A APPENDIX

A.1 COMPUTING INFRASTRUCTURE

All experiments are conducted on Ubuntu 20.04.5 LTS server equipped with NVIDIA GeForce RTX 3090 GPUs (24G Memory), Intel(R) Xeon(R) Platinum 8370C CPU @ 2.80GHz, Pytorch 1.13.0, and CUDA 11.8.

A.2 EXPERIMENTAL CONFIGURATION

In this section, we present the detailed experimental setups, including datasets, pre-processing, model architectures, and specific hyperparameters used in each task.

A.2.1 DATASETS

- **Seq-MNIST and PS-MNIST** (Le et al., 2015) are two variants of the MNIST handwritten digit recognition dataset. Each task involves reading a 28×28 grayscale digit image into the network pixel by pixel through a raster scan. In the Seq-MNIST task, the sequence order mimics the way humans read: row-by-row. Conversely, the PS-MNIST task involves a pre-processing step where the order is shuffled using a fixed random permutation matrix, which notably increases the task's complexity.
- **Sequential CIFAR10 and Sequential CIFAR100** are tasks based on the CIFAR-10/CIFAR-100 dataset introduced by Krizhevsky et al. (2009). In these tasks, a 32×32 full-color image is used as the network input. In the column-by-column task, the image's columns (32 pixels each) are sequentially fed into the network from left to right, resulting in a sequence length of 32. For the pixel-by-pixel task, the image is read through a one-dimensional raster scan, leading to a sequence length of 1024. The primary objective of these tasks is to classify the image into one of ten categories.
- **Spiking Heidelberg Digits (SHD)** (Cramer et al., 2020) is a spike-based sequence classification benchmark that consists of spoken digits from 0 to 9 in both English and German, resulting in 20 classes. The dataset comprises recordings from twelve speakers, two of whom only appear in the test set. Each original waveform is converted into spike trains over 700 input channels. The training set includes 8,332 examples, while the test set comprises 2,088 examples (no validation set). The SHD dataset is a widely used task to assess the performance of SNNs in processing and classifying speech data represented in spiking format.

A.2.2 PRE-PROCESSING

In column-by-column Sequential CIFAR10/Sequential CIFAR100, we employ the same data augmentation techniques as those utilized in Fang et al. (2021a) to ensure a fair comparison. For all other tasks, no specific augmentation method is implemented.

A.2.3 MODEL ARCHITECTURES AND HYPERPARAMETERS

We specify all architecture configurations as follows:

- **Column-by-column Sequential CIFAR10/Sequential CIFAR100**: To enhance consistency in comparison, we use the identity model architecture in Fang et al. (2023).
- **SHD**: Following the previous works on this dataset (Wang et al., 2023; Sun et al., 2023), we adopts two-hidden-layer fully connection network architecture. The hidden dimension is set to 256 and 352 to provide results under different parameter amounts.
- **Pixel-by-pixel S-MNIST, PS-MNIST, and Sequential CIFAR10:** In this task, we utilized the network architecture that could be regarded as a stack of residual blocks, which we denote as RB. Each residual block comprises a residual connection and a sequence of '1x1 convolution - Batch normalization 1D - Spiking Neuron model'. For S-MNIST/PS-MNIST tasks, we utilize 3-hidden-layer architecture (FC128-BN-PMSN)-RB128-RB128 and (FC208-BN-PMSN)-RB208-RB208, respectively, for different parameter counts. FC

represents the fully connected layer while BN signifies the 1D batch normalization layer. For more challenging Sequential CIFAR10, the architecture is expanded to (FC128-BN-PMSN)-RB128-RB128-AP4-(FC256-BN-PMSN)-RB256-RB256, where AP is the average pooling layer.

In Table 3, we list all task-specific hyperparameters as follows:

Table 3: Hyperparameters used in different tasks.

Datasets	Global Learning Rate	Neuronal Learning Rate	Weight Decay	Dropout	Batchsize	Epochs	θ	γ
S-MNIST	1e-2	1e-3	1e-2	0.1	64	200	1	1
PS-MNIST	1e-2	1e-3	1e-2	0	64	200	1	1
SHD	1e-2	1e-3	0	0.4	40	150	1	1
Sequential CIFAR10 (column-by-column)	1e-3	1e-3	0	0	128	200	1	1
Sequential CIFAR100 (column-by-column)	1e-3	1e-3	0	0	128	200	1	1
Sequential CIFAR10 (pixel-by-pixel)	1e-2	1e-3	1e-2	0.1	64	200	1	1

For all experiments, we adopt the same initialization of the PMSN model, wherein the coupling coefficient $\beta_{i+1,i} = -\beta_{i,i+1} = 5i$, decaying factor $\tau_i = 2$, step size $dt \sim U(1e-3, 1e-1)$, and current gain $\gamma_{0:n-1} = 1$, $\gamma_n \sim U(0,1)$. The AdamW optimizer and Cross-entropy (CE) loss are adopted across all datasets

A.3 Ablation Study

In Figure 5, we report the accuracy of the PMSN model with various compartment numbers when implemented to pixel-by-pixel Sequential CIFAR10 task ($T = 1024$). In this section, we further evaluate the effect of the compartment number of the PMSN model on spatiotemporal tasks ($T = 32$). As presented in Table 4, when the compartment number n increases from 2 to 5, the accuracy improves quickly. However, as n continues to increase, the accuracy sees only marginal improvement, but at the cost of a substantial increase in computational consumption.

Table 4: Ablation study of compartment number n

Dataset	Compartment number n					
	2	3	4	**5**	9	17
Sequential CIFAR10	88.79%	90.49%	90.75%	**90.97%**	91.05%	91.43%
Sequential CIFAR100	61.84%	65.16%	65.83%	**66.08%**	66.51%	66.81%

A.4 Deriviation

A.4.1 Model Discretization of our proposed PMSN model (Eqs.7,8,9)

Recall the continuous-time formulation of the PMSN model from Eq. 5 and Eq. 6 as:

$$\dot{\mathcal{V}}_h(t) = \begin{bmatrix} -\frac{1}{\tau_1} & \beta_{2,1} & 0 & \cdots & 0 \\ \beta_{1,2} & -\frac{1}{\tau_2} & \beta_{3,2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \beta_{n-2,n-1} & -\frac{1}{\tau_{n-1}} \end{bmatrix} \mathcal{V}_h(t) + \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{n-1} \end{bmatrix} I(t), \tag{18}$$

$$\dot{v}_s(t) = \begin{bmatrix} 0 & 0 & \cdots & \beta_{n-1,n} \end{bmatrix} \mathcal{V}_h(t) - \frac{1}{\tau_n} v_s(t) + \gamma_n I(t) - \theta S(t), \quad S(t) = H(v_s(t), \theta). \tag{19}$$

The first full-rank state transition matrix in Eq. 18 is denoted by $\mathcal{T} \in \mathbb{R}^{n-1 \times n-1}$, which could be diagonalized using eigenvalue decomposition $\mathcal{T} = P\Lambda P^{-1}$, where Λ is the eigenvalue matrix, and $P \in \mathbb{C}^{n-1 \times n-1}$ denotes eigenvector matrix. This yields:

$$\dot{\mathcal{V}}_h(t) = P\Lambda P^{-1}\mathcal{V}_h(t) + [\gamma_1, .., \gamma_{n-1}]^T I(t). \tag{20}$$

After multiplying both sides of Eq. 20 by P^{-1}, we could obtain the following form:

$$P^{-1}\dot{\mathcal{V}}_h(t) = \Lambda P^{-1}\mathcal{V}_h(t) + P^{-1}[\gamma_1, .., \gamma_{n-1}]^T I(t). \tag{21}$$

By replacing variable $V_h = P^{-1}\mathcal{V}_h$, $\phi_c = P^{-1}[\gamma_1, .., \gamma_{n-1}]^T$, $\Phi_s = [0, 0, .., \beta_{n-1,n}]P$, Eq. 21 and Eq. 19 could be written as:

$$\dot{V}_h(t) = \Lambda V_h(t) + \phi_c I(t), \tag{22}$$

$$\dot{v}_s(t) = \Phi_s V_h(t) - \frac{1}{\tau_n} v_s(t) + \gamma_n I(t) - \theta S(t), \quad S(t) = H(v_s(t),\ \theta). \tag{23}$$

Then, we denote the total input current compartment as $I_h(t)$:

$$I_h(t) = \Phi_s V_h(t) + \gamma_n I(t). \tag{24}$$

Plugging $I_h(t)$ into Eq. 23, we have:

$$\dot{v}_s(t) = -\frac{1}{\tau_n} v_s(t) + I_h(t) - \theta S(t), \quad S(t) = H(v_s(t),\ \theta). \tag{25}$$

By applying Zero-order Hold technique (DeCarlo, 1989), Eq. 22 can be calculated in the following closed-form:

$$\begin{aligned} V_h\left(t + dt\right) &= e^{\Lambda dt} V_h\left(t\right) + \phi_c I\left(t\right) \int_{\tau=t}^{t+dt} e^{\Lambda(t+dt-\tau)} \mathrm{d}\tau \\ &= e^{\Lambda dt} V_h\left(t\right) + \phi_c I\left(t\right) \int_{\tau=0}^{dt} e^{\Lambda\tau} \mathrm{d}\tau \\ &= e^{\Lambda dt} V_h\left(t\right) + \Lambda^{-1}\left(e^{\Lambda dt} - I\right) \phi_c I\left(t\right), \end{aligned} \tag{26}$$

where dt is the step size. Similarly, the update formula of v_s in Eq. 23 can be deduced as:

$$v_s(t + dt) = e^{-\frac{dt}{\tau_n}} v_s(t) + I_h(t + dt) - \theta S(t), \quad S(t) = H(v_s(t),\ \theta). \tag{27}$$

By substituting $\bar{\mathcal{T}} = exp(\Lambda dt)$, $\Phi_c = \Lambda^{-1}(exp(\Lambda dt) - I)\phi_c$ in Eq. 26 and $\alpha = exp(-\frac{dt}{\tau_n})$ in Eq. 27, we could finally obtain the following update formula of V_h and v_s as:

$$V_h[t] = \bar{\mathcal{T}} V_h[t-1] + \Phi_c I[t], \tag{28}$$

$$v_s[t] = \alpha v_s[t-1] + I_h[t] - \theta S[t-1], \quad S[t] = H(v_s[t],\ \theta). \tag{29}$$

A.4.2 Parallelized Computing of the Output Compartment with Reset (Eq. 15)

According to the update formula of v_s in Eq. 13, setting $\alpha = 1$, we could rewrite it as the following form:

$$v_r[t-1] = I_h[t] + v_s[t-1] - v_s[t], \tag{30}$$

Cumulating Eq. 30 from t = 0 to t yields:

$$\sum_{i=0}^{t-1} v_r[i] = \sum_{i=0}^{t} I_h[i] - v_s[t], \tag{31}$$

As $v_r[t] = \theta S[t] \cdot \lfloor v_s[t] \rfloor_\theta$, by moving $v_s[t]$ to the left side of the equation, dividing both sides of Eq. 31 by θ and rounding down, it can be expressed as:

$$\sum_{i=0}^{t-1} S[i] \cdot \lfloor v_s[i] \rfloor_\theta + \lfloor v_s[t] \rfloor_\theta = \lfloor \sum_{i=0}^{t} I_h[i] \rfloor_\theta \tag{32}$$

Given that all input is non-negative and thus $v_s[t]$ is non-negative at any given moment, we can reformulate $\lfloor v_s[t] \rfloor_\theta$ as $S[t] \cdot \lfloor v_s[t] \rfloor_\theta$ and substitute it into the preceding equation. This leads us to the conclusion that:

$$\sum_{i=0}^{t} S[i] \lfloor v_s[i] \rfloor_\theta = \lfloor \sum_{i=0}^{t} I_h[i] \rfloor_\theta. \tag{33}$$

By multiplying both sides of the above equation by θ, we could obtain:

$$\sum_{i=0}^{t} v_r[i] = \theta \sum_{i=0}^{t} S[i] \lfloor v_s[i] \rfloor_\theta = \theta \lfloor \sum_{i=0}^{t} I_h[i] \rfloor_\theta. \tag{34}$$

Ultimately, by substituting Eq. 34 into Eq. 31, we could derive the conclusion in the Eq. 15 that:

$$\sum_{i=0}^{t-1} v_r[i] = \theta \lfloor \sum_{i=0}^{t-1} I_h[i] \rfloor_\theta, \quad v_s[t] = \sum_{i=0}^{t} I_h[i] - \theta \lfloor \sum_{i=0}^{t-1} I_h[i] \rfloor_\theta, \tag{35}$$

A.4.3 Gradient Backpropagation of PMSN with Surrogate Gradient (Eq. 17)

As mentioned in Eq. 17, the gradient flow for the PMSN parameter update is formulated as:

$$\Delta \mathcal{W}^l \propto \frac{\partial \mathcal{L}}{\partial \mathcal{W}^l} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial I^l[t]} S^{l-1}[t], \quad \Delta b^l \propto \frac{\partial \mathcal{L}}{\partial b^l} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial I^l[t]}, \tag{36}$$

where $\mathcal{L}$ is the loss function. $\mathcal{W}^l$ and b^l refer to weight and bias terms of layer l, respectively. Subsequently, $\frac{\partial \mathcal{L}}{\partial I^l[t]}$ can be derived from:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial I^l[t]} &= \sum_{i=t}^{T} \frac{\partial \mathcal{L}}{\partial S^l[i]} \frac{\partial S^l[i]}{\partial v_s^l[i]} \frac{\partial v_s^l[i]}{\partial I^l[t]} + \sum_{i=t}^{T-1} \frac{\partial \mathcal{L}}{\partial v_s^l[i+1]} \frac{\partial v_s^l[i+1]}{\partial v_s^l[i]} \frac{\partial v_s^l[i]}{\partial I^l[t]} \\ &= \sum_{i=t}^{T} \frac{\partial \mathcal{L}}{\partial S^l[i]} \frac{\partial S^l[i]}{\partial v_s^l[i]} \frac{\partial v_s^l[i]}{\partial I^l[t]} + \sum_{i=t}^{T-1} \frac{\partial \mathcal{L}}{\partial v_s^l[i+1]} \left(1 - \frac{\partial v_r^l[i]}{\partial v_s^l[i]}\right) \frac{\partial v_s^l[i]}{\partial I^l[t]}. \end{aligned} \tag{37}$$

Recalling Eq. 11 and Eq. 13, we could obtain $\frac{\partial v_s^l[i]}{\partial I^l[t]}$ as:

$$\frac{\partial v_s^l[i]}{\partial I^l[t]} = \frac{\partial v_s^l[i]}{\partial I_h^l[i]} \frac{\partial I_h^l[i]}{\partial I^l[t]} = \begin{cases} \Phi_s \bar{\mathcal{T}}^{i-t} \Phi_c, & \text{if} \quad i > t, \\ \Phi_s \bar{\mathcal{T}}^{i-t} \Phi_c + \gamma_n, & \text{if} \quad i = t, \\ 0 & \text{otherwise.} \end{cases} \tag{38}$$

To overcome the discontinuity that happened during spike generation and reset, we employ triangle function (Deng et al., 2022) and straight-through estimator (Bengio et al., 2013) as surrogate gradients:

$$\frac{\partial S^l[i]}{\partial v_s^l[i]} = g'[i] = \begin{cases} (\Gamma - |\Delta|)/\Gamma^2, & \text{if} \quad |\Delta| < \Gamma \\ 0, & \text{otherwise} \end{cases}, \qquad \frac{\partial v_r^l[i]}{\partial v_s^l[i]} = 1, \tag{39}$$

where $\Delta = v_s[t] - \theta$. Γ is a hyperparameter that determines the permissible range for gradients to pass. When plugging Eq. 38 and Eq. 39 into Eq. 37, we could derive the spatiotemporal credit assignment of PMSN as:

$$\frac{\partial \mathcal{L}}{\partial I^l[t]} = \sum_{i=t}^{T} \frac{\partial \mathcal{L}}{\partial S^l[i]} g'[i] \frac{\partial v_s^l[i]}{\partial I^l[t]} = \underbrace{\frac{\partial \mathcal{L}}{\partial S^l[t]} g'[t] \gamma_n}_{\text{Spatial}} + \underbrace{\sum_{i=t}^{T} \frac{\partial \mathcal{L}}{\partial S^l[i]} g'[i] \Phi_s \bar{\mathcal{T}}^{i-t} \Phi_c}_{\text{Temporal}}, \tag{40}$$

A.5 Computation Process of Energy Cost (Figure 5)

In Section 5.4, we carry out a comparative study of computational cost and accuracy among various spiking neuron models by varying their neuronal or network dimensions. Here, we detail the computation process of the theoretical and empirical energy costs for each model.

The number of Multiply-Accumulate (MAC) operations and cheap Accumulate (AC) operations are used to quantify the energy consumption during one inference. Due to the binary nature of spikes in SNNs, the multiplication in synaptic operations between layers employs more efficient

Table 5: Comparative analysis of energy cost among diverse models with an equivalent network structure, where Fr_{in} is the average spike frequency of each presynaptic layer, h denotes input dimension, m signifies neurons numbers, t is simulation time. k is the order of PSN families (representing the receptive field scale of each neuron), and n is the compartment number of our PMSN.

Neuron Model	Dynamics	Theoretical Energy Cost
LIF	$V[t] = \alpha V[t-1] + I[t] - \theta S[t-1]$	$hmtFr_{in}E_{AC} + mtE_{MAC}$
PSN	$V[t] = \sum_{i=0}^{t} \mathcal{W}_{t,i} I[i]$	$hmtFr_{in}E_{AC} + mt^2E_{MAC}$
masked PSN	$V[t] = \sum_{i=t-k+1}^{t} \mathcal{W}_{t,i} I[i]$	$hmtFr_{in}E_{AC} + kmtE_{MAC}$
SPSN	$V[t] = \sum_{i=t-k+1}^{t} \mathcal{W}_{i} I[i]$	$hmtFr_{in}E_{AC} + kmtE_{MAC}$
PMSN(Ours)	$V_h[t] = \bar{\mathcal{T}} V_h[t-1] + \Phi_c I[t]$ $I_h(t) = \Phi_s V_h(t) + \gamma_n I(t)$ $v_s[t] = v_s[t-1] + I_h[t] - \theta S[t-1].$	$hmtFr_{in}E_{AC} + 8(n-1)mtE_{MAC}$

AC operations. However, the operations of neuronal updates, which involve the updates of full-resolution analog signals (i.e., membrane potential and input current), induce more energy-intensive MAC operations. According to the neuronal dynamics of each model, we could derive the theoretical cost of each neuron model within a layer as Table 5.

Following the data collected by Horowitz (2014) on the $45nm$ CMOS , we assign values of $E_{AC} = 0.9\ pJ$ and $E_{MAC} = 4.6\ pJ$ respectively. Additionally, to ascertain the spiking frequency of each presynaptic layer, denoted Fr_{in}, we carry out one inference process on all spiking neuron models with a randomly selected input batch. The recorded spike frequencies of all hidden layers are demonstrated in Figure 5(a). Thus, by substituting the collected values into the theoretical cost, summing up the cost of each layer, we could compute the total energy cost of each model as presented in 5(b).

A.6 Training Speed and Memory Comparison between PMSN and PSN

Regarding our PMSN model, which incorporates a higher number of neuronal compartments compared to single-compartment models, it is intuitive to reason that the PMSN model might necessitate increased computational resources during training. To shed light on how significant the difference is, we conducted a detailed comparative analysis between the PMSN and PSN models, specifically focused on evaluating the maximum memory consumption and training speed. To this end, we employed three distinct benchmarks, S-MNIST, Sequential CIFAR10 ($T = 32$), and Sequential CIFAR10 ($T = 1024$). Note that all experiments are carried out under uniform training configurations and consistent network architectures. The compartment number of the PMSN model is $n = 5$. The results are presented as follows:

Table 6: Computational metrics of PMSN and PSN on different datasets

Matrix	Model	S-MNIST (T=784)	Sequential CIFAR10 (T=32)	Sequential CIFAR10 (T=1024)
Training time (seconds/epoch)	PMSN	35.32	73.21	61.7562
	PSN	21.18	49.02	34.0634
Memory consumption (GB)	PMSN	1.44	3.14	9.40
	PSN	0.61	1.76	9.08
Accuracy (%)	PMSN	99.40	90.97	82.14
	PSN	97.90	88.45	55.24

As expected, the PMSN model requires more time and memory than the PSN model across all datasets. This increased demand is primarily attributed to a larger number of neuronal compartments being used (i.e., five in PMSN v.s. one in PSN). However, the increase in actual time and memory consumption is relatively modest, generally less than twice that of the PSN model in all experiments. This highlights the effectiveness of our proposed parallelization method. Furthermore, we believe this additional computational cost is worthwhile when considering the significantly enhanced sequential modeling capacity as demonstrated on the Sequential CIFAR10 dataset.

A.7 SUPPLEMENT OF MULTI-COMPARTMENT DYNAMICS VISUALIZATION

In Figure 6, we delve into the layer-wise dynamics of each compartment within the same neuron, as well as the distribution of dynamic coefficients across various neurons in the same layer, to shed light on the underlying mechanism of our PMSN model in preserving long-term memory and processing multi-scale temporal information. We employ the five-compartment PMSN model trained in pixel-by-pixel Sequential CIFAR task for illustration.

In the left column of the figures, we first present the impulse response of one spiking neuron in each layer. Given that each neuron possesses unique dynamics within the same layer, we average the dynamic coefficients of each neuron to derive the averaged dynamic of the layer to enhance consistency. Each compartment has its distinct decay coefficient $\lambda_i = e^{\alpha+\beta \boldsymbol{i}}$, which is one-by-one coupled with another compartment possessing a conjugated decay coefficient after training. Consequently, as shown in the figures, the dynamic of each compartment exhibits a damped oscillation pattern. The duration of the oscillatory activity within a compartment is directly proportional to the temporal extent of the long-term memory preserved. The real part α, namely the damping coefficient, indicates the decay rate of membrane potential, while the image part β determines the oscillation frequency.

The divergence between different compartment couples within the same neuron underscores the multi-scale temporal information processing mechanism inherent in a single neuron. Each couple of compartments (i.e., 1 and 4, 2 and 3) consistently displays different oscillation frequencies and decay rates. Therefore, the PMSN model is endowed with the capacity to preserve temporal information across diverse frequency domains. Notably, the $2, 3$ compartments always exhibit a longer oscillation period than the $1, 4$ compartments. This suggests that the hidden compartments tend to retain information for a longer duration when they are not in close proximity to the input or output compartments.

The variance among dynamics of different neurons within the same layer also holds significance. To highlight this, we illustrate the distribution of the oscillation frequency $\frac{\beta}{2\pi}$ and damping coefficient α among diverse neurons in the same layer within the middle column and right column of the figures, respectively. The results indicate different neuron owns different decaying rate of information, as well as restoring information in different oscillation frequency. This diversity among neurons enables the SNNs to incorporate multi-scale information integration, thereby facilitating the storage of various scale memory in sequential modeling tasks.

B CODE AVAILABILITY

The source codes developed will be publicly available after the review process.

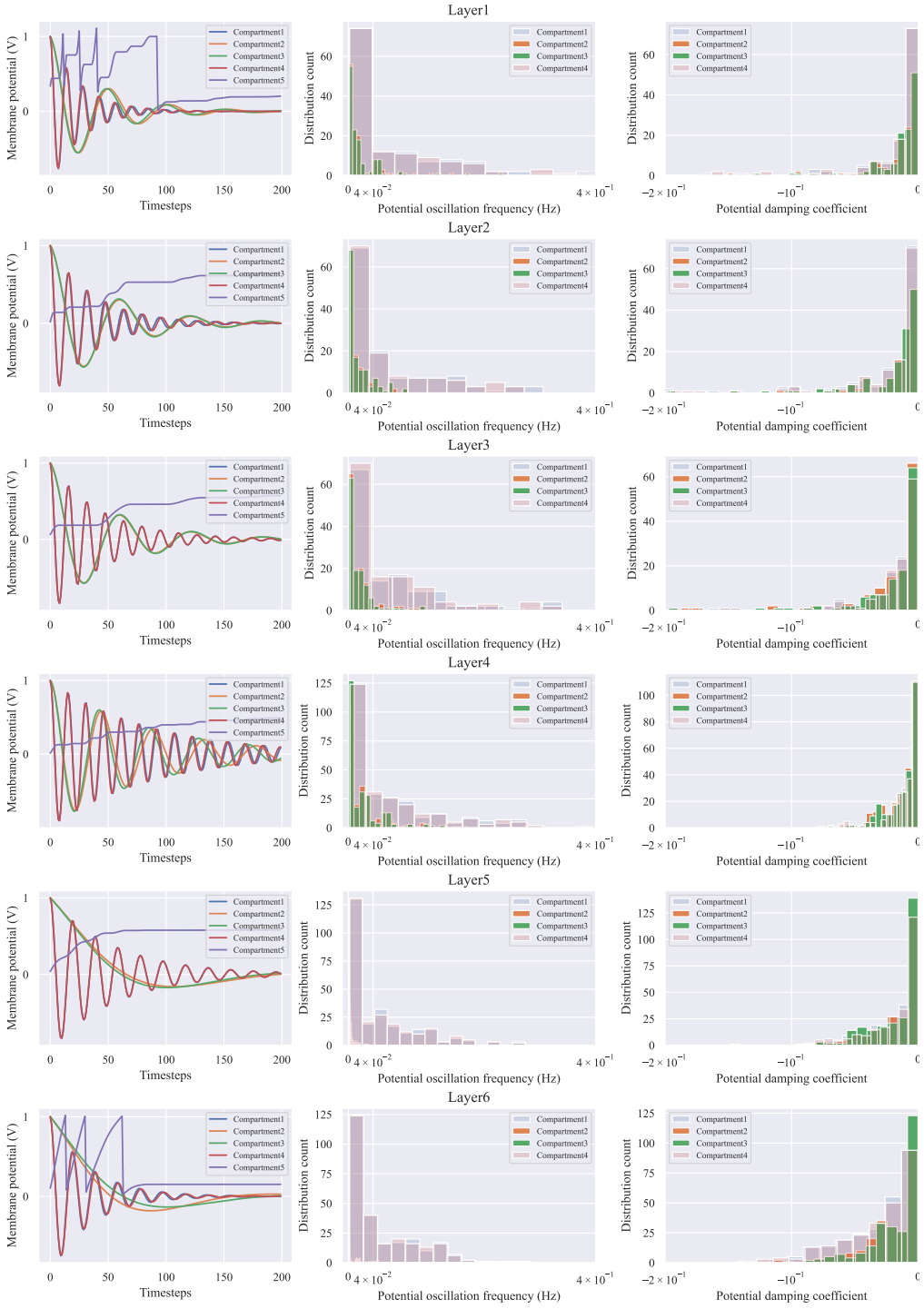

Figure 6: The visualization of PMSN model dynamics($n = 5$) within distinct layers. **Left:** the impulse response of different compartments in the same neuron. Each hidden compartment is characterized by its own dynamic coefficient $\lambda_i = e^{\alpha+\beta \boldsymbol{i}}$, exhibiting damped oscillation patterns after receiving inputs, while the compartment5 is responsible for spike generation and reset. The transient response time of the oscillation in each compartment reflects the preservation time of memory within the membrane potential. **Middle:** the distribution of oscillation frequencies $\frac{\beta}{2\pi}$ for each hidden compartment among a variety of neurons in layer i, reveals each compartment or each neuron process information in the varied frequency domains. **Right:** the distribution of damping coefficients α for each hidden compartment among diverse neurons, which indicates the information decay speeds in each compartment.