Electronic Research Archive

ERA, 34(1): 1–30.
DOI: 10.3934/era.2026001
Received: 28 October 2025
Revised: 07 December 2025
Accepted: 17 December 2025
Published: 26 December 2025

https://www.aimspress.com/journal/ERA

Research article

Path integration based on a recurrent spiking neural network

Jiaxin Lin[1], Yihong Wang[1,2,*], Xuying Xu[1,2], Xiaochuan Pan[1,2] and Rubin Wang[2,3]

[1] Institute for Cognitive Neurodynamics, School of Mathematics, East China University of Science and Technology, Shanghai 200237, China
[2] Center for Intelligent Computing, School of Mathematics, East China University of Science and Technology, Shanghai 200237, China
[3] School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China

* **Correspondence:** Email: wangyihong@ecust.edu.cn.

Abstract: Path integration refers to the process by which animals continuously update spatial position during movement by integrating locomotor stimuli such as speed and head direction, thereby enabling real-time localization in continuous space. This process relies on the cooperation of spatial neurons, including grid cells and place cells. In recent years, continuous attractor networks and recurrent neural networks have provided useful insights into the mechanisms of path integration; however, they often rely on fixed-weight connections or exhibit a lack of biological plausibility. To explore more biologically plausible mechanisms, we propose a path integrator based on the recurrent spiking neural network (RSNN). Employing recurrently connected leaky integrate and fire (LIF) neurons, the RSNN encodes spatial positions as spike sequences via membrane potential reset mechanisms, enabling robust long-term path integration. Analysis reveals the spontaneous emergence of spatial and locomotor units, with some units exhibiting spatial-locomotor conjunctive properties, indicating synergistic computations underlying path integration. Ablation experiments confirm that stripe and border units have a larger effect on performance than other unit types under our experimental conditions. Under sparse spiking conditions, the network naturally develops diverse biologically inspired representations. The RSNN's performance provides novel insights into neuronal synergistic mechanisms in biological navigation, offering a biologically grounded framework for path integration modeling and contributing to the development of brain-inspired navigation algorithms.

Keywords: path integration; recurrent spiking neural network; spatial representations; locomotor representations; neurodynamics

1. Introduction

In natural environments, the ability of animals to flexibly adapt to complex and changing spatial scenarios and perform behaviors, such as foraging, avoidance, and migration, relies on their highly evolved spatial cognitive abilities. Notably, even in the absence of external visual cues, animals are able to maintain accurate spatial orientation by continuously integrating self-motion signals [1]. This process relies heavily on the synergistic mechanism of different functional cells in the entorhinal-hippocampal circuit [2]. Specifically, speed cells exhibit a linear relationship between firing rate and speed [3]; head direction (HD) cells are selectively tuned to specific orientations in space [4]; grid cells generate hexagonally periodic firing patterns that encode the global spatial metric of the environment [5–7]; and both border cells and place cells are responsible for encoding environmental boundaries and discrete spatial locations, respectively [8,9]. The coordination of these cells allows animals to maintain stable spatial representations in the absence of significant external sensory input, revealing the neural mechanisms of spatial navigation [10].

To analyze the computational principles of path integration, researchers have extensively explored computational modeling approaches. Burak and Fiete implemented a model of path integration using a continuous attractor neural network (CANN), in which a two-dimensional neural sheet generates a localized "peak of excitation" (an activity envelope), representing the current position [11]. Locomotor stimuli drove the smooth movement of this activity envelope across the neural sheet, enabling continuous position updates and producing spatially periodic grid-like response patterns in individual neurons. However, this model had two major limitations. First, in the absence of visual cues, noise led to gradual drift of the activity envelope, resulting in accumulated position errors [7]. Second, the model depended on fixed, manually defined, near-symmetric connection weights, which limits its generalization to new environments [12,13]. To overcome these limitations, some studies adopted recurrent neural networks (RNNs) for path integration. Unlike traditional CANNs, RNNs automatically learned connection weights through backpropagation. Cueva and Wei trained an RNN to estimate current position based on locomotor stimuli and found that cells within the network had spontaneously developed diverse spatial response patterns [14]. Furthermore, Banino et al. incorporated RNN into a reinforcement learning framework for navigation tasks, where trained agents exhibited neural representations with hexagonal spatial firing patterns [15]. However, standard RNNs relied on continuous-valued signal transmission, which fundamentally differs from the event-driven dynamics of biological neurons characterized by threshold-triggered spikes and sparse firing activity [16,17].

Spiking neural networks (SNNs) are widely regarded as a biologically plausible framework for modeling neural systems due to their event-driven computation and interpretable dynamical mechanisms [16,18]. As the fundamental components of SNNs, leaky integrate and fire (LIF) neurons are commonly used to simulate the firing behavior of biological neurons. In LIF models, the membrane potential evolves by integrating input currents over time and decays exponentially due to the membrane's inherent leakiness. When the membrane potential reaches firing threshold, the neuron emits a spike and resets membrane potential to resting potential [16,19]. This mechanism not only closely mirrors the dynamics of biological neurons but also enables energy-efficient computation through sparse, event-driven activity [20,21]. The sparse spiking nature of SNN is critical for both efficient information processing and environmental adaptability [22,23]. Neurons fire only when necessary, leading to high spatiotemporal sparsity. This sparse spiking improves computational efficiency and energy utilization while effectively mitigating the influence of noise [24]. Furthermore, some studies have shown that the dynamic modulation of membrane potential decay rates and firing thresholds in LIF neurons enhances the learning capacity of SNNs in complex tasks [25,26].

Leveraging the inherent advantages of spiking neural networks (SNNs) in terms of biological plausibility and energy efficiency, we propose a recurrent spiking neural network (RSNN) for path integration that captures realistic integration mechanisms and simulates functionally relevant neuronal properties. The model utilizes parameterizable LIF neurons to simulate the dynamic evolution of membrane potentials. Spatial information is encoded via spikes, thereby enabling effective path integration. This approach offers a biologically grounded computational framework and demonstrates the feasibility of using spike sequences for path integration [27]. In the recurrent layer of the trained network, a rich variety of spatial and locomotor units spontaneously emerge, closely resembling navigation-related cells observed in biological experiments. Notably, the numerical distribution of spatial units suggests that grid units may not be the dominant contributors to path integration, challenging the prevailing view that they are primary drivers of path integration. Ablation analyses further reveal the critical contributions of stripe and border units to path integration under our experimental conditions. Spatial representations emerge robustly across different environmental geometries; although environmental structure modulates the tuning properties of these representations, it does not hinder their formation. Furthermore, the observed strong correlation between spatial and locomotor units indicates that synergistic interactions among these units are essential for path integration. From a computational perspective, the event-driven, sparsely activated architecture of SNNs confers significant energy advantages over conventional rate-based RNNs [16,19], making it well-suited for low-power neuromorphic navigation systems. By integrating back propagation techniques with biologically inspired mechanisms, we not only advance our understanding of neural mechanism of animal's navigation processes but also provide new insights and methodologies for developing brain-inspired models of navigation.

The main contributions of this work are as follows:

(i) A biologically plausible path integration model is proposed based on the RSNN with LIF dynamics, bridging biological interpretability and artificial intelligence.

(ii) Grid units have traditionally been regarded as the primary contributors to path integration, but we find that stripe units and border units play a more direct and fundamental role in path integration, providing a novel biologically grounded perspective.

(iii) We find that stripe units emerge consistently across different environments, and some formation is independent of specific boundary configurations.

(iv) We highlight the synergistic interactions between locomotor and spatial units, which are essential for accurate path integration.

2. Methods

2.1. Network architecture

The RSNN adopts a three-layer architecture comprising an input layer, a recurrent layer, and an output layer, as illustrated in Figure 1. The input layer includes N_{in} units that receive locomotor stimuli, specifically two one-dimensional scalars representing speed (x_1^t) and direction (x_2^t) at time step t. The recurrent layer consists of N_{rec} LIF units that receive both the locomotor stimuli from the input layer and binary spike signals (0/1 scalars) from other LIF units within the recurrent layer. Although classical biological LIF models typically include an explicit refractory period term that suppresses spiking immediately after a spike event, the LIF implementation adopted in this work (from the snnTorch) does not include an explicit refractory variable. Instead, the spike reset mechanism inherently introduces an implicit refractory-like behavior by reducing the membrane potential to zero after firing. This prevents the neuron from emitting consecutive spikes within a short period unless the input current is sufficiently strong. The behavior also noted in prior works adopted similar simplified LIF models [28–30]. The output layer consists of N_{out} units that generate two-dimensional scalar values (y_1^t, y_2^t), representing the estimated spatial position at time step t.

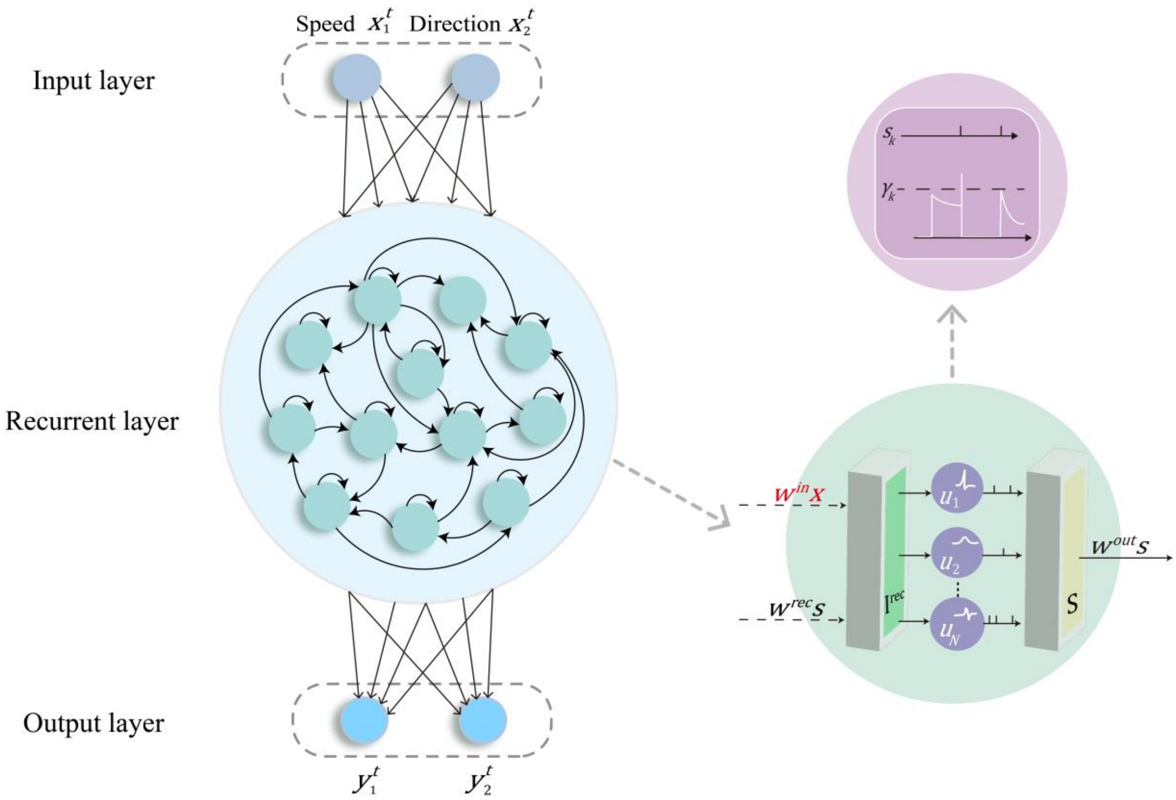


Figure 1. The whole network architecture of RSNN.

The input currents, membrane potential dynamics, and spike emissions of each unit in the recurrent layer of the network are shown in Eqs (1)–(3):

$$I_i^{t+1,rec} = \sum_{k=1}^{N_{in}} w_{ik}^{in} x_k^{t+1} + b_i^{in} + \alpha \cdot \left(\sum_{j=1}^{N_{rec}} w_{ij}^{rec} s_j^{t,rec} + b_i^{rec} \right) \tag{1}$$

$$u_i^{t+1,rec} = \left(1-\beta_i\right) u_i^{t,rec} \left(1-s_i^{t,rec}\right) + \beta_i I_i^{t+1,rec} \tag{2}$$

$$s_i^{t+1,rec} = \Theta\left(u_i^{t+1,rec}, \gamma_i^{t,rec}\right) \tag{3}$$

At discrete time step $t+1$, x_k^{t+1} represents the locomotor stimulus encoded by the input-layer unit k, where $k=1$ corresponds to speed stimulus and $k=2$ to direction stimulus. The weight from input-layer unit k to recurrent-layer unit i is denoted by w_{ik}^{in}. Likewise, $s_j^{t,rec} \in \{0,1\}$ indicates the binary spike signal of recurrent-layer unit j, transmitted to recurrent-layer unit i at time t with weight w_{ij}^{rec}. The terms b_i^{in} and b_i^{rec} are bias applied in the linear mappings from the input and the recurrent layer to recurrent-layer unit i, respectively. The input current $I_i^{t+1,rec}$ received by unit i is jointly determined by both the feedforward input and the recurrent-layer spike stimuli, with α serving as a scaling factor to balance their relative magnitudes. The membrane potential of recurrent-layer unit i at time $t+1$ is denoted as $u_i^{t+1,rec}$, and its temporal dynamics are modulated by the decay coefficient $\beta_i \in (0,1)$ which governs the rate at which the membrane potential decays over time.

$$\Theta(u, \gamma) = \begin{cases} 1, & u \geq \gamma \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

As described in Eqs (3) and (4), when the membrane potential u_i of unit i in the recurrent layer exceeds the firing threshold γ_i the unit emits a spike immediately, otherwise no spike is emitted.

To address the non-differentiability of spike generation in LIF neurons, we employ the fast-sigmoid surrogate gradient provided in snnTorch during backpropagation [31]. In the forward pass, spikes are generated using the Heaviside step function, preserving the discrete event dynamics. In the backward pass, this function is replaced by a smooth surrogate

$$\phi(m) = \sigma(qm), q = 25 \tag{5}$$

where m denotes the difference between the membrane potential u and the firing threshold γ. These trainable parameters, such as the firing threshold γ, the membrane decay rate β or other learnable components appearing in the spike-generation can be optimized using the surrogate gradient.

The surrogate derivative

$$\phi'(m) = \frac{1}{(1+q|m|)^2} \tag{6}$$

is then used to approximate the gradient of the spike function, ensuring that gradients propagate through spike events and allowing all relevant parameters to be optimized. This approach enables the learning of parameters such as membrane potential thresholds and decay rates, following standard methods in surrogate-gradient-based SNN training [28,29].

A linear mapping of the spike stimuli in the recurrent layer is performed to fit the spatial coordinates $\left(y_1^{t+1}, y_2^{t+1}\right)$ as shown in Eq (7):

$$y_k^{t+1} = \sum_{j=1}^{N_{rec}} w_{kj}^{out} s_j^{t+1,rec} + b_k^{out} \tag{7}$$

where w_{kj}^{out} denotes the weight from recurrent-layer unit j to output-layer unit k, and b_k^{out} represents the bias term in the linear mapping from the recurrent layer to output-layer unit k.

2.2. Trajectory generations

The training dataset comprises continuous, smooth, and stochastic trajectories, with the network receiving a fixed number of trajectories per training iteration, defined by the batch size (*BS*). Each trajectory lies on a two-dimensional plane and originates from a common initial position at coordinates (0,0). Both speed and displacement are expressed in dimensionless arbitrary units (*A.U.*). The RSNN is trained to predict the spatial position at each time step based on the corresponding speed and direction stimuli. Trajectories in the training dataset are generated according to the following rule.

The speed v^t at each time step t follows the distribution:

$$v^t \sim \left|N(0,\sigma^2)\right|, \sigma = 0.1 \tag{8}$$

where N denotes a normal distribution and σ is a scaling parameter to simulate the randomness in speed variations during biological movement [3].

The initial direction θ^0 is sampled from a uniform distribution:

$$\theta^0 \sim U(-\pi,\pi) \tag{9}$$

At subsequent time steps, the direction is updated according to Eq (10):

$$\theta^t = \theta^{t-1} + \pi\varepsilon^t,\ \varepsilon^t \sim N(0,\sigma^2), \sigma = \frac{1}{20} \tag{10}$$

where ε^t is a Gaussian perturbation term to simulate directional randomness during biological movement.

We construct four experimental environments: a square, a triangle, a combined (consisting of a triangle and a rectangle) and a circle arena. All environments use the above speed and direction rules to generate trajectories. Due to the presence of boundary constraints in each environment, the boundary-checking mechanism is implemented at every time step. Specifically, if the position generated at a given time step exceeds the environmental boundary, the speed and direction for that step are resampled until the resulting position falls within the permissible area.

At each time step $t \in \{1, 2, \ldots, T\}$, the RSNN receives the locomotor input $x^t = (x_1^t, x_2^t)$ and outputs the predicted spatial position $y^t = (y_1^t, y_2^t)$, while the true position is denoted as $y^{t,\ target} = (y_1^{t,\ target}, y_2^{t,\ target})$. The loss function used for training comprises two components: the prediction error and the metabolic loss [14]. The prediction error is defined as the mean squared error (MSE) between the predicted and true trajectory at each position. The metabolic loss is implemented as L2 regularization term applied to the membrane potentials of the recurrent-layer units. This L2 regularization encourages the membrane potentials of units to remain close to zero [32]. The loss function is shown in Eq (11):

$$Loss_{train} = \frac{1}{BS \cdot T \cdot N_{out}} \sum_{bs,\ t,\ j=1}^{BS,\ T,\ N_{out}} \left(y_j^t - y_j^{t,\ target}\right)^2 + \frac{\delta}{BS \cdot T \cdot N_{rec}} \sum_{bs,\ t,\ k=1}^{BS,\ T,\ N_{rec}} \left(u_k^{t,\ rec}\right)^2 \tag{11}$$

A total of BS trajectories are simultaneously fed into the network in each epoch, and δ is the weight coefficient of metabolic loss (the batch subscript is not explicitly shown in this equation to simplify the notation). To further improve the model's stability in learning the mechanism of path integration, each trajectory is trained for R_t iterations. The network's connection weights (w^{in}, w^{rec}, w^{out}), biases (b^{in}, b^{rec}, b^{out}), firing threshold (γ^{rec}), and leakage factor (β) are updated using the Adam optimization algorithm [33–35], with an initial learning rate of 0.001. The learning rate decays by a factor of 0.1 every 2000 epochs, and the weights are updated once per epoch. The weights within the recurrent layer are initialized using the Kaming's uniform distribution [36], while the firing threshold and leakage factor are initialized from a uniform distribution $U(0,1)$. For validation and testing datasets, novel trajectories are generated to evaluate the performance.

The training parameters are summarized in Table 1.

To evaluate the performance of the trained model, the testing loss of one trajectory with time step T is computed:

$$Loss_{test}=\frac{1}{T\cdot N_{out}}\sum_{t,\,j=1}^{T,\,N_{out}}\left(y_j^t-y_j^{t,\,target}\right)^2 \tag{12}$$

Table 1. Network architecture and training parameters.

Description	Parameters	Value
Number of input-layer units	N_{in}	2
Number of recurrent-layer units	N_{rec}	512
Number of output-layer units	N_{out}	2
Scaling factor	α	0.15
Coefficient of metabolic loss	δ	0.001
Batch size	BS	256
Total time step	T	300
Number of training epochs	Epoch	6500
Repetition of each trajectory	R_t	50
Boundary length of the square	$lrange_{square}$	4
Boundary length of the triangle	$lrange_{triangle}$	6
Boundary length of the combined	$lrange_{combined}$	4
Radius of the circle	$lrange_{circle}$	4

2.3. Evaluation metric of biological responses

2.3.1. Direction tuning score

The firing rates of head direction cells are strongly modulated by the animal's head orientation in the horizontal plane, independent of its position, posture, or behavior [4]. These neurons exhibit elevated activity when the head aligns with a specific preferred direction and reduced activity as the orientation deviates, indicating strong directional tuning [37]. This preferred direction corresponds to the cell's optimal orientation.

We adopt the quantization method to identify HD units in the recurrent layer [38–40]. The planar direction is uniformly divided into 36 intervals $\{[10(j\text{-}1), 10j]^{\circ}\}_{j=1}^{36}$, which spans 10° in the counterclockwise direction starting from 0° (aligned with the rightward horizontal direction), and θ_j represents the middle value of the j-th interval. For each unit i, the firing rate within each interval $[10(j\text{-}1), 10j]^{\circ}$ is computed as the ratio of spikes emitted by unit i within $[10(j\text{-}1), 10j]^{\circ}$ to the number of times the head direction occurs in the interval $[10(j\text{-}1), 10j]^{\circ}$. $R_{i,j}^{d}$ represents the firing rates of unit i over the j-th interval. These 36 values form a vector $R_i^d = [R_{i,1}^d, R_{i,2}^d, \ldots, R_{i,36}^d]$, representing the unit's firing rates across all directions.

To avoid potential confusion between the imaginary unit i in the complex exponential and the unit index i, we use the subscript p to denote the unit index in the next formula. The direction score $Direction_p$ of the recurrent unit p is calculated according to Eq (13):

$$Direction_p = \frac{\left|\sum_{j=1}^{36} R_{p,j}^{d}\, e^{\mathrm{i}\pi\theta_j/180}\right|}{\sum_{j=1}^{36} R_{p,j}^{d}} \tag{13}$$

where each direction is encoded as a complex unit vector, and the weighted sum produces a complex vector whose magnitude reflects the degree of directional tuning. If a unit's firing rates are concentrated around a particular direction, the vectors sum constructively, resulting in $Direction_p$ approaching 1, indicating that unit p exhibits a strong preference for a specific orientation (it reaches its highest firing rate within a particular direction range, with the firing rate gradually decreasing as the orientation shifts away from this range). Conversely, if firing rate is broadly distributed across directions, the vectors partially cancel, yielding $Direction_p$ approaching 0, suggesting that the unit lacks directional selectivity. A predefined score threshold $Direction_{th}$ is used to classify unit p as an HD unit if it satisfies $Direction_p > Direction_{th}$.

To aid interpretation, we provide firing-rate tuning curves and polar plots for both a HD unit and a non-HD unit in the Appendix (Figure A1). These figures illustrate the contrast between strongly directional and non-directional units, offering a more intuitive understanding of how $Direction_p$ captures directional selectivity.

2.3.2. Speed tuning score

Speed cells are neurons whose firing rates exhibit a monotonic, approximately linear relationship with speed. These neurons were initially identified in the medial entorhinal cortex of rodents [3].

We adopt the quantization method for speed units in the recurrent layer [41]. Specifically, the speed ranges from 0 to 0.2, which covers 95% of the total speed range approximately, and it is then discretized into 20 intervals $\{[0.01(j\text{-}1), 0.01j]\}_{j=1}^{20}$ and v_j represents the right value of the j interval.

For each unit i, its firing rate across these intervals, denoted as $\{R^v_{i,j}\}^{20}_{j=1}$, is computed. The speed score $Speed_i$ is then calculated by $\{R^v_{i,j}\}^{20}_{j=1}$ and $\{v_j\}^{20}_{j=1}$, as formalized in Eq (14).

$$Speed_i = \left| \frac{\sum_{j=1}^{20}(v_j - \bar{v})(R^v_{i,j} - \bar{R}^v_i)}{\sqrt{\sum_{j=1}^{20}(v_j - \bar{v})^2}\sqrt{\sum_{j=1}^{20}(R^v_{i,j} - \bar{R}^v_i)^2}} \right| \tag{14}$$

where $\bar{v}$ and $\bar{R}^v_i$ represent the mean values of $\{v_j\}^{20}_{j=1}$ and $\{R^v_{ij}\}^{20}_{j=1}$. $Speed_i$ approaching 1 indicates that unit i exhibits a strong linear relationship for speed, while the value near 0 suggests a weaker or no such relationship. A predefined score threshold $Speed_{th}$ is used to classify unit i as a speed unit if it satisfies $Speed_i > Speed_{th}$.

2.3.3. Border tuning score

In biological navigation systems, neurons that selectively respond to environmental boundaries are defined as border cells, which are primarily located in the entorhinal cortex and the parahippocampal gyrus [42,43]. We define the quantization method for border units in a square environment as follows.

The environment is first evenly partitioned into $L \times L$ ($L = 32$). In this case, for spatial regions, firing rate of each recurrent-layer unit i is computed to generate a spatial firing map $f_i\ (\in R^{L \times L})$, which is then normalized to obtain $f_i^{norm} (\in R^{L \times L})$. The process is repeated separately for each of the four boundaries (i.e., top, bottom, left, right). Take the top direction for example, a smoothing operation is applied to f_i^{norm} from top to bottom with step s ($s = 2$) to obtain $f^{smooth}_{i,top} (\in R^{(L\text{-}s+1) \times L})$. The normalization and smoothing formulations are described in Eqs (15) and (16):

$$f_i^{norm} = \frac{f_i - min(f_i)}{max(f_i) - min(f_i) + \eta} \tag{15}$$

$$f^{smooth,k}_{i,top} = \frac{1}{s}\sum_{j=0}^{s-1} f_i^{norm,\ k+j}, k = 1, \ldots, L\text{-}s+1 \tag{16}$$

Here, $f^{smooth,k}_{i,\ top}$ denotes the k-th smoothed row vector of unit i from the top direction. The terms $max(f_i)$ and $min(f_i)$ represent the maximum and minimum value of f_i, and $\eta = 1\times10^{-9}$ is added to prevent the division-by-zero case that occurs for silent units (defined as those with zero firing rates under input stimuli). For the smoothed firing rate series $\{f^{smooth,k}_{i,\ top}\}^{L\text{-}s+1}_{k=1}$, two metrics are computed:

(i) r_i^k, the mean value of smoothed row vector $f^{smooth,k}_{i,\ top}$.

(ii) m_i^k, the proportion of elements exceeding 0.3 in $f_{i,\ top}^{smooth,k}$.

Subsequently, the maximum number of consecutive rows $L_{i,\ border}$ that satisfy $r_i^k > R_{border}^{high}$ and $m_i^k > R_{border}^{valid}$ is computed, in which R_{border}^{high} and R_{border}^{valid} are the thresholds $(R_{border}^{high} = 0.3, R_{border}^{valid} = 0.5)$. On this basis, the boundary segmentation line is calculated. When $L_{i,\ border}= 0$, the length of the segmentation line is set to 1 to ensure non-negative; otherwise, the smaller value between $L_{i,\ border}$ and 8 is taken, in which 8 is the ideal length of the boundary representations. The top border score of unit i is shown in Eq (17):

$$Border_{i,\ top}=\left(F_{i,\ in}\text{-}F_{i,out}\cdot P_{i,\ out}\right)\cdot\left(1\text{-}\frac{L_{i,\ border}}{L}\right) \tag{17}$$

where $F_{i,\ in}$ denotes the average value of firing rates on the inner side of the segmentation line $L_{i,\ border}$ (the side close to the starting direction), and $F_{i,\ out}$ represents the average value of firing rates on the outer side. The term $P_{i,\ out} \in [0,1]$ indicates the proportion of firing rates greater than 0 in the outer side. The border score for each unit is computed independently along the four boundaries, and the maximum score among them is selected as the overall border score, denoted as $Border_i$. A predefined score threshold $Border_{th}$ is used to classify unit i as a border unit if it satisfies $Border_i > Border_{th}$.

2.3.4. Stripe tuning score

V. Schøyen et al. proposed that neurons exhibiting band-like responses in spatial responses may play a prominent role in path integration [44]. Similarly, we observed units whose spatial firing maps displayed band-like patterns in the recurrent layer.

We define the quantization method for stripe units in a square environment as follows. The environment is first partitioned into $L \times L$ (L = 32) spatial regions, and the spatial firing map $f_i\left(\in R^{L \times L}\right)$ and $f_i^{norm}\left(\in R^{L \times L}\right)$ are calculated for each recurrent-layer unit. Analyses were conducted separately along the vertical and horizontal directions. Take the case of the vertical direction. The firing map was smoothed from top to bottom with the step s (s = 2) to get $\{f_{i,\ vertical}^{smooth,\ k}\}_{k=1}^{L\text{-}s+1}$ as in Eq (16). For each smoothed row vector, we compute the metrics r_i^k and m_i^k as described previously.

Candidate bands $B = \{(s_k, e_k)\}_{k=1}^{N}$ are identified as continuous intervals, where each interval satisfies the criterion $r_i^k > R_{stripe}^{high}$ and $m_i^k > R_{stripe}^{valid}$ (k=s_k,...,e_k). Here, s_k and e_k denote the start and end indices of the k-th candidate band, R_{stripe}^{high} and R_{stripe}^{valid} are the thresholds (R_{stripe}^{high} = 0.3, R_{stripe}^{valid} = 0.5), and N is the total number of candidate bands. The stripe score of the unit i along the vertical direction is calculated in Eqs (18) and (19):

$$B^{temp,\,k} = \left(F_{k,\,stripe} - 0.1 \cdot max\left(r_i^{left,\,k}, r_i^{right,\,k}\right)\right) \cdot \left(1 - \frac{e_k - s_k}{L}\right) \tag{18}$$

$$Stripe_{i,\,hori} = max(\{B^{temp,\,k}\}_{k=1}^{N}) \cdot \sqrt{N} \tag{19}$$

$F_{k,stripe}$ denotes the average value of firing rates in the k-th candidate band. For each candidate band $B = (s_k, e_k)$, the adjacent mean values are defined as $r_i^{left,k} = r_i^{s_k - 1}$ (if $s_k > 0$, otherwise take 3 to reduce the stripe score of the border unit), and $r_i^{right,k} = r_i^{e_k + 1}$ (if $e_k < L - 1$, otherwise take 3). Ultimately, the stripe score $Stripe_i$ of unit i is defined as the larger of its horizontal and vertical stripe scores. A predefined score threshold $Stripe_{th}$ is used to classify unit i as a stripe unit if it satisfies $Stripe_i > Stripe_{th}$.

3. Results

3.1. Key hyperparameters

Before conducting large-scale training, we first performed a systematic evaluation of key hyperparameters to identify suitable settings for the subsequent experiments. All models were trained for 6500 epochs under each configuration. Figure 2 summarizes the effects of BS, N_{rec}, and δ on model performance and neural activity patterns.

To ensure a consistent and comparable evaluation, we used a fixed set of 100 testing trajectories generated from two random seeds. For each hyperparameter configuration in Figures 2(a)–(c), models were trained under the corresponding setting and then evaluated on the same testing trajectories. The resulting testing losses were summarized using boxplots, which visualize the distribution of testing losses under two evaluation horizons: $T = 300$ (blue boxplots) and $T = 4000$ (red boxplots).

Figure 2(a) shows the testing losses for models trained with different batch sizes ($BS = 32$, 128, 256, 512) under two evaluation horizons: $T = 300$ and $T = 4000$. Across both horizons, the testing loss decreases as the batch size increases. For $T = 300$, the median loss drops from roughly 0.28 at $BS = 32$ to approximately 0.16 at $BS = 256$, 512. A similar trend is observed for $T = 4000$, where $BS = 256$ and $BS = 512$ again yield noticeably lower testing loss than smaller batch sizes. Although $BS = 512$ achieves comparable performance, $BS = 256$ provides a favorable trade-off between model accuracy and computational cost. Therefore, $BS = 256$ is selected as the default batch size for subsequent experiments.

Figure 2(b) shows the influence of different N_{rec} on test performance. At $T = 300$, the 512-unit model achieves the lowest median testing loss (≈ 0.16), outperforming both smaller (256 units) and larger (1024 units) configurations. A similar pattern is observed at $T = 4000$, where the 512-unit model again exhibits the lowest median testing loss (≈ 0.24). The 1024-unit model shows a wider distribution range of testing losses, likely because its larger capacity requires more training epochs to fully stabilize. Within the same training epochs, this limited convergence leads to higher observed testing losses. These results indicate that a recurrent size of 512 provides a favorable balance between model capacity, temporal stability, and computational efficiency during both training and testing. Therefore, N_{rec} = 512 is selected as the default configuration for the subsequent training.

Figure 2(c) illustrates the effect of the metabolic loss coefficient (δ = 0, 0.001, 0.01) on testing losses at T = 300 and T = 4000. At T = 300, the median testing loss remains relatively stable (≈ 0.16) for δ = 0 and δ = 0.001, but increases noticeably to 0.44 for δ = 0.01, indicating that excessively strong metabolic regularization impairs performance. A similar trend is observed for T = 4000, where small δ values maintain low testing losses, while δ = 0.01 substantially elevates the testing loss. These results suggest that small but nonzero metabolic loss coefficients can regularize the network without sacrificing accuracy. Moreover, we observed that an appropriate metabolic loss coefficient also promotes the emergence of biologically plausible representations in the recurrent-layer units. Consequently, δ = 0.001 is chosen as the default value for subsequent experiments.

Figure 2(d) shows the sparsity of recurrent-layer unit activities under different settings of the recurrent-layer size N_{rec}. For each configuration, we evaluated 100 testing trajectories (generated from two random seeds) and computed the average firing rate at each time step. Specifically, the firing rate at time t is defined as

$$r(t) = \frac{1}{100 * N_{rec}} \sum_{i=1}^{100} \sum_{j=1}^{N_{rec}} s_{i,t,j} \tag{20}$$

where $s_{i,t,j} \in \{0,1\}$ denotes whether unit j fired at time t in trajectory i.

Across all configurations, the networks operate in a low-activity regime (firing rates < 0.30), consistent with the sparse dynamics typical of spiking neural network models. The relationship between activity and recurrent-layer size is nonmonotonic: the 256-unit and 1024-unit models exhibit higher firing rates, whereas the 512-unit model maintains the lowest activity level (≈ 0.08). In smaller networks (256 units), each unit must participate more heavily in representing the trajectory, leading to stronger effective recurrent drive per unit and consequently elevated firing activity. In contrast, very large networks (1024 units) dilute inputs across many units, but the increased number of recurrent connections amplifies total network-level excitation, making it harder for individual units to remain silent unless strong regularization is applied. This suggests that 512 units form a balanced regime—small enough to avoid persistent self-excitation yet large enough to distribute computation—resulting in both sparser activity and better path-integration performance.

Figure 2(e) shows how the metabolic loss coefficient δ shapes the overall firing activity of the recurrent layer. The firing rate is computed using the same definition as in Eq (20). As δ increases, the metabolic penalty progressively suppresses neuronal activity, leading to a clear downward trend across conditions. Without the penalty (δ = 0), the network exhibits the highest overall activity. Introducing a mild constraint (δ = 0.001) yields the sparsest and most stable firing pattern, indicating that this setting achieves an effective balance between metabolic efficiency and reliable recurrent dynamics. Larger penalties further reduce activity but provide diminishing benefits, as overly strong suppression can slightly destabilize the recurrent state.

Figure 2(f) shows the comparison between the predicted trajectories and the corresponding true trajectories. Overall, the model exhibits strong generalization capability under these hyperparameters, and the predicted trajectories largely match the true trajectories.

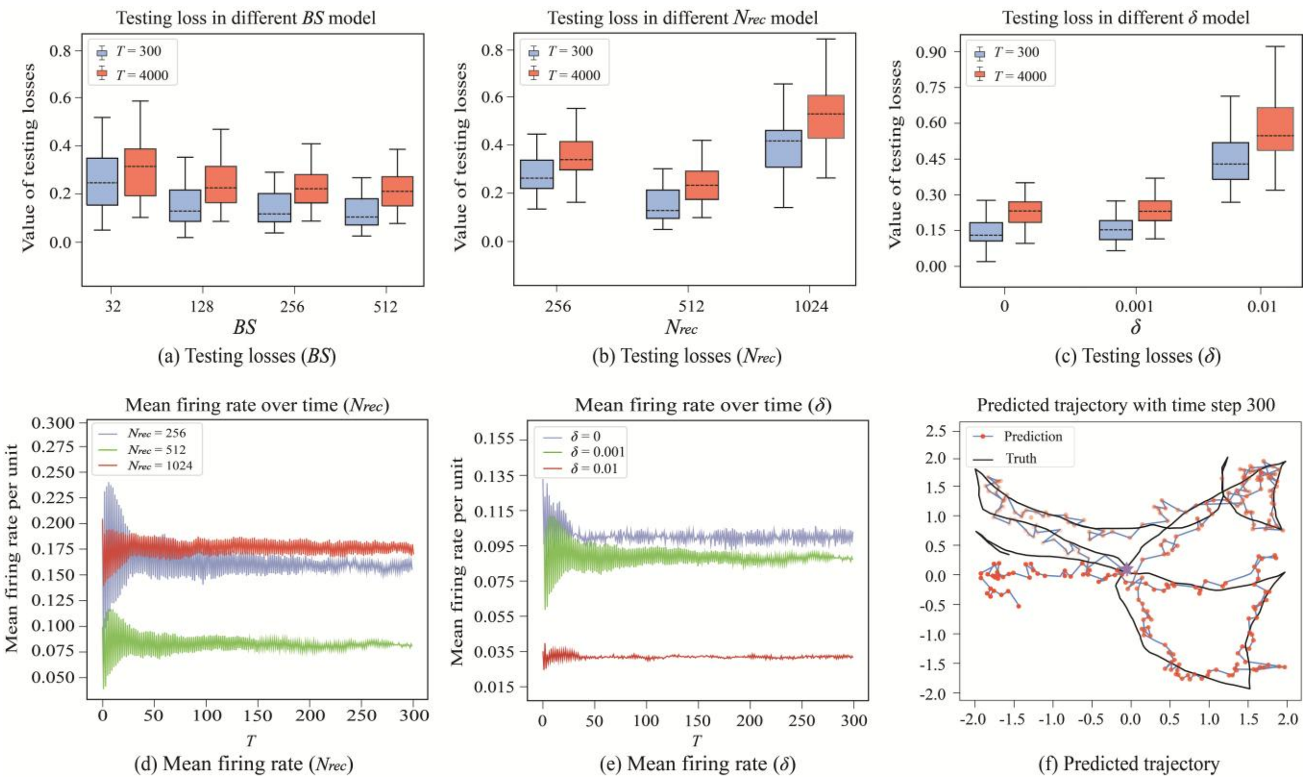


Figure 2. Visualization of model performance, firing statistics under different hyperparameter settings and trajectory prediction.

3.2. Model training and validation

After establishing the key hyperparameters and architectural settings, a total of 6500 training epochs is conducted in our work, and the dynamic changes of the network during training are illustrated in Figure 3.

Figure 3(a) shows the loss curve over training epochs. The training loss decreased sharply during the early stage (epoch < 200), from 1.4 to about 0.2. As training continued, the decline slowed, stabilizing near 0.2 around epoch 2000 and converging to 0.17 by epoch 6500, indicating that the RSNN reached a steady state.

Figure 3(b) shows spike rasters of 100 out of 512 recurrent-layer units at the early (epoch 1) and late (epoch 6500) training stages. At the early stage, the spike rasters were dense and uniform, reflecting the high sensitivity to input stimuli. By epoch 6500, the rasters became sparse and diverse: Some units fired frequently and dominated information transmission, while others were mostly silent. This result suggests that training leads to functional differentiation among units.

Figure 3(c) shows the testing loss curve on the long trajectories over 4000 time steps. The testing loss gradually rises up in the first 1000 time steps, then rises more slowly and stabilizes around 0.24. These results indicate that the model maintains good stability and generalization performance, even during long-term path integration.

Figure 3(d) shows the evolution of weight distributions over training epochs and the area of each curve is 1. Input-to-recurrent weights followed a symmetric bell-shaped distribution, with the center near zero. By the end of training, the mean value was 0.0198, with about 70% of weights in [-0.11, 0.09], indicating that only a small subset of recurrent-layer units with strong connections from the input layer were affected by input stimuli. Recurrent-to-recurrent weights exhibited a similar pattern (mean: -0.002, 70% in [-0.112, 0.119]), suggesting weak inhibitory interactions. Recurrent-to-output weights were also narrowly distributed (mean: 0.0042, 90% in [-0.03, 0.07]).

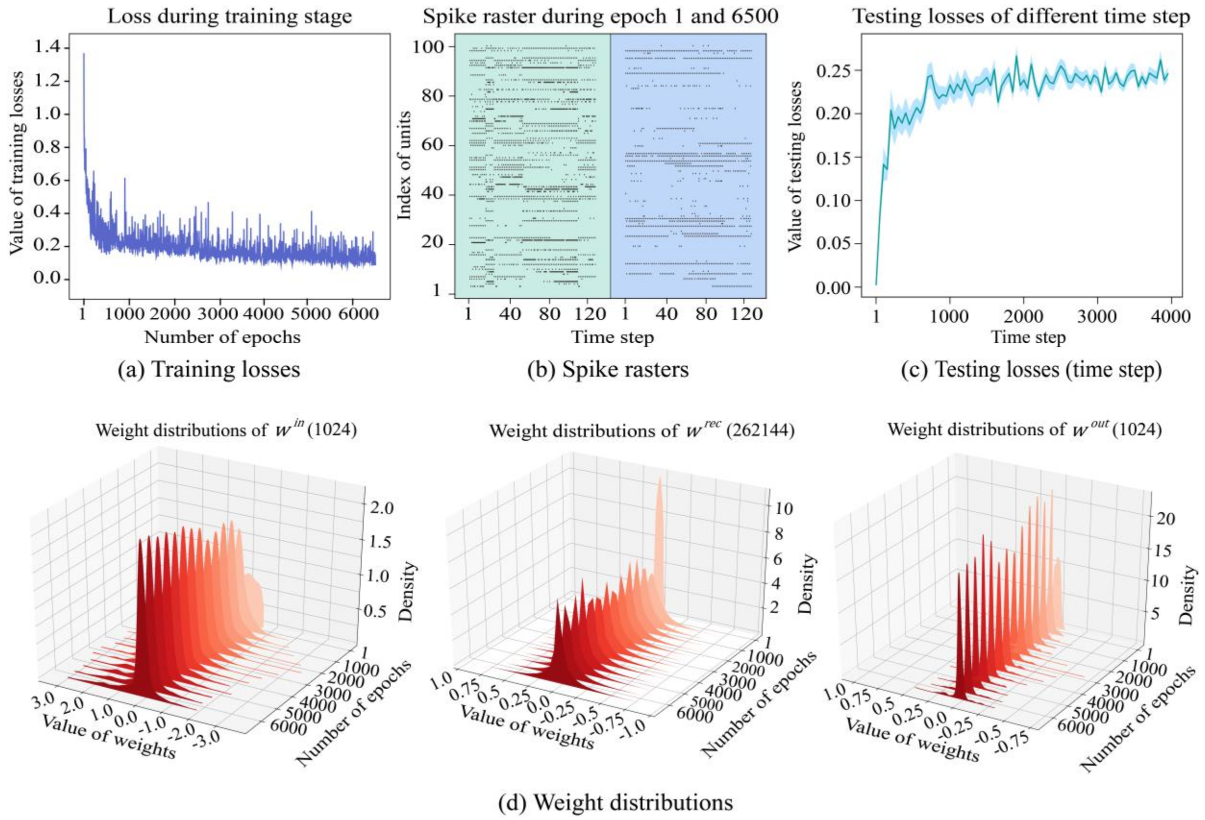


Figure 3. Visualization of RSNN training and testing stages.

3.3. Neural representations

During the execution of path integration, the RSNN spontaneously develops a variety of representational units whose firing patterns resemble those of biological neurons, including locomotor and spatial units. The diverse firing patterns of recurrent-layer units in RSNN are illustrated below. The criteria for quantifying the neural representation scores are described in methods, with score thresholds set as follows:

$$Direction_{th} = 0.3,\ Speed_{th} = 0.5,\ Border_{th} = 0.4, \text{ and } Stripe_{th} = 0.3.$$

We further verified the robustness of these thresholds by varying them within a ± 0.1 range. The number and proportion of different unit types remained largely unchanged under these perturbations, indicating that the classification of units is stable. Appendix Figure A2 compares the unit type distributions obtained using the default thresholds and the elevated thresholds, showing highly consistent patterns.

To better visualize the representations of locomotor units, we restrained the speed range from 0 to 0.2 (accounting for 95% of observed speeds), and generated 1000 trajectories, each with 10,000 time steps.

For HD units, we computed the mean firing rates across direction intervals $\{[10(j\text{-}1), 10j]^{\circ}\}_{j=1}^{36}$ while keeping speed fixed. The mean firing rate for a direction interval was calculated by dividing the total number of spikes emitted when the input direction was within that interval, by the total number of time steps it spent in that interval.

For speed units, we computed the mean firing rates across speed intervals $\{[0.01(j\text{-}1),0.01j]\}_{j=1}^{20}$ while keeping direction fixed, using calculation method like HD units.

The result of locomotor units is summarized in Figure 4.

Figure 4(a) shows examples of HD units. The horizontal axis represents direction stimuli (0° to 360°), and the vertical axis indicates firing rates. Subplot titles show the direction scores. The top two rows display HD units, while the bottom two rows show units with no clear directional preference. Figure 4(b) shows examples of speed units. The horizontal axis denotes speed stimuli (0 to 0.2), and the vertical axis indicates firing rates. Subplot titles show the speed scores. The top two rows are speed units, while the bottom two rows show units with no clear response to speed.

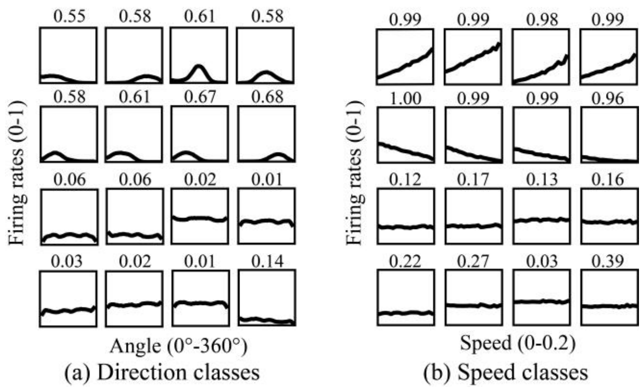


Figure 4. Firing rates and scores of direction classes and speed classes.

From the perspective of spatial units, we also generated 1000 trajectories, each with 10,000 time steps (determined by Eqs (8)–(10)). Take the square environment as an example: The environment is first evenly partitioned into $L \times L$ ($L = 32$). For each spatial region, the mean firing rate of a unit was obtained by dividing the total number of spikes emitted by the unit when the real trajectory fell within that region by the total number of time steps in which the position in the real trajectory was located in that region. The resulting mean firing rate maps were then visualized as Figure 5, where color intensity reflects the magnitude of the firing rate in each region.

(i) Locomotor units: In the square environment ($lrange_{square} = 4$), 23.6% (121 out of 512) of units in the RSNN met the criteria for HD units, as illustrated in Figure 5(a) (top two rows). Most of these HD units exhibited a clear preference for a single preferred orientation, consistent with experimental findings in rodents [40]. Units without obvious directional preference ($Direction_i < Direction_{th}$)

accounted for 13.9% (71 out of 512), excluding silent units (defined as those with zero firing rates under input stimuli). Additionally, 30.7% (157 out of 512) of units in the RSNN met the criteria of speed units, as shown in Figure 5(b) (top two rows). Among them, 21.1% (108 out of 512) exhibited positive correlation with locomotion speed, while 9.6% (49 out of 512) showed negative correlation. Units with no clear speed selectivity (exhibiting similar firing rates across different speeds, excluding silent units) accounted for 13.7% (70 out of 512).

(ii) Spatial units: We considered four types of environments — square arena, triangular arena, combined arena and circular arena — to investigate the spatial representation of units under varying spatial geometries. Specifically, for the triangle and the combined environment, the bias terms (b^{in},b^{out}) of the network were fixed to zero vectors, while all other parameters were kept consistent with those in Table 1. The results are presented in Figure 5.

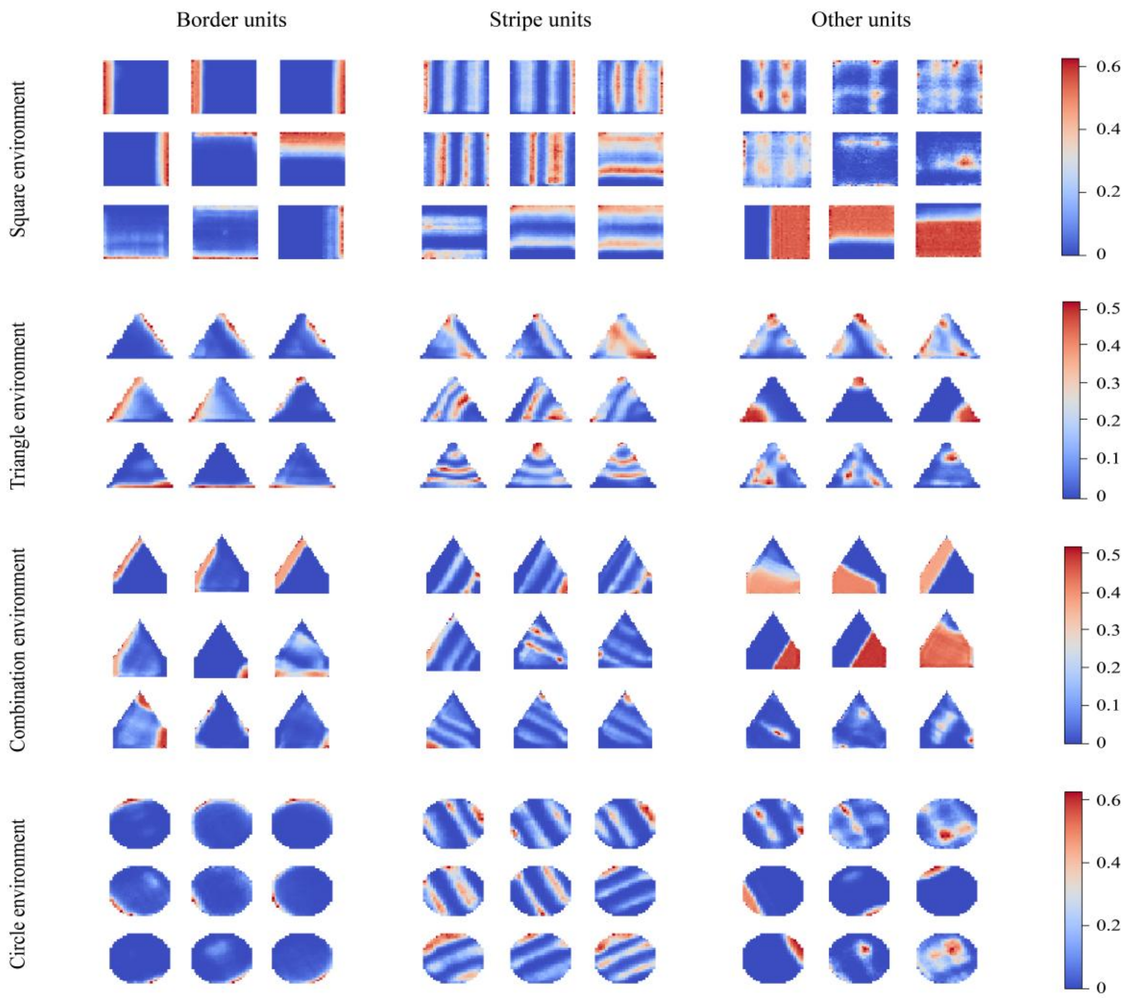


Figure 5. Spatial units in different environments: border units, stripe units, and other units.

In the square environment (top row of Figure 5) the RSNN developed border units that fired strongly near boundaries. Additionally, stripe units exhibited one to three spatially parallel bands, with their orientations largely aligned to the boundaries. A small subset of units (4 out of 512) displayed grid-like responses, as identified through visual inspection; however, unlike the typical hexagonal periodic firing, these units showed a quadrilateral pattern of activity.

In the triangle environment (second row of Figure 5), some units still showed strong firing rates near boundaries. Stripe units remained prevalent, with firing patterns showing curvature. Notably, some units exhibited boundary-selective firing patterns (i.e., the second row of other units in the triangle environment), similar to boundary vector cells (BVCs) identified in rodents [42].

To further investigate the influence of environmental geometry on spatial representations, we designed a combined environment composed of a rectangle and triangle with equal areas (third row of Figure 5). The RSNN similarly developed border and stripe units. The stripe units exhibited 2 to 3 firing bands. Unlike previous environments, band-like responses in this combined environment were tilted and did not strictly align with the edges of either the rectangle or triangle. Border units continued to fire around environmental edges, with some encoding both triangle and rectangle boundaries.

In the circle environment (bottom row of Figure 5), border units fired locally around specific arc segments, a pattern consistent with classical border cells, which typically respond to successive edges in a direction range [45]. This supports the result that the geometric structure of the environment shapes the receptive field properties of border cells [46]. The network under the circle environment also gave rise to distinct types of response units, including stripe units exhibiting one to three firing bands, as well as a subset of emergent place units, grid units, and BVCs. Notably, the band-like responses also exhibit a tilt orientation similar to BVCs, suggesting a close functional relationship between the two again.

These results demonstrate that the RSNN can develop locomotor and spatial representations across diverse environments. The emergence of such representations enables the network to infer environmental features and perform path integration effectively. This provides a new computational perspective for understanding how biological neural systems learn and adapt to different spatial contexts.

3.4. Evolution of spatial units

The spatial response patterns of both stripe and border units exhibited strong consistency across different environments, indicating that these representations are not tied to any specific geometry but instead reflect a more generalizable spatial coding mechanism. To further investigate the relationship between these two unit types and to understand how such structured responses develop over the course of training, we examined the temporal evolution of their firing patterns during training in the square environment, as shown in Figure 6. As the learning rate was reduced to 10^{-6} after epoch 6000, we extended the evaluation for another 1000 epochs to further observe the stability of the emergent biological neural representations.

In Figure 6(a), most units that developed final border sensitivity showed strong responses to environmental boundaries in early training stage and maintained stable firing patterns throughout. A few units transiently exhibited border-stripe conjunctive responses during middle training stage but eventually evolved to stable boundary responses, suggesting that border units are generally stable. In Figure 6(b), units that develop final stripe sensitivity remained largely neither border or stripe during the early training stage and gradually developed stable band-like responses in the later stages, indicating that their response patterns required extended learning and implying the crucial role of input from other types of units in the recurrent layer. Notably, some final stripe units initially exhibited boundary responses, highlighting a strong developmental link between the two unit types. Figure 6(c) presents final border-stripe conjunctive units identified at the end of training. Some units initially responded to boundaries, gradually developed band-like responses during middle training stage, and eventually stabilized as co-encode units that integrate both spatial features.

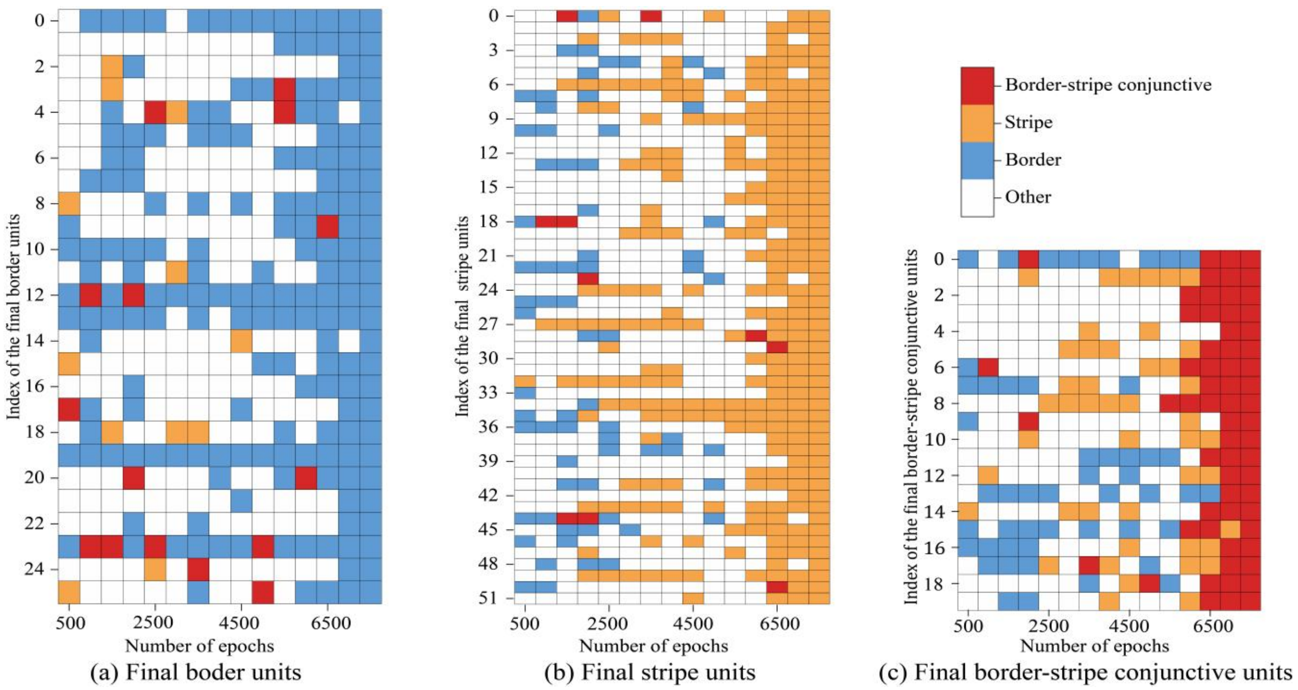


Figure 6. Evolution of spatial units during training.

3.5. Connection patterns and collaborative analysis

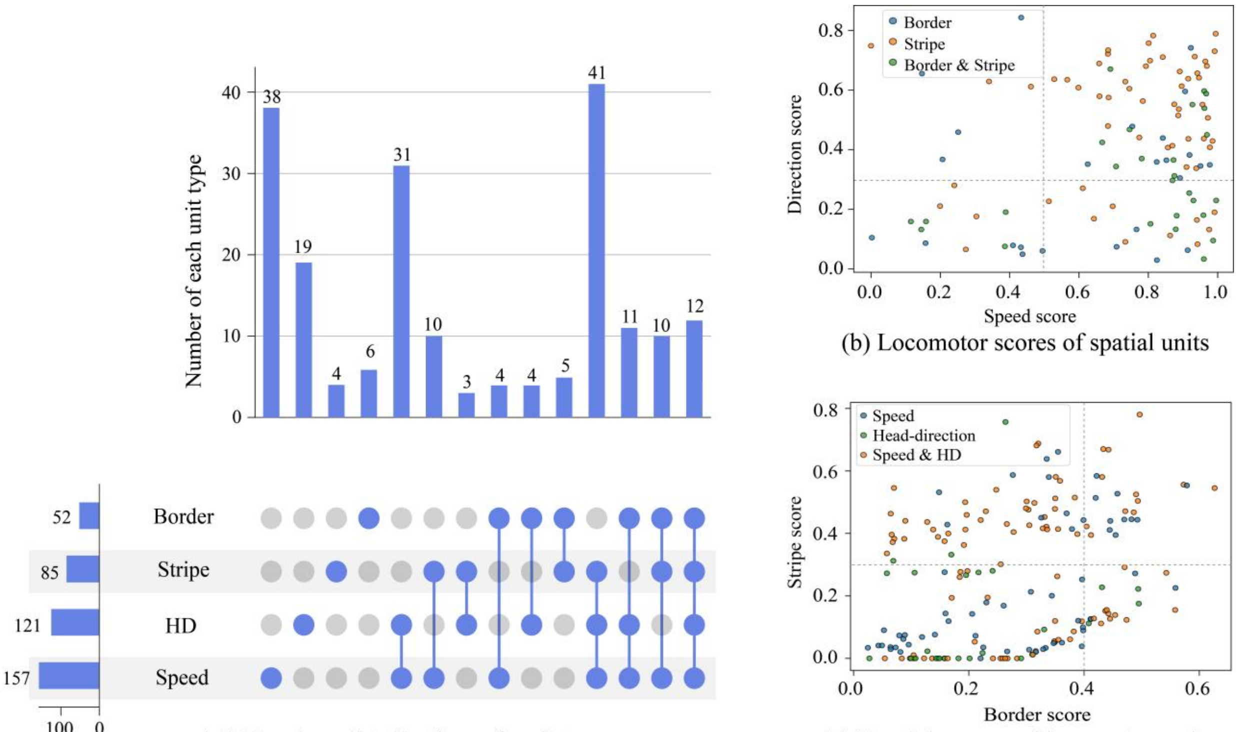
After characterizing the developmental trajectories of spatial units, we next shifted our focus to all functionally meaningful units in the network in order to better understand their collective organization. To probe the functional connectivity between spatial units and locomotor units, we first quantified the numerical composition of each unit type and visualized their overall distribution. Building on this, we examined how unit types relate to the incoming connection weights, allowing us to assess whether units with spatial selectivity also exhibit distinctive connectivity patterns. This analysis provides a more integrated view of how diverse functional representations co-emerge and interact within the network.

As shown in Figure 7(a), spatial units more often co-encode information with locomotor units than appear in isolation, whereas many locomotor units encode independently, with some also forming conjunctive representations with spatial units. Within the locomotor category, speed-HD conjunctive units make up a substantial proportion. Strikingly, stripe-speed-HD conjunctive units account for nearly half of all stripe units, a pattern absent among border units, suggesting that stripe units are strongly linked to the presence of locomotor units.

In Figure 7(b), 110 units with spatial representations were classified into border, stripe, and border-stripe conjunctive units, and subsequently were mapped based on their speed and direction scores to analyze their coupling with locomotor units. Among border units, 44.0% (11 out of 25) exhibited high scores in both speed and HD metrics, and 32.0% (8 out of 25) responded strongly to either speed or orientation alone. Notably, the units with conjunctive sensitivity to boundaries and orientations were similar to biologically observed BVCs [42]. A majority of stripe units, 70.7%

(41 out of 58), exhibited high scores in speed and direction scores, suggesting their involvement in encoding both spatial and locomotor information. Similarly, 44.4% of border-stripe conjunctive units (12 out of 27) were located in high speed and high direction score regions, suggesting integration of spatial and locomotor information, akin to units that co-encoded position and movement [47].

In Figure 7(c), 183 units with locomotor representations were classified into speed, HD and speed-HD conjunctive units, and subsequently were mapped based on their border and stripe scores to analyze their coupling with spatial units. The majority of speed units (61.3%, 38 out of 62) exhibited low scores in both dimensions, indicating that they primarily performed independent encoding. A smaller subset (38.7%, 24 out of 62) showed high stripe or border scores, facilitating involvement in spatial representation. Most HD units were also spatially independent, with 73.1% (19 out of 26) located in regions of low border and stripe scores. Only a small proportion (26.9%, 7 out of 26) showed high stripe or border scores. These results indicate that locomotor units predominantly encoded movement-related information without strong spatial associations. Nevertheless, a minority of locomotor units were spatially linked, facilitating information integration. Notably, speed-HD conjunctive units were distributed relatively evenly across the four regions defined by border and stripe score thresholds, reflecting their diverse encoding properties. Figure 7(b) and Figure 7(c) together demonstrate that locomotor units (HD/speed) are functionally upstream of the spatial units (border/stripe), since the locomotor information encoding is relatively independent and the majority of spatial units contain locomotor information, mathematically consistent with their function in path integration.



(a) Number distribution of unit types

(b) Locomotor scores of spatial units

(c) Spatial scores of locomotor units

Figure 7. Statistics of different functional unit types.

We also analyzed the incoming connection weights of spatial units. For each unit, incoming weights from other units were ranked from largest to smallest, and the largest k values (*Top-k*, $k = 1, 2, ..., 512$) were selected. We then quantified the number and weight distributions of connections from border or stripe units to different unit types (grid, stripe, border and non-spatial). In Figure 8, as k increased, grid units progressively established more connections from stripe units, although the corresponding weights remained relatively small. This pattern suggests that the two-dimensional periodicity of grid units may rely on integrating multiple one-dimensional periodic inputs from stripe units. In contrast, connections from border units exhibited higher weight values, underscoring the importance of salient boundary information in stabilizing grid activity. Border and stripe units both received more connections from border units than from stripe units, although their weight distributions showed no clear preference. Figure 8 further shows that non-spatial units received fewer connections from border and stripe units. When $k = 50$, nonspatial units had only about 0.05 ratio of connection numbers, underscoring their limited direct involvement in spatial representation.

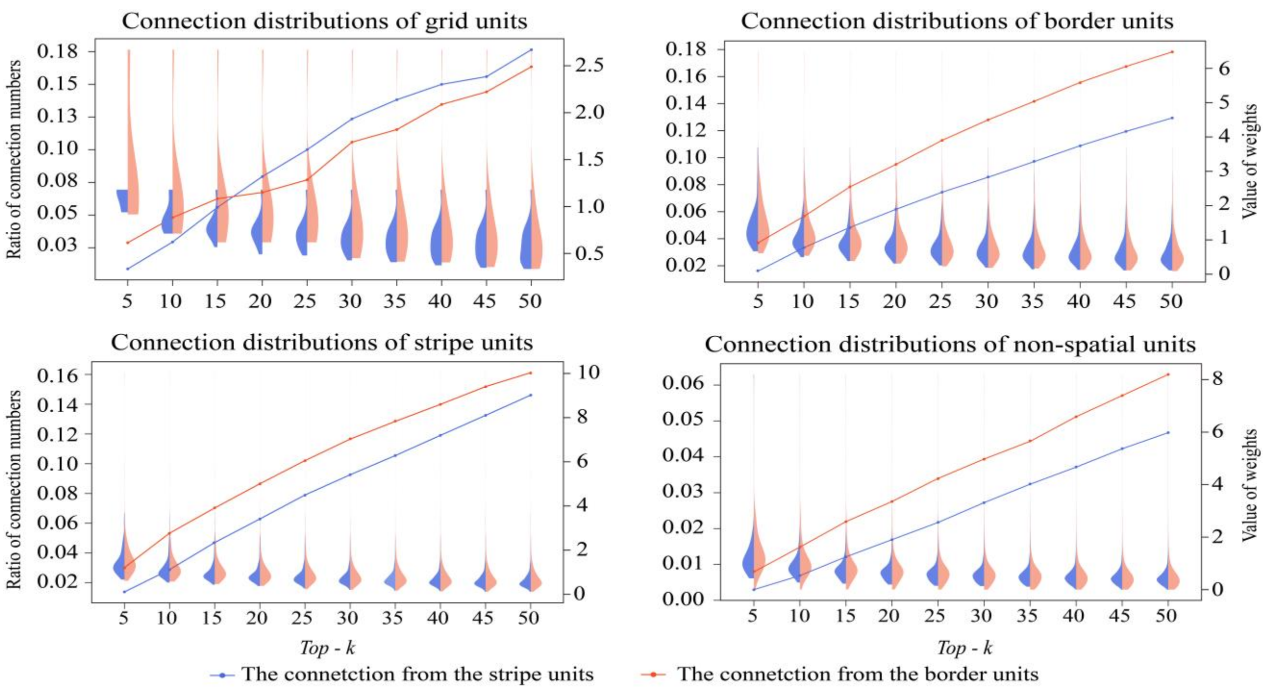


Figure 8. Connection weights of spatial units and non-spatial units.

Compared with nonspatial units, spatial units formed more connections from both border and stripe units, with an overall stronger connectivity from border units. In particular, in grid units, the preference for stripe units eventually exceeded that for border units. These results suggest that border units serve as a salient feature in navigation, and stripe units play an important role in facilitating complex spatial encoding (grid units).

Overall, spatial units, particularly border and stripe units, played a crucial role in integrating spatial and locomotor information, while locomotor units operated independently. These results suggest that locomotor units may predominantly occupy upstream positions to process primary input stimuli, and then transmit information through locomotor-spatial conjunctive units. These results provide important insights into the functional differentiation and synergistic mechanisms among unit types during path integration.

3.6. Ablation experiments

To further investigate the functional contributions of border and stripe units in path integration, we conducted an ablation experiment on spatial representation units within the RSNN. Specifically, among the 110 units exhibiting spatial representations, those with distinct spatial units (border and stripe units) were selected, and a control group was concurrently established, comprising the top 30 high weight units defined by the absolute magnitude of their incoming connection weights, then were randomly deleted at different proportions. The resulting impact on model performance in path integration was evaluated and is shown in Figure 9. Across all pruning ratios, the removal of stripe units consistently led to the most significant degradation in path integration performance, followed by border units. In contrast, pruning high weight units resulted in comparatively minor increases in testing loss. This pattern strongly suggests that, within our RSNN model, spatially tuned units — particularly stripe units — contribute more critically to successful path integration than units whose prominence is solely defined by the magnitude of their connection weights. This observation underscores the idea that spatial tuning within the RSNN arises from complex recurrent dynamics rather than being solely determined by the strength of individual connections.

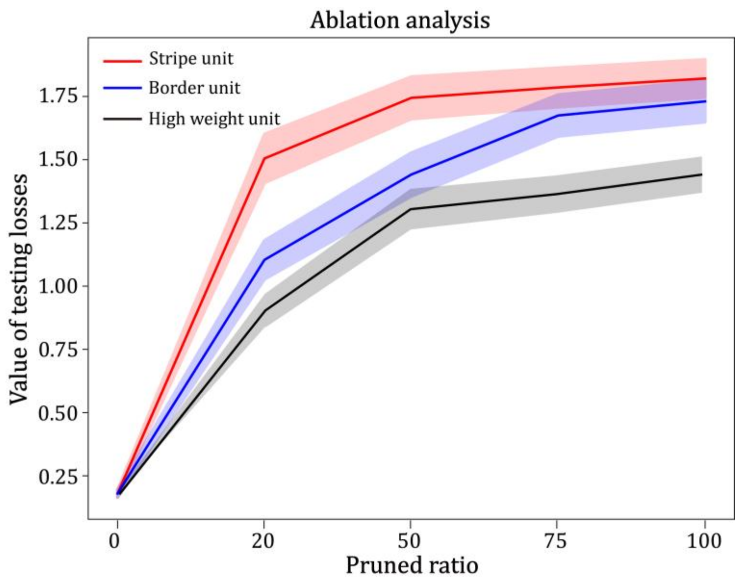


Figure 9. Ablation analysis of spatial units and high weight units.

4. Conclusions

We constructed and trained a recurrent spiking neural network (RSNN) for path integration to simulate animal navigation. The model achieves relatively accurate path integration and automatically develops locomotor and spatial firing patterns resembling those of biological neurons, indicating both functional effectiveness and biological plausibility. This aligns with an important goal of our work, which is to develop a modeling framework that is both biologically grounded and computationally efficient.

In contrast to conventional RNNs that relied on continuous-valued signal transmission, the RSNN transmits information through sparse spiking sequences, better reflecting the discrete and event-driven nature of information processing in biological neural systems [16,48,49]. To provide a direct comparison, we also trained a rate-based RNN under identical conditions, with representative results shown in Appendix Figure A3. Although the rate-based model converges more smoothly and reaches a lower training loss, it does not develop the structured spatial representations that emerge spontaneously in the RSNN, nor does it capture the discrete signaling patterns characteristic of biological units. From an engineering standpoint, this trade-off is well recognized: Spiking neural networks rarely surpass rate-based neural networks in pure accuracy under identical training regimes. Their advantages instead lie in how computations are performed — through sparse, asynchronous updates that support low-latency processing and can be deployed efficiently on neuromorphic hardware [19,50,51]. Moreover, unlike continuous attractor networks that rely on manually designed or hand-tuned connection weights, the RSNN can be trained through gradient descent to autonomously learn connection weights, while still being able to spontaneously develop diverse spatial and locomotor representations. Compared with these conventional models, the RSNN thus provides a more biologically plausible architecture for path integration.

The RSNN adopts a sparse coding scheme in which recurrent-layer units emit binary spike responses, closely mimicking the discrete signaling mechanisms of biological neurons. These units receive both recurrent spike stimuli and feedforward locomotor inputs, playing a critical role in path integration. The loss function combines prediction error with a metabolic loss that drives the membrane potential toward 0, consistent with biological efficiency principles [52].

The RSNN spontaneously develops diverse spatially selective units, quite similar to the navigation-related functional neurons found in a mammal's hippocampal formation, including border, stripe, place, and grid cells. Grid cells have traditionally been viewed as a central component for path integration [5], and some computational models equate accurate path integration with the emergence of regular grid-like patterns [11]. However, recent work in deep learning suggests that the emergence of hexagonal grid-like responses depends on constraints of specific objective functions; in the absence of such constraints, grid-like responses tend to disappear [53]. Consistent with this view, our results show a greater prevalence of stripe and border units without explicit objective function constraints, and that grid units appear less frequently. This also aligns with experiments that animals are still capable of path integration in the absence of grid cells [54,55]. In addition, the RSNN gives rise to two locomotor unit types: speed units and HD units. These demonstrate the network's ability to encode direction, speed, and spatial position.

In terms of functional connectivity among units, we conduct a systematic analysis of spatial units and locomotor units. Border units are observed to have strong connectivity with locomotor units, likely due to the frequent changes in direction and speed near boundaries. Stripe units exhibit stronger functional associations with locomotor units, suggesting that band-like responses are associated with richer locomotor information. Notably, the RSNN gives rise to stripe units exhibiting diverse firing patterns in different environments. For example, in the square environment, these patterns align with boundary orientations. More uniquely, in the triangle environment, stripe patterns can even exhibit curvature. Locomotor units predominantly encode input stimuli independently, although some also contribute to spatial localization. These findings underscore their coordinated interaction in facilitating path integration.

Across path integration tasks at short, medium, and long temporal scales, the RSNN consistently exhibits robust generalization performance without relying on specific optimization function constraints [44,53,56]. This suggests that spike-driven sparse coding enables effective generalization over extended time step horizons, aligning with empirical findings of stable long-term path integration in mammals.

Analysis of the evolution of spatial unit types during training show that border units emerge early and remain stable throughout the process, whereas stripe units typically emerge during later stages. Some units develop mixed representations combining boundary and band-like responses. These results indicate that spatial encoding exhibits plasticity during the training stages.

Despite the demonstrated advantages of the RSNN in both predictive performance and biological representation, several limitations remain. First, the training environment is relatively simple and limited to two-dimensional planar space, and future work could extend the RSNN to more complex three-dimensional settings or dynamic environments with obstacles to further evaluate the RSNN's generalization capability and the robustness of its biological representations [57].

Second, quantitative analysis of neuronal representations is limited to the square environment. In environments with complex spatial structures, the analysis is constrained to qualitative descriptions. Future work should develop a generalized metric to evaluate neuronal representations across different environment.

Finally, the RSNN demonstrates inherent robustness to noise, benefitting from the membrane reset mechanism of LIF neurons, which helps the network filter irrelevant fluctuations and extract salient information during learning. While the use of LIF neurons and sparse spiking activity allows our model to capture several key aspects of biological neuronal dynamics, it nonetheless remains a simplified abstraction of real cortical circuits [58,59]. Important physiological features — such as spike-frequency adaptation [60], dendritic nonlinearities [61], and dynamic synaptic processes including short-term plasticity [62,63] — are not explicitly incorporated. These mechanisms are known to shape firing regularity, temporal integration, and spatial tuning in biological neurons, and their inclusion could further enrich the emergent representations observed in our system. Thus, although LIF-based models provide a useful foundation for studying biological representations emergence, they may still fall short of reproducing the full complexity of biological neuronal dynamics. Extending the framework toward more biophysically detailed neural network models represents a promising direction for future work [59,61,64].

While all recurrent and feedforward connection weights in our current RSNN are fully trainable, the underlying LIF-based recurrent module is inherently compatible with reservoir-style architectures. In principle, the recurrent layer could be reconfigured into a fixed or sparsified reservoir in future work, with only the output readout remaining trainable. Such architectures have been shown to generate rich temporal dynamics while substantially reducing training complexity and energy consumption, particularly when deployed on neuromorphic hardware [65–67]. These insights suggest that our framework could be adapted for hardware-efficient implementations without altering its core computational principles.

Addressing the remaining limitations will further advance the development of navigation systems with greater biological interpretability and adaptive learning capacity. By more closely emulating the structural and functional organization of biological neural circuits, RSNN-based models may achieve improved generalization and provide a stronger theoretical and practical foundation for efficient, low-power intelligent navigation systems [68,69].

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This study is supported by the National Natural Science Foundation of China (Grant Nos. 12172132, 12272136 and 12472054) and Science and Technology Commission of Shanghai Municipality (No. 24JS2810400).

Conflict of interest

The authors declare there are no conflicts of interest.

References

1. A. S. Etienne, K. J. Jeffery, Path integration in mammals, *Hippocampus*, **14** (2004), 180–192. https://doi.org/10.1002/hipo.10173
2. B. L. McNaughton, F. P. Battaglia, O. Jensen, E. I. Moser, M. B. Moser, Path integration and the neural basis of the "cognitive map", *Nat. Rev. Neurosci.*, **7** (2006), 663–678. https://doi.org/10.1038/nrn1932
3. E. Kropff, J. E. Carmichael, M. B. Moser, E. I. Moser, Speed cells in the medial entorhinal cortex, *Nature*, **523** (2015), 419–424. https://doi.org/10.1038/nature14622
4. J. S. Taube, R. U. Muller, J. B. Ranck, Head-direction cells recorded from the postsubiculum in freely moving rats. I. Description and quantitative analysis, *J. Neurosci.*, **10** (1990), 420–435. https://doi.org/10.1523/JNEUROSCI.10-02-00420.1990
5. T. Hafting, M. Fyhn, S. Molden, M. B. Moser, E. I. Moser, Microstructure of a spatial map in the entorhinal cortex, *Nature*, **436** (2005), 801–806. https://doi.org/10.1038/nature03721
6. Y. Wang, X. Xu, R. Wang, Modeling the grid cell activity on non-horizontal surfaces based on oscillatory interference modulated by gravity, *Neural Networks*, **141** (2021), 199–210. https://doi.org/10.1016/j.neunet.2021.04.009
7. M. C. Fuhs, D. S. Touretzky, A spin glass model of path integration in rat medial entorhinal cortex, *J. Neurosci.*, **26** (2006), 4266–4276. https://doi.org/10.1523/JNEUROSCI.4353-05.2006
8. Q. Shao, Y. Wang, X. Xu, Y. Wang, X. Pan, Y. Du, et al., The neural computational and dynamical mechanisms of reward-modulated spatial coding in hippocampal place cells, *Cognit. Neurodyn.*, **19** (2025), 99. https://doi.org/10.1007/s11571-025-10282-6
9. T. Solstad, C. N. Boccara, E. Kropff, M. B. Moser, E. I. Moser, Representation of geometric borders in the entorhinal cortex, *Science*, **322** (2008), 1865–1868. https://doi.org/10.1126/science.116646

10. E. I. Moser, E. Kropff, M. B. Moser, Place cells, grid cells, and the brain's spatial representation system, *Annu. Rev. Neurosci.*, **31** (2008), 69–89. https://doi.org/10.1146/annurev.neuro.31.061307.090723
11. Y. Burak, I. R. Fiete, Accurate path integration in continuous attractor network models of grid cells, *PLoS Comput. Biol.*, **5** (2009), e1000291. https://doi.org/10.1371/journal.pcbi.1000291
12. E. Kropff, A. Treves, The emergence of grid cells: Intelligent design or just adaptation? *Hippocampus*, **18** (2008), 1256–1269. https://doi.org/10.1002/hipo.20520
13. A. Guanella, D. Kiper, P. Verschure, A model of grid cells based on a twisted torus topology, *Int. J. Neural Syst.*, **17** (2007), 231–240. https://doi.org/10.1142/S0129065707001093
14. C. J. Cueva, X. X. Wei, Emergence of grid-like representations by training recurrent neural networks to perform spatial localization, preprint, arXiv:1803.07770.
15. A. Banino, C. Barry, B. Uria, C. Blundell, T. Lillicrap, P. Mirowski, et al., Vector-based navigation using grid-like representations in artificial agents, *Nature*, **557** (2018), 429–433. https://doi.org/10.1038/s41586-018-0102-6
16. W. Maass, Networks of spiking neurons: the third generation of neural network models, *Neural Networks*, **10** (1997), 1659–1671. https://doi.org/10.1016/S0893-6080(97)00011-7
17. C. Siettos, J. Starke, Multiscale modeling of brain dynamics: from single neurons and networks to mathematical tools, *Wiley Interdiscip. Rev. Syst. Biol. Med.*, **8** (2016), 438–458. https://doi.org/10.1002/wsbm.1348
18. C. Hong, X. Wei, J. Wang, B. Deng, H. Yu, Y. Che, Training spiking neural networks for cognitive tasks: A versatile framework compatible with various temporal codes, *IEEE Trans. Neural Networks Learn. Syst.*, **31** (2019), 1285–1296. https://doi.org/10.1109/TNNLS.2019.2919662
19. K. Roy, A. Jaiswal, P. Panda, Towards spike-based machine intelligence with neuromorphic computing, *Nature*, **575** (2019), 607–617. https://doi.org/10.1038/s41586-019-1677-2
20. M. Bouvier, A. Valentian, T. Mesquida, F. Rummens, M. Reyboz, E. Vianello, et al., Spiking neural networks hardware implementations and challenges: A survey, *ACM J. Emerging Technol. Comput. Syst.*, **15** (2019), 1–35. https://doi.org/10.1145/3304103
21. G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, et al., Neuromorphic silicon neuron circuits, *Front. Neurosci.*, **5** (2011), 73. https://doi.org/10.3389/fnins.2011.00073
22. A. V. Kurbako, D. M. Ezhov, V. I. Ponomarenko, M. D. Prokhorov, Spike-timing dependent plasticity learning of small spiking neural network for image recognition, *Eur. Phys. J. Spec. Top.*, **234** (2025), 3659–3668. https://doi.org/10.1140/epjs/s11734-025-01512-3
23. Q. Yu, R. Yan, H. Tang, K. C. Tan, H. Li, A spiking neural network system for robust sequence recognition, *IEEE Trans. Neural Networks Learn. Syst.*, **27** (2015), 621–635. https://doi.org/10.1109/TNNLS.2015.2416771
24. D. Zambrano, R. Nusselder, H. S. Scholte, S. M. Bohté, Sparse computation in adaptive spiking neural networks, *Front. Neurosci.*, **12** (2018), 987. https://doi.org/10.3389/fnins.2018.00987
25. J. Ding, B. Dong, F. Heide, Y. Ding, Y. Zhou, B. Yin, et al., Biologically inspired dynamic thresholds for spiking neural networks, in *Advances in Neural Information Processing Systems*, **35** (2022), 6090–6103.
26. S. Liu, P. L. Dragotti, Enhanced accuracy in first-spike coding using current-based adaptive LIF neuron, *Neural Networks*, **184** (2025), 107043. https://doi.org/10.1016/j.neunet.2024.107043

27. D. Xue, W. Li, C. Yuan, H. Liu, M. Yao, W. Liu, et al., MSFI: Multi-timescale Spatio-temporal Features Integration in spiking neural networks, *Neural Networks*, **193** (2026), 108024. https://doi.org/10.1016/j.neunet.2025.108024
28. G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, W. Maass, Long short-term memory and learning-to-learn in networks of spiking neurons, in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, (2018), 795–805.
29. E. O. Neftci, H. Mostafa, F. Zenke, Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks, *IEEE Signal Process Mag.*, **36** (2019), 51–63. https://doi.org/10.1109/MSP.2019.2931595
30. W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, Y. Tian, Deep residual learning in spiking neural networks, in *Advances in Neural Information Processing Systems*, **34** (2021), 21056–21069.
31. J. K. Eshraghian, M. Ward, E. O. Neftci, X. Wang, G. Lenz, G. Dwivedi, et al., Training spiking neural networks using lessons from deep learning, *Proc. IEEE*, **111** (2023), 1016–1054. https://doi.org/10.1109/JPROC.2023.3308088
32. Y. Yan, H. Chu, Y. Jin, Y. Huan, Z. Zou, L. Zheng, Backpropagation with sparsity regularization for spiking neural network learning, *Front. Neurosci.*, **16** (2022), 760298. https://doi.org/10.3389/fnins.2022.760298
33. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, preprint, arXiv:1412.6980.
34. W. Ye, S. Chen, H. Liu, Y. Liu, Y. Chen, Y. Cui, et al., The architecture design and training optimization of spiking neural network with low-latency and high-performance for classification and segmentation, *Neural Networks*, **191** (2025), 107790. https://doi.org/10.1016/j.neunet.2025.107790
35. M. Yao, H. Zhang, G. Zhao, X. Zhang, D. Wang, G. Cao, et al., Sparser spiking activity can be better: Feature refine-and-mask spiking neural network for event-based visual recognition, *Neural Networks*, **166** (2023), 410–423. https://doi.org/10.1016/j.neunet.2023.07.008
36. K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, (2015), 1026–1034. https://doi.org/10.1109/ICCV.2015.123
37. P. E. Sharp, A. Tinkelman, J. Cho, Angular velocity and head direction signals recorded from the dorsal tegmental nucleus of gudden in the rat: implications for path integration in the head direction cell circuit, *Behav. Neurosci.*, **115** (2001), 571–588. https://doi.org/10.1037/0735-7044.115.3.571
38. D. L. Ringach, R. M. Shapley, M. J. Hawken, Orientation selectivity in macaque V1: diversity and laminar dependence, *J. Neurosci.*, **22** (2002), 5639–5651. https://doi.org/10.1523/JNEUROSCI.22-13-05639.2002
39. J. S. Taube, Head direction cells and the neurophysiological basis for a sense of direction, *Prog. Neurobiol.*, **55** (1998), 225–256. https://doi.org/10.1016/S0301-0082(98)00004-5
40. D. Yoganarasimha, X. Yu, J. J. Knierim, Head direction cell representations maintain internal coherence during conflicting proximal and distal cue rotations: comparison with hippocampal place cells, *J. Neurosci.*, **26** (2006), 622–631. https://doi.org/10.1523/JNEUROSCI.3885-05.2006
41. K. Pearson, VII. Note on regression and inheritance in the case of two parents, *Proc. R. Soc.*, **58** (1895), 240–242. https://doi.org/10.1098/rspl.1895.0041

42. C. Barry, C. Lever, R. Hayman, T. Hartley, S. Burton, J. O'Keefe, et al., The boundary vector cell model of place cell firing and spatial memory, *Rev. Neurosci.*, **17** (2006), 71–98. https://doi.org/10.1515/REVNEURO.2006.17.1-2.71
43. F. Savelli, D. Yoganarasimha, J. J. Knierim, Influence of boundary removal on the spatial representations of the medial entorhinal cortex, *Hippocampus*, **18** (2008), 1270–1282. https://doi.org/10.1002/hipo.20511
44. V. Schøyen, M. B. Pettersen, K. Holzhausen, M. Fyhn, A. Malthe-Sørenssen, M. E. Lepperød, Coherently remapping toroidal cells but not grid cells are responsible for path integration in virtual agents, *iScience*, **26** (2023), 108102. https://doi.org/10.1016/j.isci.2023.108102
45. C. Lever, S. Burton, A. Jeewajee, J. O'Keefe, N. Burgess, Boundary vector cells in the subiculum of the hippocampal formation, *J. Neurosci.*, **29** (2009), 9771–9777. https://doi.org/10.1523/JNEUROSCI.1319-09.2009
46. L. Muessig, F. R. Rodrigues, T. L. Bjerknes, B. W. Towse, C. Barry, N. Burgess, et al., Environment geometry alters subiculum boundary vector cell receptive fields in adulthood and early development, *Nat. Commun.*, **15** (2024), 982. https://doi.org/10.1038/s41467-024-45098-1
47. F. Sargolini, M. Fyhn, T. Hafting, B. L. McNaughton, M. P. Witter, M. B. Moser, et al., Conjunctive representation of position, direction, and velocity in entorhinal cortex, *Science*, **312** (2006), 758–762. https://doi.org/10.1126/science.1125572
48. A. M. Andrew, Spiking neuron models: Single neurons, populations, plasticity, *Kybernetes*, **32** (2003), 1278–1280. https://doi.org/10.1108/k.2003.06732gae.003
49. R. Wang, Y. Wang, X. Xu, Y. Li, X. Pan, Brain works principle followed by neural information processing: a review of novel brain theory, *Artif. Intell. Rev.*, **56** (2023), 285–350. https://doi.org/10.1007/s10462-023-10520-5
50. M. Koo, G. Srinivasan, Y. Shim, K. Roy, sBSNN: Stochastic-bits enabled binary spiking neural network with on-chip learning for energy efficient neuromorphic computing at the edge, *IEEE Trans. Circuits Syst. I Regul. Pap.*, **67** (2020), 2546–2555. https://doi.org/10.1109/TCSI.2020.2979826
51. B. Yin, F. Corradi, S. M. Bohté, Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks, *Nat. Mach. Intell.*, **3** (2021), 905–913. https://doi.org/10.1038/s42256-021-00397-w
52. S. B. Laughlin, Energy as a constraint on the coding and processing of sensory information, *Curr. Opin. Neurobiol.*, **11** (2001), 475–480. https://doi.org/10.1016/S0959-4388(00)00237-3
53. R. Schaeffer, M. Khona, I. Fiete, No free lunch from deep learning in neuroscience: A case study through models of the entorhinal-hippocampal circuit, in *Advances in Neural Information Processing Systems*, **35** (2022), 16052–16067.
54. Y. Shrager, C. B. Kirwan, L. R. Squire, Neural basis of the cognitive map: path integration does not require hippocampus or entorhinal cortex, *Proc. Natl. Acad. Sci. U.S.A.*, **105** (2008), 12034–12038. https://doi.org/10.1073/pnas.0805414105
55. M. Pettersen, V. S. Schøyen, M. D. Østby, A. Malthe-Sørenssen, M. E. Lepperød, Self-supervised grid cells without path integration, *bioRxiv*, 2024. https://doi.org/10.1101/2024.05.30.596577
56. R. Schaeffer, M. Khona, T. Ma, C. Eyzaguirre, S. Koyejo, I. Fiete, Self-supervised learning of representations for space generates multi-modular grid cells, in *Advances in Neural Information Processing Systems*, **36** (2023), 23140–23157.

57. M. Osswald, S. H. Ieng, R. Benosman, G. Indiveri, A spiking neural network model of 3D perception for event-based neuromorphic stereo vision systems, *Sci. Rep.*, **7** (2017), 40703. https://doi.org/10.1038/srep40703
58. W. Gerstner, W. M. Kistler, *Spiking Neuron Models: Single Neurons*, *Populations, Plasticity*, Cambridge University Press, 2002. https://doi.org/10.1017/CBO9780511815706
59. E. M. Izhikevich, Simple model of spiking neurons, *IEEE Trans. Neural Networks*, **14** (2003), 1569–1572. https://doi.org/10.1109/TNN.2003.820440
60. J. Benda, A. V. M. Herz, A universal model for spike-frequency adaptation, *Neural Comput.*, **15** (2003), 2523–2564. https://doi.org/10.1162/089976603322385063
61. M. London, M. Häusser, Dendritic computation, *Annu. Rev. Neurosci.*, **28** (2005), 503–532. https://doi.org/10.1146/annurev.neuro.28.061604.135703
62. M. Tsodyks, S. Wu, Short-term synaptic plasticity, *Scholarpedia*, **8** (2013), 3153. https://doi.org/10.4249/scholarpedia.3153
63. H. Markram, Y. Wang, M. Tsodyks, Differential signaling via the same axon of neocortical pyramidal neurons, *Proc. Natl. Acad. Sci. U.S.A.*, **95** (1998), 5323–5328. https://doi.org/10.1073/pnas.95.9.5323
64. Q. Zhu, F. Han, Q. Wang, A spinal circuit model with an asymmetric cervical-lumbar layout for limb coordination and gait control in quadrupeds, *Appl. Math. Mech.*, **46** (2025), 1433–1450. https://doi.org/10.1007/s10483-025-3282-9
65. Z. Liao, H. Yamahara, K. Terao, K. Ma, M. Seki, H. Tabata, Short-term memory capacity analysis of $Lu_3Fe_4Co_{0.5}Si_{0.5}O_{12}$-based spin cluster glass towards reservoir computing, *Sci. Rep.*, **13** (2023), 5260. https://doi.org/10.1038/s41598-023-32084-8
66. Z. Shi, Z. Liao, H. Tabata, Boosting learning ability of overdamped bistable stochastic resonance system based physical reservoir computing model by time-delayed feedback, *Chaos, Solitons Fractals*, **161** (2022), 112314. https://doi.org/10.1016/j.chaos.2022.112314
67. W. Maass, T. Natschläger, H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations, *Neural Comput.*, **14** (2002), 2531–2560. https://doi.org/10.1162/089976602760407955
68. R. Li, Y. Wang, X. Xu, F. Li, F. Tang, X. Pan, A neuro-inspired visual SLAM approach using AKAZE feature extraction in complex and dynamic environments, *Cognit. Neurodyn.*, **20** (2026), 15. https://doi.org/10.1007/s11571-025-10386-z
69. K. Yamazaki, V. Vo-Ho, D. Bulsara, N. Le, Spiking neural networks and their applications: A review, *Brain Sci.*, **12** (2022), 863. https://doi.org/10.3390/brainsci12070863

Appendix

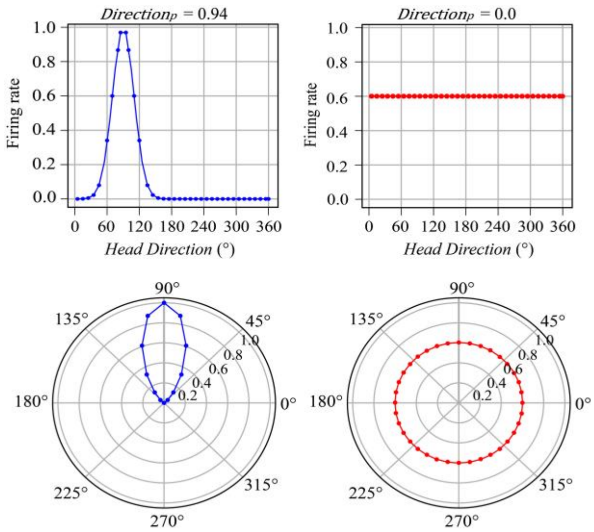


Figure A1. Illustration of $Direction_p$ for an HD unit (left) and a non-HD unit (right). The top panels show firing rate tuning curves. The horizontal axis indicates head direction (0°–360°), and the vertical axis shows firing rate. Sharp peaks indicate strong directional tuning, whereas flat curves indicate weak or absent tuning. The bottom panels provide an intuitive view of directional tuning, where the angle shows head direction and the radial distance reflects firing rate, making it easier to understand how the unit's activity is concentrated in its preferred direction.

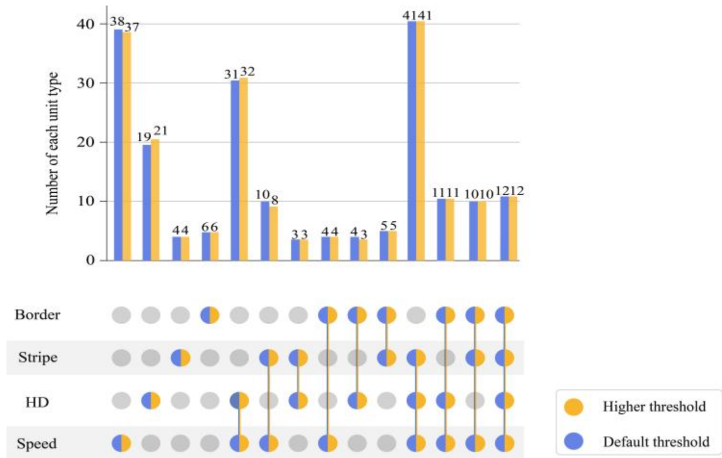


Figure A2. Number distribution of unit types under different thresholds. The bar plot shows the number of units of each functional type under different threshold conditions. Blue bars represent results using the default threshold, while yellow bars correspond to a slightly higher threshold (+ 0.1). The results indicate that small threshold adjustments have minimal impact on the number of units in each category, suggesting robustness of the classification. Circles below indicate units of each type, corresponding to the labels on the left ("Border", "Stripe", "HD", "Speed"). Connected circles represent units that belong to multiple functional types.

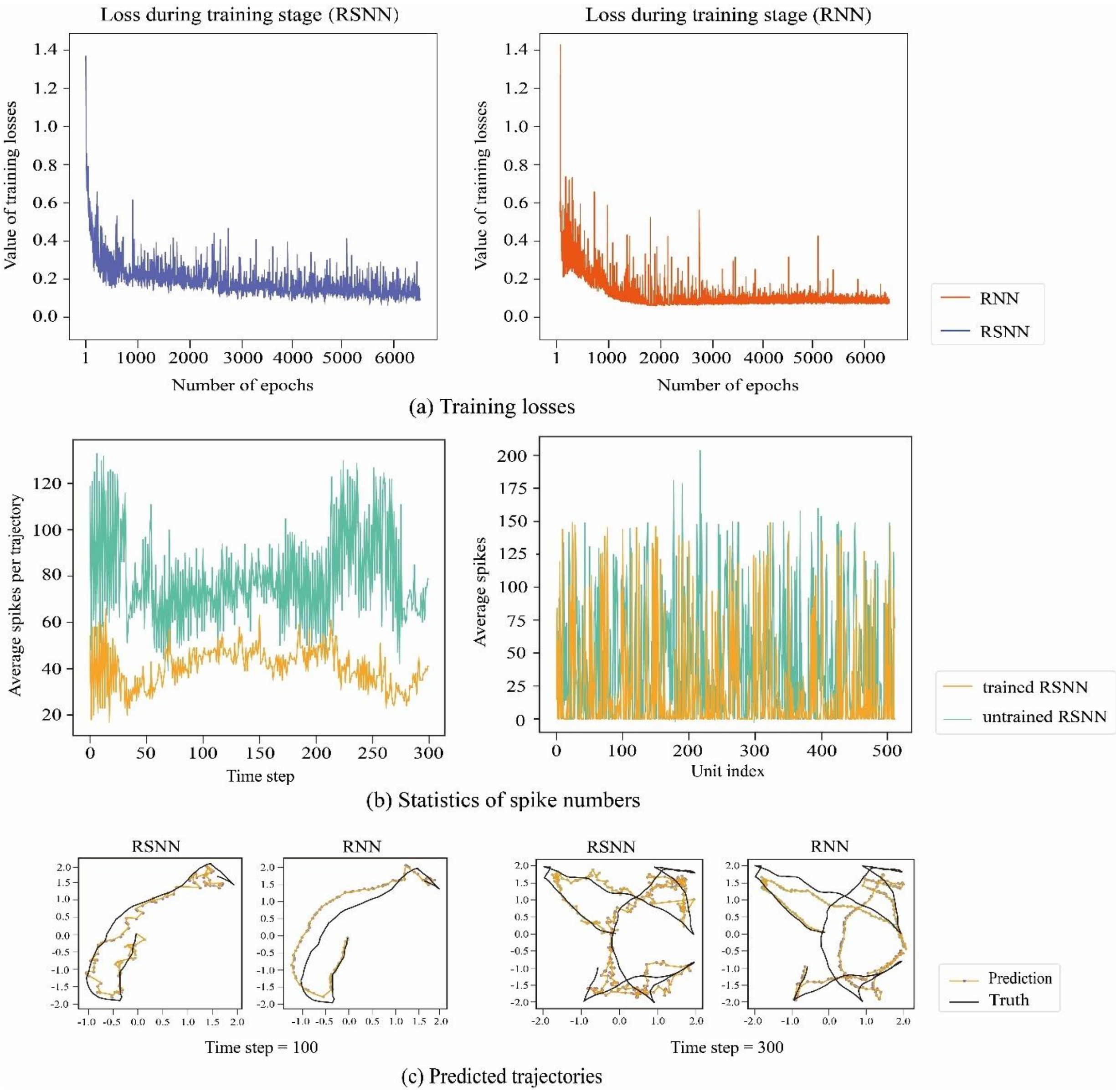


Figure A3. Comparison between RSNN and rate-based RNN during training and predicted trajectories. (a) Training losses: The RNN converges smoothly to a lower final loss ($\approx$ 0.10), while the RSNN shows larger fluctuations due to LIF neuron dynamics, settling at $\approx$ 0.17. (b) RSNN spike statistics: Left, average spikes per trajectory; right, average spikes per unit. The RSNN achieves path integration with sparse, event-driven spiking. (c) Predicted trajectories: On some trajectories, RSNN predictions may outperform those of the rate-based RNN, capturing finer structure and better aligning with the ground truths.