entropy

MDPI

Article

Representation Learning for Dynamic Functional Connectivities via Variational Dynamic Graph Latent Variable Models

Yicong Huang and Zhuliang Yu *

College of Automation Science and Technology, South China University of Technology, Guangzhou 510641, China; 201920116424@mail.scut.edu.cn
* Correspondence: zlyu@scut.edu.cn

Abstract: Latent variable models (LVMs) for neural population spikes have revealed informative low-dimensional dynamics about the neural data and have become powerful tools for analyzing and interpreting neural activity. However, these approaches are unable to determine the neurophysiological meaning of the inferred latent dynamics. On the other hand, emerging evidence suggests that dynamic functional connectivities (DFC) may be responsible for neural activity patterns underlying cognition or behavior. We are interested in studying how DFC are associated with the low-dimensional structure of neural activities. Most existing LVMs are based on a point process and fail to model evolving relationships. In this work, we introduce a dynamic graph as the latent variable and develop a Variational Dynamic Graph Latent Variable Model (VDGLVM), a representation learning model based on the variational information bottleneck framework. VDGLVM utilizes a graph generative model and a graph neural network to capture dynamic communication between nodes that one has no access to from the observed data. The proposed computational model provides guaranteed behavior-decoding performance and improves LVMs by associating the inferred latent dynamics with probable DFC.

Keywords: neural latent variable models; dynamic functional connectivities; variational information bottleneck; dynamic graphs

Citation: Huang, Y.; Yu, Z. Representation Learning for Dynamic Functional Connectivities via Variational Dynamic Graph Latent Variable Models. *Entropy* **2022**, *24*, 152. https://doi.org/10.3390/e24020152

Academic Editor: Luis Javier García Villalba

Received: 6 December 2021
Accepted: 14 January 2022
Published: 19 January 2022

1. Introduction

Recent progress in invasive recording technologies, such as high-density microelectrode arrays [1–3] and optical fibers [4], allows access to large-scale neural population activities with the precision of single neurons. While these data are usually high-dimensional [5,6], the literature has shown that dimensionality-reduction approaches and generative models can faithfully explain population spike activities [7] and infer single-trial neural firing rates [8] with stable low-dimensional latent dynamics. At present, with rapid developments in machine learning and deep learning, the community has proposed several latent variable models (LVMs) with better efficiency and performance to extract the low-dimensional structure [9–11]. They bring novel insights into neuroscience [12,13] and facilitate the development of brain–computer interfaces [14].

However, existing LVMs consider the latent dynamics as abstract trajectories in the vector space without neurophysiological meaning. We are interested in how the coordination of different brain functions produces such low-dimensional trajectories. In other words, can we specify the exact meaning for each dimension of the trajectories based on neurophysiology? This question may help with understanding how cognitive or behavioral processes emerge. Recently, advances in network neuroscience [15,16] suggest that dynamic functional connectivities (DFC) may be a more accurate representation of functional brain networks [17–20]. This refers to dynamic relationships between distributed signals. DFC may be the mechanism for the coordination of activity between different neural networks to accomplish a complex task. Hence, it may also be a proper explanation for the latent

dynamics discovered by LVMs. Nevertheless, while LVMs assume that latent dynamics follow a point process, DFC involve modeling dynamic relationships between variables. The community needs a more general framework to associate LVMs with DFC.

In this work, we improve LVMs and bridge LVMs and DFC by drawing on recent progress in graph representation learning and graph generative models [21], proposing a Variational Dynamic Graph Latent Variable Model (VDGLVM). We first generalize LVMs with the variational information bottleneck [22] framework to enable the design of non-Euclidean latent variables. We then leverage a dynamic graph as the latent variable to model population spiking and behavior simultaneously. The inferred evolution of relationships between nodes in the inferred dynamic graph can be associated with DFC implicitly. Thus, our model improves LVMs via specifying a probable explanation for the latent dynamics based on DFC. We evaluate our framework with real-world neural data. The results show that our method can not only decode behaviors with high performance but also identify simple yet interpretable representations that are informative of the task.

2. Background and Related Work

In this section, we describe the notations used in the present work, and we provide some background and related work on latent variable models, dynamic functional connectivity, and graph neural networks.

2.1. Latent Variable Models (LVMs)

We consider recorded spike count data from N neurons at T time steps, which results in an observation matrix $\mathbf{X} \in \mathbb{N}^{N \times T}$, with elements $x_{i,t}$ denoting the spike count of neuron $i \in \{1, \ldots, N\}$ at time $t \in \{1, \ldots, T\}$, and the corresponding behavior $\mathbf{Y} \in \mathbb{R}^{B \times T}$, with B as the dimension of the behavior. The literature assumes that the spikes are generated from an inhomogeneous Poisson process based on an underlying non-negative value firing rate. Most LVMs try to find a lower-dimensional trajectory $\mathbf{Z} \in \mathbb{R}^{L \times T}$ with $L \leqslant N$, which explains the observed neural data. In this paper, we focus on dynamical models for behavior decoding. Usually, this is accomplished within the auto-encoder framework. Different choices of the encoder and the decoder may result in different LVMs, including the Gaussian Process (GP) [23–27], linear dynamical system (LDS) [28,29], multi-layer perceptrons (MLP) [30,31], recurrent neural networks (RNN) [8,26], etc. Moreover, since most LVMs are generative models, some approaches extract complex latent structures by carefully designing constraints on the latent space. For example, pi-VAE explicitly uses brain states as labels and models them with the latent variable in a supervised manner [31] and Swap-VAE leverages self-supervised learning techniques to decompose the latent space [30].

In this work, we consider the dynamic communications of nodes as the underlying dynamics. While existing LVMs only use a point process without exact neurophysiological meanings, our model introduces DFC to explain the latent dynamics. A slight yet significant difference between the proposed model and LVMs mentioned above is that we train our model to decode behavior in a supervised manner. This fits more closely with the information-based framework described below.

2.2. Dynamic Functional Connectivities (DFC)

Recently, the focus of the systems neuroscience community tends to shift from the analysis of the static correlation between signals to the study of dynamic relationships between different brain functions evolving with time [32]. The communications are often referred to as dynamic functional connectivities (DFC). To capture undirectional statistical dependencies between different neural signals, a common approach is investigating the correlations based on sliding windows [33–36]. To model directional dependencies, the most frequently used method is to estimate Granger causality [37–39]. Since DFC can be conveniently described using a dynamic graph, some works on optimizing the dynamic graph structure based on graph measures or designed constraints have been developed [40–42]. These

approaches have been applied on modeling DFC based on functional magnetic resonance imaging (fMRI) [36], electroencephalograms [43,44], and neurons [42].

For these methods of studying DFC, the nodes are directly defined in the data, and the challenge accurately estimates the correlations or edges based on the data. However, to associate LVMs with DFC, the nodes and node information for determining the edges are not available. To bridge this gap and reveal a probable neurophysiological meaning to improve LVMs, the present work extends LVMs and explores a computational model to infer a dynamic graph that can be associated with DFC.

2.3. Graph Neural Networks (GNNs)

A graph $G = (V, E)$ is defined by its node set $V = \{v_1, v_2, \ldots, v_{|V|}\}$ and edge set $E \subseteq V \times V$. The numbers of nodes and edges are denoted by $|V|$ and $|E|$, respectively. The structure of the graph can be described by an adjacency matrix, $\mathbf{A} \in \{0,1\}^{|V|\times|V|}$. We also have d-dimensional features for each node in the graph, denoted as $\mathbf{X_g} \in \mathbb{R}^{|V|\times d}$.

GNNs are representation learning neural networks for graph-structured data [21,45,46]. They have achieved remarkable success in different areas, such as molecular graph generation [47], brain connectome analysis [48], etc. GNNs leverage a message-passing [49] procedure to obtain a representation for each node v_i. It first aggregates features of its neighbor nodes $\mathcal{N}_i$ to obtain a message $\mathbf{m}_i$ for v_i, with a multi-set function $AGGREGATE$ as:

$$\mathbf{m}_i = AGGREGATE\big(\{\mathbf{h}_j, v_j \in \mathcal{N}_i\}\big), \tag{1}$$

where $\mathbf{h}_j$ is the intermediate representation of node v_j. Second, GNNs update the representation for each node v_i with the message via a function $COMBINE$ as:

$$\mathbf{h}_i = COMBINE(\mathbf{h}_i, \mathbf{m}_i). \tag{2}$$

GNNs repeat these two steps to iteratively aggregate neighbor information to obtain better node representations. For graph-level representation learning, GNNs use an additional multi-set function $READOUT$ on the node embeddings to have a graph representation vector $\mathbf{h}_g$:

$$\mathbf{h}_g = READOUT(\mathbf{h}_k, v_k \in V). \tag{3}$$

In this work, we define DFC as the latent variable instead of constructing edges on nodes predefined on the data. Thus, we leverage graph generative models to infer a dynamic graph as the latent variable. We also use GNNs to map the dynamic relationships between nodes and behavior.

3. Methodology

In this section, we introduce VDGLVM. We start by introducing the learning objectives of LVMs based on the variational information bottleneck framework to understand LVMs at a higher level. Within the framework, we build a generative model to utilize a dynamic graph as the latent variable. VDGLVM inherits the expressiveness of deep neural networks and attempts to interpret neural latent dynamics by LVMs based on DFC.

3.1. Variational Information Bottleneck (VIB)

The information-theoretical framework explains deep neural networks as an information bottleneck [50]. A network is optimized to find the representation mapping that maintains the maximum mutual information between the input and output [51]. The literature has shown that deep neural networks that constrain information from the input to an intermediate representation tend to be more robust [22].

For LVMs, we find the latent point process $\mathbf{Z}$ by maximizing the mutual information between the latent variable $\mathbf{Z}$ and the behavior variable $\mathbf{Y}$, restricting the mutual informa-

tion between the spikes variable $\mathbf{X}$ and the latent variable $\mathbf{Z}$ with a constant I_c, resulting in a constrained optimization problem:

$$\begin{aligned} &\max MI(\mathbf{Z};\mathbf{Y}) \\ &s.t.\ MI(\mathbf{X};\mathbf{Z}) \leqslant I_c, \end{aligned} \tag{4}$$

where the exact form of $MI(\mathbf{Z};\mathbf{Y})$ is:

$$\begin{aligned} MI(\mathbf{Z};\mathbf{Y}) &= \mathbb{E}_{p(\mathbf{Z},\mathbf{Y})}\left[\log \frac{p(\mathbf{Z}|\mathbf{Y})}{p(\mathbf{Z})}\right] \\ &= \mathbb{E}_{p(\mathbf{Z},\mathbf{Y})}\left[\log \frac{p(\mathbf{Y}|\mathbf{Z})}{p(\mathbf{Y})}\right]. \end{aligned} \tag{5}$$

We can be further formulate the objective by introducing a Lagrange multiplier $\beta > 0$ as:

$$\min\ -MI(\mathbf{Z},\mathbf{Y}) + \beta MI(\mathbf{X},\mathbf{Z}). \tag{6}$$

This suggests that we can tune β to find a suitable bottleneck for deep neural networks to restrict the mutual information between $\mathbf{X}$ and $\mathbf{Z}$. However, the direct optimization of the above objective function is intractable since mutual information is notoriously difficult to compute when we only have access to samples but not the exact distributions.

VIB [22] solves this by maximizing a lower bound of $MI(\mathbf{Z},\mathbf{Y})$ and minimizing an upper bound of $MI(\mathbf{X},\mathbf{Z})$ simultaneously. The key is to use a variational distribution $q(\mathbf{Y}|\mathbf{Z})$ to approximate the intractable conditional distribution $p(\mathbf{Y}|\mathbf{Z})$. This approximation indicates that $KL(p(\mathbf{Y}|\mathbf{Z})||q(\mathbf{Y}|\mathbf{Z})) \geqslant 0$, where $KL(\cdot||\cdot)$ is the Kullback–Leibler divergence. Therefore, we have:

$$\begin{aligned} MI(\mathbf{Z};\mathbf{Y}) &= \mathbb{E}_{p(\mathbf{Z},\mathbf{Y})}\left[\log \frac{p(\mathbf{Y}|\mathbf{Z})}{p(\mathbf{Y})}\right] \\ &\geqslant \mathbb{E}_{p(\mathbf{Z},\mathbf{Y})}\left[\log \frac{q(\mathbf{Y}|\mathbf{Z})}{p(\mathbf{Y})}\right] \\ &= \mathbb{E}_{p(\mathbf{Z},\mathbf{Y})}[\log q(\mathbf{Y}|\mathbf{Z})] + H(\mathbf{Y}), \end{aligned} \tag{7}$$

where $H(\mathbf{Y})$ is the differential entropy of the output variable $\mathbf{Y}$. We neglect this term in optimization since it is irrelevant to the model once given the data. This bound is tight when $q(\mathbf{Y}|\mathbf{Z}) = p(\mathbf{Y}|\mathbf{Z})$. Based on the graphical model of the information bottleneck principle, we have a lower bound of $MI(\mathbf{Z};\mathbf{Y})$ as:

$$MI(\mathbf{Z};\mathbf{Y}) \geqslant \int_{\mathbf{X}}\int_{\mathbf{Y}}\int_{\mathbf{Z}} p(\mathbf{X},\mathbf{Y})p(\mathbf{Z}|\mathbf{X})\log q(\mathbf{Y}|\mathbf{Z})d\mathbf{X}d\mathbf{Y}d\mathbf{Z}. \tag{8}$$

Similarly, an upper bound of $MI(\mathbf{X};\mathbf{Z})$ is:

$$MI(\mathbf{X};\mathbf{Z}) \leqslant \mathbb{E}_{p(\mathbf{X})}[KL(p(\mathbf{Z}|\mathbf{X})||q(\mathbf{Z}))], \tag{9}$$

where $q(\mathbf{Z})$ is a variational approximation of the latent distribution $p(\mathbf{Z})$. The bound is tight when $q(\mathbf{Z}) = p(\mathbf{Z})$. Using the following empirical data distribution,

$$p(\mathbf{X},\mathbf{Y}) = \frac{1}{N_s}\sum_{n=1}^{N_s} \delta_{\mathbf{X}^{(i)}}(\mathbf{X}^{(i)})\delta_{\mathbf{Y}^{(i)}}(\mathbf{Y}^{(i)}), \tag{10}$$

where N_s is the number of samples and $\delta(\cdot)$ is the Dirac delta function, we have the objective for LVMs based on VIB when the latent variable is a point process:

$$\min\ \frac{1}{N_s}\sum_{n=1}^{N_s}\left[-\int_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}^{(\mathbf{i})})\log q(\mathbf{Y}^{(\mathbf{i})}|\mathbf{Z})d\mathbf{Z} + \beta KL(p(\mathbf{Z}|\mathbf{X}^{(i)})||q(\mathbf{Z}))\right], \tag{11}$$

where $p(\mathbf{Z}|\mathbf{X})$ is the encoder and $q(\mathbf{Y}|\mathbf{Z})$ is the decoder. The first term is the likelihood of output $\mathbf{Y}$, given latent variable $\mathbf{Z}$, inferred from input $\mathbf{X}$. In this work, $\mathbf{Y}$ is the behaviors of the subject, and thus, we formulate this term based on a Gaussian distribution. The second term is regarded as a regularization term for the latent variable $\mathbf{Z}$, given the prior distribution $q(\mathbf{Z})$.

3.2. Dynamic Graphs as Dynamic Functional Connectivities

Most LVMs study neural activities in a particular region based on the behavior, while the behavior may be completed as a complex cognitive process involving communications between different brain regions. However, for LVMs, nodes and edges for defining a graph are not available on the data, and thus, one needs to infer them, given the data. This can be formulated as inferring the dynamic relationships between variables that are not defined in the observed data. Meanwhile, graphs are suitable to model dependencies. Recent progress on graph-based deep learning significantly promotes graph modeling both in performance and efficiency. We then wonder whether a graph is feasible as the latent variable of LVMs, such that it corresponds to DFC and provides a probable explanation for the latent dynamics based on graph generative models.

To simplify, we assume that the dynamic relationships faithfully capture the evolving nature of the observations. The present work aims to capture pair-wise dependencies between variables and emphasize the dynamic relationships in generating neural activities and behaviors. Thus, we do not consider node dynamics, but assume that the dynamic relationships already summarize all necessary information about the data. The nodes only serve as labels for the variables we are interested in.

Let $G_t = (V, E_t)$ be a dynamic graph, described by its dynamic adjacency matrix $\mathbf{A}_t$ and static node features $\mathbf{X_g}$, and let $\mathbf{G} = \{G_1, \ldots, G_T\}$ be a graph process. Our objective is to replace the point process $\mathbf{Z}$ with a graph process $\mathbf{G}$ within the VIB framework. However, it is intractable to directly solve the problem due to the discrete nature of graphs. We start by assuming that the graph is a Gilbert random graph [52], such that every possible edge of a graph is conditionally independent of each other. Let $e_{ij} \sim Bernoulli(\theta_{ij})$ be the binary variable indicating whether the edge (i, j) exists, where θ_{ij} is the probability that edge (i, j) occurs in a graph G. Based on the theory of Gilbert random graphs, the distribution of random graph variable G can be factorized as:

$$p(G) = \prod_{(i,j)\in E} p(e_{ij}). \tag{12}$$

We relax the edge weights from binary variables to continuous variables in the range $(0, 1)$, such that we can optimize the objective based on gradient descent efficiently. Specifically, we give a weight $\hat{e}_{ij}$ to every edge (i, j) to reformulate G as a smoothed-graph variable $\hat{G}$. The edge weights can be summarized in a modified adjacency matrix $\hat{\mathbf{A}} \in [0, 1]^{n\times n}$. Therefore, with a smoothed-graph process $\hat{\mathbf{G}} = \{\hat{G}_1, \ldots, \hat{G}_T\}$, the objective function becomes:

$$\min \frac{1}{N_s} \sum_{n=1}^{N_s} \left[- \int_{\hat{\mathbf{G}}} p(\hat{\mathbf{G}}|\mathbf{X}^{(\mathbf{i})}) \log q(\mathbf{Y}^{(\mathbf{i})}|\hat{\mathbf{G}}) d\hat{\mathbf{G}} + \beta KL(p(\hat{\mathbf{G}}|\mathbf{X}^{(i)})||q(\hat{\mathbf{G}})) \right], \tag{13}$$

where $p(\hat{\mathbf{G}}|\mathbf{X}^{(i)})$ is a sequential graph generative model serving as the encoder, and $q(\mathbf{Y}^{(\mathbf{i})}|\hat{\mathbf{G}})$ is a sequential graph neural network serving as the decoder.

3.3. VDGLVM

In the VIB framework, we propose the use of dynamic graphs as the latent variable to study the relationship dynamics between static variables with VDGLVM. The architecture of VDGLVM is shown in Figure 1. The network is comprised of 4 layers. The encoder $p(\hat{\mathbf{G}}|\mathbf{X}^{(i)})$ includes the encoding block, extracting features from neural population spikes, and the graph generative model, generating graphs. The decoder $q(\mathbf{Y}^{(\mathbf{i})}|\hat{\mathbf{G}})$ includes the

graph neural network, which performs graph-level representation learning, and the output layer, which maps graph embeddings to output behavior.

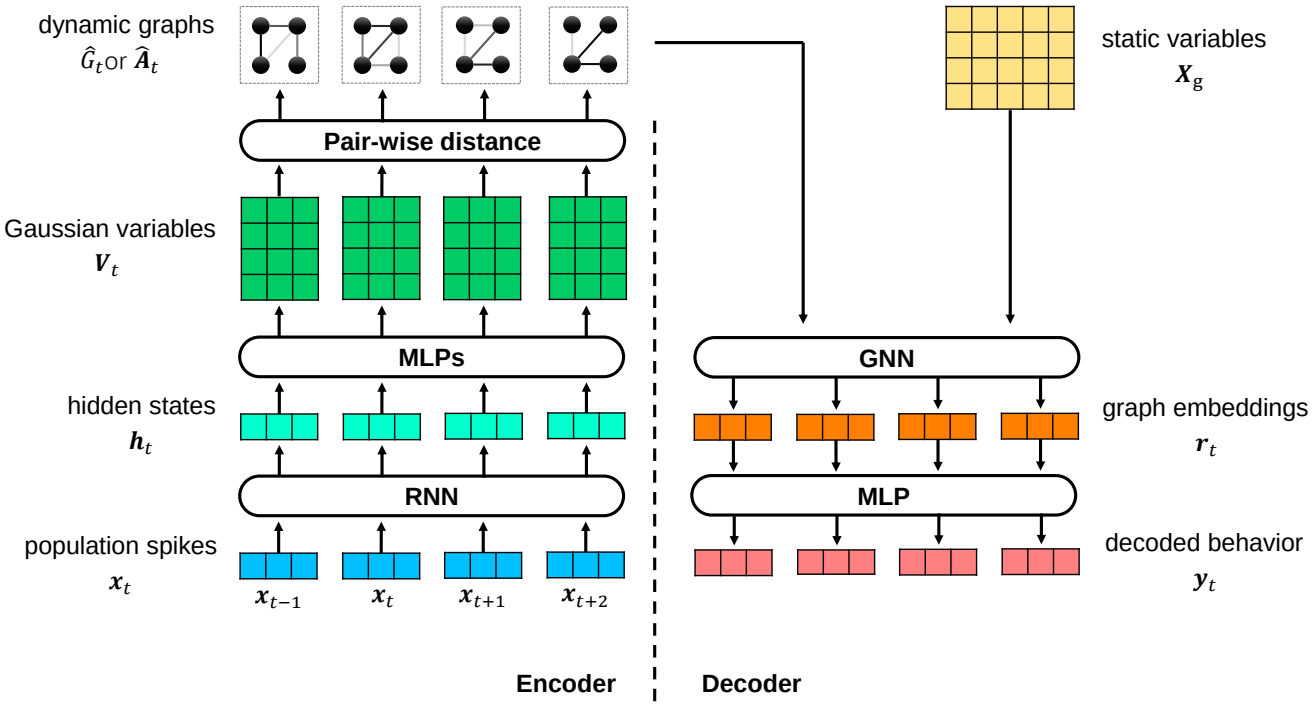

Figure 1. Illustration of VDGLVM using graphs to model the dynamic relationships between static variables. The left side is the encoder, consisting of a dynamical model extracting features from spikes data and a graph generative network building graphs for every time step. The right side is the decoder, including a GNN computing embeddings for graphs and a MLP projector mapping embeddings to behaviors.

3.3.1. Encoder

We instantiate $p(\hat{\mathbf{G}}|\mathbf{X}^{(i)})$ based on dynamical models. To capture nonlinear, non-Markovian, long short-term time-dependent dynamics, we utilize the RNN to extract features. At each time step t, RNN reads the spikes observation $\mathbf{x}_t$ and updates its hidden state $\mathbf{h}_t \in \mathbb{R}^H$ by:

$$\mathbf{h}_{t+1} = RNN_\theta(\mathbf{x}_t, \mathbf{h}_t), \tag{14}$$

where RNN_θ is a deterministic non-linear transition function parameterized by θ. In practice, the vanilla RNN may suffer from gradient issues; it becomes hard to train and leads to performance degeneration. Thus, normally, RNN_θ is implemented via long short-term memory (LSTM) or a gated recurrent unit (GRU). It models the dynamic graph $\hat{G}_t$ by parameterizing a factorization of the joint sequence probability distribution as a product of conditional probabilities, such that:

$$\begin{aligned} p(\hat{G}_1, \ldots, \hat{G}_T) &= \prod_{t=1}^{T} p(\hat{G}_t|\hat{G}_{<t}) \\ p(\hat{G}_t|\hat{G}_{<t}) &= p(\hat{G}_t|\mathbf{h}_t), \end{aligned} \tag{15}$$

where we instantiate $p_\phi(\hat{G}_t|\mathbf{h}_t)$ as a graph generative model parameterized by ϕ, which maps the RNN hidden state $\mathbf{h}_t$ to a probability distribution of the graph variable $\hat{G}_t$, which can be factorized as:

$$p(\hat{G}_t) = \prod_{i=1}^{n}\prod_{j=1}^{n} p(\hat{e}_{ij_t}). \tag{16}$$

To compute egde weights, we first use n MLPs to generate n Gaussian latent variables $\mathbf{v}_{i_t} \in \mathbb{R}^d$, summarized in $\mathbf{V}_t \in \mathbb{R}^{n \times d}$, from the RNN hidden state, as:

$$\begin{aligned} p(\mathbf{v}_{i_t}|\mathbf{h}_t) &= \mathcal{N}\left(\mu_{\mathbf{v}_{i_t}}, \sigma^2_{\mathbf{v}_{i_t}}\right) \\ \left[\mu_{\mathbf{v}_{i_t}}, \sigma^2_{\mathbf{v}_{i_t}}\right] &= MLP_i(\mathbf{h}_t), \end{aligned} \tag{17}$$

where, in practice, we share the parameters of MLP_i, except for the last layer. We apply the reparameterization trick to optimize the objective function with gradient-based methods. Specifically, we sample ϵ from a standard normal distribution, and then we generate $\mathbf{v}_{it}$ via:

$$\begin{aligned} \epsilon &\sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ \mathbf{v}_{i_t} &= \mu_{\mathbf{v}_{i_t}} + \epsilon \odot \sigma_{\mathbf{v}_{i_t}}. \end{aligned} \tag{18}$$

To perform link prediction, we compute the edge weights $\hat{e}_{ij_t}$ based on pair-wise distance defined by cosine similarity:

$$\begin{aligned} p(\hat{G}_t|\mathbf{V}_t) &= \prod_{i=1}^{n}\prod_{j=1}^{n} p(\hat{e}_{ij_t}|\mathbf{v}_{i_t}, \mathbf{v}_{j_t}) \\ \hat{e}_{ij_t} &= \sigma\left(\frac{\log w_{ij_t} - \log\left(1 - w_{ij_t}\right)}{\tau}\right) \\ w_{ij_t} &= \frac{1}{2}\left(\frac{{\mathbf{v}_{i_t}}^T \mathbf{v}_{j_t}}{\left\|\mathbf{v}_{i_t}\right\| \times \left\|\mathbf{v}_{j_t}\right\|} + 1\right), \end{aligned} \tag{19}$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the sigmoid function and $\tau \in (0, 1)$ is temperature. When $\tau \to 0$, we have:

$$\begin{aligned} \lim_{\tau \to 0} p(\hat{e}_{ij_t}|\mathbf{v}_{i_t}, \mathbf{v}_{j_t}) &= 0, \; if \; w_{ij_t} < 0.5, \\ \lim_{\tau \to 0} p(\hat{e}_{ij_t}|\mathbf{v}_{i_t}, \mathbf{v}_{j_t}) &= 1, \; if \; w_{ij_t} > 0.5, \end{aligned} \tag{20}$$

and $p(\hat{e}_{ij_t}|\mathbf{v}_{i_t}, \mathbf{v}_{j_t})$ becomes more sensitive when w_{ij_t} is around 0.5. When $\tau = 1$, $p(\hat{e}_{ij_t}|\mathbf{v}_{it}, \mathbf{v}_{j_t}) = w_{ij_t}$. Our solution differs from previous works, generating edges such as: $p(\hat{e}_{ij_t}|\mathbf{v}_{it}, \mathbf{v}_{j_t}) = \sigma\left({\mathbf{v}_{it}}^T \mathbf{v}_{j_t}\right)$. Although both procedures use a continuous distribution to approximate the Bernoulli distribution, the logarithm and the normalization provide more stable gradients. With a proper temperature τ, the objective function is smoothed with a well-defined gradient $\frac{\partial \hat{e}_{ij_t}}{\partial w_{ij_t}}$. This is similar to the binary concrete distribution, but we do not involve the reparameterization trick and sample from the distribution, and the formula is slightly different. We provide this scheme as a better alternative to generate connectivities from a set of vectors.

Moreover, when $\tau \to \infty$, the weight $\hat{e}_{ij_t}$ tends to be irrelevant to w_{ij_t} as:

$$\lim_{\tau \to \infty} p(\hat{e}_{ij_t}|\mathbf{v}_{i_t}, \mathbf{v}_{j_t}) = 0.5. \tag{21}$$

This means that, when $\tau > 1$, this computation is similar to the VIB, which controls information from w_{ij_t} to $\hat{e}_{ij_t}$ given a prior $q(\hat{e}_{ij_t}) = 0.5$. Nevertheless, a focal solution for $\hat{G}_t$, such that $\hat{e}_{ij_t}$ stays around 0 or 1, may be more informative about the data, since each functional connectivities may distribute more separately. Hence, here, we study $\tau \in (0, 1)$.

3.3.2. Decoder

We design $q(\mathbf{Y}^{(\mathbf{i})}|\hat{\mathbf{G}})$ as a combination of a sequential graph-level GNN $p_{\omega_0}(\mathbf{r}_t|\hat{G}_t)$ parameterized by ω_0, and a multi-layer neural network MLP_{ω_1} parameterized by ω_1, where

$\mathbf{r}_t \in \mathbb{R}^d$ is the graph-embedding vector for $\hat{G}_t$. We implement $p_{\omega_0}(\mathbf{r}_t|\hat{G}_t)$ based on a 1-layer Graph Isomorphism Network (GIN) [53]:

$$\begin{aligned} \mathbf{X_g}' &= MLP_\eta\big((\hat{\mathbf{A}}_t + (1+\gamma)\mathbf{I})\mathbf{X_g}\big) \\ \mathbf{r}_t &= sum(\mathbf{X_g}'), \end{aligned} \tag{22}$$

where MLP_η is a multi-layer neural network parameterized by η, γ is a hyperparameter, $\mathbf{X_g} \in \mathbb{R}^{n\times d}$ is the matrix for n d-dimensional static variables defined by the delta function, $\hat{\mathbf{A}}_t$ is the dynamic adjacency matrix, and sum is a summation across nodes serving as the $READOUT$ function. Without a loss of generality, we treat static variables $\mathbf{X_g}$ as learnable parameters, optimized jointly with the model.

Finally, we map the graph-level representation to the output behaviors via a multi-layer neural network MLP_ω:

$$\mathbf{y}_t = MLP_\omega(\mathbf{r}_t). \tag{23}$$

We do not utilize time information in the decoder explicitly. Instead, we decode output behaviors $\mathbf{y}_t$ from graphs $\hat{G}_t$ for every time step t separately, since the time-dependent dynamics are already extracted by RNN_θ.

3.4. Differences from Existing Work

In this subsection, we briefly discuss the differences between our work and existing dynamic graph-learning methods based on optimization.

It is of great interest to learn the underlying graph structure of observed multivariate time series data. Chepuri et al. [54] propose a general learning scheme for noisy signals with a prior smoothness. Specifically, this learning approach finds denoised signals and a K-sparse graph structure from the data via optimizing:

$$\begin{aligned} \arg\min_{\hat{\mathbf{X}},\hat{\mathbf{A}}} &\|\hat{\mathbf{X}} - \mathbf{X}\|^2 + \gamma tr(\hat{\mathbf{X}}^T \mathcal{L}(\hat{\mathbf{A}})\hat{\mathbf{X}}) \\ \text{s.t.} &\sum_{i,j} \hat{e}_{ij} = K, \end{aligned} \tag{24}$$

where $\mathbf{X} \in \mathbb{R}^{N\times T}$ is the noisy data, $\hat{\mathbf{X}}$ is the denoised signals, $\mathcal{L}(\hat{\mathbf{A}})$ is the graph Laplacian matrix given the structure $\hat{\mathbf{A}}$, and γ and K are hyperparameters. To generalize this learning procedure to dynamic graphs learning to study DFC, Jiang et al. [42] find the solution based on:

$$\begin{aligned} \arg\min_{\{\hat{\mathbf{x}}_t\},\hat{\mathbf{A}}_t} &\sum_{t=1}^{T}\Big[\|\hat{\mathbf{x}}_t - \mathbf{x}_t\|^2 + \gamma_1 tr(\hat{\mathbf{x}}_t^T \mathcal{L}(\hat{\mathbf{A}}_t)\hat{\mathbf{x}}_t)\Big] + \gamma_2 \sum_{t=1}^{T-1} \|\hat{\mathbf{A}}_t - \hat{\mathbf{A}}_{t+1}\|_1 \\ \text{s.t.} &\sum_{i,j} \hat{e}_{ij_t} = K, \end{aligned} \tag{25}$$

where $\mathbf{x}_t \in \mathbb{R}^N$ is the noisy signals on time step t, and the additional constraint serves as temporal smoothness. One should notice two important differences between the optimization approaches and VDGLVM, namely, the modeling goal and the learning framework.

First, we define the graph structure on Gaussian latent variables $\mathbf{V}_t \in \mathbb{R}^{n\times d}$ inferred from the observed data $\mathbf{x}_t$, rather than on the data directly. The interest of the present work is not learning correlations of the observed signals, but finding probable dynamic communications between unseen latent variables underlying the observed data to interpret latent dynamics by LVMs. Since variables for DFC are less than the dimension of the data ($n < N$), we find the variables via a sequential VAE and perform link prediction on the variables to infer a graph from the data. This is why we introduce a dynamic graph as the latent variable and approximate the intractable posterior distribution with graph generative models. This is the most predominant difference.

Second, the learning framework is different. The optimization approach attempts to find the graph structure and eliminate additive Gaussian noise directly. To find the solution, the optimization framework requires varying forms of prior assumptions to constrain the solution space of the optimization problem. Consequently, it relies heavily on the formulation of the regularization terms, reflecting simplified prior knowledge of graph signals. Moreover, as indicated in Jiang et al. [42], computational complexity may be increased considerably with a more accurate modeling analysis. On the other hand, deep generative models can learn sophisticated mappings to find a more informative graph structure for its expressiveness and efficiency in a data-driven manner.

Nevertheless, the graph construction procedures are both based on the smoothness prior to the variables. $tr(\hat{\mathbf{X}}^T \mathcal{L}(\hat{\mathbf{A}})\hat{\mathbf{X}})$ is the Dirichlet energy of a graph, given the structure $\hat{\mathbf{A}}$, which encourages similar graph signals to be connected. It describes the distance between two signals as $e_{ij}\|\mathbf{x}_i - \mathbf{x}_j\|^2$, which is essentially consistent with our link-prediction scheme based on cosine similarity in Equation (19). Furthermore, the temporal smoothness priors are fundamentally consistent as well. While the optimization approach applies the assumption on the temporal graph property directly, VDGLVM merges the assumption in the RNN with an appropriate regularization on the parameters.

4. Experimental Setup

The validity and usefulness of our model depend on its performance when decoding behavior from neural spikes, and on whether the inferred representations preserve task-relevant structures. In this section, we first describe the datasets we use in the experiments. Then, we compare VDGLVM's performance in decoding behaviors against other typical methods to demonstrate its feasibility. We further analyze qualitatively the representations that VDGLVM learned. Finally, we show the effects of two important hyperparameters on VDGLVM.

4.1. Dataset Description

We used two real-world neural datasets from the Neural Latents Benchmark [55] to demonstrate the interpretability of VDGLVM. All datasets are publicly available.

The primary dataset we conducted experiments on is `Area2_Bump` [56]. The data were recorded as a macaque was performing a center–out reaching task. `Area2_Bump` includes hand kinematics as behaviors and corresponding neural population spikes of neurons from area 2 of the somatosensory cortex.

We also used `MC_Maze` [57], and its scaled versions, `MC_Maze_L`, `MC_Maze_M`, and `MC_Maze_S`, as the secondary datasets to compare model performance on different datasets. The datasets consist of neural population spikes and the associated hand kinematics when a monkey made reaches with an instructed delay to visually presented targets while avoiding the boundaries of a virtual maze. The neurons were from the primary motor and dorsal premotor cortices, which directly involve motor control.

We used the recorded hand position as target behaviors and built models to map the neural spikes to the behaviors. We binned the ensemble spike activities into 20-ms bins.

4.2. Model Configurations

We used two unidirectional GRU layers as the feature extractor. The space of the features was 128-dimensional. We applied dropout with $p = 0.5$ and layer normalization to avoid over-fitting. MLPs were implemented with two hidden layers. To further accelerate model training, we used ELU as the nonlinear activation function. In our model, we used three nodes for the graph, resulting in three dynamic edges. The static node features were in 128-dimensional space. We set the temperature $\tau = 0.03$ and VIB regularization multiplier $\beta = 1 \times 10^{-4}$ as defaults. The model was trained in a single Nvidia Titan RTX GPU for 256 epochs. We used the Adam optimizer with a learning rate of 0.001 to optimize the parameters. In our experiments, we split the dataset into 75% as the training set and 25% as the test set.

5. Results

5.1. Performance of Decoding Kinematics

We first wondered whether VDGLVM decodes behavior from neural spikes with a promised performance. We measured the performance by computing R^2 between the decoded behavior and normalized ground-truth behavior. To validate the expressiveness of VDGLVM, we assessed our model on different datasets and compared it against classic LVMs, including spike smoothing, GPFA, and SLDS. Since our model is based on nonlinear dynamical models and deep neural networks, it should significantly outperform these baselines. We also included a GRU-based RNN to compare, which is the backbone of VDGLVM without graphs and VIB. We trained these models in a supervised manner. The results are summarized in Table 1 (best results presented in bold font).

Table 1. Comparison of different model performances with R^2.

Model	`Area2_Bump`	`MC_Maze`	`MC_Maze_L`	`MC_Maze_M`	`MC_Maze_S`
Smoothing	0.595	0.642	0.597	0.546	0.481
GPFA	0.613	0.669	0.598	0.571	0.533
SLDS	0.762	0.812	0.792	0.772	0.667
RNN	0.901	0.896	0.862	0.794	0.710
VDGLVM	**0.927**	**0.912**	**0.898**	**0.818**	**0.794**

Spike smoothing, GPFA, and SLDS are essentially linear dynamical models. GPFA gives better results than spike smoothing because it accounts for the spikes across neurons and time in a probabilistic framework simultaneously. While both spike smoothing and GPFA benefit from the smoothness, SLDS uses a switching mechanism to approximate nonlinear dynamics with disparate linear dynamics and, thus, better models complex neural dynamics. RNN can capture more complex dynamics of nonlinear systems. Thus, we have the following observation from the table: in general, both RNN and VDGLVM outperform the methods based on linear models significantly. Another implicit advantage of RNN and VDGLVM is that they are end-to-end data-driven models, which are convenient to learn.

Furthermore, VDGLVM consistently achieves better performance than RNN. The predominant difference between VDGLVM and RNN is the design of the latent variable. This result proves the feasibility of utilizing graphs as the latent variable. It also implies that the introduced graph generative model and graph neural network may induce additional inductive bias. These modules ultimately improve expressiveness.

Deep neural networks tend to overfit. The results on `MC_Maze` and its scaled versions illustrate that both RNN and VDGLVM suffer from performance degeneration when samples are insufficient. However, VDGLVM is relatively alleviated, even though the model has extra trainable parameters for tackling graphs. Owing to the VIB framework, VDGLVM is less affected by noise and, thus, provides better generalization capability.

5.2. Latent Structure for Behavior Decoding

We are interested in the latent structure learned by VDGLVM. We include a three-dimensional latent space learned by RNN to compare, which represents existing LVMs based on a point process. Since we do not apply additional assumptions about the trajectories, the latent dynamics learned by VDGLVM should be similar to that by RNN. We visualize the dynamics after training VDGLVM and RNN on the `Area2_Bump` dataset with active trials in Figure 2.

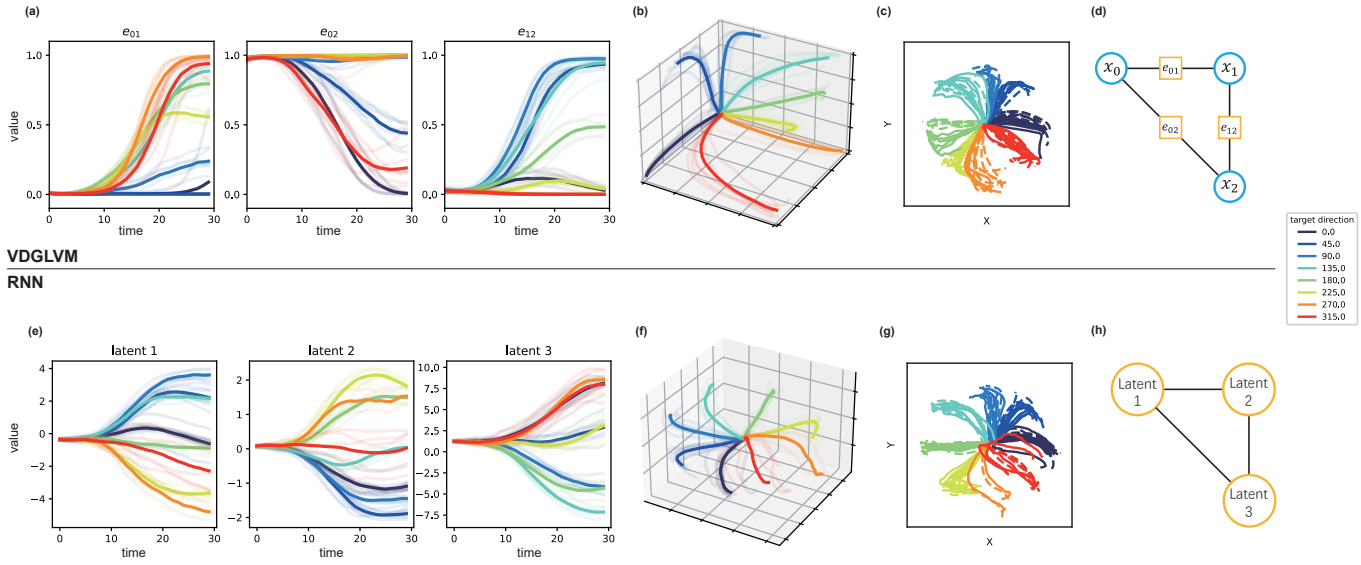

Figure 2. Inferred edge weights dynamics and behavior reconstructions by VDGLVM and inferred dynamics by RNN after training, where we use RNN to represent LVMs based on a point process. We visualize the edges evolving with time in (**a**). We combine them and show three-dimensional trajectories in (**b**), where each axis corresponds to an edge. We show behavior reconstructions for VDGLVM in (**c**), where the ground-truth trajectories are dash lines and decoding ones are solid lines. The graph with three dynamic edges and three nodes is shown in (**d**), where blue circles are static node variables and orange squares are dynamic edges. We show the three-dimensional trajectories inferred by RNN in (**e**,**f**). We show behavior reconstructions for RNN in (**g**). The equivalent graph structure of RNN is presented in (**h**), where orange circles are dynamic nodes.

Deep neural networks learn representations favorable to prediction. Since our main objective is decoding behavior from neural spikes, the actual dynamic functional connectivities may be further mapped to the representations favor for predictions. Therefore, as can be seen in Figure 2b,f, the latent trajectories learned by both VDGLVM and RNN preserve the task structure (reaching trajectories) in three-dimensional space.

However, the representations by RNN are only a low-dimensional embedding of the prediction, while the representations by VDGLVM contain a more informative structure. Looking at Figure 2a, we found that at least one dimension of the latent trajectories stays around 0 or 1 and provides no information about the dynamics. The results indicate that the dynamic relationships are local, governed by a subset of relationships. This sparsity agrees with the results of the constructed graph in previous work studying DFC [20]. The literature has shown that communication efficiency is one of the fundamental attributes that support neural function. In the language of complex networks, sparser connections between modules enable efficient inter-module integrations of information. We capture this by properly parameterizing the edge weights. The property is missed in most LVMs based on a point process, as shown in Figure 2e. Therefore, the dynamics graph inferred by our model provides a probable explanation, close to DFC, for latent dynamics by LVMs.

This illustrates that using a dynamic graph as the latent variable does associate latent dynamics with probable DFC. Meanwhile, as shown in Figure 2h, one may interpret the point process as a complete graph with node dynamics, where each node corresponds to one dimension of the trajectories. However, this trivial consideration does not have useful and practical insight into neural dynamics. On the contrary, as shown in Figure 2d, the graph by VDGLVM specifies an exact meaning for the dynamics as communications between nodes. One can conveniently associate the nodes with variables defined in the brain. Thus, we conclude that VDGLVM provides feasible neurophysiological interpretability to the inferred latent dynamics by existing LVMs.

5.3. Hyperparameter Tuning

We empirically analyzed two key hyperparameters by adjusting them and evaluating the resulting decoding performance.

5.3.1. β in VIB Objective Function

Our model is based on the VIB framework. It restricts the information flow from the observations $\mathbf{X}$ to the latent variable $\hat{\mathbf{G}}$ to minimize the VIB objective function. Given a proper β, this encourages the model to focus on information relevant to output $\mathbf{Y}$ in $\mathbf{X}$. Meanwhile, the model learns to drop useless information, such as noise. Therefore, compared with the vanilla auto-encoder, the VIB-based model improves the generalization performance. However, an inappropriately large β will be harmful to the model. Since VIB forces the latent coding towards a given prior, the bottleneck becomes so small that the information flow is blocked. With extremely large β values, the model learns to transform the distribution of input $p(\mathbf{X})$ to the prior $q(\hat{\mathbf{G}})$, which is completely uninformative for decoding.

The property of β has been extensively studied in the literature. We verified this by studying the relationship between β and model performance. We present the result in Figure 3a. We manually adjusted β from typical settings to illustrate its effects on the model performance revealed by R^2. As shown in the figure, when we use $\beta \geqslant 0$, increasing β improves the decoding performance against the baseline. The network attains the best when $10^{-5} \geqslant \beta \geqslant 10^{-4}$. Nevertheless, with a larger β, more and more discriminative information about input and output are blocked from input to latent coding. Thus, the model becomes seriously degenerated.

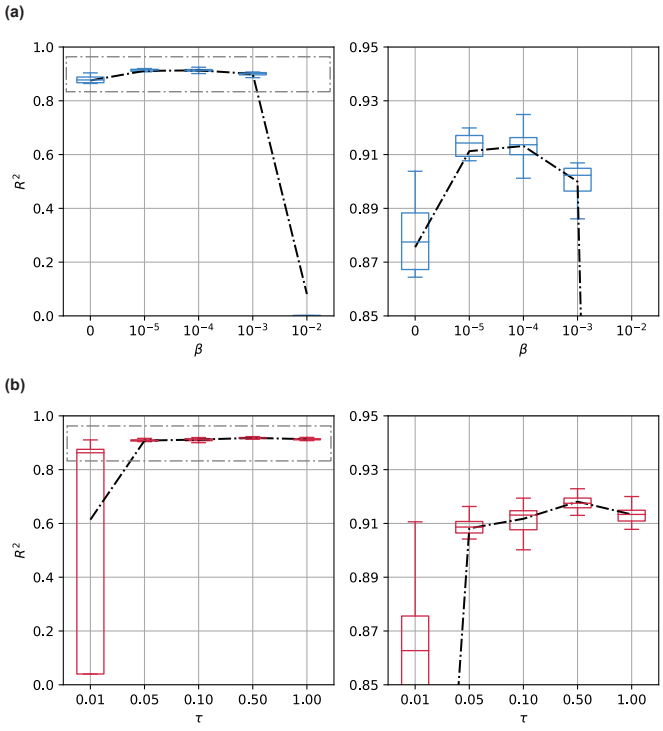

Figure 3. Decoding performance with the changing (**a**) β and (**b**) τ. Results are assessed with typical settings of the hyperparameters across 32 Monte Carlo simulations. The results are shown on the left side as two box plots, and the means are shown as lines. To present a more clear illustration, the results are zoomed in and shown on the right side.

5.3.2. τ in Graph Generation

Another hyperparameter significant for the model is the temperature τ. As discussed above, we only study $\tau \in (0, 1)$. A small τ makes the sigmoid function steeper; it tends

to be a step function and, thus, is more sensitive for the input around 0.5. More mathematically, with a smaller τ, the function has a higher Lipschitz constant. Since we directly involve this term in the propagation process, the Lipschitz constant of the model increases accordingly. While a smaller τ extracts more stable graph structures, an improperly small τ may make the model over-confident and, thus, less robust. We studied the association between τ and model performance. We present the result in Figure 3b. We found that the model may become unstable when $\tau < 0.01$, as the objective function becomes hard to optimize. Moreover, necessary dynamics are severely suppressed and thus insufficient to decode the target behavior. Therefore, the model tends to learn a trivial solution for the objective function.

6. Conclusions

Latent variable models (LVMs) for modeling neural data fail to determine the exact meaning of the inferred latent dynamics, while the literature has shown that dynamic functional connectivities (DFC) may be a possible mechanism of governing cognitive or perceptual tasks. In the present work, we propose Variational Dynamic Graph Latent Variable Model (VDGLVM), a representation learning model improving existing LVMs by interpreting the latent dynamics as DFC. To accomplish this, we design our model based on the variational information bottleneck and propose the use of a dynamic graph as the latent variable. Our model provides a probable explanation for latent dynamics captured by LVMs based on neurophysiology. In addition to the behavior dataset tested in the paper, we hope that the proposed model has the potential to apply to the analysis of neural activities in other perceptive and cognitive processes.

Author Contributions: Conceptualization, Y.H.; methodology, Y.H.; software, Y.H.; validation, Y.H.; formal analysis, Y.H.; investigation, Y.H.; resources, Z.Y.; data curation, Y.H.; writing—original draft preparation, Y.H.; writing—review and editing, Y.H. and Z.Y.; visualization, Y.H.; supervision, Z.Y.; project administration, Y.H.; funding acquisition, Z.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under grants 61836003, 61573150, and 61573152, as well as the International Cooperation open Project of the State Key Laboratory of Subtropical Building Science, South China University of Technology (2019ZA01).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. The datasets can be found in https://gui.dandiarchive.org/#/dandiset (accessed on 5 December 2021).

Acknowledgments: The authors acknowledge all of the anonymous reviewers for their constructive comments that helped to improve the quality of this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jun, J.J.; Steinmetz, N.A.; Siegle, J.H.; Denman, D.J.; Bauza, M.; Barbarits, B.; Lee, A.K.; Anastassiou, C.A.; Andrei, A.; Aydın, Ç.; et al. Fully integrated silicon probes for high-density recording of neural activity. *Nature* **2017**, *551*, 232–236. [CrossRef]
2. Hong, G.; Lieber, C.M. Novel electrode technologies for neural recordings. *Nat. Rev. Neurosci.* **2019**, *20*, 330–345. [CrossRef]
3. Steinmetz, N.A.; Aydin, C.; Lebedeva, A.; Okun, M.; Pachitariu, M.; Bauza, M.; Beau, M.; Bhagat, J.; Böhm, C.; Broux, M.; et al. Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *Science* **2021**, *372*, 6539. [CrossRef] [PubMed]
4. Sych, Y.; Chernysheva, M.; Sumanovski, L.T.; Helmchen, F. High-density multi-fiber photometry for studying large-scale brain circuit dynamics. *Nat. Methods* **2019**, *16*, 553–560. [CrossRef]
5. DiCarlo, J.J.; Zoccolan, D.; Rust, N.C. How does the brain solve visual object recognition? *Neuron* **2012**, *73*, 415–434. [CrossRef] [PubMed]
6. Stringer, C.; Pachitariu, M.; Steinmetz, N.; Carandini, M.; Harris, K.D. High-dimensional geometry of population responses in visual cortex. *Nature* **2019**, *571*, 361–365. [CrossRef]

7. Cunningham, J.P.; Byron, M.Y. Dimensionality reduction for large-scale neural recordings. *Nat. Neurosci.* **2014**, *17*, 1500–1509. [CrossRef]
8. Pandarinath, C.; O'Shea, D.J.; Collins, J.; Jozefowicz, R.; Stavisky, S.D.; Kao, J.C.; Trautmann, E.M.; Kaufman, M.T.; Ryu, S.I.; Hochberg, L.R.; et al. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nat. Methods* **2018**, *15*, 805–815. [CrossRef]
9. Keshtkaran, M.R.; Pandarinath, C. Enabling hyperparameter optimization in sequential autoencoders for spiking neural data. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 15937–15947.
10. Ye, J.; Pandarinath, C. Representation learning for neural population activity with Neural Data Transformers. *Neurons Behav. Data Anal. Theory* **2021**. [CrossRef]
11. Hurwitz, C.; Kudryashova, N.; Onken, A.; Hennig, M.H. Building population models for large-scale neural recordings: Opportunities and pitfalls. *arXiv* **2021**, arXiv:2102.01807.
12. Gallego, J.A.; Perich, M.G.; Naufel, S.N.; Ethier, C.; Solla, S.A.; Miller, L.E. Cortical population activity within a preserved neural manifold underlies multiple motor behaviors. *Nat. Commun.* **2018**, *9*, 1–13. [CrossRef]
13. Perich, M.G.; Gallego, J.A.; Miller, L.E. A neural population mechanism for rapid learning. *Neuron* **2018**, *100*, 964–976. [CrossRef] [PubMed]
14. Degenhart, A.D.; Bishop, W.E.; Oby, E.R.; Tyler-Kabara, E.C.; Chase, S.M.; Batista, A.P.; Byron, M.Y. Stabilization of a brain–computer interface via the alignment of low-dimensional spaces of neural activity. *Nat. Biomed. Eng.* **2020**, *4*, 672–685. [CrossRef] [PubMed]
15. Bassett, D.S.; Sporns, O. Network neuroscience. *Nat. Neurosci.* **2017**, *20*, 353–364. [CrossRef]
16. Bassett, D.S.; Zurn, P.; Gold, J.I. On the nature and use of models in network neuroscience. *Nat. Rev. Neurosci.* **2018**, *19*, 566–578. [CrossRef] [PubMed]
17. Breakspear, M. "Dynamic" connectivity in neural systems. *Neuroinformatics* **2004**, *2*, 205–224. [CrossRef]
18. Hutchison, R.M.; Womelsdorf, T.; Allen, E.A.; Bandettini, P.A.; Calhoun, V.D.; Corbetta, M.; Della Penna, S.; Duyn, J.H.; Glover, G.H.; Gonzalez-Castillo, J.; et al. Dynamic functional connectivity: Promise, issues, and interpretations. *Neuroimage* **2013**, *80*, 360–378. [CrossRef] [PubMed]
19. Gonzalez-Castillo, J.; Bandettini, P.A. Task-based dynamic functional connectivity: Recent findings and open questions. *Neuroimage* **2018**, *180*, 526–533. [CrossRef]
20. Avena-Koenigsberger, A.; Misic, B.; Sporns, O. Communication dynamics in complex brain networks. *Nat. Rev. Neurosci.* **2018**, *19*, 17–33. [CrossRef]
21. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Philip, S.Y. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 4–24. [CrossRef]
22. Alemi, A.A.; Fischer, I.; Dillon, V.J.; Murphy, K. Deep Variational Information Bottleneck. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
23. Byron, M.Y.; Cunningham, J.P.; Santhanam, G.; Ryu, S.I.; Shenoy, K.V.; Sahani, M. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. In Proceedings of the Advances in Neural Information Processing Systems 22 (NIPS 2009), Vancouver, BC, Canada, 7–10 December 2009; pp. 1881–1888.
24. Zhao, Y.; Park, I.M. Variational latent gaussian process for recovering single-trial dynamics from population spike trains. *Neural Comput.* **2017**, *29*, 1293–1316. [CrossRef] [PubMed]
25. Wu, A.; Roy, N.A.; Keeley, S.; Pillow, J.W. Gaussian process based nonlinear latent structure discovery in multivariate spike train data. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 3499–3508.
26. She, Q.; Wu, A. Neural dynamics discovery via gaussian process recurrent neural networks. In Proceedings of the 35th Uncertainty in Artificial Intelligence Conference, Virtual online, 3–6 August 2020; pp. 454–464.
27. Liu, D.; Lengyel, M. A universal probabilistic spike count model reveals ongoing modulation of neural variability. In Proceedings of the Thirty-Fifth Conference on Neural Information Processing Systems, Online, 6–14 December 2021.
28. Macke, J.H.; Buesing, L.; Cunningham, J.P.; Yu, B.M.; Shenoy, K.V.; Sahani, M. Empirical models of spiking in neural populations. In Proceedings of the Advances in Neural Information Processing Systems 24: 25th Conference on Neural Information Processing Systems (NIPS 2011), Granada, Spain, 12–15 December 2011; pp. 1350–1358.
29. Gao, Y.; Archer, E.W.; Paninski, L.; Cunningham, J.P. Linear dynamical neural population models through nonlinear embeddings. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 163–171.
30. Liu, R.; Azabou, M.; Dabagia, M.; Lin, C.H.; Gheshlaghi Azar, M.; Hengen, K.; Valko, M.; Dyer, E. Drop, Swap, and Generate: A Self-Supervised Approach for Generating Neural Activity. In Proceedings of the Advances in Neural Information Processing Systems, Online, 6–14 December 2021; Volume 34.
31. Zhou, D.; Wei, X.X. Learning identifiable and interpretable latent models of high-dimensional neural activity using pi-VAE. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 7234–7247.
32. Bastos, A.M.; Schoffelen, J.M. A tutorial review of functional connectivity analysis methods and their interpretational pitfalls. *Front. Syst. Neurosci.* **2016**, *9*, 175. [CrossRef]
33. Handwerker, D.A.; Roopchansingh, V.; Gonzalez-Castillo, J.; Bandettini, P.A. Periodic changes in fMRI connectivity. *Neuroimage* **2012**, *63*, 1712–1719. [CrossRef]

34. Thompson, G.J.; Magnuson, M.E.; Merritt, M.D.; Schwarb, H.; Pan, W.J.; McKinley, A.; Tripp, L.D.; Schumacher, E.H.; Keilholz, S.D. Short-time windows of correlation between large-scale functional brain networks predict vigilance intraindividually and interindividually. *Hum. Brain Mapp.* **2013**, *34*, 3280–3298. [CrossRef]
35. Zalesky, A.; Fornito, A.; Cocchi, L.; Gollo, L.L.; Breakspear, M. Time-resolved resting-state brain networks. *Proc. Natl. Acad. Sci. USA* **2014**, *111*, 10341–10346. [CrossRef] [PubMed]
36. Hindriks, R.; Adhikari, M.H.; Murayama, Y.; Ganzetti, M.; Mantini, D.; Logothetis, N.K.; Deco, G. Can sliding-window correlations reveal dynamic functional connectivity in resting-state fMRI? *Neuroimage* **2016**, *127*, 242–256. [CrossRef] [PubMed]
37. Granger, C.W. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica* **1969**, *37*, 424–438. [CrossRef]
38. Dhamala, M.; Rangarajan, G.; Ding, M. Analyzing information flow in brain networks with nonparametric Granger causality. *Neuroimage* **2008**, *41*, 354–362. [CrossRef] [PubMed]
39. West, T.O.; Halliday, D.M.; Bressler, S.L.; Farmer, S.F.; Litvak, V. Measuring directed functional connectivity using non-parametric directionality analysis: Validation and comparison with non-parametric Granger Causality. *NeuroImage* **2020**, *218*, 116796. [CrossRef] [PubMed]
40. Fallahi, A.; Pooyan, M.; Lotfi, N.; Baniasad, F.; Tapak, L.; Mohammadi-Mobarakeh, N.; Hashemi-Fesharaki, S.S.; Mehvari-Habibabadi, J.; Ay, M.R.; Nazem-Zadeh, M.R. Dynamic functional connectivity in temporal lobe epilepsy: A graph theoretical and machine learning approach. *Neurol. Sci.* **2021**, *42*, 2379–2390. [CrossRef]
41. Qiao, C.; Hu, X.Y.; Xiao, L.; Calhoun, V.D.; Wang, Y.P. A deep autoencoder with sparse and graph Laplacian regularization for characterizing dynamic functional connectivity during brain development. *Neurocomputing* **2021**, *456*, 97–108. [CrossRef]
42. Jiang, B.; Huang, Y.; Panahi, A.; Yu, Y.; Krim, H.; Smith, S.L. Dynamic Graph Learning: A Structure-Driven Approach. *Mathematics* **2021**, *9*, 168. [CrossRef]
43. Dimitriadis, S.I.; Laskaris, N.A.; Del Rio-Portilla, Y.; Koudounis, G.C. Characterizing dynamic functional connectivity across sleep stages from EEG. *Brain Topogr.* **2009**, *22*, 119–133. [CrossRef]
44. Allen, E.; Damaraju, E.; Eichele, T.; Wu, L.; Calhoun, V.D. EEG signatures of dynamic functional network connectivity states. *Brain Topogr.* **2018**, *31*, 101–116. [CrossRef]
45. Gori, M.; Monfardini, G.; Scarselli, F. A new model for learning in graph domains. In Proceedings of the 2005 IEEE International Joint Conference on Neural Networks, Montreal, QC, Canada, 31 July–4 August 2005; Volume 2, pp. 729–734.
46. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The graph neural network model. *IEEE Trans. Neural Netw.* **2008**, *20*, 61–80. [CrossRef]
47. Jin, W.; Barzilay, R.; Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 2323–2332.
48. Kawahara, J.; Brown, C.J.; Miller, S.P.; Booth, B.G.; Chau, V.; Grunau, R.E.; Zwicker, J.G.; Hamarneh, G. BrainNetCNN: Convolutional neural networks for brain networks; towards predicting neurodevelopment. *NeuroImage* **2017**, *146*, 1038–1049. [CrossRef] [PubMed]
49. Gilmer, J.; Schoenholz, S.S.; Riley, P.F.; Vinyals, O.; Dahl, G.E. Neural message passing for quantum chemistry. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 1263–1272.
50. Tishby, N.; Zaslavsky, N. Deep learning and the information bottleneck principle. In Proceedings of the 2015 IEEE Information Theory Workshop (ITW), Jerusalem, Israel, 26 April–1 May 2015; pp. 1–5.
51. Shwartz-Ziv, R.; Tishby, N. Opening the black box of deep neural networks via information. *arXiv* **2017**, arXiv:1703.00810.
52. Gilbert, E.N. Random graphs. *Ann. Math. Stat.* **1959**, *30*, 1141–1144. [CrossRef]
53. Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How Powerful are Graph Neural Networks? In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
54. Chepuri, S.P.; Liu, S.; Leus, G.; Hero, A.O. Learning sparse graphs under smoothness prior. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 5–9 March 2017; pp. 6508–6512.
55. Pei, F.C.; Ye, J.; Zoltowski, D.M.; Wu, A.; Chowdhury, R.H.; Sohn, H.; O'Doherty, J.E.; Shenoy, K.V.; Kaufman, M.; Churchland, M.M.; et al. Neural Latents Benchmark '21: Evaluating latent variable models of neural population activity. In Proceedings of the Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2), Virtual, 6–14 December 2021.
56. Chowdhury, R.H.; Glaser, J.I.; Miller, L.E. Area 2 of primary somatosensory cortex encodes kinematics of the whole arm. *eLife* **2020**, *9*, e48198. [CrossRef] [PubMed]
57. Churchland, M.M.; Cunningham, J.P.; Kaufman, M.T.; Ryu, S.I.; Shenoy, K.V. Cortical preparatory activity: Representation of movement or first cog in a dynamical machine? *Neuron* **2010**, *68*, 387–400. [CrossRef] [PubMed]