Supplement for Linear-Nonlinear-Time-Warp-Poisson models of neural activity

Patrick N. Lawlor **Matthew G. Perich** **Lee E. Miller** **Konrad P. Kording**

Supplemental Results

Simulations

In order to test whether our model could correctly capture time-warp variability, we simulated data with known parameters and attempted to recover those parameters. The simulations were designed to be as similar as possible to the real neural data analyzed in the next section (PMd neurons recorded during a reaching task; matched for firing rates). To do this, we simulated spiking activity in two parts. First, we simulated data according to an LNP model of movement-related activity in PMd. Next, we simulated random, shared time-warp variability for each reach (see Supplemental Methods for details). Finally, we fit our model to this simulated data, and asked whether it could accurately infer both sets of parameters – the receptive field parameters for each neuron, $\Theta^{(c)}$, as well as the shared reach-specific time-warp variability, $\tau^{(j)}$.

We explored a number of questions related to model performance. We asked whether our model could detect time-warp variability when it was actually present, and whether the model would erroneously detect time-warp variability if it were actually absent. Further, we asked whether time-warp inference depended on the strength of the simulated effect. We asked how inference accuracy depended on the number of neurons recorded; because we model time-warp as *shared* temporal variability, more neurons should lead to higher accuracy. We also asked how inference accuracy depended on the strength of the DTW prior. Finally, we asked if our joint LNP-DTW model was better able to recover receptive field parameters than the LNP alone.

With respect to the time-warp parameters, we found that our algorithm could accurately recover the simulated model parameters (Fig. S1). The accuracy of the time-warp parameter inference increased with the number of simulated neurons simulated, and plateaued after approximately 100 neurons (Fig. S1A,B,C). Importantly, the model's predictive power only increased when time-warp variability was actually present (Fig. S1E,F). This is because we use cross validation, which guards against apparent benefit due to overfitting. Finally, we found that DTW prior strength predictably affected time-warp inference. A prior that biased the algorithm against finding time-warp variability (i.e., a bias for diagonal steps) improved predictions when time-warp variability was not present (Fig. S1A,D, blue markers), presumably by preventing overfitting. Such a prior also decreased the effect size of time-warp variability when the effect was present (Fig. S1B,C,E,F, red markers), presumably by biasing the algorithm against the effect. The effect of the DTW prior in all cases decreased in importance as the number of neurons increased. This was likely because the information contained in the neural data (the "likelihood") outweighed that contained in the prior.

With respect to the neuron-specific receptive field parameters (i.e., the LNP portion of the model), we found that the joint LNP+DTW model estimated the receptive field parameters more accurately than the LNP alone when time-warp variability was present (Fig. S2B,C), but that their estimates were similar when time-warp variability was absent (Fig. S2A), as expected. In summary, we find that our model accurately recovers the simulation parameters and does so better than the base model alone.

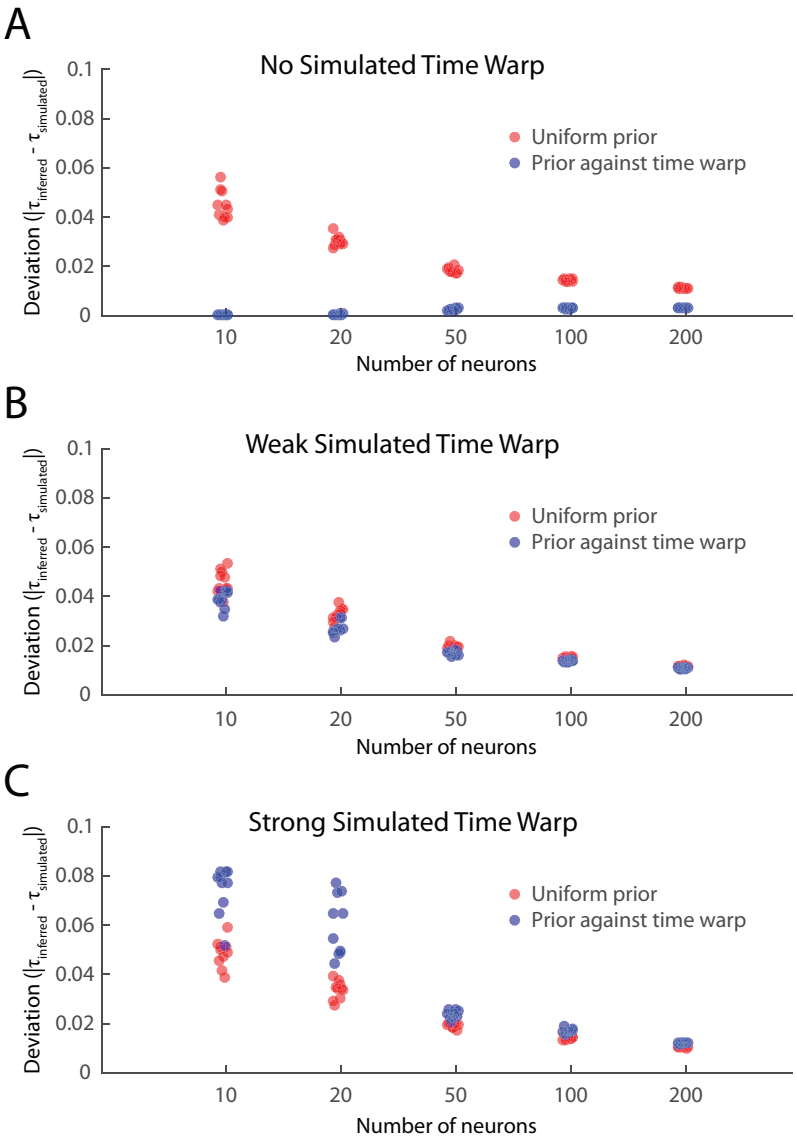

Fig. 1: **Simulation results, time-warp parameters. A-C)** Deviation between simulated and inferred DTW model parameters. Deviation (y-axis) measured by normalized distance between simulated warp path and inferred warp path (see Methods). Each dot represents a model fit using a random subset of simulated neurons; 10 repetitions per condition, jittered along x-axis for visualization. Deviation decreases with the number of neurons. When no warp is present, a uniform prior (lack of prior against time warp) leads to increased deviation. When strong warp is present, a prior against time warp leads to increased deviation.

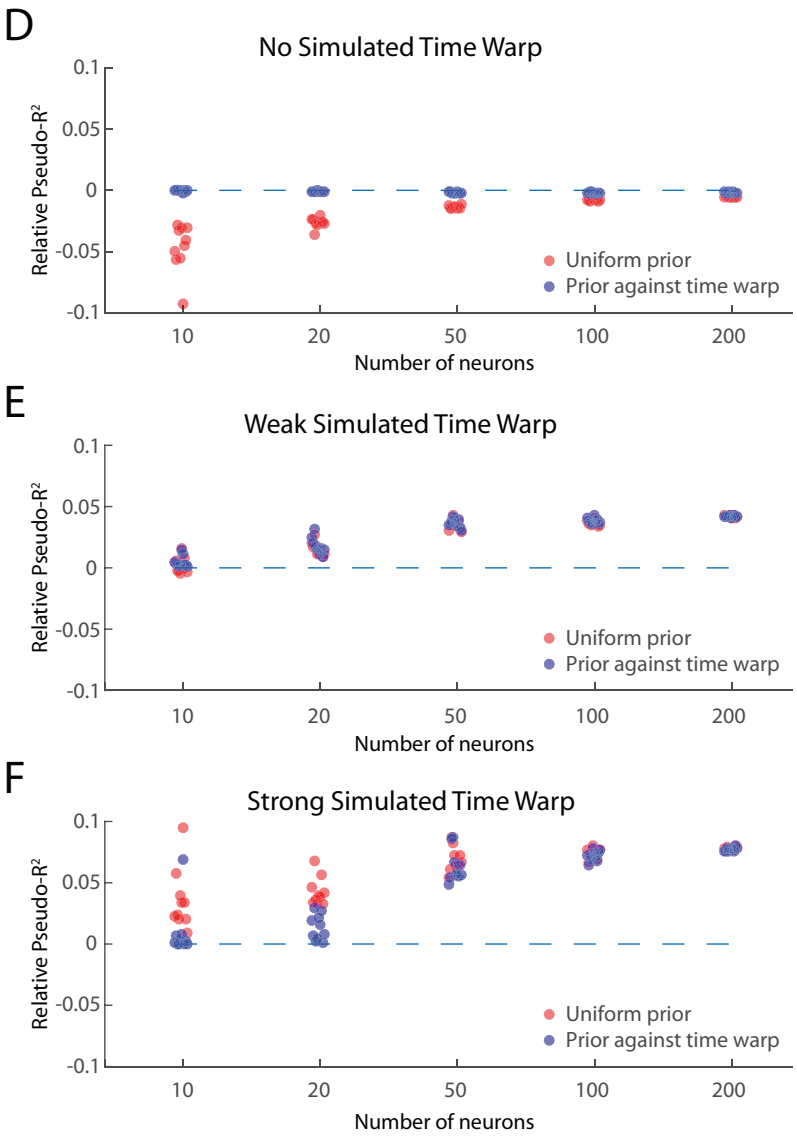

Fig. 1: **Simulation results, time-warp parameters. D-F)** Effect size of time-warp variability. Relative Pseudo-R2 (y-axis) shows marginal improvement in predictive power of LNP+DTW model over LNP model alone. Values shown represent median across neurons. Positive values indicate added predictive power, and negative values indicate decreased predictive power. When no warp is present, a lack of prior against time warp leads to overfitting and thus worsened predictive power on test data. When strong warp is present, a prior against time warp decreases the effect size of time warp.

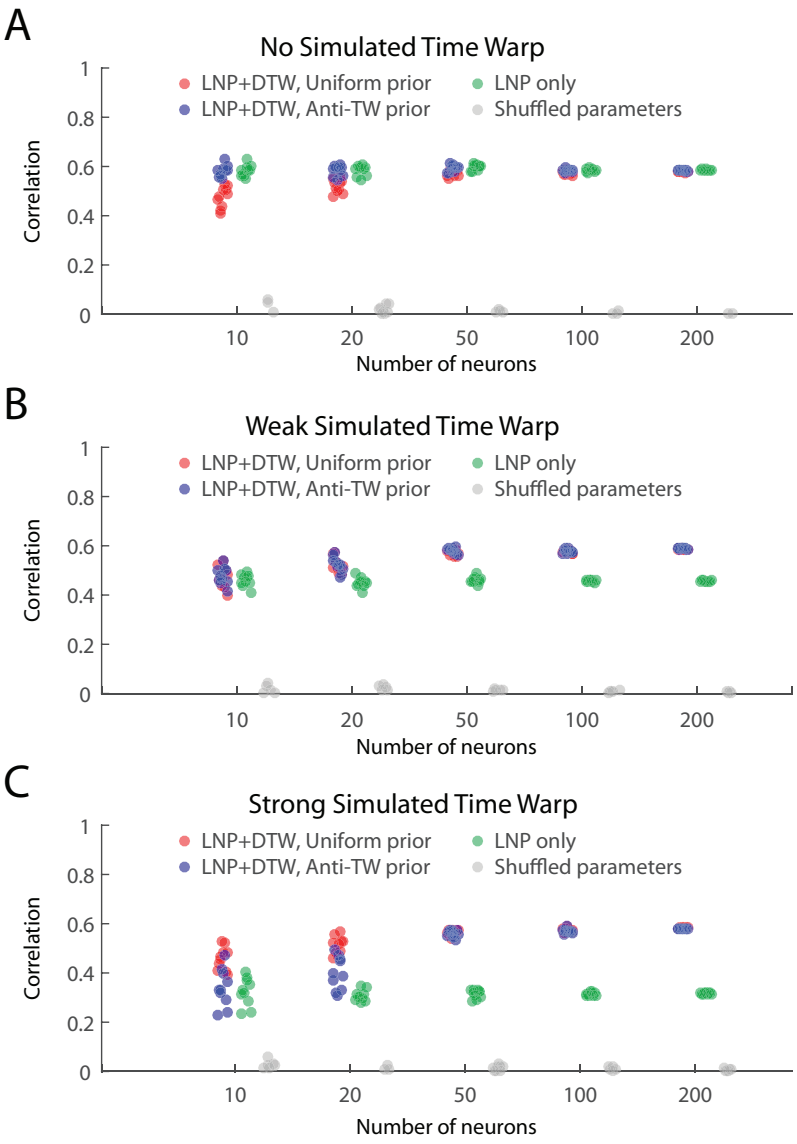

Figure 2: **Simulation results, receptive field parameters. A-C)** Correlation between simulated and inferred LNP receptive field parameters. Each dot represents a model fit using a random subset of simulated neurons; 10 repetitions per condition, jittered along x-axis for visualization. Results shown for LNP+DTW model with uniform (red) and anti-TW (blue) prior, compared with LNP model alone (green), and shuffled parameters (gray). LNP+DTW receptive field estimates are equivalent to LNP model receptive fields when no time warp is present. LNP+DTW receptive field estimates are superior to LNP model receptive field estimates with increasing strength of time warp variability.

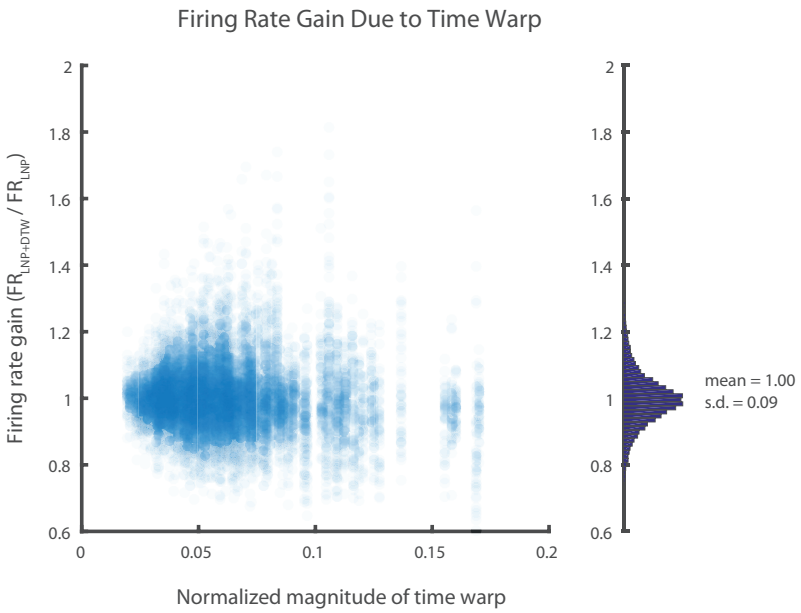

Figure 3: **Firing Rate gain changes due to time warp.** To analyze the effect of gain in our LNP+DTW model, we extracted a gain-like parameter from each neuron and reach (y-axis). We found that our model effectively captured a considerable amount of gain (summarized in the histogram on the right).

Firing Rate Gain Analysis

Motivated by prior work, in particular by Goris et al (2014), we sought to analyze whether our model captured firing rate gain effects. Their prior work enhanced the standard LNP model with a scalar gain term, where each trial was allowed a different gain value. Our
LNP+DTW model allows for a gain effect (e.g., by a "stretch"), and ideally we would like to compare the two models directly. However the time resolution of our model compared with that of Goris et al (2014) makes that comparison difficult. We use time bins of 10 ms (with 61 time bins per reach), whereas they use time bins on the order of 1 second in order to encompass an entire trial. Because in their work there is one time bin per trial, a scalar parameter is sufficient to capture the gain effect for an entire trial. In our case, however, we would require multiple gain parameters per reach (61 parameters, one for each time bin) due to having multiple time bins per reach.

So in order to compare the two models, we extracted a gain-like term from each reach for each neuron. To do this, we calculated the ratio of predicted number of spikes by the LNP+DTW model compared to the LNP model alone ($FR_{LNP+DTW}/FR_{LNP}$) for each reach and neuron. If a given reach had only a "shift" effect (a change in relative timing of neural activity without a change in number of spikes), the gain should be nearly one. If, however, a given reach had a "stretch"/gain effect, the gain would be different than one. In essence, this quantified the gain effect due to time warping.

We examined both the distribution of the gain-like term as well as its dependence on the strength of the time-warp effect. We found that the distribution of the gain-like term was nearly normal with a mean of 1.00 and a standard deviation of 0.09. In other words, the gain was on average nearly one, as expected, and its variability across reaches was considerable. Importantly, even for some reaches with large-magnitude time warps, the gain-like effect was close to one. Such reaches likely correspond to "shifts", highlighting the need to model more than just gain changes.

Control Analysis: Choosing a Speed Threshold

Here we label the "start" of a reach using a speed threshold. This is important because, for the basic reach model, we align the LNP model covariates to this threshold crossing. We chose a threshold of 8 cm/s. The nature of the task (fluid series of variable reaches) often meant that the hand did not reach a complete stop between reaches, making lower thresholds untenable. While reasonable, labeling

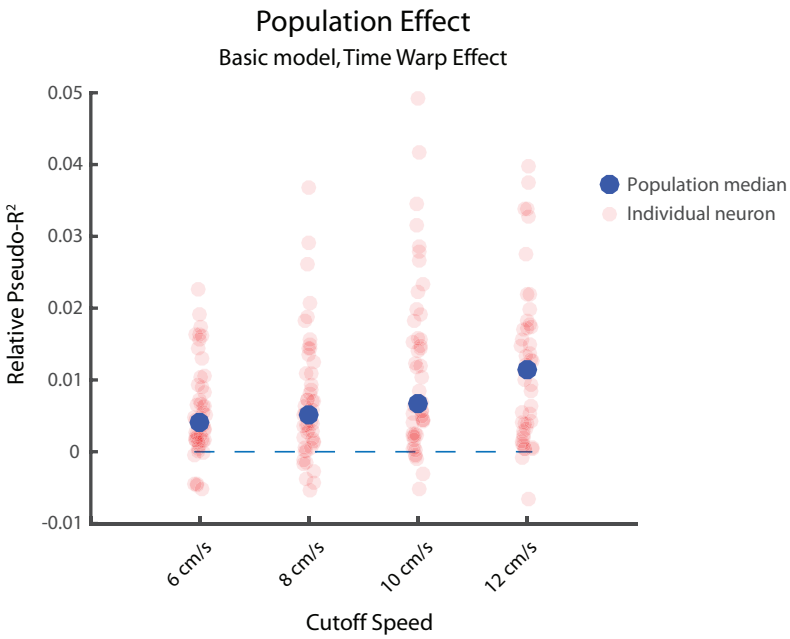

Figure 4: **Time-warp variability robustness to speed threshold cutoff.** Shown is the effect size of time-warp variability as a function of speed threshold used for labeling movement onset. Effect size is given in terms of relative pseudo-R2 between a model including both DTW and LNP, and a model including only the LNP.

the start of a complex kinematic sequence using only a speed threshold crossing is a simplification. We thus tested the robustness of the time-warp effect to the choice of speed threshold (Fig. S4). We found that a lower threshold, 6 cm/s led to a similar effect size, while higher thresholds led to somewhat larger effect sizes. We speculate that higher thresholds led to larger effect sizes because higher thresholds are worse approximations of movement onset. If PMd activity is related to planning movement onset, a higher speed threshold will lead to greater temporal jitter (temporal offset) between PMd activity and movement onset due to kinematic variability between reaches.

Control Analysis: Effect of Time Bin Duration

One important design consideration in models of neural activity is time resolution. In this study, we use 10 ms time bins because we believe that it offers a desirable balance between two factors: 1) the firing rate of a neuron should change more slowly than the duration of a time bin, favoring shorter time bins; and 2) the number of time bins with no spikes should be reasonably minimized in order to better balance the dataset and decrease file size in memory, favoring longer time bins. We have used 10 ms time bins in numerous other studies (e.g., Fernandes et al, 2013; Ramkumar et al, 2016), but given the special importance of temporal variability in this work, we examined the effect of using time bins with different durations.

We found that the effect size of time-warp variability was reasonably consistent across four different time bin durations (Fig. S5). We thus believe that our central result — that time-warp variability is a considerable effect in this data — is robust to this aspect of model design.

Supplementary Methods

LNP + DTW: Model fitting

To fit the combined LNP+DTW model, we used an EM-like approach in which we alternated between fitting the LNP and fitting the DTW portions of the model. I.e., we first fit the LNP; we then fit DTW while holding the LNP parameters constant; we then re-fit the LNP parameters while holding the

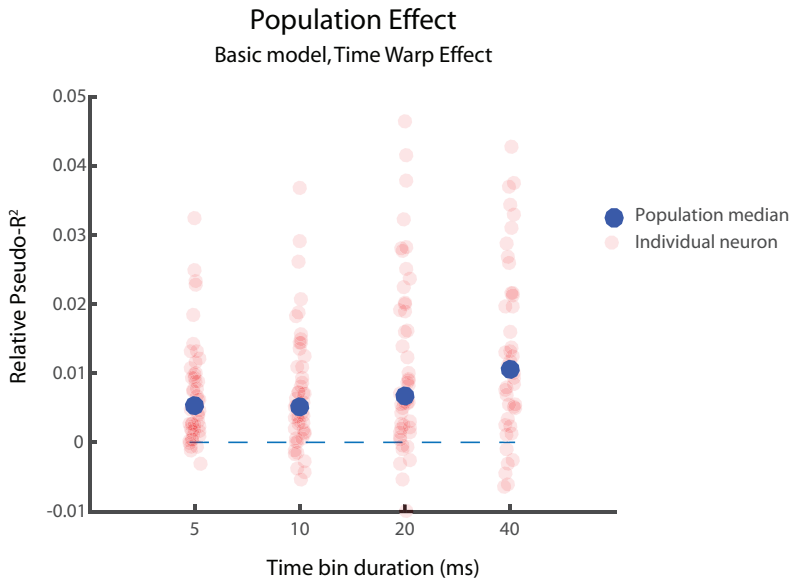

Figure 5: **Time-warp variability robustness to time bin duration.** Shown is the effect size of time-warp variability as a function of time bin duration. Effect size is given in terms of Relative Pseudo-R^2 between a model including both DTW and LNP, and a model including only the LNP.

DTW parameters constant, and so on. We chose this approach because it is difficult to optimize for both the $\Theta^{(c)}$ and $\tau^{(j)}$ directly. This would amount to solving

$$\left\{\hat{\Theta}^{(c)}, \hat{\tau}^{(j)}\right\} = \underset{\{\Theta,\tau\}}{\operatorname{argmax}} P\left(\left\{\Theta^{(c)}\right\}, \left\{\tau^{(j)}\right\} | X, \left\{n_t^{(c,j)}\right\}\right) \tag{1}$$

for all neurons c, reaches j, and time bins t (where curly braces indicate the set of all corresponding parameters). Instead, we find the most likely values of the $\{\Theta^{(c)}\}$ conditioned upon the $\{\tau^{(j)}\}$. We then find the most likely values of the $\{\tau^{(j)}\}$ conditioned upon the $\{\Theta^{(c)}\}$.

For the LNP portion of the model, this corresponds to finding

$$\left\{\hat{\Theta}^{(c)}\right\} = \underset{\{\Theta\}}{\operatorname{argmax}} P\left(\left\{\Theta^{(c)}\right\} | X, \left\{n_t^{(c,j)}\right\}_{c,j,t}, \left\{\tau^{(j)}\right\}\right) \tag{2}$$

We use the notation $\{n_t^{c,j}\}_{c,j,t}$ for disambiguation when the set encompasses multiple indices; $\{n_t^{c,j}\}_{c,j,t}$ denotes the set of $n_t^{(c,j)}$ for all c, j, and t which contains $C \times J \times T$ items.

Applying Bayes' rule yields

$$\left\{\hat{\Theta}^{(c)}\right\} = \underset{\{\Theta\}}{\operatorname{argmax}} \frac{P\left(\left\{n_t^{(c,j)}\right\}_{c,j,t} | X, \left\{\Theta^{(c)}\right\}, \left\{\tau^{(j)}\right\}\right) P\left(\left\{\Theta^{(c)}\right\}\right)}{P\left(\left\{n_t^{(c,j)}\right\}_{c,j,t}\right)} \tag{3}$$

Using the independence of individual cells c and their $\Theta^{(c)}$ (conditioned on the time warp) per the generative model

$$\left\{\hat{\Theta}^{(c)}\right\} = \underset{\{\Theta\}}{\operatorname{argmax}} \frac{\prod_{c=1}^{C} P\left(\left\{n_t^{(c,j)}\right\}_{j,t} | X, \left\{\Theta^{(c)}\right\}, \left\{\tau^{(j)}\right\}\right) P\left(\Theta^{(c)}\right)}{P\left(\left\{n_t^{(c,j)}\right\}_{c,j,t}\right)} \tag{4}$$

Which, because $\Theta^{(c)}$ is independent of the denominator, is equivalent to

$$\begin{aligned}\left\{\hat{\Theta}^{(c)}\right\} &= \underset{\{\Theta\}}{\operatorname{argmax}} \prod_{c=1}^{C} P\left(\left\{n_t^{(c,j)}\right\}_{j,t} | X, \left\{\Theta^{(c)}\right\}, \left\{\tau^{(j)}\right\}\right) P\left(\Theta^{(c)}\right) \\ &= \underset{\{\Theta\}}{\operatorname{argmax}} \prod_{c=1}^{C} \left[P\left(\left\{n_t^{(c,j)}\right\}_{j,t} | X, \left\{\Theta^{(c)}\right\}, \left\{\tau^{(j)}\right\}\right)\right] P\left(\Theta^{(c)}\right)\end{aligned} \tag{5}$$

Using the independence of reaches j, and time bins t

$$\left\{\hat{\Theta}^{(c)}\right\} = \underset{\{\Theta\}}{\operatorname{argmax}} \prod_{c=1}^{C} \left[\prod_{j=1}^{J}\prod_{t=1}^{T} P\left(n_t^{(c,j)} | X, \Theta^{(c)}, \tau^{(j)}\right)\right] P\left(\Theta^{(c)}\right) \tag{6}$$

Taking the log of this expression yields

$$\left\{\hat{\Theta}^{(c)}\right\} = \underset{\{\Theta\}}{\operatorname{argmax}} \sum_{c=1}^{C}\sum_{j=1}^{J}\sum_{t=1}^{T} \log P\left(n_t^{(c,j)} | X, \Theta^{(c)}, \tau^{(j)}\right) + \sum_{c=1}^{C} \log P\left(\Theta^{(c)}\right) \tag{7}$$

We combine the notation of X and $\tau^{(j)}$ into $X_{\tau(t)}$ to denote the value of X in the alignment matrix at position (t, τ). Note that $\tau^{(j)}$ is fixed for this step of the iteration.

$$\begin{aligned}\left\{\hat{\Theta}^{(c)}\right\} &= \underset{\{\Theta\}}{\operatorname{argmax}} \sum_{c=1}^{C}\sum_{j=1}^{J}\sum_{t=1}^{T} \log P\left(n_t^{(c,j)} | X, \Theta^{(c)}, \tau^{(j)}\right) + \sum_{c=1}^{C} \log P\left(\Theta^{(c)}\right) \\ &= \underset{\{\Theta\}}{\operatorname{argmax}} \sum_{c=1}^{C}\sum_{j=1}^{J}\sum_{t=1}^{T} \log \text{Poisson}\left(X_{\tau(t)}\Theta^{(c)}, n_t^{(c,j)}\right) + \sum_{c=1}^{C} \log P\left(\Theta^{(c)}\right)\end{aligned} \tag{8}$$

This optimization problem can be solved by standard methods (e.g., gradient ascent). Because the cells are modeled as independent given the time warp, they can be fit separately of each other.

Next, fitting the DTW parameters $\left\{\tau^{(j)}\right\}$ corresponds to finding

$$\left\{\tau^{(j)}\right\} = \underset{\{\tau\}}{\operatorname{argmax}}\, P\left(\left\{\tau^{(j)}\right\} | \left\{\Theta^{(c)}\right\}, X, \left\{n_t^{(c,j)}\right\}_{c,j,t}\right) \tag{9}$$

Applying Bayes' rule yields

$$\left\{\tau^{(j)}\right\} = \underset{\{\tau\}}{\operatorname{argmax}} \frac{P\left(\left\{n_t^{(c,j)}\right\}_{c,j,t} | \left\{\Theta^{(c)}\right\}, X, \left\{\tau^{(j)}\right\}\right) P\left(\left\{\tau^{(j)}\right\}\right)}{P\left(\left\{n_t^{(c,j)}\right\}_{c,j,t}\right)} \tag{10}$$

Which, because $\{\tau^{(j)}\}$ is independent of the denominator, is equivalent to

$$\left\{\tau^{(j)}\right\} = \underset{\{\tau\}}{\operatorname{argmax}} P\left(\left\{n_t^{(c,j)}\right\}_{c,j,t} | \left\{\Theta^{(c)}\right\}, X, \left\{\tau^{(j)}\right\}\right) P\left(\left\{\tau^{(j)}\right\}\right) \tag{11}$$

Using the independence of reaches j as per the generative model

$$\begin{aligned}\left\{\tau^{(j)}\right\} &= \underset{\{\tau\}}{\operatorname{argmax}} \prod_{j=1}^{J} P\left(\left\{n_t^{(c,j)}\right\}_{c,t} | \left\{\Theta^{(c)}\right\}, X, \tau^{(j)}\right) P\left(\tau^{(j)}\right) \\ &= \underset{\{\tau\}}{\operatorname{argmax}} \prod_{j=1}^{J} \left[P\left(\left\{n_t^{(c,j)}\right\}_{c,t} | \left\{\Theta^{(c)}\right\}, X, \tau^{(j)}\right)\right] P\left(\tau^{(j)}\right)\end{aligned} \tag{12}$$

Using the conditional independence of cells c given the time warp

$$\left\{\tau^{(j)}\right\} = \underset{\{\tau\}}{\operatorname{argmax}} \prod_{j=1}^{J} \left[\prod_{c=1}^{C} P\left(\left\{n_t^{(c,j)}\right\}_{t} | \Theta^{(c)}, X, \tau^{(j)}\right)\right] P\left(\tau^{(j)}\right) \tag{13}$$

And now using the assumption of DTW that time bin t depends only on the previous time bin $t-1$; i.e., the conditional independence of time bins t given their previous time bins $t-1$

$$\left\{\tau^{(j)}\right\} = \underset{\{\tau\}}{\operatorname{argmax}} \prod_{j=1}^{J} \left[\prod_{c=1}^{C}\prod_{t=1}^{T} P\left(n_t^{(c,j)} | \Theta^{(c)}, X, \tau_t^{(j)}\right)\right] \left[\prod_{t=1}^{T} P\left(\tau_t^{(j)} | \tau_{t-1}^{(j)}\right)\right] \tag{14}$$

Taking the log of this expression yields

$$\begin{aligned}\left\{\tau^{(j)}\right\} &= \underset{\{\tau\}}{\operatorname{argmax}} \sum_{j=1}^{J}\sum_{c=1}^{C}\sum_{t=1}^{T} \log P\left(n_t^{(c,j)} | \Theta^{(c)}, X, \tau_t^{(j)}\right) + \sum_{j=1}^{J}\sum_{t=1}^{T} \log P\left(\tau_t^{(j)} | \tau_{t-1}^{(j)}\right) \\ &= \underset{\{\tau\}}{\operatorname{argmax}} \sum_{j=1}^{J}\sum_{c=1}^{C}\sum_{t=1}^{T} \log \text{Poisson}\left(X_{\tau(t)}\Theta^{(c)}, n_t^{(c,j)}\right) + \sum_{j=1}^{J}\sum_{t=1}^{T} \log P\left(\tau_t^{(j)} | \tau_{t-1}^{(j)}\right)\end{aligned} \tag{15}$$

The first term in the first row of Eqn. 15, $\log P\left(n_t^{(c,j)} | \tau_t^{(j)}, \Theta^{(c)}, X\right)$, is the log likelihood function for a Poisson variable. The second term in the first row of Eqn. 15, $\log P\left(\tau_t^{(j)} | \tau_{t-1}^{(j)}\right)$, is a prior over the allowable steps. The time-warp parameter for each reach is modeled as independent of the others, so they can be found separately for each reach using dynamic programming.

It is important to note that separately, each model has a global optimum. The LNP's cost function is convex, and the global optimum can be found by gradient methods. DTW's cost function is not convex (its parameters are discrete), but the global optimum can be found using dynamic programming methods (see section on DTW below). The joint model, however, has no guarantee of an easy-to-find global optimum. But by alternating, we can guarantee that the solution will improve (on training data). This is because each step finds the global optimum given the parameters of the other part of the model. The model typically converged after a small number of alternations (Fig. S6). We terminated alternation after the relative change in log likelihood fell below a pre-specified threshold.

Neither our approach nor any EM-type algorithm guarantees a global optimum, meaning that the inferred parameters will depend on their initialization. We suspect that fitting the standard LNP first provides a good initialization of receptive field parameters (at least when the effect of time warp is not extremely strong), and DTW merely corrects the temporal properties of the LNP filters on a reach-by-reach basis. If the LNP parameters were randomly initialized, we suspect that the solution would likely be a worse local optimum (further from the global optimum).

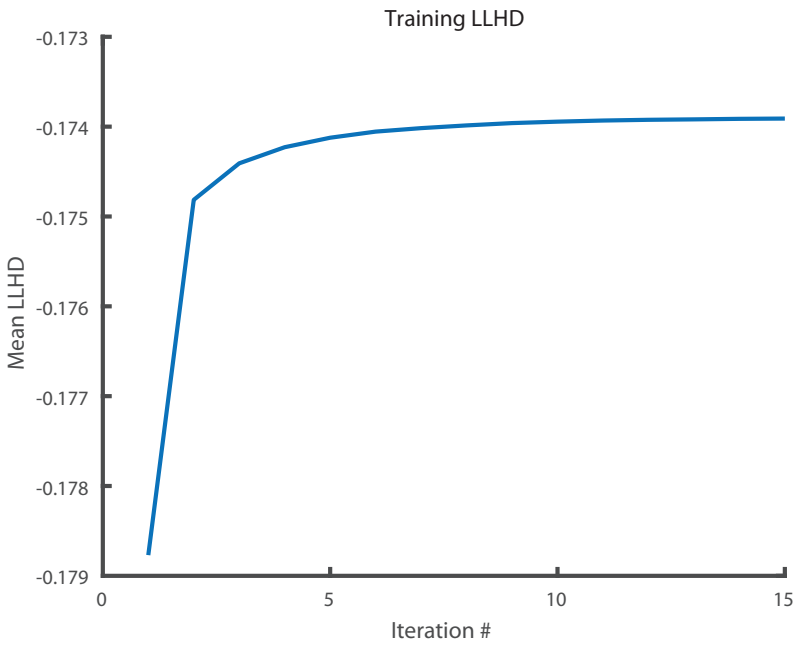

Figure 6: **Log likelihood improvement with increasing number of iterations.** We fit the LNP+DTW model using an EM-like approach; by alternating between fitting the LNP model (while holding the DTW parameters constant) and fitting the DTW model (while holding the LNP parameters constant). Because each of these two fitting procedures finds the global optimum for its corresponding problem, the log likelihood of the joint model should increase with each step of each iteration. Shown is the mean log likelihood calculated on training data as a function of iteration number. Iteration 1 corresponds to the LNP model alone (i.e., prior to time-warp correction). The model converges after 5-10 iterations.

We implemented and fit this model in Matlab. We fit the LNP part of the model using the *lassoglm()* function, a routine for fitting Generalized Linear Models with regularization penalties. For the DTW part of the model, we used a custom implementation of the algorithm.

The basic model takes approximately 30 minutes to fit on the author's computer (64 bit i7 2.3 GHz with 4 virtual cores, 20 GB RAM). Fitting is parallelized across neurons and reaches, so the model can be fit considerably faster on a cluster with many cores.

Parameter selection

For the LNP portion of the model, we used L2 regularization to improve its stability. We used the *lassoglm()* routine in Matlab, which minimizes the following cost function:

$$\frac{1}{N}\text{Deviance} + \lambda P \tag{16}$$

Where Deviance measures model fit, N is the number of data points, and P is the regularization penalty, defined as:

$$P = \frac{1-\alpha}{2}||\beta||_2^2 + \alpha||\beta||_1 \tag{17}$$

Regularization type is specified by two parameters, α and λ. Technically, one could determine the optimal value for each parameter for each individual neuron. This would, however, be extremely computationally demanding (requiring a separately cross validated, two-dimensional grid search for each neuron; not yet considering the DTW portion of the model). We opted to use the same value for all neurons for each of α and λ.

Regularization type is set by the parameter, α (L1, $\alpha = 1$ vs L2, $\alpha \to 0$); we used a value of 0.01. This parameter is often chosen on philosophical grounds; L1 is preferred when many parameters are expected to be exactly zero, whereas L2 is preferred when a small contribution is expected from many covariates. In our experience, the time course of neural activity is often well approximated by a linear combination of multiple temporal basis functions, rather than a single basis function, leading us to prefer L2 regularization (i.e., α small).

For LNP regularization strength, λ, we used a value of 0.05 for all neurons. We have found this value to work well in previous models with similar data (number of data points, number of covariates, firing rates) Ramkumar et al (2016). For this parameter, we also did a sensitivity analysis by examining the cross-validated model performance of the LNP model (Pseudo-R^2) achieved by multiple lambda values around 0.05 spanning multiple orders of magnitude. The population's model performance was not particularly sensitive to choice of this parameter, so we used 0.05.

Regarding the DTW parameters when analyzing the PMd data, we used a uniform prior over the DTW steps: over 1, up 1 and over 1, and up 2 and over 1 (Fig. 4). These parameters were selected by cross validation. More specifically, we tested three values for the relative probability of diagonal steps to non-diagonal steps: 1:1:1, 2:1:1, and 3:1:1 (up 1 and over 1:up 2 and over 1:over 1). We selected the value which led to the highest cross-validated model fit, 1:1:1.

When analyzing the simulated data, we tested and plotted two different settings of prior distributions: a uniform prior, and a prior favoring diagonal steps (a prior against time warp) with a relative probability of 2:1:1 (up 1 and over 1:up 2 and over 1:over 1).

We also note that all of these fitting parameters were determined using the recordings of only one animal (MM). When we used the same parameters with recordings from the second animal (MT), the results were very similar. We feel that this model generalization is evidence that we have chosen good parameter values, and that we have identified a strong, reproducible effect.

Cross validation

For the LNP portion of the model, we used two-fold, reach-wise cross validation for each neuron. We used the odd-numbered reaches in one cross-validation fold, and the even-numbered reaches in the other cross-validation fold. The model fit from the odd-numbered reaches allowed us to make predictions on the even-numbered reaches, and vice versa. We used these test-set predictions in the next step.

For the DTW portion of the model, we used 10-fold, neuron-wise cross validation for each reach. We used 90% of the neurons in each cross-validation fold to fit the model, and made predictions on the remaining 10% of the neurons. The inputs needed to fit the DTW portion of the model are:

1. The actual spike trains for the training-set neurons. These are simply the experimental recordings.
2. The predictions of the spike trains for the training-set neurons. These come from the LNP, and are the test-set reach predictions.
3. Fitting parameters (i.e., DTW transition probabilities)

We then used the inferred time warp (from the training-set neurons) to correct the LNP predictions (of the test-set neurons). The to-be-corrected LNP predictions are themselves test-set predictions (made on test-set reaches). Thus, at no point do training-set predictions leak into the final predictions which we use for model evaluation. This comes at the cost of increased computational resources, but we believe it provided the fairest assessment of model performance.

Simulations

Because we proposed a new type of model, we sought to characterize its behavior using simulations before applying it to real data. To do this, we simulated spike trains using the generative model, and then attempted to recover the model parameters without knowledge of the simulations.

Simulations: LNP model

First, we simulated neural data using the LNP model that was matched to the real PMd data in terms of firing rates. We used the same model that was fit to the real data (see LNP spatial tuning and temporal response parameterization). For each neuron, we randomly chose the following parameters (within a range consistent with the real data): baseline firing rate, preferred reach direction, tuned model component weight, untuned model component weight, and weights of the various temporal basis functions. To simplify the interpretation of the results, we did not simulate a dependence on spike history.

Simulations: DTW model

Next, we simulated a time warp for each reach. To do this, we randomly generated time warp paths. To test the robustness of the model, we generated paths with a variety of behaviors: no time warp, weak time warp, and strong time warp. In the case of no time warp, we always chose the path's steps to be diagonal (up 1 and over 1). This corresponds to the time moving forward at the same rate in spike train time t and LNP model time τ. In the case of the weak time warp, we sampled the steps randomly from among the following possible steps (equi-probable, but only up to the constraint that the time-warp path begin at $(t, \tau) = (1, 1)$ and end at $(t, \tau) = (T, T)$: over 1, up 1 and over 1 (diagonal), and up 2 and over 1. Because adjacent steps were independent by design, it was unlikely to see "runs" of a single type of step; this is why we label this type of simulated time warp weak. In the case of the strong warp, however, we generated time warps with "runs" of a particular type of step. To do this, we simply required that the first step of a run was followed by four steps of exactly the same type. For the first step of each run, the type of step was randomly drawn from the same distribution as those from the weak time-warp distribution. Overall, this had the effect of creating substantial time-warp variability.

We then applied each simulated time warp to the model covariates for a single reach. Following our generative model, we linearly combined these warped covariates with the LNP model parameters. We then passed the resultant signal through the exponential nonlinearity to give the simulated mean function ("underlying firing rate"). We then sampled spikes from the Poisson distribution.

To test the LNP+DTW model, we fit it to this simulated data. To explore the behavior of the model, we fit it under multiple conditions. To test its sensitivity, we fit the model to varying numbers of neurons (10, 20, 50, 100, 200 neurons). To test the model's dependence on the chosen prior distribution (DTW's step parameters), we tested two prior strengths (diagonal:non-diagonal probabilities of 1:1 and 2:1). These corresponded to no preference for diagonal steps (1:1) and to a preference for diagonal steps (2:1).

Simulations: Performance metrics

To quantify the performance of our model, we used two metrics: a deviation metric, and Relative Pseudo-R^2 (Eqn. 15 in main text).

We designed the deviation metric to measure the distance between two time-warp paths. In this case, we wanted to quantify the deviation between the simulated time-warp path and the inferred time-warp path in order to assess our parameter inference procedure. We build this metric by starting with a single reach:

$$\left|\tau^{(j)} - \hat{\tau}^{(j)}\right| = \sum_{t=1}^{T} \left|\tau_t^{(j)} - \hat{\tau}_t^{(j)}\right| \tag{18}$$

which measures the absolute value of the difference between the simulated and inferred warp path time τ for each time bin t. We generalize this to multiple reaches J, and normalize so that its value is consistent across an arbitrary number of reaches. We divide by T^2 because otherwise the metric

would accumulate with both the length of the spike train T and with the possible values of the single-bin deviation (between 0 and T).

$$\text{Deviation} = \frac{1}{J}\sum_{j=1}^{J}\left(\frac{1}{T^2}\sum_{t=1}^{T}\left|\tau_t^{(j)} - \hat{\tau}_t^{(j)}\right|\right) \tag{19}$$

Dynamic Time Warping

Intuition

In DTW (Sakoe and Chiba, 1978), we start with two signals that may be "misaligned", and our goal is to find a way to align them. DTW views this as a matching problem: for every time bin t of T in signal 1, which time bins τ of T in signal 2 do those best match up with? This motivates viewing the problem as a $T \times T$ matrix in which we measure the correspondence/agreement between every pair of time bins t and τ. The best alignment between the two signals amounts to finding the path through the $T \times T$ matrix that starts at $(t, \tau) = (1, 1)$ and ends at $(t, \tau) = (T, T)$ and maximizes the total correspondence between the two signals.

One begins by calculating the correspondence between every pair of time bins t and τ, and placing the resulting value in the $T \times T$ matrix we call *logLikelihoodMatrix*. We call it this because in our work, we use the log likelihood function of a Poisson variable to measure the extent to which signal 1 and signal 2 match.

After populating the *logLikelihoodMatrix*, we calculate the accumulated log likelihood matrix, *acclogLikelihoodMatrix*. Each entry in this $T \times T$ matrix will effectively keep track of the best alignment path up to that point. More specifically, each entry will contain the sum of the log likelihood values along the best path up to that point in the matrix as well as the sum of the log transition probabilities of that path. This is accomplished by dynamic programming; the solution to the full problem is found by solving the component sub-problems. Intuitively, this makes sense because the best alignment path from $(t, \tau) = (1, 1)$ to $(t, \tau) = (T, T)$ will use the best path to the second-to-last position in the alignment matrix along with the best final step. The best path to the second-to-last position in the alignment matrix will use the best path to the third-to-last position in the alignment matrix along with the best second-to-last step.

The basic algorithm itself can be found in Sakoe and Chiba (1978) and is widely available online.

References

Fernandes HL, Stevenson IH, Phillips AN, Segraves Ma, Kording KP (2013) Saliency and saccade encoding in the frontal eye field during natural scene search. Cerebral cortex (New York, NY : 1991) pp 1–14, DOI 10.1093/cercor/bht179

Goris RLT, Movshon JA, Simoncelli EP (2014) Partitioning neuronal variability. Nature Neuroscience (April), DOI 10.1038/nn.3711

Ramkumar P, Lawlor PN, Glaser JI, Wood DK, Phillips AN, Segraves MA, Kording KP (2016) Feature-based attention and spatial selection in frontal eye fields during natural scene search. Journal of Neurophysiology 116(3):1328–1343, DOI 10.1152/jn.01044.2015, URL `http://jn.physiology.org/lookup/doi/10.1152/jn.01044.2015`

Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing ASSP-26(1):43–49, DOI 10.1109/TASSP.1978.1163055