Appendix to "Extraction and Recovery of Spatio-Temporal Structure in Latent Dynamics Alignment with Diffusion Model"

A Methodology details

A.1 DM architecture details

We adopt the architecture of DM mainly derived from Diffusion Transformer (DiT) [36]. The canonical DiT architecture is based on techniques like patchify [57] and transformer layer for tokens [58], which are well-suited for image feature extraction. This is because the above techniques focus on local feature extraction and the global feature can also be implicitly captured through the stacking of token-based Transformer layers. However, considering the neural observations and latent dynamics are in the format of multi-variate time series, patchify and local feature extraction loses their semantic meaning. There doesn't exist a theoretical guarantee or bio-plausible observation that adjacent latent dimensions of the data matrix have a stronger connection than far-apart latent dimensions. Thus, directly adopting the traditional DiT architecture into this setting may lead to sub-optimal solutions.

To fully utilize the domain knowledge of our task, we propose to use the Spatio-Temporal Transformer Block (STBlock). Each STBlock is mainly composed of a Spatio Transformer layer followed by a Temporal Transformer layer, which are 1-layer encoders based on multi-head self-attention. Since there exists underlying dependency and structure between latent state dimensions, the Spatio Transformer layer takes latent states of each time bin as inputs to extract their spatial structure. Whereas the Temporal Transformer layer takes the entire latent trajectory of each latent space dimension as inputs to extract its underlying temporal structure. We note that we use the sinusoidal position embeddings [59] to encode the timestep t (i.e., noise scale) into the deep neural network of DM. In each STBlock, the input sequentially goes through:

- **Spatio Transformer**: Layer Normalization $\rightarrow$ Multi-head Self-attention Layer (along time bins) $\rightarrow$ Point-wise Feed-forward.
- **Temporal Transformer**: Layer Normalization $\rightarrow$ Multi-head Self-attention Layer (along latent space dimensions) $\rightarrow$ Point-wise Feed-forward.

We illustrate the main architecture of DM in Figure 2(A), and implement the DM in Pytorch [60].

A.2 VAE and DM cooperative source domain learning details

In DM training, we note that $\mathbf{Z}_0^{(s)}$ here are actually latent dynamics inferred via VAE in Eq. 6. Considering the limited number of source-domain latent dynamics, we wish to perform data augmentation so that the DM can adequately estimate $p_s(\mathbf{Z})$. Here we propose to enrich the input samples by learning the VAE objective (Eq. 6) and the diffusion objective (Eq. 8) cooperatively. Through the learning process of the VAE objective (i.e., ELBO), the optimization process with stochastic gradient descent (SGD) adds auxiliary perturbation to the original data samples $\mathbf{Z}_0^{(s)}$ rather than pure Gaussian noise. This technique further fills the sample space of $\mathbf{Z}_0^{(s)}$, leading to better density estimation. Specifically, in each training iteration, conditioning on the current value of $\boldsymbol{\phi}_s$, we infer a set of $\mathbf{Z}_0 = h(\mathbf{X}^{(s)}; \boldsymbol{\phi}_s)$ and use it as the temporal $\mathbf{Z}_0^{(s)}$ to optimize Eq. 8. The traditionally sequential approach is that we fully

Table 2: The coefficient of determination values ($R^2 \uparrow$, in %) and RMSE $\downarrow$ of sequential source domain learning and cooperative source domain learning on the primate motor cortex dataset. Boldface denotes the highest score. Each experiment condition is repeated with 5 different random seeds, and their mean and standard deviation are listed.

	Metric	Sequential	Cooperative
M1-M2	$R^2(\%)$	18.96 ($\pm$2.27)	**20.47** ($\pm$2.71)
	RMSE	7.76 ($\pm$0.40)	**7.62** ($\pm$0.42)
M2-M3	$R^2(\%)$	21.73 ($\pm$2.46)	**22.62** ($\pm$2.66)
	RMSE	7.94 ($\pm$0.47)	**7.73** ($\pm$0.50)
M2-C2	$R^2(\%)$	6.96 ($\pm$2.88)	**8.57** ($\pm$2.96)
	RMSE	11.85 ($\pm$0.42)	**11.64** ($\pm$0.51)

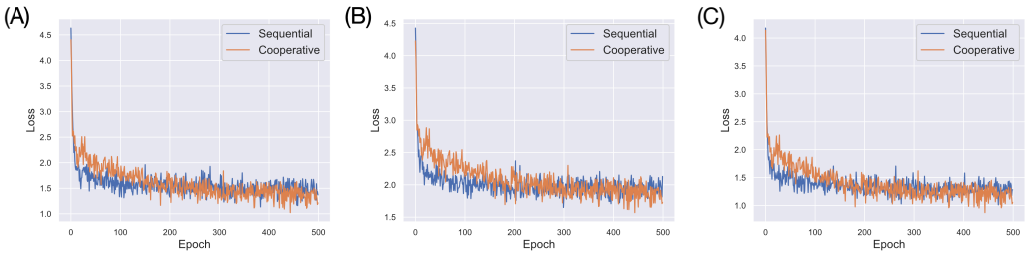

Figure 6: **Training loss curves under different neural sessions**. **(A)** Neural session: M-1. **(B)** Neural session: M-3. **(C)** Neural session: C-2.

optimize VAE first, obtain an optimal ϕ_s, and use it to optimize Eq. 8. Experimental results show that the former approach achieves higher density estimation and alignment performance. Figure 6 manifests the training loss curve in three source-target neural recording session pairs. We observe that despite the relatively under-fitting at the early stage, the cooperative source domain learning paradigm converges to solutions with lower losses and better fits. Table 2 manifests that our cooperative source domain learning paradigm leads to higher distribution alignment and neural decoding performance.

B Detailed derivation of maximum likelihood alignment

B.1 Relationship between KL-Divergence and DSM Loss

Under assumptions in Appendix A of [41], the KL divergence between the ground truth density $q\left(\mathbf{Z} \mid \mathbf{X}^{(t)}; \boldsymbol{\phi}\right)$ and the DM marginal distribution $p_0\left(\mathbf{Z}; \boldsymbol{\theta}_s\right)$ can be derived as:

$$
\begin{aligned}
& \mathbb{D}_{\mathrm{KL}}\left(q(\mathbf{Z} \mid \mathbf{X}^{(t)}; \boldsymbol{\phi}) \| p_0\left(\mathbf{Z}; \boldsymbol{\theta}_s\right)\right) \\
\overset{(i)}{\leqslant} & \mathbb{E}_{t \sim \mathcal{U}[0,T]} \mathbb{E}_{\mathbf{Z} \sim q(\cdot \mid \mathbf{X}^{(t)}; \boldsymbol{\phi})}\left[\lambda(t)\left(\nabla_{\mathbf{Z}} \log p_t(\mathbf{Z}; \boldsymbol{\theta}_s) - \boldsymbol{s}(\mathbf{Z}, t; \boldsymbol{\theta}_s)\right) \mathrm{d}\overline{\mathbf{w}}\right] + \mathbb{D}_{\mathrm{KL}}\left(p_T\left(\mathbf{Z}; \boldsymbol{\theta}_s\right) \| \pi(\mathbf{Z})\right) \\
& + \frac{1}{2} \mathbb{E}_{t \sim \mathcal{U}[0,T]} \mathbb{E}_{\mathbf{Z} \sim q(\cdot \mid \mathbf{X}^{(t)}; \boldsymbol{\phi})}\left[\lambda(t)^2 \left\|\nabla_{\mathbf{Z}} \log p_t(\mathbf{Z}; \boldsymbol{\theta}_s) - \boldsymbol{s}(\mathbf{Z}, t; \boldsymbol{\theta}_s)\right\|_2^2 \mathrm{~d}t\right] \\
\overset{(ii)}{=} & \mathbb{E}_{t \sim \mathcal{U}[0,T]} \mathbb{E}_{\mathbf{Z} \sim q(\cdot \mid \mathbf{X}^{(t)}; \boldsymbol{\phi})}\left[\lambda(t)^2 \left\|\nabla_{\mathbf{Z}} \log p_t(\mathbf{Z}; \boldsymbol{\theta}_s) - \boldsymbol{s}(\mathbf{Z}, t; \boldsymbol{\theta}_s)\right\|_2^2 \mathrm{~d}t\right] + \mathbb{D}_{\mathrm{KL}}\left(p_T\left(\mathbf{Z}; \boldsymbol{\theta}_s\right) \| \pi(\mathbf{Z})\right) \\
\overset{(iii)}{=} & \mathbb{E}_{t \sim \mathcal{U}[0,T]} \mathbb{E}_{\mathbf{Z}_0 \sim q(\cdot \mid \mathbf{X}^{(t)}; \boldsymbol{\phi}), p_{0t}(\mathbf{Z}_t \mid \mathbf{Z}_0)}\left[\lambda(t)^2 \left\|\nabla_{\mathbf{Z}_t} \log p_{0t}\left(\mathbf{Z}_t \mid \mathbf{Z}_0\right) - \boldsymbol{s}\left(\mathbf{Z}_t, t; \boldsymbol{\theta}\right)\right\|_2^2\right] \\
& + \mathbb{D}_{\mathrm{KL}}\left(p_T\left(\mathbf{Z}; \boldsymbol{\theta}_s\right) \| \pi(\mathbf{Z})\right) \\
= & \mathcal{L}_{\mathrm{DSM}}\left(\boldsymbol{\phi}, \boldsymbol{\theta}_s\right) + \mathbb{D}_{\mathrm{KL}}\left(p_T(\mathbf{Z}; \boldsymbol{\theta}_s) \| \pi(\mathbf{Z})\right),
\end{aligned}
\tag{15}
$$

in which (i) is due to Girsanov Theorem [40], in (ii) we invoke the martingale property of Itô integrals [61], and in (iii) we use the denoising score matching (DSM) technique [29]. Thus we can draw to Eq. 11.

B.2 Upper bound of maximum likelihood alignment objective

In Section 3.3, by substituting Eq. 11 into Eq. 10, we have

$$
-\mathbb{E}_{\mathbf{Z} \sim q(\mathbf{Z} \mid \mathbf{X}^{(t)}; \boldsymbol{\phi})}\left[\log p_0(\mathbf{Z}; \boldsymbol{\theta}_s)\right] \leqslant \mathcal{L}_{\mathrm{DSM}}\left(\boldsymbol{\phi}, \boldsymbol{\theta}_s\right) + \mathbb{D}_{\mathrm{KL}}\left(p_T(\mathbf{Z}; \boldsymbol{\theta}_s) \| \pi(\mathbf{Z})\right) + \mathbb{H}\left(q(\mathbf{Z} \mid \mathbf{X}^{(t)}; \boldsymbol{\phi})\right).
\tag{16}
$$

We note that the third term $\mathbb{H}(\cdot)$ depends on the parameter set $\boldsymbol{\phi}$ of the probabilistic encoder. As $q(\mathbf{Z} \mid \mathbf{X}^{(t)}; \boldsymbol{\phi}) \approx p_0(\mathbf{Z}; \boldsymbol{\theta}_s)$, we have

$$\mathbb{H}\left(q(\mathbf{Z} \mid \mathbf{X}^{(t)}; \boldsymbol{\phi})\right) - \mathbb{H}\left(p_T(\mathbf{Z}; \boldsymbol{\theta}_s)\right) \approx \int_T^0 \frac{\partial}{\partial t} \mathbb{H}\left(p_t(\mathbf{Z}; \boldsymbol{\theta}_s)\right) \mathrm{d}t \tag{17}$$

$$\begin{aligned}
&\overset{(i)}{=} \mathbb{E}_{t\sim\mathcal{U}[0,T]} \mathbb{E}_{\mathbf{Z}\sim q(\mathbf{Z}|\mathbf{X}^{(t)};\boldsymbol{\phi})} \left[2\boldsymbol{f}(\mathbf{Z}, t)^\top \nabla_{\mathbf{Z}} \log p_t(\mathbf{Z}; \boldsymbol{\theta}_s) - \lambda(t)^2 \left\|\nabla_{\mathbf{Z}} \log p_t(\mathbf{Z}; \boldsymbol{\theta}_s)\right\|_2^2\right] \mathrm{d}t \\
&\overset{(ii)}{=} - \mathbb{E}_{t\sim\mathcal{U}[0,T]} \mathbb{E}_{\mathbf{Z}\sim q(\mathbf{Z}|\mathbf{X}^{(t)};\boldsymbol{\phi})} \left[2\nabla_{\mathbf{Z}} \cdot \boldsymbol{f}(\mathbf{Z}, t) + \lambda(t)^2 \left\|\nabla_{\mathbf{Z}} \log p_t(\mathbf{Z}; \boldsymbol{\theta}_s)\right\|_2^2\right] \mathrm{d}t \\
&\overset{(iii)}{=} - \mathbb{E}_{t\sim\mathcal{U}[0,T]} \mathbb{E}_{\mathbf{Z}_0\sim q(\cdot|\mathbf{X}^{(t)};\boldsymbol{\phi}), p_{0t}(\mathbf{Z}_t|\mathbf{Z}_0)} \left[2\nabla_{\mathbf{Z}_t} \cdot \boldsymbol{f}(\mathbf{Z}_t, t) + \lambda(t)^2 \left\|\nabla_{\mathbf{Z}_t} \log p_{0t}\left(\mathbf{Z}_t \mid \mathbf{Z}_0\right)\right\|_2^2\right] \mathrm{d}t
\end{aligned} \tag{18}$$

where in both (i) and (ii) we use integration by parts, and in (iii) we use denoising score matching (DSM) [29]. Putting the second term on the LHS of Eq. 17 into RHS and then substituting the third term on the RHS of Eq. 16, we have

$$-\mathbb{E}_{\mathbf{Z}\sim q(\mathbf{Z}|\mathbf{X}^{(t)};\boldsymbol{\phi})}\left[\log p_0(\mathbf{Z}; \boldsymbol{\theta}_s)\right] \leqslant \mathbb{D}_{\mathrm{KL}}\left(p_T(\mathbf{Z}; \boldsymbol{\theta}_s) \| \pi(\mathbf{Z})\right) \tag{19}$$

$$-\mathbb{E}_{t\sim\mathcal{U}[0,T]} \mathbb{E}_{\mathbf{Z}_0\sim q(\cdot|\mathbf{X}^{(t)};\boldsymbol{\phi}), p_{0t}(\mathbf{Z}_t|\mathbf{Z}_0)} \left[\lambda(t)^2 \left\|\nabla_{\mathbf{Z}_t} \log p_{0t}\left(\mathbf{Z}_t \mid \mathbf{Z}_0\right)\right\|_2^2\right] \tag{20}$$

$$+\mathbb{E}_{t\sim\mathcal{U}[0,T]} \mathbb{E}_{\mathbf{Z}_0\sim q(\cdot|\mathbf{X}^{(t)};\boldsymbol{\phi}), p_{0t}(\mathbf{Z}_t|\mathbf{Z}_0)} \left[\lambda(t)^2 \left\|\nabla_{\mathbf{Z}_t} \log p_{0t}\left(\mathbf{Z}_t \mid \mathbf{Z}_0\right) - \boldsymbol{s}\left(\mathbf{Z}_t, t; \boldsymbol{\theta}\right)\right\|_2^2\right] \tag{21}$$

$$-\mathbb{E}_{t\sim\mathcal{U}[0,T]} \mathbb{E}_{\mathbf{Z}_0\sim q(\cdot|\mathbf{X}^{(t)};\boldsymbol{\phi}), p_{0t}(\mathbf{Z}_t|\mathbf{Z}_0)} \left[-2\nabla_{\mathbf{Z}_t} \cdot \boldsymbol{f}\left(\mathbf{Z}_t, t\right)\right]. \tag{22}$$

Since the transition probability $p_{0t}\left(\mathbf{Z}_t \mid \mathbf{Z}_0\right)$ is a fixed Gaussian distribution and it is independent of the parameter set $\boldsymbol{\phi}$, we can eliminate the term in Eq. 20 and rewrite the above Eqs as:

$$-\mathbb{E}_{\mathbf{Z}\sim q(\mathbf{Z}|\mathbf{X}^{(t)};\boldsymbol{\phi})}\left[\log p_0(\mathbf{Z}; \boldsymbol{\theta}_s)\right] \leqslant \mathbb{D}_{\mathrm{KL}}\left(p_T(\mathbf{Z}; \boldsymbol{\theta}_s) \| \pi(\mathbf{Z})\right) \tag{23}$$

$$+\mathbb{E}_{t\sim\mathcal{U}[0,T]} \mathbb{E}_{\mathbf{Z}_0\sim q(\cdot|\mathbf{X}^{(t)};\boldsymbol{\phi}), p_{0t}(\mathbf{Z}_t|\mathbf{Z}_0)} \left[\lambda(t)^2 \left\|\nabla_{\mathbf{Z}_t} \log p_{0t}\left(\mathbf{Z}_t \mid \mathbf{Z}_0\right) - \boldsymbol{s}\left(\mathbf{Z}_t, t; \boldsymbol{\theta}\right)\right\|_2^2\right] \tag{24}$$

$$-\mathbb{E}_{t\sim\mathcal{U}[0,T]} \mathbb{E}_{\mathbf{Z}_0\sim q(\cdot|\mathbf{X}^{(t)};\boldsymbol{\phi}), p_{0t}(\mathbf{Z}_t|\mathbf{Z}_0)} \left[-2\nabla_{\mathbf{Z}_t} \cdot \boldsymbol{f}\left(\mathbf{Z}_t, t\right)\right]. \tag{25}$$

By substituting denoiser function $\boldsymbol{\epsilon}(\mathbf{Z}_t, t; \boldsymbol{\theta})$ into score function $\boldsymbol{s}(\mathbf{Z}_t, t; \boldsymbol{\theta})$ of Eq. 24, we have:

$$\begin{aligned}
&-\mathbb{E}_{\mathbf{Z}\sim q(\mathbf{Z}|\mathbf{X}^{(t)};\boldsymbol{\phi})}\left[\log p_0(\mathbf{Z}; \boldsymbol{\theta}_s)\right] \leqslant \mathbb{D}_{\mathrm{KL}}\left(p_T(\mathbf{Z}; \boldsymbol{\theta}_s) \| \pi(\mathbf{Z})\right) \\
&+\mathbb{E}_{t\sim\mathcal{U}[0,T]} \mathbb{E}_{\mathbf{Z}_0\sim q(\mathbf{Z}|\mathbf{X}^{(t)};\boldsymbol{\phi}), \boldsymbol{\epsilon}\sim\mathcal{N}(0, \boldsymbol{I}_{l\times d})} \left[w(t)^2 \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}(\mathbf{Z}_t, t; \boldsymbol{\theta}_s)\right\|_2^2 - 2\nabla_{\mathbf{Z}_t} \cdot \boldsymbol{f}\left(\mathbf{Z}_t, t\right)\right].
\end{aligned}$$

C Detailed algorithm

C.1 Overall alignment loss function in ERDiff

Extracting the latter two terms from Eq. 12, we have the following main Maximum Likelihood Alignment (MLA) loss:

$$\mathcal{L}_{\mathrm{MLA}}(\boldsymbol{\phi}) = \mathbb{E}_{t\sim\mathcal{U}[0,T]} \mathbb{E}_{\mathbf{Z}_0\sim q(\mathbf{Z}|\mathbf{X}^{(t)};\boldsymbol{\phi}), \boldsymbol{\epsilon}\sim\mathcal{N}(0, \boldsymbol{I}_{l\times d})} \left[w(t)^2 \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}(\mathbf{Z}_t, t; \boldsymbol{\theta}_s)\right\|_2^2 - 2\nabla_{\mathbf{Z}_t} \cdot \boldsymbol{f}\left(\mathbf{Z}_t, t\right)\right]. \tag{26}$$

Then, considering the Sinkhorn Regularizer term in Eq. 13:

$$\mathcal{L}_{\mathrm{SHD}}(\boldsymbol{\phi}) = \min_{\boldsymbol{\gamma}} \langle \boldsymbol{\gamma}, \mathbf{C} \rangle_F + \lambda \mathbb{H}(\boldsymbol{\gamma}), \tag{27}$$

where each value $\mathbf{C}[i][j] = \left\|\mathbf{Z}_i^{(s)} - \mathbf{Z}_j^{(t)}\right\|_2^2$ in cost matrix $\mathbf{C}$ denotes the squared Euclidean cost from $\mathbf{Z}_i^{(s)}$ to $\mathbf{Z}_j^{(t)}$, $\mathbb{H}(\boldsymbol{\gamma})$ computes the entropy of transport plan $\boldsymbol{\gamma}$, and λ refers to the weight of the entropy term. Then, we can find the optimal $\boldsymbol{\phi}_t$ via the minimization of the following total loss function:

$$\boldsymbol{\phi}_t = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[(1-\alpha)\mathcal{L}_{\mathrm{MLA}} + \alpha \mathcal{L}_{\mathrm{SHD}}\right], \tag{28}$$

where $\alpha \in [0, 1]$ is a trade-off parameter that weights the importance of Sinkhorn Regularizer term.

C.2 Algorithm for source domain learning of ERDiff

Algorithm 1 Source Domain Learning of ERDiff

Input: Source-domain neural observations $\mathbf{X}^{(s)}$; Learning rate η and all other hyperparameters.
Output: Parameter set $\phi^{(s)}$ of source-domain probabilistic encoder; Parameter set $\psi^{(s)}$ of source-domain probabilistic decoder; Parameter set $\boldsymbol{\theta}^{(s)}$ of source-domain diffusion model.

Initialize ϕ, ψ, and $\boldsymbol{\theta}$;
while not converge **do**
  $\mathcal{L}_{\mathrm{ELBO}} \leftarrow$ Compute the loss function based on evidence lower bound according to Eq. 6;
  Parameter Update: $\phi \leftarrow \phi - \eta \cdot \partial \mathcal{L}_{\mathrm{ELBO}} / \partial \phi$;
  Parameter Update: $\psi \leftarrow \psi - \eta \cdot \partial \mathcal{L}_{\mathrm{ELBO}} / \partial \psi$;
  Inference $\mathbf{Z}_0^{(s)} \sim q(\cdot \mid \mathbf{X}^{(s)}; \phi)$, and Noise Sampling $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I}_{l \times d})$;
  $\mathcal{L}_{\mathrm{DSM}} \leftarrow$ Compute the denoising score matching loss according to Eq. 5;
  Parameter Update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \cdot \partial \mathcal{L}_{\mathrm{DSM}} / \partial \boldsymbol{\theta}$;
end while
return ϕ, ψ, and $\boldsymbol{\theta}$.

C.3 Algorithm for maximum likelihood alignment of ERDiff

Algorithm 2 Maximum Likelihood Alignment of ERDiff

Input: Target-domain neural observations $\mathbf{X}^{(t)}$; Learning rate η and all other hyperparameters.
Output: Parameter set $\phi^{(t)}$ of target-domain probabilistic encoder;

while not converge **do**
  Inference $\mathbf{Z}_0^{(t)} \sim q(\cdot \mid \mathbf{X}^{(t)}; \phi)$, and Noise Sampling $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I}_{l \times d})$;
  $\mathcal{L}_{\mathrm{MLA}} \leftarrow$ Compute the main Maximum Likelihood Alignment Loss according to Eq. 26;
  $\mathcal{L}_{\mathrm{SHD}} \leftarrow$ Compute the Sinkhorn Regularizer according to Eq. 27;
  Parameter Update: $\phi \leftarrow \phi - \eta \cdot \partial \left[(1-\alpha)\mathcal{L}_{\mathrm{MLA}} + \alpha \mathcal{L}_{\mathrm{SHD}} \right] / \partial \phi$;
end while
return ϕ.

D Experimental results on rat hippocampal CA1 dataset

We further verify the effectiveness of ERDiff on a publicly available rat hippocampus dataset [27]. In this study, a rat navigated a 1.6m linear track, receiving rewards at both terminals (L&R) as depicted in Figure 4 (A). Concurrently, neural activity from the hippocampal CA1 region was captured, in which the neuron numbers across all recorded sessions are around 120. The neural spiking activities were binned into 25ms intervals. A single round trip by the rat from one end of the track to the opposite was categorized as one lap. We chose 5 sessions in total, and in each session, 80 laps were sampled. The 2−dimensional neural latent manifolds can be observed in Figure 7 (B) and (C). We also list the neural decoding results of the rat's position in Table 3.

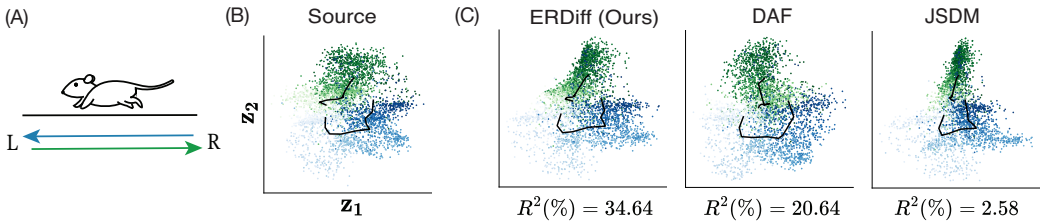

Figure 7: **Rat hippocampus dataset and experimental results**. **(A)** Illustration of the experiment setting. **(B)** 2D visualization of inferred latents in the source domain, in which the bold lines manifest the mean location of latent states corresponding to the rat's position for two opposite directions. **(C)** On target domain, 2D visualization of inferred latents after alignment by ERDiff, DAF, and JSDM, respectively. We observe that ERDiff preserves the spatio-temporal structure of latent dynamics well.

Table 3: Decoding results after neural distribution alignment on the rat hippocampus dataset across sessions. Each experiment condition is repeated with 5 different random seeds, and their mean and standard deviation are listed.

Method	$R^2(\%) \uparrow$	RMSE $\downarrow$
Cycle-GAN	4.81 ($\pm$2.47)	7.65 ($\pm$0.34)
JSDM	-2.88 ($\pm$1.59)	7.65 ($\pm$0.23)
SASA	18.51 ($\pm$1.65)	6.98 ($\pm$0.20)
DANN	15.04 ($\pm$2.32)	7.30 ($\pm$0.26)
RDA-MMD	21.73 ($\pm$2.45)	7.22 ($\pm$0.26)
DAF	20.20 ($\pm$2.26)	7.36 ($\pm$0.28)
ERDiff	$\mathbf{32.69}$($\pm$2.19)	$\mathbf{6.84}$($\pm$0.26)

Table 4: Comparative analyses of computational cost between ERDiff and baseline methods during alignment. ERDiff has a comparable computational cost and maintains the stability of alignment.

Method	Cycle-GAN	JSDM	SASA	RDA-MMD	DAF	**ERDiff**
Add'l. Param	26K	0K	33K	65K	91K	28K
Add'l. Size	117KB	0KB	187KB	314KB	367KB	139KB
Align. Time	103ms	77ms	155ms	264ms	251ms	183ms
Stability	✗	✔	✔	✗	✗	✔

E Computational cost analysis

Here we conduct time complexity analysis with respect to the batch size B for the alignment phase. The ERDiff's alignment objective function is composed of two main terms: Diffusion Noise Residual and Sinkhorn Divergence. We note that in the diffusion noise residual computation, it does not go through the entire T diffusion steps. Instead, it just samples a single time step (noise scale) t and calculates the noise residual specific to that step. Thus, the total complexity of this part takes $\mathcal{O}(K_1 * B * d)$, in which the coefficient K_1 relates to the inference complexity of the DM denoiser $\boldsymbol{\epsilon}(\mathbf{Z}, t)$; d denotes the latent dimension size. For the Sinkhorn Divergence, it has to compute the distance matrix, costing $\mathcal{O}(K_2 * B^2)$; K_2 is a relatively small coefficient in magnitude. By summing up, the total complexity of ERDiff is given by $\mathcal{O}(K_1 * B * d + K_2 * B^2)$. This $\mathcal{O}(B^2)$ complexity is applicable since the non-adversarial baseline methods we compared (i.e., JSDM, and SASA) require quadratic complexities as well.

For any given target domain, ERDiff can stably align it to the source domain in a comparable overhead with baselines. In Table 4, we conduct a comparative analysis between ERDiff and baseline methods in terms of additional parameter number, additional model size, stability, and alignment time. The demonstrated alignment time corresponds to the execution time for aligning one iteration (a batch of size 64) on a MacBook Pro (2019 equipped with 8-Core Intel Core i9 and 4 GB RAM).