Bubblewrap: Online tiling and real-time flow prediction on neural manifolds

Anonymous Author(s)
Affiliation
Address
email

Abstract

While most classic studies of function in experimental neuroscience have focused on the coding properties of individual neurons, recent developments in recording technologies have resulted in an increasing emphasis on the dynamics of neural populations. This has given rise to a wide variety of models for analyzing population activity in relation to experimental variables, but direct testing of many neural population hypotheses requires intervening in the system based on current neural state, necessitating models capable of inferring neural state online. Existing approaches, primarily based on dynamical systems, require strong parametric assumptions that are easily violated in the noise-dominated regime and do not scale well to the thousands of data channels in modern experiments. To address this problem, we propose a method that combines fast, stable dimensionality reduction with a soft tiling of the resulting neural manifold, allowing dynamics to be approximated as a probability flow between tiles. This method can be fit efficiently using online expectation maximization, scales to tens of thousands of tiles, and outperforms existing methods when dynamics are noise-dominated or feature multi-modal transition probabilities. The resulting model can be trained at kiloHertz data rates, produces accurate approximations of neural dynamics within minutes, and generates predictions on submillisecond time scales, fast enough to serve as a component of closed-loop causal experiments.

1 Introduction

Systems neuroscience is in the midst of a data explosion. Advances in microscopy [1, 2] and probe technology [3, 4, 5] have made it possible to record thousands of neurons simultaneously in behaving animals. At the same time, growing interest in naturalistic behaviors has increased both the volume and complexity of jointly recorded behavioral data. On the neural side, this has resulted in a host of new modeling and analysis approaches that aim to match the complexity of these data, typically using artificial neural network models as proxies for biological neural computation [6, 7, 8].

At the same time, this increase in data volume has resulted in increasing emphasis on methods for dimensionality reduction [9] and a focus on neural populations in preference to the coding properties of individual neurons [10]. However, given the complexity of neural dynamics, it remains difficult to anticipate what experimental conditions will be needed to test population hypotheses in *post hoc* analyses, complicating experimental design and reducing power. Conversely, adaptive experiments, those in which the conditions tested change in response to incoming data, have been used in neuroscience to optimize stimuli for experimental testing [11, 12, 13, 14], in closed-loop designs [15, 16, 17], and even to scale up holographic photostimulation for inferring functional connectivity in large circuits [18].

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Yet, despite their promise, adaptive methods are rarely applied in practice for two reasons: First, although efficient online methods for dimensionality reduction exist [19, 20, 21, 22, 23], these methods do not typically identify *stable* dimensions to allow low-dimensional representations of data to be compared across time points. That is, when the spectral properties of the data are changing in time, methods like incremental SVD may be projecting the data into an unstable basis, rendering these projections unsuitable as inputs to further modeling. Second, while many methods based on the dynamical systems approach exist [6, 24, 25, 26, 27, 28, 29] — even online approaches [30, 31, 16, 32] — these approaches typically assume a system with lawful dynamics perturbed by Gaussian noise. However, many neural systems of interest are noise-dominated, with multimodal transition kernels between states.

Here, we propose an alternative to the linear systems approach that combines a fast, stable, online dimensionality reduction with a semiparametric tiling of the low-dimensional neural manifold. This tiling introduces a discretization of the neural state space that allows dynamics to be modeled as a Hidden Markov Model defined by a sparse transition graph. The entire model, which we call "Bubblewrap," can be learned online using a simple EM algorithm and handles tilings and graphs of up to thousands of cells at kiloHertz data rates. Most importantly, this model outperforms methods based on dynamical systems in high-noise regimes when the dynamics are more diffusion-like. Training can be performed at a low, fixed latency ≈10ms using a GPU, while a cached copy of the model in main memory is capable of predicting upcoming states at <1ms latency. As a result, Bubblewrap offers a flexible method for online learning of neural dynamics performant enough to serve as a prediction engine for causal feedback experiments.

2 Stable subspaces from streaming SVD

As detailed above, one of the most pressing issues in online neural modeling is dealing with the increasingly large dimensionality of collected data — hundreds of channels per Neuropixels probe [4, 5], tens of thousands of pixels for calcium imaging. However, as theoretical work has shown [33, 34], true neural dynamics often lie on a low-dimensional manifold, so that population activity can be accurately captured by analyzing only a few variables.

Here, we combine two approaches to data reduction: In the first stage, we use sparse random projections to reduce dimensionality from an initial d dimensions (thousands) to n (a few hundred) [35, 36]. By simple scaling, for a fixed budget of N cells in our manifold tiling, we expect density (and thus predictive accuracy) to scale as $N^{\frac{1}{n}}$ in dimension n, and so we desire n to be as small as possible. However, by the Johnson-Lindenstrauss Lemma [37, 36], when reducing from d to n dimensions, the distance between vectors u_* and v_* in the reduced space is related to the distance between their original versions u and v by

$$(1-\varepsilon)\|u-v\|^2 \leq \|u_* - v_*\|^2 \leq (1+\varepsilon)\|u-v\|^2 \tag{1}$$

with probability $1-\delta$ if $n > \mathcal{O}(\log(1/\delta)/\varepsilon^2)$. Unfortunately, even for $\varepsilon \sim 0.1$ (10% relative error), the required n may be quite large, making this inappropriate for reducing to the very small numbers of effective dimensions characterizing neural datasets.

Thus, in the second stage, we reduce from $n \sim \mathcal{O}(100)$ to $k \sim \mathcal{O}(10)$ dimensions using a streaming singular value decomposition. This method is based on the incremental block update method of [20, 22] with an important difference: While the block update method aims to return the top-k SVD at every time point, the directions of the singular vectors may be quite variable during the course of an experiment (Figure 1d–h), which implies an unstable representation of the neural manifold. However, as we show below, the top-k *subspace* spanned by these vectors stabilizes in seconds on typical neural datasets and remains so throughout the experiment. Therefore, by selecting a stable basis (instead of the singular vector basis) for the top-k subspace, we preserve the same information while ensuring a stable representation of the data for subsequent model fitting.

More specifically, let $\mathbf{x}_t \in \mathbb{R}^n$ be a vector of input data after random projections. In our streaming setup, these are processed b samples at a time, with $b = 1$ reasonable for slower methods like calcium imaging and $b = 40$ more appropriate for electrophysiological sampling rates of ∼20kHz. Then, if the data matrix X has dimension $n \times T$, adding columns over time, the incremental method of [20, 22] produces at each time step a factorization $X = QRW^\top$, where the columns of the orthogonal matrices Q and W span the left and right top-k singular subspaces, respectively. If the matrix R were

Algorithm 1 Stable Streaming SVD (ssSVD)

1: **Given:** Initial data X_0, decay parameter $\alpha \in (0, 1]$
2: **Initialize:** QR Factorization: $X_0 = Q_0 R_0$
3:
4: **for** $t = 1 \ldots$ **do**
5: Fetch b new columns of data, X_+
6: $C \leftarrow Q_{t-1}^\top X_+, \quad X_\perp \leftarrow X_+ - Q_{t-1}C, \quad Q_\perp, R_\perp \leftarrow \texttt{QR}(X_\perp)$ ▷ Gram-Schmidt
7: $\hat{Q} \leftarrow [Q_{t-1} \quad Q_\perp], \quad \hat{R} \leftarrow \begin{bmatrix} R_{t-1} & C \\ 0 & R_\perp \end{bmatrix}$ ▷ QR of augmented data
8: $U, \Sigma, V \leftarrow \texttt{SVD}(\hat{R})$
9: $\Sigma \leftarrow \alpha\Sigma$ ▷ Discount old data
10: $M \leftarrow Q_{t-1}^\top \hat{Q} U_1$ ▷ U_1 contains the first k columns of U
11: $\tilde{U}, \tilde{\Sigma}, \tilde{V} \leftarrow \texttt{SVD}(M), \quad T \leftarrow \tilde{U}\tilde{V}^\top$ ▷ Orthogonal Procrustes: $\min_T \|\hat{Q}U_1T^\top - Q_{t-1}\|_F$
12: $Q_t \leftarrow \hat{Q}U_1T^\top$ ▷ Update left subspace basis
13: **end for**

diagonal, this would be equivalent to the SVD. In the incremental algorithm, R is augmented at each timestep based on new data to form $\hat{R}$, which is block diagonalized via an orthogonal matrix and truncated to the top-k subspace, allowing for an exact reduced-rank SVD (Appendix A).

However, as reviewed in [20, 22], since there are multiple choices of basis Q for for the top-k singular subspace, there are likewise multiple choices of block diagonalization for $\hat{R}$. In [20, 22], the form of this operation is chosen for computational efficiency. But an equally valid option is to select the orthogonal matrix that minimizes the change in the singular subspace basis Q from one timestep to the next:

$$\min \|Q_t - Q_{t-1}\|_F = \min_T \|\hat{Q}U_1T^\top - Q_{t-1}\|_F, \tag{2}$$

where $\hat{Q}$ is an augmented basis for the top-$(k + b)$ singular subspace, U_1 contains the first k left singular vectors of $\hat{R}$, and T is an orthogonal matrix (Appendix A). This minimization is known as the Orthogonal Procrustes problem and has a well-known solution [38]: $T = \tilde{U}\tilde{V}^\top$, where $\tilde{U}$ and $\tilde{V}$ are the left and right singular vectors, respectively, of $M \equiv Q_{t-1}^\top \hat{Q} U_1$. This procedure is summarized in Algorithm 1. There, lines 1–8 follow [20, 22], while lines 10 and 11 perform the Orthogonal Procrustes procedure. Line 9 serves as a leak term that discounts past data as in [39].

Figures 1a-c illustrates the performance of the two-stage dimension reduction algorithm for a case of $d = 10^4$ randomly generated Gaussian data. While ssSVD yields minimal distortion (due to truncation of the spectrum to $k = 6$), random projections require $k \sim \mathcal{O}(100)$ to achieve the same result (Figure 1a). By contrast, random projections are much faster (Figure 1b). Thus, we can trade off distortion against time by adjusting n, the number of intermediate dimensions. As Figure 1c shows, the optimal tradeoff occurs somewhere around $n = 200$ for this example.

Figures 1d-i show results for neural data from recorded from monkey motor cortex [26] in a cued reach task. While projection of the data into the basis defined by streaming SVD remains unstable early in data collection (top), the ssSVD representation is nearly equivalent to the full offline result after only a few trials (≈15s, middle). This is due to the fact that, in all data sets we examined, the top-k SVD *subspace* was identified extremely quickly; ssSVD simply ensures the choice of a stable basis for that subspace.

3 Bubblewrap: a soft manifold tiling for online modeling

As reviewed above, most neural population modeling approaches are based on the dynamical systems framework, assuming a lawful equation of motion corrupted by noise. However, for animals engaged in task-free natural behavior [40, 41, 42], trajectories are likely to be sufficiently complex that simple dynamical models fail. For instance, state space models with Gaussian noise necessarily produce unimodal transition probabilities centered around the mean prediction, while neural trajectories may exhibit multimodal distributions beginning at the same system state. By contrast, we pursue an

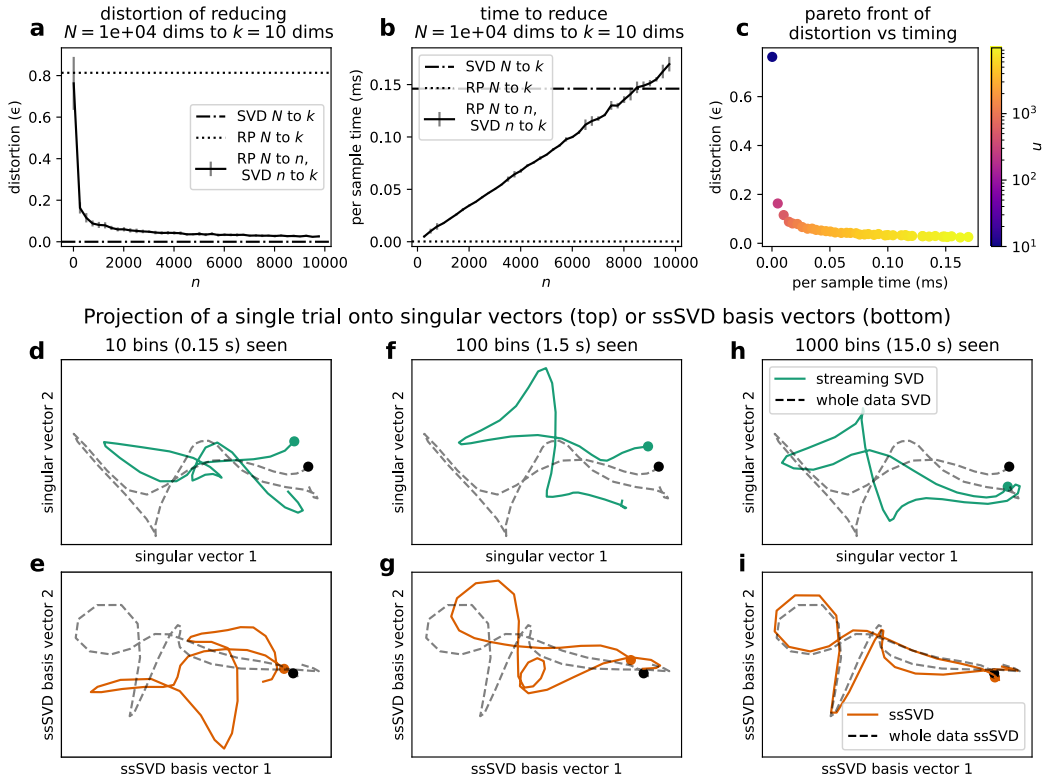

Figure 1: **Timing and stability of two-stage dimension reduction. a)** Distortion (ε) as a function of number of dimensions retained (n) for both sparse random projections and ssSVD on random Gaussian data with batch size $b = 1000$. **b)** Time required for the dimensionality reduction in **(a)**, amortized for batch size. While random projections are extremely efficient, ssSVD time costs grow with the number of dimensions retained. **c)** Pareto front for the time-distortion tradeoff of random projections followed by ssSVD. Color indicates n, the number of dimensions retained by random projections. **d–f)** Embedding of a single trial (green line) into the basis defined by streaming SVD for different amounts of data seen. Dotted line indicates the same trial embedded using SVD on the full data set. Rapid changes in estimates of singular vectors early on lead to an unstable representation. **g–i)** Same trial and conventions as **(d–f)** for the ssSVD embedding. Dotted lines in the two rows are the same curve in different projections.

alternative method that trades some accuracy in estimating instantaneous system state for flexibility in modeling the manifold describing neural activity.

Our approach is to produce a soft tiling of the neural manifold in the form of a Gaussian mixture model (GMM). We then approximate the transitions between cells of this tiling via a Hidden Markov Model (HMM), which allows us to capture multimodal probability flows. As the number of components in the GMM is increased, the model produces an increasingly finer-grained description of dynamics that assumes neither an underlying state space model nor a particular distribution of noise.

More specifically, let x_t be the low-dimensional system state and let $z_t \in 1 \dots N$ index the mixture components of the GMM. Then we have for the dynamics

$$p(z_t = j | z_{t-1} = i) = A_{ij} \quad p(x_t | z_t) = \mathcal{N}(\mu_{z_t}, \Sigma_{z_t}) \quad p(\mu_j, \Sigma_j) = \mathrm{NIW}(\mu_{0j}, \lambda_j, \Psi_j, \nu_j), \tag{3}$$

where we have assumed Normal-inverse-Wishart priors on the parameters of the mixture components. Given its exponential family form and the conjugacy of the priors, online expectation maximization updates are available in closed form [43, 44, 45] for each new datum, though we opt, as in [44] for a

gradient-based optimization of an estimate of the evidence lower bound

$$\begin{aligned}\hat{Q}(A,\mu,\Sigma) = &\sum_{ij}(\hat{N}_{ij}(T)+\beta_{ij}-1)\log A_{ij} + \sum_j(\hat{S}_{1j}(T)+\lambda_j\mu_{0j})^\top\Sigma_j^{-1}\mu_j \\ &-\frac{1}{2}\sum_j \mathrm{tr}((\Psi_j+\hat{S}_{2j}(T)+\lambda_j\mu_{0j}\mu_{0j}^\top+(\lambda_j+\hat{n}_j(T))\mu_j\mu_j^\top)\Sigma_j^{-1}) \\ &-\frac{1}{2}\sum_j(\nu_j+\hat{n}_j(T)+d+2)\log\det\Sigma_j\end{aligned} \tag{4}$$

with accumulating (estimated) sufficient statistics

$$\begin{aligned}\alpha_j(t) &= \sum_i \alpha_i(t-1)\Gamma_{ij}(t) & \hat{N}_{ij}(t) &= (1-\varepsilon_t)\hat{N}_{ij}(t-1)+\alpha_i(t-1)\Gamma_{ij}(t) \\ \hat{n}_j(t) &= \sum_i \hat{N}_{ij}(t) & \hat{S}_{1j}(t) &= (1-\varepsilon_t)\hat{S}_{1j}(t-1)+\alpha_j(t)x_t \\ & & \hat{S}_{2j}(t) &= (1-\varepsilon_t)\hat{S}_{2j}(t-1)+\alpha_j(t)x_tx_t^\top\end{aligned} \tag{5}$$

where $\alpha_j(t) = p(z_t = j|x_{1:t})$ is the filtered posterior, $\Gamma_{ij}(t)$ is the update matrix from the forward algorithm [43], and ε_t is a forgetting term that discounts previous data. Note that even for $\varepsilon = 0$, $\hat{Q}$ is only an estimate of the evidence lower bound because the sufficient statistics are calculated using $\alpha(t)$ and not the posterior over all observed data.

In setting Normal-Inverse-Wishart priors over the Gaussian mixture components, we take an empirical Bayes approach by setting prior means μ_{0j} to the current estimate of the data center of mass and prior covariance parameters Ψ_j to $N^{-\frac{2}{k}}$ times the current estimate of the data covariance (Appendix B). For initializing the model we use a small data buffer $M \sim \mathcal{O}(10)$. We chose effective observation numbers $(\lambda, \nu) = 10^{-3}$ and trained this model to maximize $\hat{Q}(A, \mu, \Sigma)$ using Adam [46], enforcing parameter constraints by replacing them with unconstrained variables a_{ij} and lower triangular L_j with positive diagonal: $A_{ij} = \exp(a_{ij})/\sum_j \exp(a_{ij})$, $\Lambda_j = \Sigma_j^{-1} = L_jL_j^\top$.

Finally, in order to prevent the model from becoming stuck in local minima and to encourage more effective tilings, we implemented two additional heuristics as part of Bubblewrap: First, whenever a new observation was highly unlikely to be in any existing mixture component ($\log p(x_t|z_t) < \theta_n$ for all z_t), we teleported a node at this data point by assigning $\alpha_J(t) = 1$ for an unused index J. For initial learning this results in a "breadcrumbing" approach where nodes are placed at the locations of each new observed data. Second, when the number of active nodes was equal to our total node budget N, we chose to reclaim the node with the lowest value of $\hat{n}(t)$ and zeroed out its existing sufficient statistics before teleporting it to a new location. In practice, these heuristics substantially improved performance, especially early in training (Appendix D). The full algorithm is summarized in Algorithm 2.

4 Experiments

We demonstrated the performance of Bubblewrap on both simulated non-linear dynamical systems and experimental neural data. We compared these results to two existing online learning models for neural data, both of which are based on dynamical systems [30, 32]. To simulate low-dimensional systems, we generated noisy trajectories from a two-dimensional Van der Pol oscillator and a three-dimensional Lorenz attractor. For experimental data, we used four publicly available datasets from a range of applications: 1) trial-based spiking data recorded from primary motor cortex in monkeys performing a reach task [47, 48] preprocessed by performing online jPCA [48]; 2) continuous video data and 3) trial-based wide-field calcium imaging from a rodent decision-making task [49, 50]; 4) high-throughput Neuropixels data [51, 52]. For each data set, we gave each model the same data as reduced by random projections and ssSVD and calculated log predictive probability in the low-dimensional space (Appendix C). We quantified overall model performance by taking the mean log predictive probability over the last half of each data set (Table 1). For Bubblewrap, which is focused on coarser transitions between tiles, we also report the entropy of predicted transitions between tiles. Additional detailed experimental results and benchmarking of our GPU implementation in JAX [53]

Algorithm 2 Bubblewrap

Given: Hyperparameters $\lambda_j, \nu_j, \beta_t$, forgetting rate ε_t, teleport threshold θ, step size δ, initial data buffer M

Initialize with $x_1 : x_M$**:** $\mu_j \leftarrow \bar{\mu}$, $\Sigma_j \leftarrow \bar{\Sigma}$, $a_{ij} \leftarrow \frac{1}{N}$, $\alpha_j \leftarrow \lambda_j \frac{1}{N}$.

for $t = 1 \ldots$ **do**

  Observe new data point x_t.

  if $\log p(x_t|z_t) < \theta \; \forall z_t$ **then** ▷ Teleport

    $\mu_J = x_t, \alpha_J(t) = 1$ for $J = \arg\min_j \hat{n}_j(t)$

  end if

  Calculate $\Gamma_{ij}(t)$ via forward filtering [43].

  Update sufficient statistics via (5). ▷ E step

  $\bar{\mu} \leftarrow \frac{\sum_j \hat{S}_{1j}}{\sum_j \hat{n}_j}, \overline{\Sigma} \leftarrow \frac{\sum_j \hat{S}_{2j}}{\sum_j \hat{n}_j} - \bar{\mu}\bar{\mu}^T$ ▷ Global mean and covariance update

  $\eta_j \sim \mathcal{N}(0, s^2), \mu_{0j} \leftarrow a\mu_{0j} + (1-a)\bar{x} + \eta_j, \Psi_j \leftarrow \frac{\overline{\Sigma}}{N^{\frac{2}{k}}}$ ▷ Update priors (Appendix B)

  Perform gradient-based update of $\hat{Q}(A, \mu, L)$ (4) ▷ M step

end for

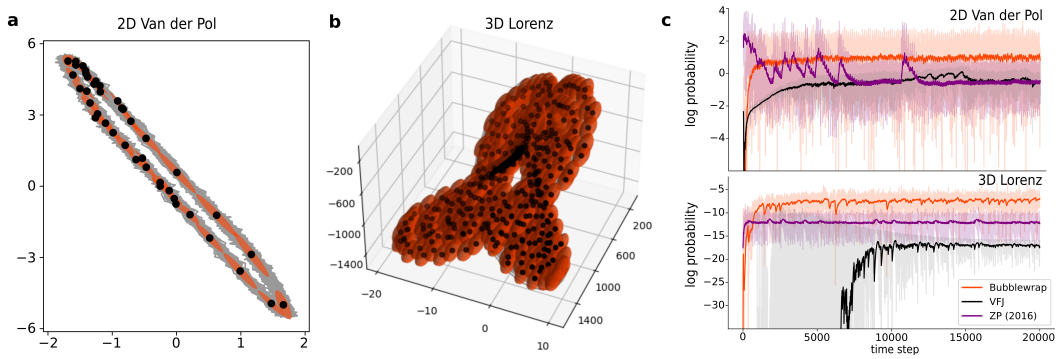

Figure 2: **Modeling of low-dimensional dynamical systems.** **a)** Bubblewrap end tiling of a 2D Van der Pol oscillator (data in gray; 5% noise case corresponding to line 1 of Table 1). Tile center locations are in black with covariance 'bubbles' for 3 sigma in orange. **b)** Bubblewrap end tiling of a 3D Lorenz attractor (5% noise), where tiles are plotted similarly to (a). **c)** Log predictive probability across all timepoints for each comparative model for the 2D Van der Pol, 0.05 case (top) and for the 3D Lorenz, 0.05 case (bottom).

are in Appendix D. We compared performance of our algorithm against both [30] (using our own implementation in JAX) and Variational Joint Filtering [32] (using the authors' own implementation).

When tested on low-dimensional dynamical systems, Bubblewrap successfully learned tilings of both neural manifolds, outperforming VJF [32] on both datasets (Figure 2a,b) while it comparable to the algorithm of [30] on one of the 2D (but neither of the 3D) cases (Figure 2). This is surprising, since both comparison methods assume an underlying dynamical system and attempt to predict differences between individual data points, while Bubblewrap only attempts to localize data to within a coarse area of the manifold.

We next tested each algorithm on more complex data collected from neuroscience experiments. These data exhibited a variety of structure, from organized rotations (Figure 3a) to rapid transitions between noise clusters (Figure 3b) to slow dynamics (Figure 3c). In each case, Bubblewrap learned a tiling of the data that allowed it to equal or outperform state predictions from the comparison algorithms (Figure 3d–f, blue). In some cases, as with the mouse dataset, the algorithm of [30] produced predictions for x_t by degenerating to a random walk model (Table 1 marked with *; Appendix D). Regardless, Bubblewrap's tiling generated transition predictions with entropies far below those of a random walk across tiles (Figure 3d–f, green), indicating it successfully identified coarse structure, even in challenging data sets. Thus, even though these data are noise-dominated and lack much of the

Table 1: Model comparison results as mean $\pm$ standard deviation of the log predictive probability over the last half of the dataset.

	Log predictive probability		
Dataset	Bubblewrap	VJF [32]	ZP (2016) [30]
2D Van der Pol, 0.05	0.965 ± 1.123	-0.338 ± 0.427	-0.471 ± 0.857
2D Van der Pol, 0.20	-1.088 ± 1.184	-1.140 ± 0.879	-0.287 ± 0.893
3D Lorenz, 0.05	-7.338 ± 1.289	-16.98 ± 1.923	$-12.09 \pm 1.517*$
3D Lorenz, 0.20	-7.474 ± 1.279	-17.30 ± 2.112	$-12.13 \pm 1.497*$
Monkey reach	3.046 ± 4.959	-5.159 ± 0.987	5.476 ± 4.212
Wide-field calcium	5.974 ± 2.979	3.768 ± 6.204	2.044 ± 2.884
Mouse video	-10.927 ± 2.386	-15.86 ± 1.084	$-8.295 \pm 2.834*$
Neuropixels	-12.838 ± 6.017	-12.06 ± 5.244	-11.39 ± 3.284

traditional structure identified by neural population models, coarse-graining identifies some reliable patterns.

Finally we considered the capability of our algorithm to scale to high-dimensional or high-sampling rate data. As a case study, we considered real-time processing (including random projections, ssSVD, and Bubblewrap learning) of Neuropixels data comprising 2688 units with 74,330 timepoints from 30 ms bins. As Figure 4 shows, Bubblewrap once again learns a tiling of the data manifold (a), capturing structure in the probability flow within the space (b) with predictive performance comparable to finer-grained methods (Table 1). More importantly, all these steps can be performed well within the 30ms per sample time of the data (c). In fact, when testing on representative examples of $d = 10^4$ dimensions, 1 kHz sampling rates, or $N = 20,000$ tiles, our algorithm was able to maintain amortized per-sample processing times below those of data acquisition. In practice, we found that even in higher-dimensional datasets (as in the Neuropixels case), only 1 or 2 thousand tiles were used by the model, making it easy to run at kHz data rates. What's more, while learning involved round trip GPU latencies to perform gradient updates, online predictions using slightly stale estimates of Bubblewrap parameters could be performed far faster, in tens of microseconds.

5 Discussion

While increasing attention has been paid in neuroscience to population hypotheses of neural function [10], and while many methods for modeling these data offline exist, surprisingly few methods function online, though presumably online methods will be needed to test some population dynamics hypotheses [17]. While the neural engineering literature has long used online methods based on Kalman filtering, (e.g., [16]), and these methods are known to work well in many practical cases, they also imply strong assumptions about the evolution of activity within these systems. Thus, many studies that employ less constrained behavior or study neural activity with less robust dynamics may benefit from more flexible models that can be trained while the experiment is running.

Here, to address this need, we have introduced both a new dimension reduction method that rapidly produces stable estimates of features and a method for rapidly mapping and charting transitions on neural manifolds. Rather than focus on moment-by-moment prediction, we focus on estimating a coarse tiling and probability flow among these tiles, rendering us less accurate than methods based on linear dynamical systems when such assumptions are met and potentially more accurate when noise dominates and only large-scale probability flow is discernible over multi-step timescales. We saw this in our experiments, where the model of [30] exhibited better overall performance in the mouse video dataset (Figure 3e) when it did not learn to predict and degenerated to a random walk. Indeed, the most relevant comparison to the two approaches is the duality between stochastic differential equations and Fokker-Planck equations, where ours is a (softly) discretized analog of the latter. Nonetheless, in many of the cases we consider, Bubblewrap produces superior results even for state prediction. Nonetheless, like many similar models, ours includes multiple hyperparameters that require setting. While we did not experience catastrophic failure or sensitive dependence on parameters in our testing, and while our methods adapt to the scale and drift of the data, some tuning was required in practice.

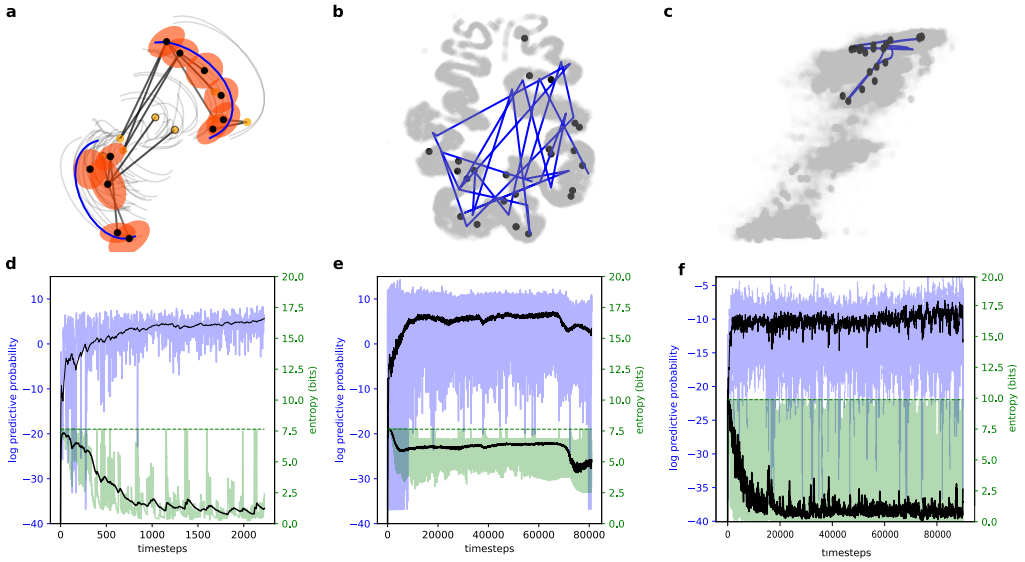

Figure 3: **Bubblewrap results on experimental datasets. a)** Bubblewrap results for example trials (blue) from the monkey reach dataset [47, 48], projected onto the first jPCA plane. All trials are shown in gray. The tile center locations which were closest to the trajectories are plotted along with their covariance "bubbles." Additionally, large transition probabilities from each tile center are plotted as black lines connecting the nodes. Bubblewrap learns both within-trial and across-trial transitions, as shown by the probability weights. **b)** Bubblewrap results on widefield calcium imaging from [49, 50], visualized with UMAP. A single trajectory comprising ≈1.5s of data is shown in blue. Covariance "bubbles" and transition probabilities omitted for clarity. **c)** Bubblewrap results when applied to videos of mouse behavior [49, 50], visualized by projection onto the first SVD plane. Blue line: 3.3s of data. **d, e, f)** Log predictive probability (blue) and entropy (green) over time for the respective datasets in (a,b,c). Dashed green line indicates maximum entropy ($\log_2(N)$).

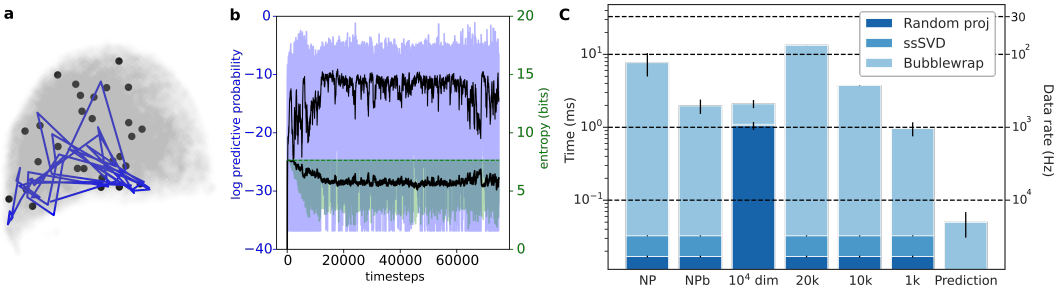

Figure 4: **High-throughput data & benchmarking. a)** Bubblewrap results for example trajectories (blue) in the Neuropixels dataset [51, 52] (data in gray) visualized with UMAP. **b)** Log predictive probability (blue) and entropy (green) over time. Dashed green line indicates maximum entropy. **c)** Average cycle time (log scale) during learning or prediction (last bar) for each timepoint. Neuropixels (NP) is run as in (a,b) with no optimization and all heuristics, and Bubblewrap is easily able to learn at rates much faster than acquisition (30 ms). By turning off the global mean and covariance and priors updates and only taking a gradient step for $\hat{Q}$ every 30 timepoints, we are able to run at close to 1 kHz (NPb). All other bars show example timings from Van der Pol synthetic datasets optimized for speed: 10^4 dim, where we randomly project down to 200 dimensions and used ssSVD to project to 10 dimensions for subsequent Bubblewrap modeling learning; N = 20k, 10k, and 1k nodes, showing how our algorithm scales with the number of tiles; and Prediction, showing the time cost to predict one step ahead.

As detailed above, while many methods target population dynamics, and a few target closed-loop settings [31, 16, 54], very few models are capable of being trained online. Thus, the most closely related approaches are those in [30, 32], to which we provide extensive comparisons. However, these comparisons are somewhat strained by the fact that we provided all models with the same ssSVD-reduced low-dimensional data, while [32] is capable of modeling high-dimensional data in its own right and [30] was targeted at inferring neural computations from dynamical systems. We thus view this work as complementary to the dynamical systems approach, one that may be preferred when small distinctions among population dynamics are less important than characterizing highly noisy, non-repeating neural behavior.

Finally, we showed that online training of Bubblewrap can be performed fast enough for even kiloHertz data acquisition rates if small latencies are tolerable and gradient steps can be performed for small numbers of samples at a time. Yet, for real-time applications, it is not training time but the time required to make predictions that is relevant, and we demonstrate prediction times of tens of microseconds. Thus, coarse-graining methods like ours open the door to online manipulation and steering of neural systems.

References

[1] Misha B Ahrens, Michael B Orger, Drew N Robson, Jennifer M Li, and Philipp J Keller. Whole-brain functional imaging at cellular resolution using light-sheet microscopy. *Nature methods*, 10(5):413–420, 2013.

[2] Valentina Emiliani, Adam E Cohen, Karl Deisseroth, and Michael Häusser. All-optical interrogation of neural circuits. *Journal of Neuroscience*, 35(41):13917–13926, 2015.

[3] Ian H Stevenson and Konrad P Kording. How advances in neural recording affect data analysis. *Nature neuroscience*, 14(2):139–142, 2011.

[4] Nicholas A Steinmetz, Christof Koch, Kenneth D Harris, and Matteo Carandini. Challenges and opportunities for large-scale electrophysiology with neuropixels probes. *Current opinion in neurobiology*, 50:92–100, 2018.

[5] Nicholas A Steinmetz, Cagatay Aydin, Anna Lebedeva, Michael Okun, Marius Pachitariu, Marius Bauza, Maxime Beau, Jai Bhagat, Claudia Böhm, Martijn Broux, et al. Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *bioRxiv*, 2020.

[6] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *nature*, 503(7474):78–84, 2013.

[7] Kanaka Rajan, Christopher D Harvey, and David W Tank. Recurrent network models of sequence generation and memory. *Neuron*, 90(1):128–142, 2016.

[8] H Francis Song, Guangyu R Yang, and Xiao-Jing Wang. Training excitatory-inhibitory recurrent neural networks for cognitive tasks: a simple and flexible framework. *PLoS computational biology*, 12(2):e1004792, 2016.

[9] John P Cunningham and M Yu Byron. Dimensionality reduction for large-scale neural recordings. *Nature neuroscience*, 17(11):1500–1509, 2014.

[10] R Becket Ebitz and Benjamin Y Hayden. The population doctrine revolution in cognitive neurophysiology. *arXiv preprint arXiv:2104.00145*, 2021.

[11] Eric T Carlson, Russell J Rasquinha, Kechen Zhang, and Charles E Connor. A sparse object coding scheme in area v4. *Current Biology*, 21(4):288–293, 2011.

[12] Christopher DiMattina and Kechen Zhang. Adaptive stimulus optimization for sensory systems neuroscience. *Frontiers in neural circuits*, 7:101, 2013.

[13] Benjamin Cowley, Ryan Williamson, Katerina Clemens, Matthew Smith, and Byron M Yu. Adaptive stimulus selection for optimizing neural population responses. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[14] Reza Abbasi-Asl, Yuansi Chen, Adam Bloniarz, Michael Oliver, Ben DB Willmore, Jack L Gallant, and Bin Yu. The deeptune framework for modeling and characterizing neurons in visual cortex area v4. *bioRxiv*, page 465534, 2018.

[15] Zihui Zhang, Lloyd E Russell, Adam M Packer, Oliver M Gauld, and Michael Häusser. Closed-loop all-optical interrogation of neural circuits in vivo. *Nature methods*, 15(12):1037–1040, 2018.

[16] Michael F Bolus, Adam A Willats, Christopher J Rozell, and Garrett B Stanley. State-space optimal feedback control of optogenetically driven neural activity. *Journal of Neural Engineering*, 2020.

[17] Diogo Peixoto, Jessica R Verhein, Roozbeh Kiani, Jonathan C Kao, Paul Nuyujukian, Chandramouli Chandrasekaran, Julian Brown, Sania Fong, Stephen I Ryu, Krishna V Shenoy, et al. Decoding and perturbing decision states in real time. *Nature*, pages 1–6, 2021.

[18] Anne Draelos and John Pearson. Online neural connectivity estimation with noisy group testing. *Advances in Neural Information Processing Systems*, 33, 2020.

[19] Matthew Brand. Incremental singular value decomposition of uncertain data with missing values. *Computer Vision?ECCV 2002*, pages 707–720, 2002.

[20] Christopher G Baker. A block incremental algorithm for computing dominant singular subspaces. Master's thesis, Florida State University, 2004.

[21] Matthew Brand. Fast low-rank modifications of the thin singular value decomposition. *Linear algebra and its applications*, 415(1):20–30, 2006.

[22] Christopher G Baker, Kyle A Gallivan, and Paul Van Dooren. Low-rank incremental methods for computing dominant singular subspaces. *Linear Algebra and its Applications*, 436(8):2866–2888, 2012.

[23] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(1), 2010.

[24] Evan Archer, Il Memming Park, Lars Buesing, John Cunningham, and Liam Paninski. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

[25] Yuanjun Gao, Evan Archer, Liam Paninski, and John P Cunningham. Linear dynamical neural population models through nonlinear embeddings. *arXiv preprint arXiv:1605.08454*, 2016.

[26] Chethan Pandarinath, Daniel J O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D Stavisky, Jonathan C Kao, Eric M Trautmann, Matthew T Kaufman, Stephen I Ryu, Leigh R Hochberg, et al. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature methods*, 15(10):805–815, 2018.

[27] Scott Linderman, Matthew Johnson, Andrew Miller, Ryan Adams, David Blei, and Liam Paninski. Bayesian learning and inference in recurrent switching linear dynamical systems. In *Artificial Intelligence and Statistics*, pages 914–922, 2017.

[28] Scott W. Linderman*, Matthew J. Johnson*, Andrew C. Miller, Ryan P. Adams, David M. Blei, and Liam Paninski. Bayesian learning and inference in recurrent switching linear dynamical systems. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[29] Josue Nassar, Scott W Linderman, Monica Bugallo, and Il Memming Park. Tree-structured recurrent switching linear dynamical systems for multi-scale modeling. *arXiv preprint arXiv:1811.12386*, 2018.

[30] Yuan Zhao and Il Memming Park. Interpretable nonlinear dynamic modeling of neural trajectories. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[31] Yuxiao Yang, Allison T Connolly, and Maryam M Shanechi. A control-theoretic system identification framework and a real-time closed-loop clinical simulation testbed for electrical brain stimulation. *Journal of neural engineering*, 15(6):066007, 2018.

[32] Yuan Zhao and Il Memming Park. Variational online learning of neural dynamics. *Frontiers in computational neuroscience*, 14, 2020.

[33] Peiran Gao, Eric Trautmann, Byron Yu, Gopal Santhanam, Stephen Ryu, Krishna Shenoy, and Surya Ganguli. A theory of multineuronal dimensionality, dynamics and measurement. *BioRxiv*, page 214262, 2017.

[34] Eric M Trautmann, Sergey D Stavisky, Subhaneil Lahiri, Katherine C Ames, Matthew T Kaufman, Daniel J O'Shea, Saurabh Vyas, Xulu Sun, Stephen I Ryu, Surya Ganguli, et al. Accurate estimation of neural population dynamics without spike sorting. *Neuron*, 103(2):292–308, 2019.

[35] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of computer and System Sciences*, 66(4):671–687, 2003.

[36] Ping Li, Trevor J Hastie, and Kenneth W Church. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 287–296, 2006.

[37] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[38] Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[39] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International journal of computer vision*, 77(1):125–141, 2008.

[40] Gordon J Berman, Daniel M Choi, William Bialek, and Joshua W Shaevitz. Mapping the stereotyped behaviour of freely moving fruit flies. *Journal of The Royal Society Interface*, 11(99):20140672, 2014.

[41] Gordon J Berman, William Bialek, and Joshua W Shaevitz. Predictability and hierarchy in drosophila behavior. *Proceedings of the National Academy of Sciences*, 113(42):11943–11948, 2016.

[42] Talmo D Pereira, Joshua W Shaevitz, and Mala Murthy. Quantifying behavior to understand the brain. *Nature Neuroscience*, pages 1–13, 2020.

[43] Gianluigi Mongillo and Sophie Deneve. Online learning with hidden markov models. *Neural computation*, 20(7):1706–1716, 2008.

[44] Olivier Cappé and Eric Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613, 2009.

[45] Sylvain Le Corff, Gersende Fort, et al. Online expectation maximization based algorithms for inference in hidden markov models. *Electronic Journal of Statistics*, 7:763–792, 2013.

[46] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[47] Mark Churchland. Churchland lab code. `"https://churchland.zuckermaninstitute.columbia.edu/content/code"`.

[48] Mark M Churchland, John P Cunningham, Matthew T Kaufman, Justin D Foster, Paul Nuyujukian, Stephen I Ryu, and Krishna V Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51–56, 2012.

[49] Simon Musall, Matthew T Kaufman, Ashley L Juavinett, Steven Gluf, and Anne K Churchland. Single-trial neural dynamics are dominated by richly varied movements. *Nature neuroscience*, 22(10):1677–1686, 2019.

[50] Simon Musall, Matthew T. Kaufman, Ashley L. Juavinett, Steven Gluf, and Anne K. Churchland. Single-trial neural dynamics are dominated by richly varied movements: dataset. Technical report, October 2019.

[51] Nick Steinmetz, Marius Pachitariu, Carsen Stringer, Matteo Carandini, and Kenneth Harris. Eight-probe neuropixels recordings during spontaneous behaviors, Mar 2019.

[52] Carsen Stringer, Marius Pachitariu, Nicholas Steinmetz, Charu Bai Reddy, Matteo Carandini, and Kenneth D Harris. Spontaneous behaviors drive multidimensional, brainwide activity. *Science*, 364(6437), 2019.

[53] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[54] Omid G Sani, Hamidreza Abbaspourazad, Yan T Wong, Bijan Pesaran, and Maryam M Shanechi. Modeling behaviorally relevant neural dynamics enabled by preferential subspace identification. Technical report, Nature Publishing Group, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Discussion section
   (c) Did you discuss any potential negative societal impacts of your work? [No] The focus of this work is a new online modeling approach for neuroscience experiments. As such we do not anticipate any direct, negative societal impacts. In the future we hope that our advance may benefit health care research such as with online analysis and tuning of brain stimulation devices.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] All code is shared in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Some detail is in the main text; additional details on parameter choice are in the supplementary.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Mean and standard deviation across time were reported in the main text. Additional experimental runs across seeds with error bars are in the supplementary.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In the supplementary, Appendix D
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Our work used numerous exisiting datasets, all of which are cited in the text.

(b) Did you mention the license of the assets? [Yes] Details of where to find the assets and their licenses are in the supplementary.

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]