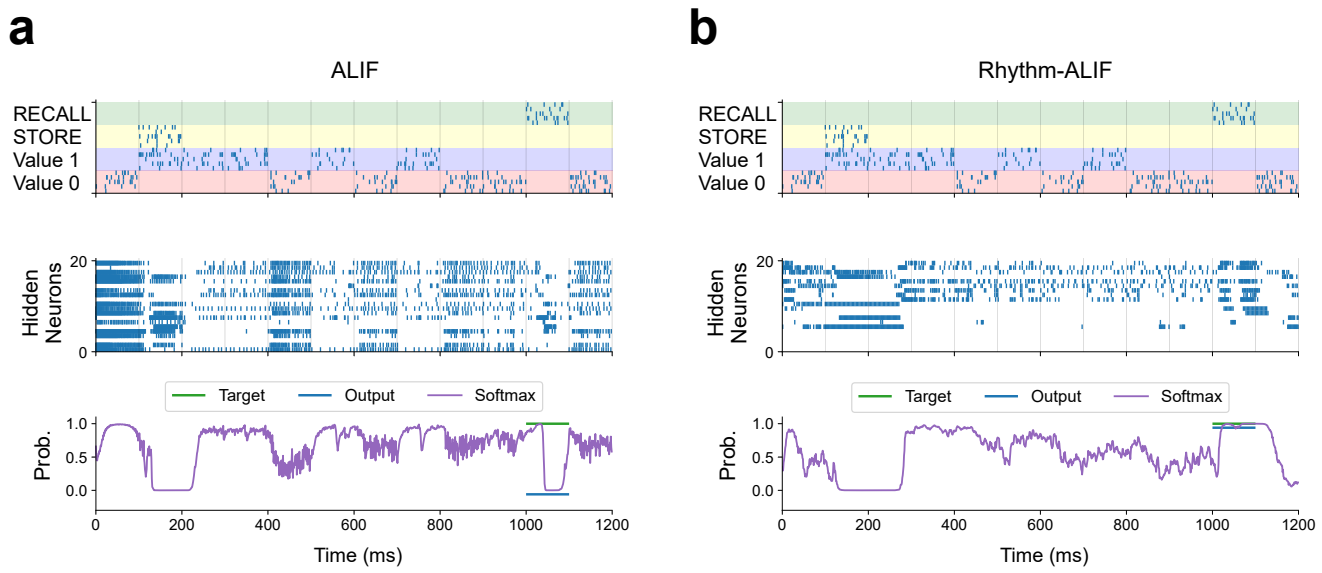

Figure S5. Comparison of ALIF and Rhythm-ALIF on the STORE-RECALL task. The spike raster of hidden neurons and evolution curves of the corresponding outputs of **(a)** ALIF and **(b)** Rhythm-ALIF on the STORE-RECALL task.

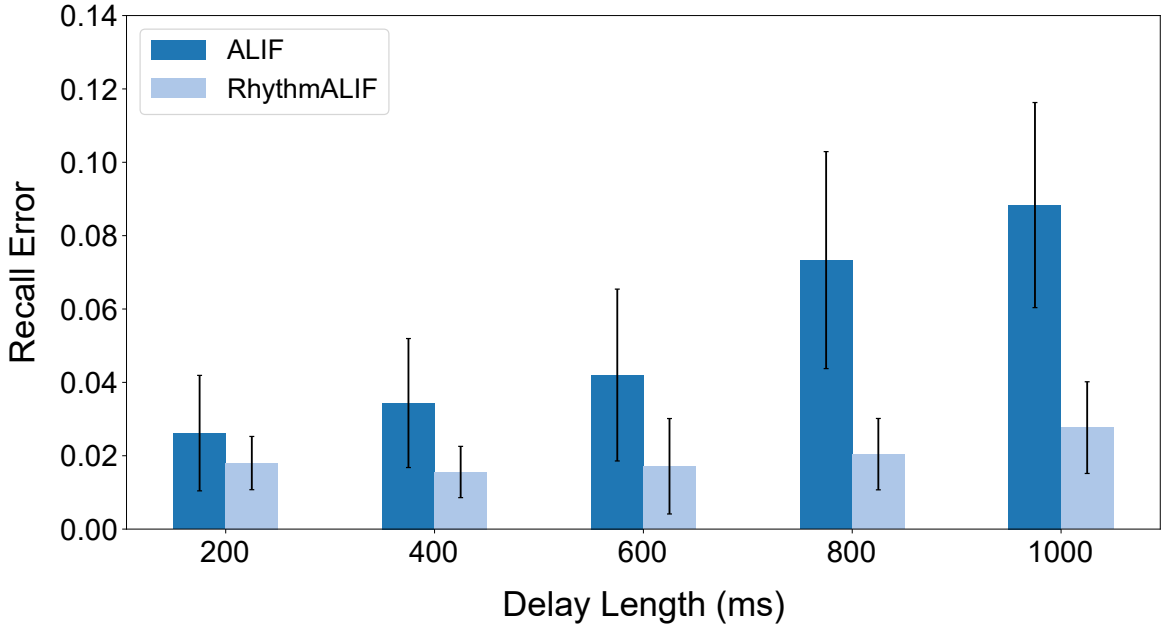

Figure S6. Extended experiments on the STORE-RECALL task with varying delay length. Our results are reported as averages over three experimental runs with different random seeds.

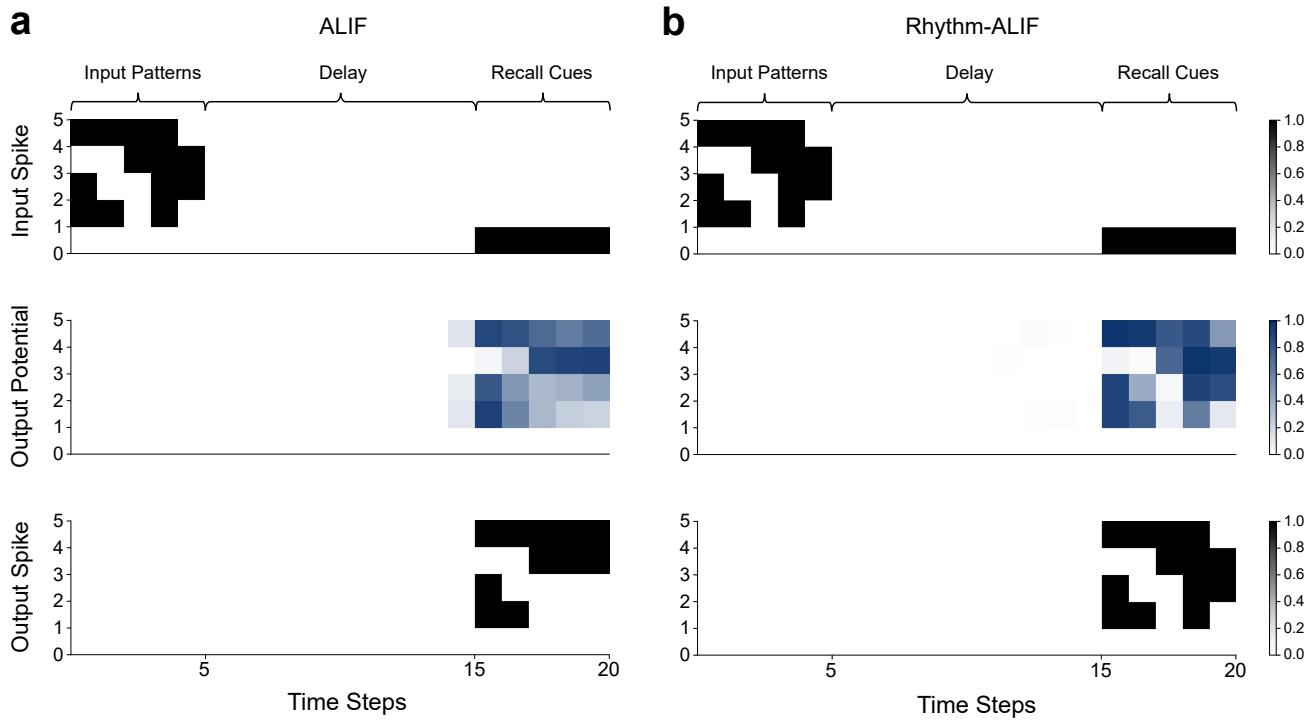

Figure S7. Comparison of (a) ALIF and (b) Rhythm-ALIF on the delayed recall task. The final recalled results are obtained by rounding the membrane potential of neurons in the output layer. It can be observed that the ALIF model fails to recall the original input pattern, whereas Rhythm-ALIF successfully recalls it after a delay period of 10 time steps.

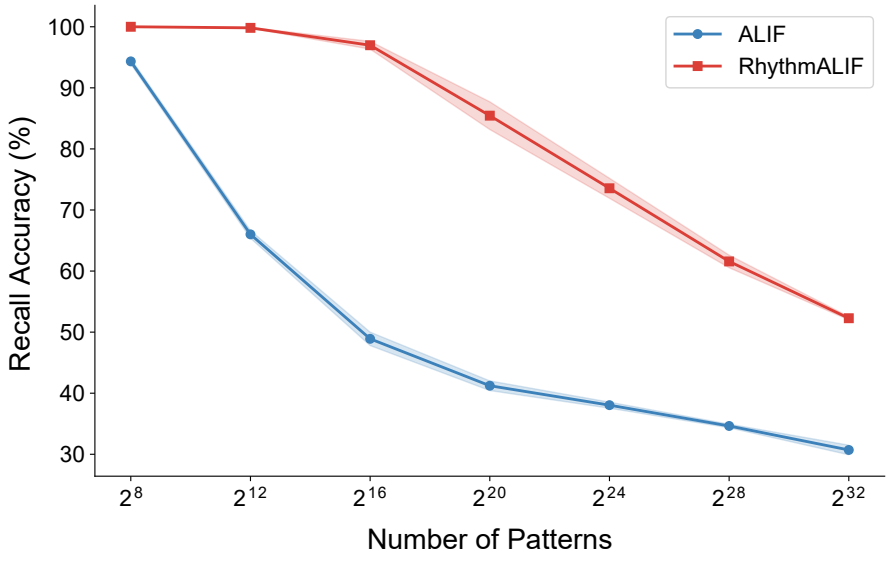

Figure S8. Comparison of (a) ALIF and (b) Rhythm-ALIF's recall accuracy on the delayed recall task with respect to number of patterns. It is evident that Rhythm-ALIF is able to maintain a high recall accuracy as the number of patterns increases, whereas ALIF rapidly drops to a much lower level.

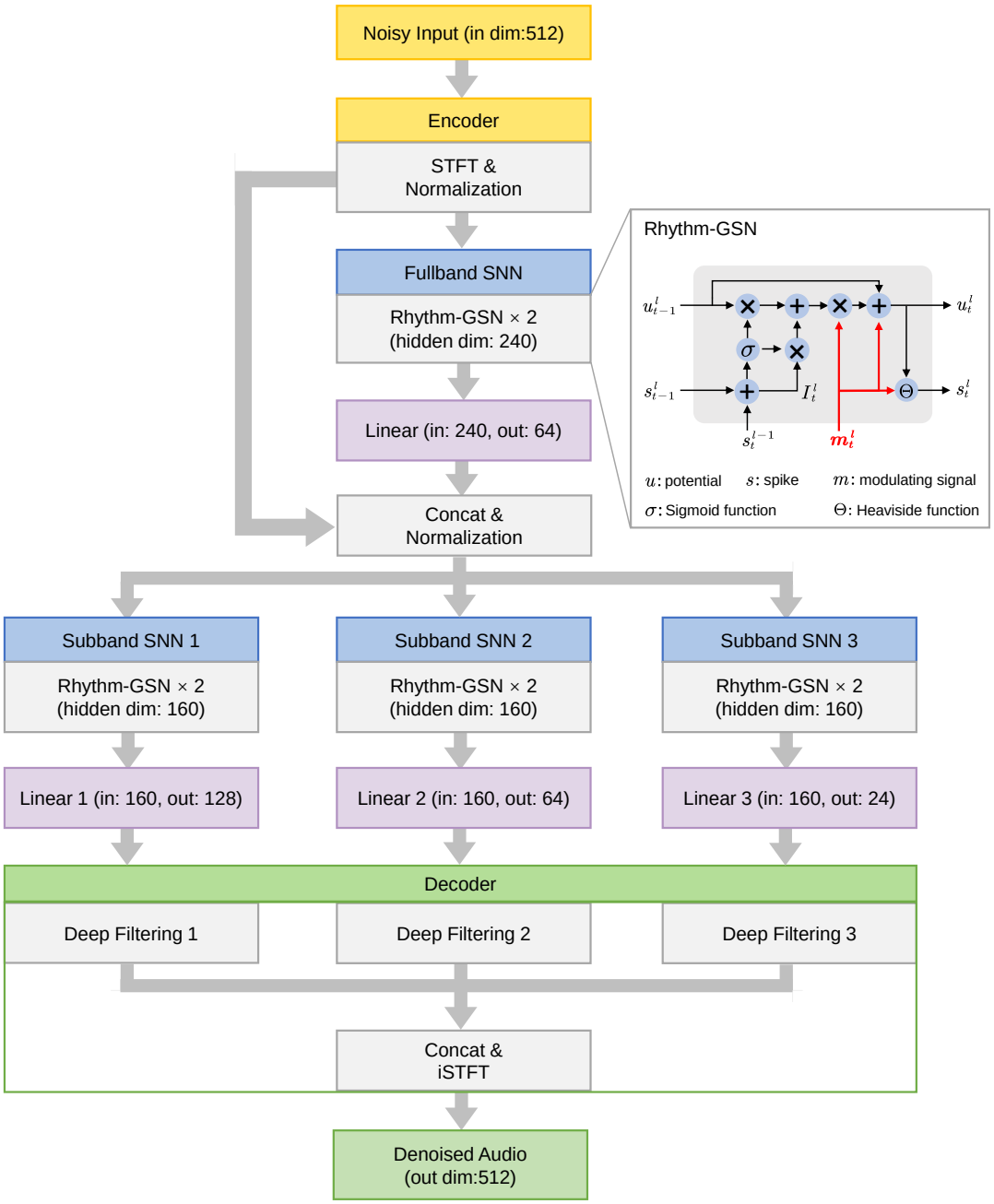

Figure S9. The overall network architecture of Rhythm-GSNN utilized in this work for the speech enhancement task. The Rhythm-GSNN consists of one Fullband SNN module and three Subband SNN modules, with each module comprising two layers of Rhythm-GSN and one linear layer. The Rhythm-GSNN utilizes the Rhythm-GSN model as its basic building block. The introduced rhythmic signal m modulates the membrane potential updates and the firing activity of the spiking neuron, as illustrated by the red lines. Regarding the denoising pipeline, this model first encodes noisy speech using a Short-Time Fourier Transform (STFT) encoder. The encoded signals are then processed by one Fullband SNN module and three Subband SNN modules. Finally, the denoised audio is reconstructed by decoding the outputs from the subband modules into audio signals via an inverse STFT (iSTFT) decoder.

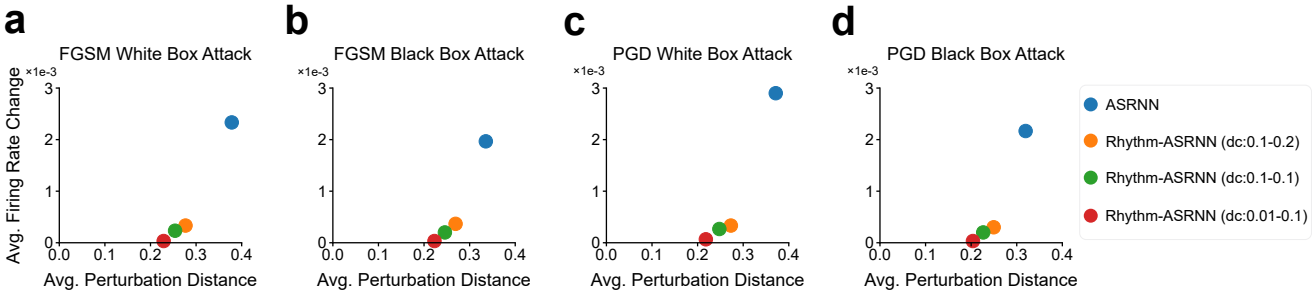

Figure S10. Comparison of the changes in average firing rate and average perturbation distance for ASRNNs and Rhythm-ASRNNs under adversarial attacks. Comparison of ASRNNs and Rhythm-ASRNNs with different duty cycle parameters (denoted as 'dc' in the legend) against (**a**) FGSM under white box attack condition, (**b**) FGSM under black box attack condition, (**c**) PGD under white box attack condition, and (**d**) PGD under black box attack condition. Experimental results demonstrate that Rhythm-ASRNN models with lower duty cycle ('dc') parameters achieve smaller average firing rate changes and reduced average perturbation distances.

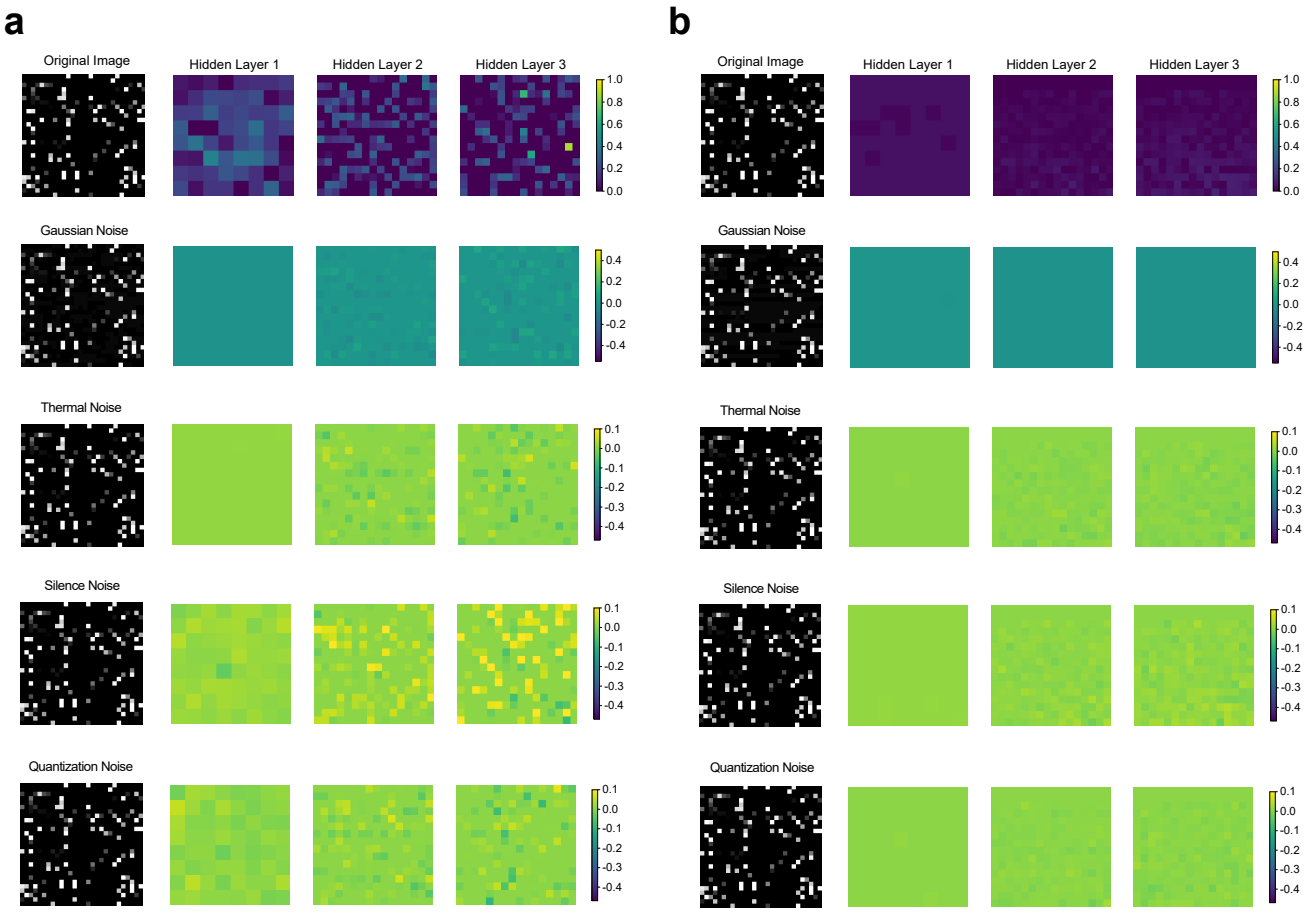

Figure S11. Visualization of inputs and hidden layers' representations under various types of noises. First row: Original inputs and hidden layers' representations of **(a)** ASRNN and **(b)** Rhythm-ASRNN. **Second row**: Gaussian noise perturbed inputs and the difference between perturbed and original representation. **Third to fifth rows**: the difference between perturbed and original representation for Thermal, Silence, and Quantization noise, respectively.

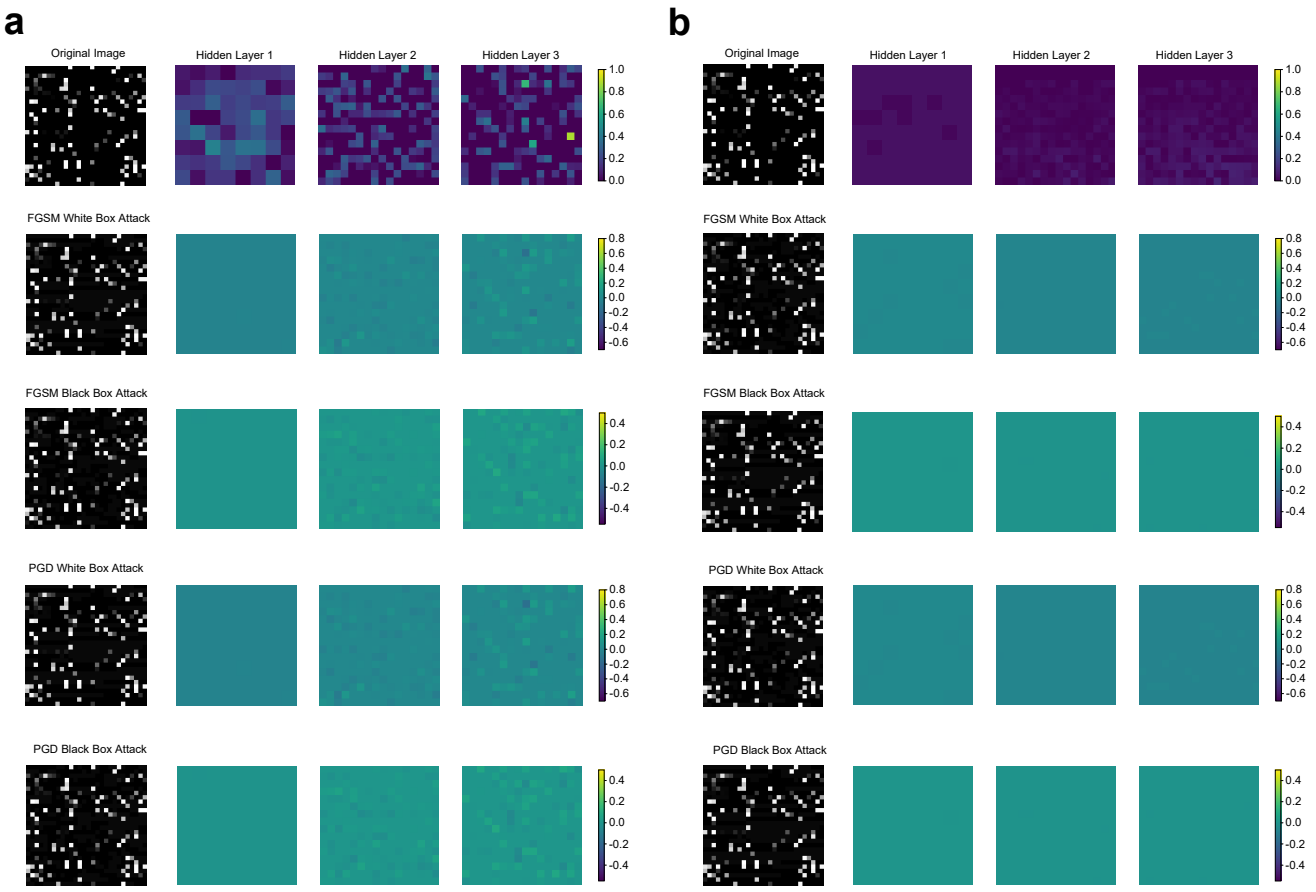

Figure S12. Visualization of inputs and hidden layers' representations under adversarial attacks. First row: Original inputs and hidden layers' representations of **(a)** ASRNN and **(b)** Rhythm-ASRNN. **Second and third rows**: Inputs under FGSM white and black box attack conditions, and the differences between the perturbed and original representations. **forth to fifth rows**: Inputs under PGD white and black box attacks, and the differences between the perturbed and original representations.

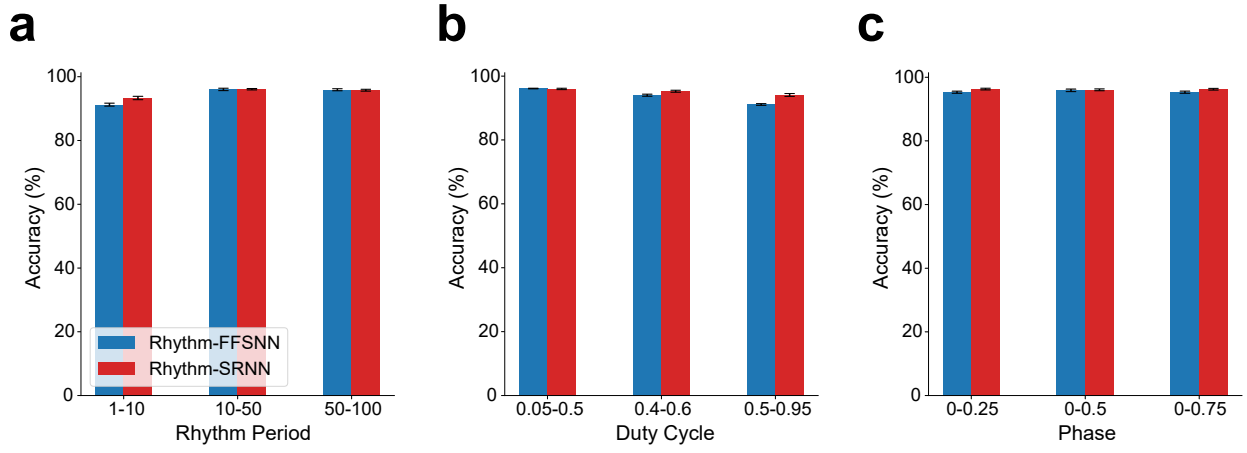

Figure S13. Ablation study of rhythm hyperparameters in Rhythm-FFSNN and Rhythm-SRNN. Accuracy of Rhythm-FFSNN and Rhythm-SRNN under different (**a**) rhythm period, (**b**) duty cycle, and (**c**) phase settings on the PS-MNIST dataset across three runs with different random seeds.

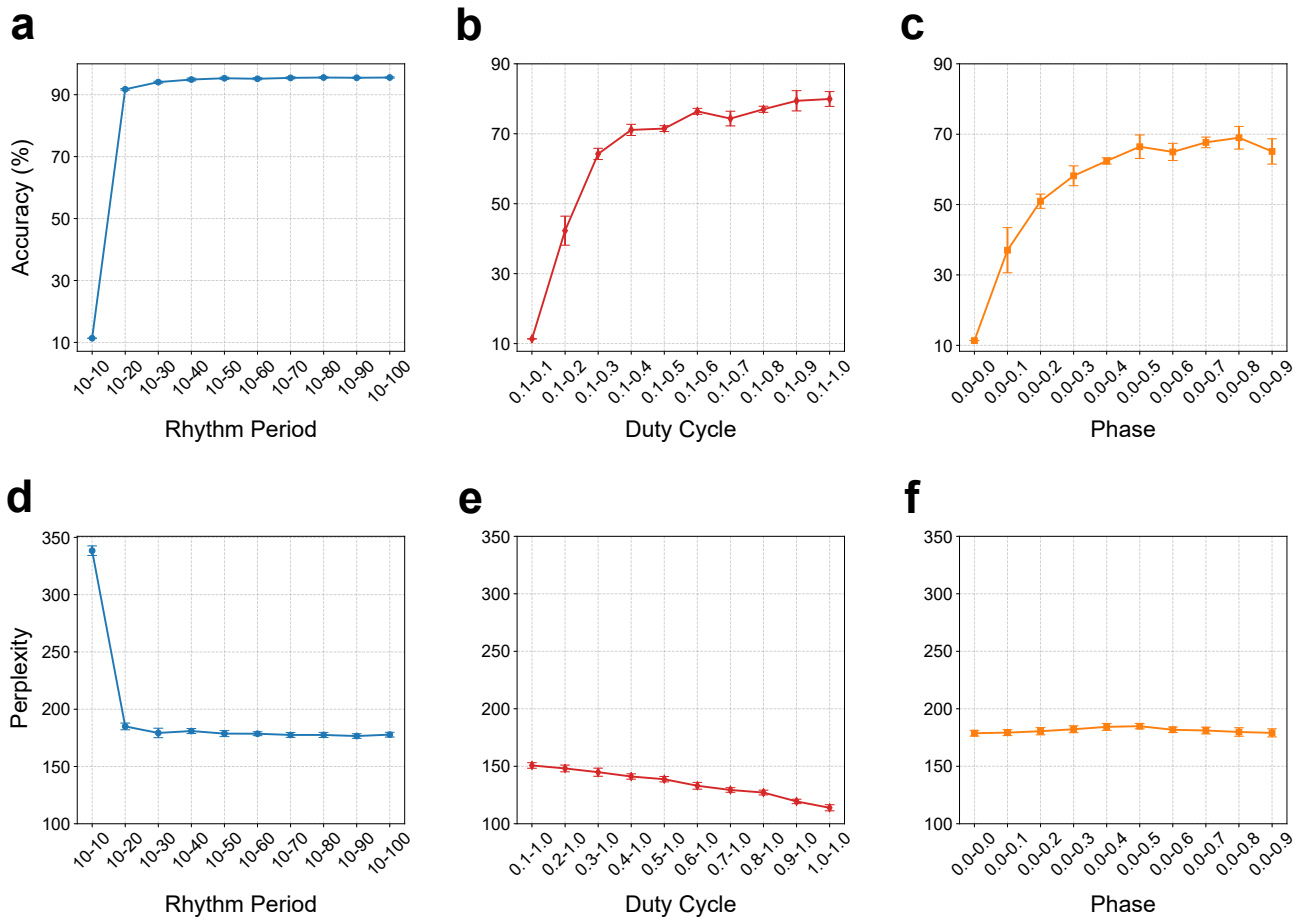

Figure S14. Ablation study on the impact of the heterogeneity of oscillatory signals on Rhythm-SNNs' performance. **a-c**, Test accuracy of Rhythm-FFSNN under different **(a)** rhythm period, **(b)** duty cycle, and **(c)** phase settings on the PS-MNIST dataset. **d-f**, Test perplexity of Rhythm-ASRNN under different **(d)** rhythm period, **(e)** duty cycle, and **(f)** phase settings on the PTB dataset. The experiments were conducted over three runs with different random seeds, and the error bars represent the standard deviation.

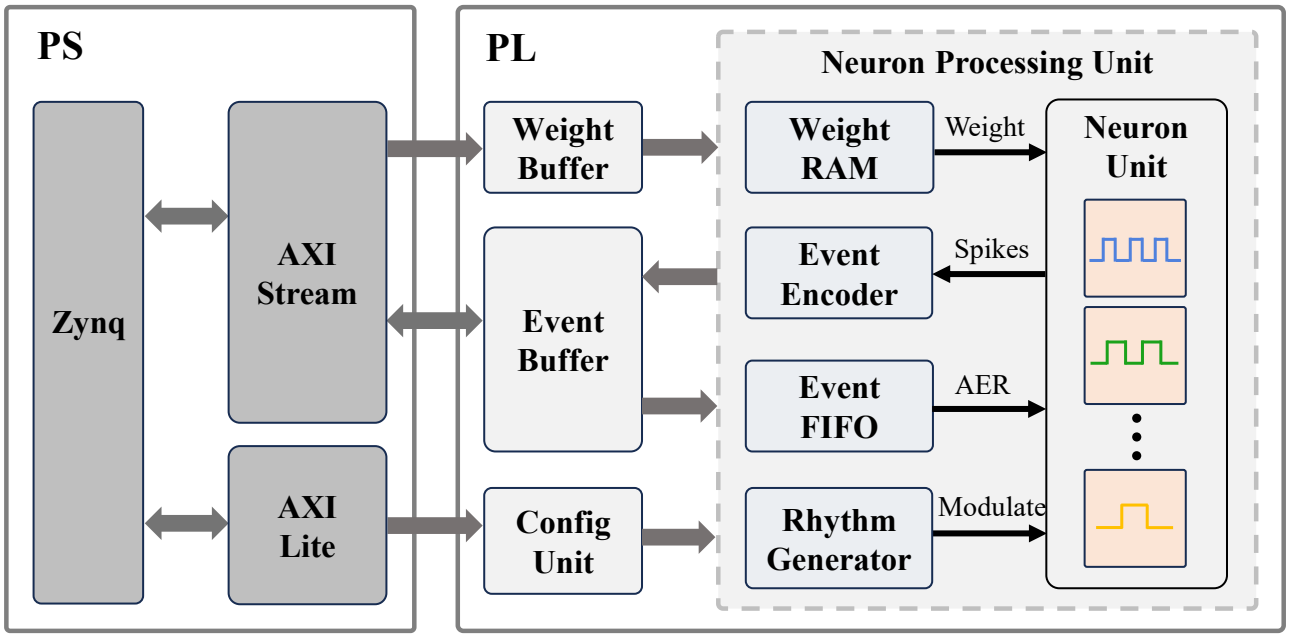

Figure S15. The FPGA-based neuromorphic system architecture designed for implementing the proposed Rhythm-SNN. The overall architecture comprises the processing system (PS) and the programmable logic (PL). The spiking neurons in the Neuron Unit can alternatively transition between silent and activated states under the modulation of oscillatory signals generated by the Rhythm Generator.

References

1. Bellec, G., Salaj, D., Subramoney, A., Legenstein, R. & Maass, W. Long short-term memory and learning-to-learn in networks of spiking neurons. *Adv. neural information processing systems* **31** (2018).
2. Yin, B., Corradi, F. & Bohté, S. M. Effective and efficient computation with multiple-timescale spiking recurrent neural networks. In *International Conference on Neuromorphic Systems 2020*, 1–8 (2020).
3. Zheng, H. *et al.* Temporal dendritic heterogeneity incorporated with spiking neural networks for learning multi-timescale dynamics. *Nat. Commun.* **15**, 277 (2024).
4. Shaban, A., Bezugam, S. S. & Suri, M. An adaptive threshold neuron for recurrent spiking neural networks with nanodevice hardware implementation. *Nat. Commun.* **12**, 4234 (2021).
5. Yin, B., Corradi, F. & Bohté, S. M. Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. *Nat. Mach. Intell.* **3**, 905–913 (2021).
6. Yang, Q. *et al.* Training spiking neural networks with local tandem learning. *Adv. Neural Inf. Process. Syst.* **35**, 12662–12676 (2022).
7. Goodfellow, I. J., Shlens, J. & Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).
8. Madry, A., Makelov, A., Schmidt, L., Tsipras, D. & Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).
9. Bahmaninezhad, F. *et al.* A comprehensive study of speech separation: spectrogram vs waveform separation. *arXiv preprint arXiv:1905.07497* (2019).
10. Reddy, C. K., Gopal, V. & Cutler, R. Dnsmos p. 835: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 886–890 (IEEE, 2022).
11. Ding, J., Bu, T., Yu, Z., Huang, T. & Liu, J. Snn-rat: Robustness-enhanced spiking neural network through regularized adversarial training. *Adv. Neural Inf. Process. Syst.* **35**, 24780–24793 (2022).
12. Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y. & Usunier, N. Parseval networks: Improving robustness to adversarial examples. In *International conference on machine learning*, 854–863 (PMLR, 2017).
13. Weng, L. *et al.* Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning*, 5276–5285 (PMLR, 2018).
14. Fazlyab, M., Robey, A., Hassani, H., Morari, M. & Pappas, G. Efficient and accurate estimation of lipschitz constants for deep neural networks. *Adv. Neural Inf. Process. Syst.* **32** (2019).
15. Horowitz, M. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE international solid-state circuits conference digest of technical papers (ISSCC)*, 10–14 (IEEE, 2014).
16. Han, S., Pool, J., Tran, J. & Dally, W. Learning both weights and connections for efficient neural network. *Adv. neural information processing systems* **28** (2015).
17. Hu, Y. *et al.* Dccrn: Deep complex convolution recurrent network for phase-aware speech enhancement. *arXiv preprint arXiv:2008.00264* (2020).
18. Hao, X., Su, X., Horaud, R. & Li, X. Fullsubnet: A full-band and sub-band fusion model for real-time single-channel speech enhancement. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6633–6637 (IEEE, 2021).
19. Dubey, H. *et al.* Icassp 2023 deep noise suppression challenge. *IEEE Open J. Signal Process.* (2024).
20. Timcheck, J. *et al.* The intel neuromorphic dns challenge. *Neuromorphic Comput. Eng.* **3**, 034005 (2023).
21. Hao, X., Ma, C., Yang, Q., Wu, J. & Tan, K. C. Toward ultralow-power neuromorphic speech enhancement with spiking-fullsubnet. *IEEE transactions on neural networks learning systems* (2025).