Appendix of *Temporal Conditioning Spiking Latent Variable Models of the Neural Response to Natural Visual Scenes*

A Hidden State and Latent Space Experiments

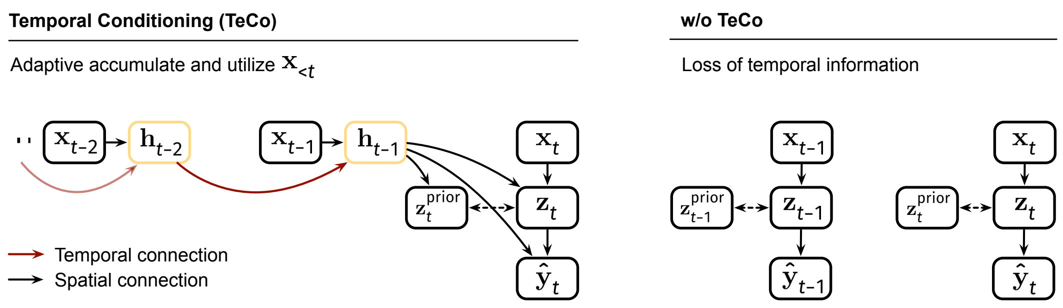
Temporal Conditioning (TeCo)
Adaptive accumulate and utilize $\mathbf{x}_{<t}$
w/o TeCo
Loss of temporal information
Temporal connection
Spatial connection

Figure 5: Graphical illustration of the temporal conditioning operation (TeCo). After completely excluding the temporal dimension from the model parameter space, we introduced the temporal conditioning operation to handle the temporal information. In particular, this operation enables *memory-dependent processing* as in biological coding circuits.

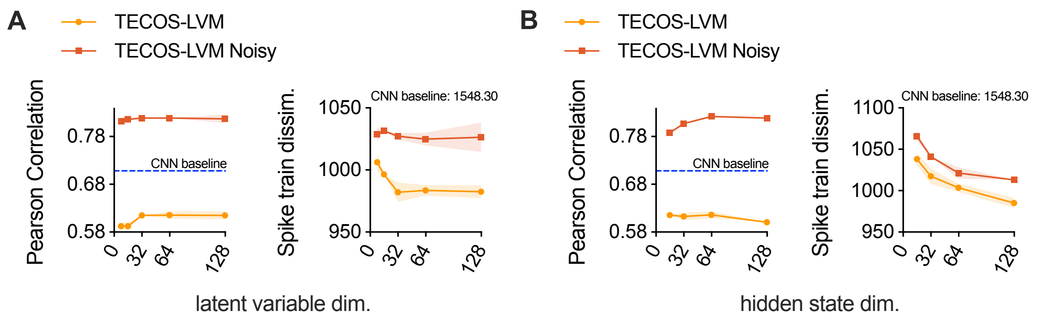

Figure 6: Performances under different hidden state and latent space dimension settings on Movie 2 Retina 2 data. For hidden state experiments, the latent space dimension is set to 32. And for latent space experiments, the hidden state dimension is 64. The error bars (SD) were calculated via various random seeds.

To further study the sensitivity to the choice of hyperparameters, we evaluated the performance of TeCoS-LVM models under different hidden state and latent space dimensionality settings. The results, as shown in Appendix Fig. 6A, indicate that increasing the latent space dimension improved performance. Still, further increases (larger than 32) had little effect when the latent space dimension was large. This also suggests that the number of latent factors for the visual stimuli coding task we considered is not very large. On the other hand, increasing the hidden state dimension enhances sensory memory capacity and benefits temporal conditioning operations. As shown in Appendix Fig. 6B, the performance does not increase significantly when hidden state dimensionality increases to a certain extent (around 64). In particular, although the spike train dissimilarity decreases, the firing rate correlation score almost no longer increases. This is consistent with our observation in the main text, namely, a lower spike train dissimilarity does not always indicate a higher firing rate correlation score (refer to Fig. 2,3). The results in Appendix Fig. 6B also indicate that the proposed temporal conditioning mechanism can effectively utilize sensory memory and achieve good results even when the hidden state dimension is limited.

B Surrogate Gradient Learning in Spiking Networks

In conventional SNN Surrogate Gradient Learning (SGL, pseudo derivative, derivative approximation) [29], the derivative of the firing function $\partial o/\partial u$ is replaced by a smooth function (pseudo derivative function) SG to mesh with the backpropagation scheme [78]. The use of this ad-hoc technique in SNN research is popular, particularly for large-scale networks. It allows for compatibility with popular automatic differentiation packages such as PyTorch and TensorFlow, simplifying the implementation of SNNs. This surrogate gradient function can be a triangular, rectangular (gate), sigmoidal, or ERF function [78]. In SGL, the gradient g_l w.r.t. synaptic weights of layer l is calculated by

$$\text{SGL: } \hat{g}_l = \sum_m \nabla_{\theta_l} u_t^{l,m} \underbrace{\text{SG}(u_t^{l,m} - v_{\text{th}})}_{\text{Surrogate the exact derivative } \partial o_t^{l,m} / \partial u_t^{l,m}} \nabla_{o_t^{l,m}} \mathcal{L}_t, \tag{10}$$

where l, m denotes neuron m in layer l, and $\mathcal{L}_t$ is the instant loss value.

C Leveraging Noisy Spiking Neural Models

Here we use the implementation in [53] to leverage the power of noisy spiking neural models. Spiking neurons with noisy neuronal dynamics have been extensively studied in prior literature [25]. Recent research of Ma et al. [53] extended them to larger networks by providing a general formularization and demonstrating their computational advantages theoretically and empirically. The Noisy LIF presented here is based on previous works that use diffusive approximation [79, 80, 25], where the sub-threshold dynamic is described by the Ornstein-Uhlenbeck process:

$$\tau_m \frac{\mathrm{d}u}{\mathrm{d}t} = -(u - u_{\text{reset}}) + RI(t) + \xi(t), \text{ eq. } \mathrm{d}u = -(u - u_{\text{reset}}) \frac{\mathrm{d}t}{\tau_m} + RI(t) \frac{\mathrm{d}t}{\tau_m} + \sigma \mathrm{d}W_t, \tag{11}$$

the white noise ξ is a stochastic process, σ is the amplitude of the noise and $\mathrm{d}W_t$ are the increments of the Wiener process in $\mathrm{d}t$ [25]. As $\sigma \mathrm{d}W_t$ are random variables drawn from a zero-mean Gaussian, this formulation is directly applicable to discrete-time simulations. Specifically, using the Euler-Maruyama method, we get a Gaussian noise term added on the right-hand side of the noise-free LIF dynamic. Without loss of generality, we extend the additive noise term in the discrete form to general continuous noise [81], the sub-threshold dynamic of Noisy LIF can be represented as:

$$\text{Noisy LIF sub-threshold dynamic: } u_t = \tau u_{t-1} + I_t + \epsilon, \tag{12}$$

where I_t is the input, the noise ϵ is independently drawn from a known distribution and satisfies $\mathbb{E}[\epsilon] = 0$ and $p(\epsilon) = p(-\epsilon)$. The constant τ here combines the simulation timestep length and the real membrane decay τ_m, which is a simplification when the timestep we cope with is fixed. This work considers the Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.2^2)$.

The membrane potentials and spike outputs become random variables due to random noise injection. Leveraging noise as a medium, we naturally obtain the firing probability distribution of Noisy LIF based on the threshold firing mechanism:

$$\mathbb{P}[\text{firing at time } t] = \mathbb{P} \underbrace{[u_t + \epsilon > v_{\text{th}}]}_{\text{Threshold-based firing}} = \underbrace{\mathbb{P}[\epsilon < u_t - v_{\text{th}}] \triangleq F_\epsilon(u_t - v_{\text{th}})}_{\text{Cumulative Distribution Function definition}},$$

where F denotes the cumulative distribution function. Therefore, we have that,

$$o_t = \begin{cases} 1, \text{with probability} & F_\epsilon(u_t - v_{\text{th}}), \\ 0, \text{with probability} & (1 - F_\epsilon(u_t - v_{\text{th}})). \end{cases} \tag{13}$$

The expressions above exemplify how noise acts as a resource for computation [82]. Thereby, we can formulate the firing process of Noisy LIF as

$$\text{Noisy LIF probabilistic firing: } o_t \sim \text{Bernoulli}\left(F_\epsilon(u_t - v_{\text{th}})\right), \tag{14}$$

Specifically, it relates to previous literature on noise escape models, in which the difference $u - v_{\text{th}}$ governs the neuron firing probabilities [83, 79, 25]. In addition, Noisy LIF employs the same resetting mechanism as the LIF model.

C.1 Noise-Driven Learning in Networks of Noisy LIF Neurons

The Noise-Driven Learning (NDL) rule [53] in networks of Noisy LIF neurons is a theoretically sound general form of Surrogate Gradient Learning. In particular, the gradient w.r.t to synaptic weights in layer l is computed by

$$\text{NDL: } \hat{g}_l = \sum_m \underbrace{\nabla_{\theta_l} u_t^{l,m}}_{\text{Pre-synaptic factor}} \overbrace{F'_\epsilon(u_t^{l,m} - v_{\text{th}})}^{\text{Post-synaptic factor}} \underbrace{\nabla_{o_t^{l,m}} \mathcal{L}_t}_{\text{Global learning signal}}, \tag{15}$$

where the superscript l, m denotes neuron m in layer l, $\mathcal{L}$ is the loss value. Here the post-synaptic factor in NDL is calculated by the probability distribution function of the postsynaptic neuron's membrane potential noise.

As shown in Equation 15, NDL is well-compatible with the backpropagation computation paradigm in trending libraries like PyTorch. Therefore we can wrap the inference and learning of Noisy LIF neurons into a module. By replacing the original LIF neuron module (with Surrogate Gradient Learning) with the Noisy LIF module (with NDL), we can easily implement noisy spiking neural networks of arbitrary architectures in a *plug-and-play* manner. An example of that can be found at `https://github.com/genema/Noisy-Spiking-Neuron-Nets`.

D Derivation of the Optimization Objective of TeCoS-LVM Models

Basic formulation – from an efficient coding [38] perspective We denote a sequence of visual stimuli as $\mathbf{x} = (\mathbf{x}_t)_{t=1\cdots T}$, where $\mathbf{x}_t \in \mathbb{R}^{\mathtt{dim}[\mathbf{x}_t]}$, $\mathtt{dim}[\mathbf{x}_t]$ stands for the dimension of $\mathbf{x}_t$. Similarly, we denote neural population response (target) as $\mathbf{y} = (\mathbf{y}_t) \in \{0, 1\}^{T\times \mathtt{dim}[\mathbf{y}_t]}$, where $\mathtt{dim}[\mathbf{y}_t]$ denotes the number of retinal ganglion cells (RGCs). At each timestep t, one high-dimensional visual stimulus $\mathbf{x}_t$ is received, and we want to predict the neural population response $\mathbf{y}_t$. This is implemented by an LVM which first compresses the visual stimuli into a low-dimensional latent representation $\mathbf{z}_t \in \mathbb{R}^{\mathtt{dim}[\mathbf{z}_t]}$, and then decodes the neural population response from it. Inspired by the information compression feature in the neural coding process [34, 35, 36, 37], we further encourage this LVM to construct a latent space in which $\mathbf{z}$ have maximal predictive power regarding $\mathbf{y}$ while being maximally compressive about $\mathbf{x}$. Therefore, our target to model the neural coding of visual stimuli turns into an optimization problem within the IB framework. *Note that stimulus from other modalities can be processed in a similar vein, but here we use the visual case as an example.*

Following previous IB literature [31, 32, 33], we assume a factorization of the joint distribution as follows,

$$p(\mathbf{x}, \mathbf{y}, \mathbf{z}) = p(\mathbf{z}|\mathbf{x}, \mathbf{y})p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) = p(\mathbf{z}|\mathbf{x})p(\mathbf{y}|\mathbf{x})p(\mathbf{x}), \tag{16}$$

namely, we assume a Markov chain $\mathbf{y} \leftrightarrow \mathbf{x} \leftrightarrow \mathbf{z}$, which implies $p(\mathbf{z}|\mathbf{x}, \mathbf{y}) = p(\mathbf{z}|\mathbf{x})$, indicating that the latent representation $\mathbf{z}$ cannot directly depend on the target response $\mathbf{y}$. Recall that, according to the IB principle [31], our objective has the form

$$\text{IB objective: } \max[\underbrace{I(\mathbf{z}, \mathbf{y})}_{\text{Predictive term}} \underbrace{-\beta I(\mathbf{z}, \mathbf{x})}_{\text{Compressive term}}]. \tag{17}$$

The predictive term encourages predictive power, while the compressive term enforces information compression. And it is equivalent to minimizing a loss function $-I(\mathbf{z}, \mathbf{y}) + \beta I(\mathbf{z}, \mathbf{x})$.

Let us examine the predictive term $I(\mathbf{z}, \mathbf{y})$ first. The mutual information between $\mathbf{z}$ and $\mathbf{y}$ is given by

$$\begin{aligned} I(\mathbf{z}, \mathbf{y}) &= \int d\mathbf{y} d\mathbf{z} p(\mathbf{y}, \mathbf{z}) \log \frac{p(\mathbf{y}, \mathbf{z})}{p(\mathbf{y})p(\mathbf{z})} \\ &= \int d\mathbf{y} d\mathbf{z} p(\mathbf{y}, \mathbf{z}) \log \frac{p(\mathbf{y}|\mathbf{z})}{p(\mathbf{y})}. \end{aligned} \tag{18}$$

According to the assumed Markov chain (16), the likelihood $p(\mathbf{y}|\mathbf{z})$ is defined by

$$p(\mathbf{y}|\mathbf{z}) = \int d\mathbf{x} p(\mathbf{x}, \mathbf{y}|\mathbf{z}) = \int d\mathbf{x} p(\mathbf{y}|\mathbf{x}) p(\mathbf{x}|\mathbf{z}) = \int d\mathbf{x} p(\mathbf{y}|\mathbf{x}) \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z})}, \tag{19}$$

and is approximated by a variational decoder $p(\mathbf{y}|\mathbf{z}; \psi^{\text{dec}})$ in our case. Given the fact that $\text{KL}[p(\mathbf{y}|\mathbf{z})||p(\mathbf{y}|\mathbf{z}; \psi^{\text{dec}})] \geq 0$, we can write

$$\begin{aligned} &\int d\mathbf{y} p(\mathbf{y}|\mathbf{z}) \log \frac{p(\mathbf{y}|\mathbf{z})}{p(\mathbf{y}|\mathbf{z}; \psi^{\text{dec}})} \geq 0 \\ \Rightarrow &\int d\mathbf{y} p(\mathbf{y}|\mathbf{z}) \log p(\mathbf{y}|\mathbf{z}) \geq \int d\mathbf{y} p(\mathbf{y}|\mathbf{z}) \log p(\mathbf{y}|\mathbf{z}; \psi^{\text{dec}}). \end{aligned} \tag{20}$$

Therefore,

$$\begin{aligned} I(\mathbf{z},\mathbf{y}) &\geq \int \mathrm{d}\mathbf{y}\mathrm{d}\mathbf{z}p(\mathbf{y},\mathbf{z}) \log \frac{p(\mathbf{y}|\mathbf{z};\psi^{\mathrm{dec}})}{p(\mathbf{y})} \\ &= \int \mathrm{d}\mathbf{y}\mathrm{d}\mathbf{z}p(\mathbf{y},\mathbf{z}) \log p(\mathbf{y}|\mathbf{z};\psi^{\mathrm{dec}}) + H(\mathbf{y}). \end{aligned} \tag{21}$$

Since the target information entropy $H(\mathbf{y})$ is independent of the optimization procedure of the parametric model, it can be ignored. Thus, $\max I(\mathbf{z},\mathbf{y}) = \max \int \mathrm{d}\mathbf{y}\mathrm{d}\mathbf{z}p(\mathbf{y},\mathbf{z}) \log p(\mathbf{y}|\mathbf{z};\psi^{\mathrm{dec}})$. By Eq. 16, $p(\mathbf{y},\mathbf{z}) = \int \mathrm{d}\mathbf{x}p(\mathbf{x})p(\mathbf{y}|\mathbf{x})p(\mathbf{z}|\mathbf{x})$, therefore,

$$\max I(\mathbf{z},\mathbf{y}) = \max \int \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y}\mathrm{d}\mathbf{z}p(\mathbf{x})p(\mathbf{y}|\mathbf{x})p(\mathbf{z}|\mathbf{x}) \log p(\mathbf{y}|\mathbf{z};\psi^{\mathrm{dec}}). \tag{22}$$

We now consider the compressive term $\beta I(\mathbf{z},\mathbf{x})$ in the IB objective (17), and we temporally discard the constant factor β. The mutual information between input stimuli and latent representation is given by

$$\begin{aligned} I(\mathbf{z},\mathbf{x}) &= \int \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{z}p(\mathbf{x},\mathbf{z}) \log \frac{p(\mathbf{z}|\mathbf{x})}{p(\mathbf{z})} \\ &= \int \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{z}p(\mathbf{x},\mathbf{z}) \log p(\mathbf{z}|\mathbf{x}) - \int \mathrm{d}\mathbf{z}p(\mathbf{z}) \log p(\mathbf{z}). \end{aligned} \tag{23}$$

Let $p(\mathbf{z};\phi^{\mathrm{prior}})$ be a variational approximation to the marginal $p(\mathbf{z})$, because $\mathrm{KL}[p(\mathbf{z})||p(\mathbf{z};\phi^{\mathrm{prior}})] \geq 0$, we have that

$$\int \mathrm{d}\mathbf{z}p(\mathbf{z}) \log p(\mathbf{z}) \geq \int \mathrm{d}\mathbf{z}p(\mathbf{z}) \log p(\mathbf{z};\phi^{\mathrm{prior}}). \tag{24}$$

With Eq. 23 in tow and using a parametric encoder $q(\mathbf{z}|\mathbf{x};\psi^{\mathrm{enc}})$, we have the following upper bound:

$$I(\mathbf{z},\mathbf{x}) \leq \int \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{z}p(\mathbf{x})q(\mathbf{z}|\mathbf{x};\psi^{\mathrm{enc}}) \log \frac{q(\mathbf{z}|\mathbf{x};\psi^{\mathrm{enc}})}{p(\mathbf{z};\phi^{\mathrm{prior}})}. \tag{25}$$

By Eqs. 22, 25, we have a lower bound for the IB objective as follows,

$$\begin{aligned} I(\mathbf{z},\mathbf{y}) - \beta I(\mathbf{z},\mathbf{x}) \geq &\int \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y}\mathrm{d}\mathbf{z}p(\mathbf{x})p(\mathbf{y}|\mathbf{x})q(\mathbf{z}|\mathbf{x};\psi^{\mathrm{enc}}) \log p(\mathbf{y}|\mathbf{z};\psi^{\mathrm{dec}}) \\ &- \beta \int \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{z}p(\mathbf{x})q(\mathbf{z}|\mathbf{x};\psi^{\mathrm{enc}}) \log \frac{q(\mathbf{z}|\mathbf{x};\psi^{\mathrm{enc}})}{p(\mathbf{z};\phi^{\mathrm{prior}})}. \end{aligned} \tag{26}$$

Using $\boldsymbol{\theta}$ to denote all the parameters ($\phi^{\mathrm{prior}}, \psi^{\mathrm{enc}}, \psi^{\mathrm{dec}}$ and other learnable parameters, like those of the feature extractor) of the model following the main text, we have that

$$\max_{\boldsymbol{\theta}} [I(\mathbf{z},\mathbf{y};\boldsymbol{\theta}) - \beta I(\mathbf{z},\mathbf{x};\boldsymbol{\theta})] = \min_{\boldsymbol{\theta}} \mathcal{L}, \tag{27}$$

where

$$\begin{aligned} \mathcal{L} = &\underbrace{-\int \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{y}\mathrm{d}\mathbf{z}p(\mathbf{x})p(\mathbf{y}|\mathbf{x})q(\mathbf{z}|\mathbf{x};\psi^{\mathrm{enc}}) \log p(\mathbf{y}|\mathbf{z};\psi^{\mathrm{dec}})}_{\mathcal{L}^{\mathrm{pred}}\text{: encouraging predictive power}} \\ &+ \beta \underbrace{\int \mathrm{d}\mathbf{x}\mathrm{d}\mathbf{z}p(\mathbf{x})q(\mathbf{z}|\mathbf{x};\psi^{\mathrm{enc}}) \log \frac{q(\mathbf{z}|\mathbf{x};\psi^{\mathrm{enc}})}{p(\mathbf{z};\phi^{\mathrm{prior}})}}_{\mathcal{L}^{\mathrm{comp}}\text{: encouraging compression}}. \end{aligned} \tag{28}$$

As for now, we have the formulation presented in Eq. 3 in the main text. We proceed to derive the exact loss function for TeCoS-LVM model learning. Following previous literature [32], we can approximate the data distribution $p(\mathbf{x},\mathbf{y})$ using the empirical data distribution $\frac{1}{T}\sum_{t=1}^{T} \delta(\mathbf{x}-\mathbf{x}_{1:t})\delta(\mathbf{y}-\mathbf{y}_t)$, where δ is the dirac delta function. Hence, we have that

$$\begin{aligned} \mathcal{L} &\approx \frac{1}{T}\sum_{t=1}^{T} \Big[\underbrace{\mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_{1:t};\psi^{\mathrm{enc}})}[-\log p(\mathbf{y}_t|\mathbf{z}_t;\psi^{\mathrm{dec}})]}_{\mathcal{L}_t^{\mathrm{pred}}} + \beta \underbrace{\mathrm{KL}[q(\mathbf{z}_t|\mathbf{x}_{1:t};\psi^{\mathrm{enc}})||p(\mathbf{z}_t;\phi^{\mathrm{prior}})]}_{\mathcal{L}_t^{\mathrm{comp}}} \Big] \\ &= \underbrace{\frac{1}{T}\sum_t \mathcal{L}_t^{\mathrm{pred}}}_{\text{Predictive term (total): } \mathcal{L}^{\mathrm{pred}}} + \beta \underbrace{\frac{1}{T}\sum_t \mathcal{L}_t^{\mathrm{comp}}}_{\text{Compressive term (total): } \mathcal{L}^{\mathrm{comp}}}. \end{aligned} \tag{29}$$

As we adopt Gaussian distributions in our temporal conditioning prior and encoder, we can analytically compute the compressive loss term $\mathcal{L}^{\text{comp}}$ composed of Kullback-Leibler divergences.

We then turn to the predictive term in our objective (17). As our model directly produces simulated spike trains, we compute the spike train dissimilarity between the prediction $\hat{\mathbf{y}}_{1:t}$ and the real record $\mathbf{y}_{1:t}$ to assess the predictive power of our model directly. This dissimilarity is used as the predictive loss term at each timestep. In particular, we employ the Maximum Mean Discrepancy (MMD) to measure the distance between spike trains. This approach has been proven suitable for spike trains in previous literature [26, 50]. Following ref. [50], we use a postsynaptic potential (PSP) function kernel for MMD. We employ the first-order synaptic model as the PSP function to capture the temporal dependencies in spike train data effectively [51]. The PSP kernel we shall use is given by

$$\kappa_{\text{PSP}}(\hat{\mathbf{y}}_{1:t}, \mathbf{y}_{1:t}) = \sum_{\tau=1}^{t} \text{PSP}(\hat{\mathbf{y}}_{1:\tau})\text{PSP}(\mathbf{y}_{1:\tau}), \text{where } \text{PSP}(\mathbf{y}_{1:\tau}) = (1 - \frac{1}{\tau_s})\text{PSP}(\mathbf{y}_{1:\tau-1}) + \frac{1}{\tau_s}\mathbf{y}_{\tau}, \tag{30}$$

here τ_s is a synaptic time constant set to 2 by default. We can write the (squared) PSP kernel MMD between the empirical data distribution and the predictive distribution as

$$\text{MMD}[p_{\psi^{\text{dec}}}(\hat{\mathbf{y}}_{1:t}), p(\mathbf{y}_{1:t})]^2 = \sum_{\tau=1}^{t} \left\| \mathbb{E}_{\hat{\mathbf{y}}_{1:\tau}\sim p_{\psi^{\text{dec}}}}[\text{PSP}(\hat{\mathbf{y}}_{1:\tau})] - \mathbb{E}_{\mathbf{y}_{1:\tau}\sim p}[\text{PSP}(\mathbf{y}_{1:\tau})] \right\|^2 \tag{31}$$

In practice, we approximate the spike train dissimilarity, which is measured by the squared PSP kernel MMD in Eq. 31 by $\sum_{\tau} ||\text{PSP}(\hat{\mathbf{y}}_{1:\tau}) - \text{PSP}(\mathbf{y}_{1:\tau})||^2$ [50]. Therefore, the predictive loss term is given by

$$\mathcal{L}_t^{\text{pred}} = \sum_{\tau=1}^{t} ||\text{PSP}(\hat{\mathbf{y}}_{1:\tau}) - \text{PSP}(\mathbf{y}_{1:\tau})||^2 \,. \tag{32}$$

Together with the compressive term $\mathcal{L}_t^{\text{comp}} = \text{KL}[q_{\psi^{\text{enc}}}(\mathbf{z}_t)||p_{\phi^{\text{prior}}}(\mathbf{z}_t)]$, by Eq. 29, we can calculate the loss function and optimize TeCoS-LVM models. To allow direct backpropagation through a single sample of the stochastic latent representation, we use the reparameterization trick as described in [84].

E Data Description

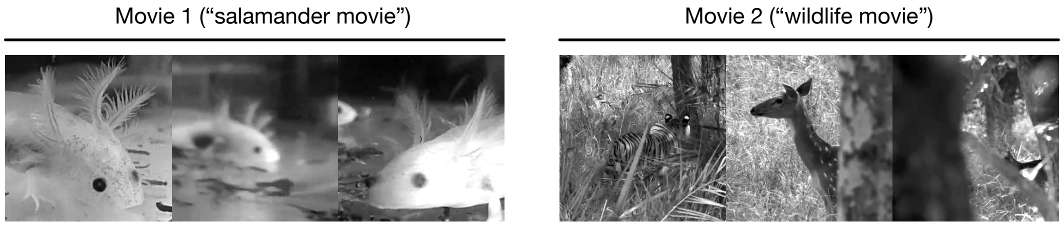

Figure 7: Example frames from Movie 1 and Movie 2 in the data we used.

We perform evaluations and analyses on real neural recordings from RGCs of dark-adapted axolotl salamander retinas. The original dataset [57] contains the spike neural responses (collected using multi-electrode arrays [85, 86]) of two retinas on two movies. Movie 1 contains natural scenes of salamanders swimming in the water. Movie 2 contains complex natural scenes of a tiger on a prey hunt. Both movies were roughly 60 s long and were discretized into bins of 33 ms. All movie frames were converted to grayscale with a resolution of 360 pixel×360 pixel at 7.5 μm×7.5 μm per pixel, covering a 2700 μm×2700 μm area on the retina. For retina 1, we have 75 repetitions for movie 1 and 107 repetitions for movie 2. For retina 2, we have 30 and 42 repetitions for movie 1 and movie 2, respectively. Some example frames are shown in Appendix Fig. 7.

F Metrics, Features used in Evaluations and Visualizations

F.1 Evaluation Metrics

Pearson correlation coefficient (Pearson CC, CC) This metric evaluates the model performance by calculating the Pearson correlation coefficient between the recorded and predicted firing rates [9, 14, 15]. The higher the value, the better the performance. For spike-output TeCoS-LVM models, the firing rates are calculated using 20 repeated trials.

Spike train dissimilarity (Spike train dissim.) This metric assesses the model performance by computing the dissimilarity between recorded and predicted spike trains. A lower value indicates better model performance. We use the MMD with a first-order PSP kernel [51, 52] to measure the spike train dissimilarity [27, 50, 58]. The first-order PSP function is given by $\text{PSP}(\mathbf{y}_{1:t}) = (1 - \frac{1}{\tau_s})\text{PSP}(\mathbf{y}_{1:t-1}) + \frac{1}{\tau_s}\mathbf{y}_t$, where τ_s is a synaptic constant and is set to 2. Given a recorded spike train $\mathbf{y}_{1:T}$ and a predicted spike train $\hat{\mathbf{y}}_{1:T}$, this metric is calculated by $\sum_{t=1}^{T} \|\text{PSP}(\mathbf{y}_{1:t}) - \text{PSP}(\hat{\mathbf{y}}_{1:t})\|^2$. Because of the variability of neural activities, we randomly selected ten (trials) recorded spike trains and used their average value in our evaluations.

van Rossum distance (van Rossum) This spike train distance was introduced in ref. [87], where the discrete spike trains are convolved by an exponential kernel $\texttt{Heaviside}(t)\exp(-t/\tau_R)$, here we use $\tau_R = 10$. The final scores are computed by averaging results calculated using ten recorded spike trains.

Victor-Purpura distance (V.-P.) This spike train distance [88] measures the dissimilarity between two spike trains by summing up the minimum cost of transforming one spike train into the other by insertion, deletion, and shifting operations. We use the average results from ten trials as the final metric.

SPIKE distance (SPIKE) The SPIKE distance [89] is a time-scale independent metric for quantifying the dissimilarity between spike trains. Its value is bounded in the interval [0, 1], and zero is obtained only for perfectly identical trains.

F.2 Spike Feature

Spike autocorrelogram The spike autocorrelogram is computed by counting the number of spikes that occur around each spike within a predefined time window [14, 9]. The resulting trace is then normalized to its maximum value (which occurs at the origin of the time axis by construction). In the main text, the maximum value is set to zero for better visualization and comparison.

G Experimental Details

G.1 Experimental Platform

The models are implemented using Python and PyTorch. Our experiments were conducted on a workstation with an Intel-10400, one NVIDIA 3090, and 64 GB RAM.

G.2 Implementation Details

TeCoS-LVM models For TeCoS-LVM models, all hyper-parameters on all datasets are fixed to be the same. We set the latent variable dimension to 32 and the hidden state dimension to 64 by default. We used the Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$) with a cosine-decay learning rate of 0.0003 with a mini-batch size 64. Training of these models is carried out for 64 epochs. We used the same architectures to implement all the TeCoS-LVM models (see Table 3). For LIF neuron TeCoS-LVM models (denoted as `TeCoS-LVM`), we used surrogate gradient learning (SGL) with an ERF surrogate gradient (see also Eq. 10) $\text{SG}_{\text{ERF}}(x) = \frac{1}{\sqrt{\pi}}\exp(-x^2)$. For Noisy LIF neuron TeCoS-LVM models (denoted as `TeCoS-LVM Noisy`), we used the Gaussian noise $\mathcal{N}(\epsilon; 0, 0.2^2)$ and the corresponding Noise-Driven Learning (a theoretically well-defined general form of SGL), which is described in

Appendix C, Eq. 15. Since random latent variables are involved in our model, we also employed the reparameterization trick for efficient training.

Baselines We followed the settings in their original implementations for the CNN [9, 15] and IB-Disjoint [21] models. Some of these settings leverage the prior statistical structure information of the firing rate, thus improving the performance of these models [9]. In particular, the CNN model is trained with Gaussian noise injection, L-2 norm regularization (0.001) over the model parameters, and L1 norm regularization (0.001) over the predicted activations [9, 15]. The IB-Disjoint model is optimized with $\beta = 0.01$, which has proven to lead to better predictive power [21]. We also use the reparameterization trick for efficient training for the IB-Disjoint model. We use a constant learning rate of 0.001, a mini-batch size of 64, and a default Adam optimizer for these two models. Following their original implementations, we used the early stopping technique in training. The network architectures of these models are listed in Appendix Table 3.

Table 3: List of network architectures (functional models) in our experiments. `conv` for the convolutional layer, `fc` for the fully-connected layer, `GRU` for the gated recurrent unit layer.

Model Name	Description
`TeCoS-LVM`/`TeCoS-LVM Noisy` (LIF/Noisy LIF spiking neurons, input channel=1); `TeCoS-LVM Rate` (LIF-Rate neurons input channel=1); `TeCoS-LVM Noisy Rate` (Noisy LIF-Rate neurons, input channel=1).	(feature extractor) 16`conv`25-32`conv`11-`fc`64 (real-valued RNN) `GRU`64 (encoder) `fc`64-`fc`64 (encoder mean) `fc`32 (encoder std) `fc`32 (prior) `fc`64-`fc`64 (prior mean) `fc`32 (prior std) `fc`32 (decoder) `fc`64-`fc`#RGCs
CNN (ReLU neurons, input channel=T)	32`conv`25-BatchNorm 16`conv`11-BatchNorm `fc`#RGCs-BatchNorm ParametricSoftPlus
IB-Disjoint (ReLU neurons, input channel=T)	16`conv`25-BatchNorm-32`conv`11-BatchNorm `fc`64-BatchNorm (encoder) `fc`64-BatchNorm-`fc`32-BatchNorm (encoder mean) `fc`32-BatchNorm (encoder std) `fc`32-BatchNorm (decoder) `fc`64-BatchNorm-`fc`#RGCs-BatchNorm ParametricSoftPlus

Symbol descriptions (parameter `type` parameter): channel number `conv` kernel size; `fc` channel number; `GRU` hidden state dimension.

H Details of Ablation Experiments

H.1 The effect of using spiking neurons

In this part, we constructed two variants (`TeCoS-LVM Rate` and `TeCoS-LVM Noisy Rate`) by replacing all the hidden spiking neurons in the TeCoS-LVM model with neurons that exhibit the same internal recurrence but provide non-spiking output. In other words, in these two variants, the activation of our hidden neurons transitioned from discrete spiking Heaviside functions to continuous functions. Specifically, we adopted the rate-output neuron model named GLIFR introduced in ref.[65]. In particular, the modified LIF-Rate neuron model we used here still uses the membrane update rules of LIF (as well as Noisy LIF). The output activation function of the LIF-Rate model is described by $o_t = \text{sigmoid}\left(\frac{u_t - v_{\text{th}}}{\sigma_u}\right)$, where the parameter σ_u controls the smoothness of the membrane voltage-spike relationship. In doing so, the internal representation is constructed in a real-valued space, rather than in a sparse spike space as TeCoS-LVM models with all LIF and Noisy LIF neurons.