Fitting summary statistics of neural data with a differentiable spiking network simulator

Guillaume Bellec*, Shuqi Wang*, Alireza Modirshanechi, Johanni Brea◇, Wulfram Gerstner◇

Laboratory of Computational Neuroscience
École polytechnique fédérale de Lausanne (EPFL)
first.lastname@epfl.ch

Abstract

Fitting network models to neural activity is an important tool in neuroscience. A popular approach is to model a brain area with a probabilistic recurrent spiking network whose parameters maximize the likelihood of the recorded activity. Although this is widely used, we show that the resulting model does not produce realistic neural activity. To correct for this, we suggest to augment the log-likelihood with terms that measure the dissimilarity between simulated and recorded activity. This dissimilarity is defined via summary statistics commonly used in neuroscience and the optimization is efficient because it relies on back-propagation through the stochastically simulated spike trains. We analyze this method theoretically and show empirically that it generates more realistic activity statistics. We find that it improves upon other fitting algorithms for spiking network models like GLMs (Generalized Linear Models) which do not usually rely on back-propagation. This new fitting algorithm also enables the consideration of hidden neurons which is otherwise notoriously hard, and we show that it can be crucial when trying to infer the network connectivity from spike recordings.

1 Introduction

Modelling neural recordings has been a fundamental tool to advance our understanding of the brain. It is now possible to fit recurrent spiking neural networks (RSNNs) to recorded spiking activity [1, 2, 3, 4, 5]. The resulting network models are used to study neural properties [6, 7, 8, 9] or to reconstruct the anatomical circuitry of biological neural networks [4, 5, 10].

Traditionally a biological RSNN is modelled using a specific Generalized Linear Model [11] (GLM) often referred to as the Spike Response Model (SRM) [12, 11, 13]. The parameters of this RSNN model are fitted to data with the maximum likelihood estimator (MLE). The MLE is consistent, meaning that if the amount of recorded data becomes infinite it converges to the true network parameters when they exist. However, when fitting neural activity in practice the MLE solutions are often reported to generate unrealistic activity [14, 15, 16] showing that this method is not perfect despite it's popularity in neuroscience. We also find that these unrealistic solution emerge more easily when hidden neurons outside of the recorded units have a substantial impact on the recorded neurons. This is particularly problematic because the likelihood is not tractable with hidden neurons and it raises the need for new methods to tackle the problem.

To address this, we optimize *sample-and-measure* loss functions in addition to the likelihood: these loss functions require *sampling* spiking data from the model and *measuring* the dissimilarity between recorded and simulated data. To measure this dissimilarity we suggest to compare summary statistics

*Equal contributions. ◇ Senior authors.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

popular in neuroscience like the peristimulus time histogram (PSTH) and the noise-correlation (NC). Without hidden neurons, this method constrains the network to generate realistic neural activity but without biasing the MLE solution in the theoretical limit of infinite data. In practice, it leads to network models generating more realistic activity than the MLE. With hidden neurons, the sample-and-measure loss functions can be approximated efficiently whereas the likelihood function is intractable. Although recovering the exact network connectivity from the recorded spikes remains difficult [10], we show on artificial data that modelling hidden neurons in this way is crucial to recover the connectivity parameters. In comparison, methods like MLE which ignore the hidden activity wrongly estimate the connectivity matrix.

In practice the method is simple to optimize with automatic differentiation but there were theoretical and technical barriers which have prevented earlier attempts. The first necessary component is to design an efficient implementation of back-propagation in stochastic RSNN inspired by straight-through gradient estimators [17, 18] and numerical tricks from deterministic RSNNs [19]. Previous generative models of spiking activity relying on back-propagation used artificial neural networks [20, 21] which are not interpretable model in terms of connectivity and neural dynamics. Previous attempts to include hidden neurons in RSNN models did not rely on back-prop but relied on expectation maximization [11, 22] or reinforce-style gradients [23, 24, 25, 26] which have a higher variance [27]. There exist other methods to fit neural data using back-propagation and deep learning frameworks but they do not back-propagate through the RSNN simulator itself, rather they require to engineer and train a separate deep network to estimate a posterior distribution [28, 29, 30, 31] or as a GAN discriminator [32, 33, 21]. The absence of a discriminator in the sample-and-measure loss function connects it with other simple generative techniques used outside of the context of neural data [34, 35, 36].

Our implementation of the algorithm is published online openly [2].

2 A recurrent spiking neural network (RSNN) model

We will compare different fitting techniques using datasets of spiking neural activity. We denote a tensor of $K^{\mathcal{D}}$ recorded spike trains as $\boldsymbol{z}^{\mathcal{D}} \in \{0,1\}^{K^{\mathcal{D}} \times T \times n_{\mathcal{V}}}$ where $n_{\mathcal{V}}$ is the total number of visible neurons recorded simultaneously and T is the number of time steps. To model the biological network which produced that activity, we consider a simple model that can capture the recurrent interactions between neurons and the intrinsic dynamics of each neuron. This recurrent network contains $n_{\mathcal{V}+\mathcal{H}}$ neurons connected arbitrarily and split into a visible and a hidden population of sizes $n_{\mathcal{V}}$ and $n_{\mathcal{H}}$. Similarly to [1, 4, 37, 14] we use a GLM where each unit is modelled with a SRM neuron [12] with $u_{t,j}$ being the distance to the threshold of neuron j and its spike $z_{t,j}$ is sampled at time step t from a Bernoulli distribution $\mathcal{B}$ with mean $\sigma(u_{t,j})$ where σ is the sigmoid function. The dynamics of the stochastic recurrent spiking neural network (RSNN) are described by:

$$z_{t,j} \sim \mathcal{B}\left(\sigma(u_{t,j})\right) \qquad \text{with} \qquad u_{t,j} = \frac{v_{t,j} - v_{\text{thr}}}{v_{\text{thr}}} \tag{1}$$

$$v_{t,j} = \sum_{i=1}^{n_{\mathcal{V}+\mathcal{H}}} \sum_{d=1}^{d_{\max}} W^{d}_{j,i} z_{t-d,i} + b_j + \mathcal{C}_{t,j} \ , \tag{2}$$

where $\boldsymbol{W}$ defines the spike-history and coupling filters spanning $d_{\max}$ time-bins, $\boldsymbol{b}$ defines the biases, $v_{\text{thr}} = 0.4$ is a constant, and $\mathcal{C}$ is a spatio-temporal stimulus filter processing a few movie frames and implemented here as a convolutional neural network (CNN) (this improves the fit accuracy as seen in [14, 38] and in Figure S1 from the appendix). Equations (1) and (2) define the probability $\mathcal{P}_\phi(\boldsymbol{z})$ of simulating the spike trains $\boldsymbol{z}$ with this model and ϕ represents the concatenation of all the network parameters ($\boldsymbol{W}$, $\boldsymbol{b}$ and the CNN parameters). Traditionally the parameters which best explain the data are given by the MLE: $\text{argmax}_{\phi} \mathcal{P}_\phi(\boldsymbol{z}^{\mathcal{D}})$ [1, 7, 37, 5, 4]. When all neurons are visible, the likelihood factorizes as $\prod_t \mathcal{P}_\phi(\boldsymbol{z}^{\mathcal{D}}_t | \boldsymbol{z}^{\mathcal{D}}_{1:t-1})$, therefore the log-likelihood can be written as the negative cross-entropy (CE) between $\boldsymbol{z}^{\mathcal{D}}_t$ and $\sigma(\boldsymbol{u}^{\mathcal{D}}_t) = \mathcal{P}_\phi(\boldsymbol{z}^{\mathcal{D}}_t = \mathbf{1} | \boldsymbol{z}^{\mathcal{D}}_{1:t-1})$ where $\boldsymbol{u}^{\mathcal{D}}_t$ is computed as in equation (2) with $\boldsymbol{z} = \boldsymbol{z}^{\mathcal{D}}$. So when all neurons are recorded and $\boldsymbol{z}^{\mathcal{D}}$ are provided in the dataset the computation of the MLE never needs to simulate spikes from the model and it is sufficient to minimize the loss function:

$$\mathcal{L}_{MLE} = -\log \mathcal{P}_\phi(\boldsymbol{z}^{\mathcal{D}}) = CE(\boldsymbol{z}^{\mathcal{D}}, \sigma(\boldsymbol{u}^{\mathcal{D}})) \ . \tag{3}$$

[2] Code repository: `https://github.com/EPFL-LCN/pub-bellec-wang-2021-sample-and-measure`

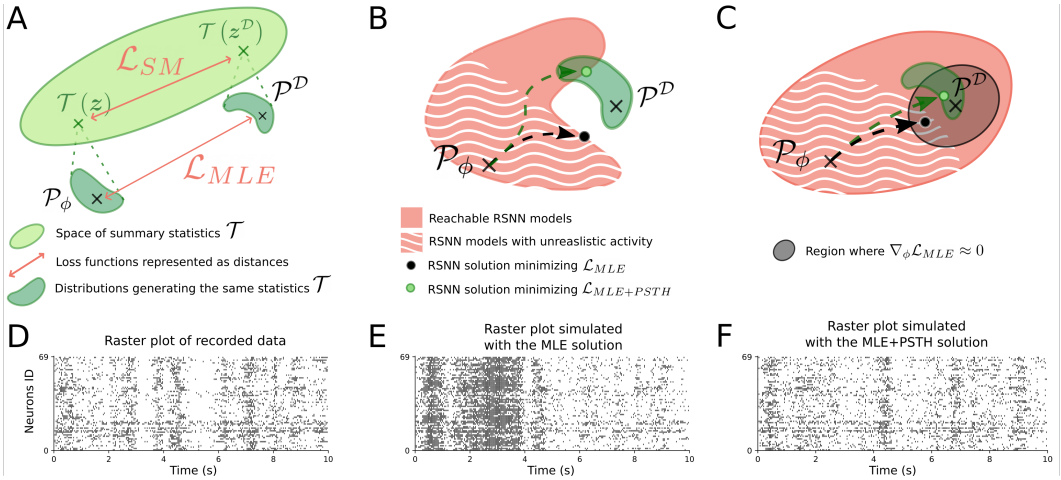

Figure 1: **A)** The distribution $\mathcal{P}_\phi$ represents the RSNN model and $\mathcal{P}_\mathcal{D}$ represents the true biological network, the goal is to bring $\mathcal{P}_\phi$ close to $\mathcal{P}_\mathcal{D}$. The loss function $\mathcal{L}_{MLE}$ is represented as the distance between $\mathcal{P}_\phi$ and $\mathcal{P}_\mathcal{D}$ because it is equal up to a constant to $D_{KL}(\mathcal{P}_\mathcal{D}, \mathcal{P}_\phi)$. We draw the space of summary statistics $\mathcal{T}$ to represent the loss function $\mathcal{L}_{PSTH}$ as the distance between the statistics $\mathcal{T}(\boldsymbol{z})$ simulated from the RSNN model $\mathcal{P}_\phi$ and measured from the data $\mathcal{P}_\mathcal{D}$. **B)** Even if the model is misspecified and $\mathcal{P}_\phi$ cannot perfectly match the true distribution $\mathcal{P}_\mathcal{D}$, when $\mathcal{L}_{MLE+PSTH}$ is minimized the statistics $\mathcal{T}$ are indistinguishable between simulation and data (the green dot lands in the dark green area). When minimizing MLE alone, the solution (black dot) might generate unrealistic activity. **C)** When the RSNN model is expressive enough to represent the true distribution but the data is insufficient, so there is some uncertainty about the true network parameters (black ellipse): minimizing $\mathcal{L}_{MLE+PSTH}$ favours solutions with realistic PSTH. **D)** Spikes recorded simultaneously from the V1-dataset [39]. **E)** Simulated spiking response from an RSNN minimizing $\mathcal{L}_{MLE}$. **F)** Same as **E** but with $\mathcal{L}_{MLE+PSTH}$.

3 The sample-and-measure loss functions

In this section we describe the sample-and-measure loss functions which include simulated data in a differentiable fashion in the optimization objective, a direct benefit is to enable the consideration of hidden neurons. We define the *sample-and-measure* loss functions as those which require *sampling* spike trains $\boldsymbol{z} \in \{0,1\}^{K \times T \times n_{\mathcal{V}+\mathcal{H}}}$ from the model $\mathcal{P}_\phi$ and *measuring* the dissimilarity between the recorded and simulated data. This dissimilarity is defined using some statistics $\mathcal{T}(\boldsymbol{z})$ and the generic form of the sample-and-measure loss functions is:

$$\mathcal{L}_{SM} = d\Big(\mathcal{T}(\boldsymbol{z}^\mathcal{D}),\ \mathbb{E}_{\mathcal{P}_\phi}\left[\, \mathcal{T}(\boldsymbol{z}) \,\right] \Big)\ , \tag{4}$$

where d is a dissimilarity function, like the mean-squared error or the cross entropy. To compute the expectations $\mathbb{E}_{\mathcal{P}_\phi}$ we use Monte-Carlo estimates from a batch of simulated trials $\boldsymbol{z}$. For example to match the PSTH between the simulated and recorded data, we consider the statistics $\mathcal{T}(\boldsymbol{z})_{t,i} = \frac{1}{K}\sum_k \boldsymbol{z}^k_{t,i}$ and evaluate the expectation with the unbiased estimate $\bar{\sigma}_{t,i} = \frac{1}{K}\sum_k \sigma(u^k_{i,t})$. Denoting the PSTH of the data as $\bar{z}^\mathcal{D}_{t,i} = \frac{1}{K}\sum_k \boldsymbol{z}^{k,\mathcal{D}}_{t,i}$ and choosing d to be the cross-entropy, we define the sample-and-measure loss function for the PSTH:

$$\mathcal{L}_{\text{PSTH}} = CE(\bar{\boldsymbol{z}}^\mathcal{D}, \bar{\boldsymbol{\sigma}})\ . \tag{5}$$

When all neurons are visible, we minimize the loss function $\mathcal{L}_{MLE+SM} = \mu_{MLE}\mathcal{L}_{MLE} + \mu_{SM}\mathcal{L}_{SM}$ where $\mu_{MLE}, \mu_{SM} > 0$ are constant scalars. When there are hidden neurons, the log-likelihood is intractable. Instead we minimize the negative of a lower bound of the log-likelihood (see appendix D for a derivation inspired by [40, 23, 24, 25]):

$$\mathcal{L}_{ELBO} = CE(\boldsymbol{z}^\mathcal{D}, \sigma(\boldsymbol{u}^\mathcal{V}))\ , \tag{6}$$

with $\sigma(\boldsymbol{u}^\mathcal{V})$ being the firing probability of the visible neurons, where the visible spikes $\boldsymbol{z}^\mathcal{V}$ are clamped to the recorded data $\boldsymbol{z}^\mathcal{D}$ and the hidden spikes $\boldsymbol{z}^\mathcal{H}$ are sampled according to the model

dynamics. Hence the implementation of $\mathcal{L}_{ELBO}$ and $\mathcal{L}_{SM}$ are very similar with the difference that the samples used in $\mathcal{L}_{SM}$ are not clamped (but all our results about $\mathcal{L}_{SM}$ are also valid when they are clamped if we use the extended definition given in Appendix D). To compute the gradients with respect to these loss functions we use back-propagation which requires the propagation of gradients through the stochastic samples $\boldsymbol{z}$. If they were continuous random variables, one could use the reparametrization trick [27], but extending this to discrete distributions is harder [17, 18, 41, 42, 43]. One way to approximate these gradients is to relax the discrete dynamics into continuous ones [41] or to use relaxed control variates [42], but we expect that the relaxed approximations become more distant from the true spiking dynamics as the network architecture gets very deep or if the network is recurrent. Instead, we choose to simulate the exact spiking activity in the forward pass and use straight-through gradient estimates [17, 18] in the backward pass by defining a pseudo-derivative $\frac{\partial z_{t,i}}{\partial u_{t,i}}$ over the binary random variables $z_{t,i} \sim \mathcal{B}(\sigma(\boldsymbol{u}_{t,i}))$. We use here the same pseudo-derivative $\frac{\partial z_{t,i}}{\partial u_{t,i}} = \gamma \max(0, 1 - |u_{t,i}|)$ as in deterministic RNNs [19] because the dampening factor (here $\gamma = 0.3$) can avoid the explosive accumulation of approximation errors through the recurrent dynamics [44]. Although the resulting gradients are biased, they work well in practice.

A geometrical description of a sample-and-measure loss function In the remaining paragraphs of this section we provide a geometrical representation and a mathematical analysis of the loss function $\mathcal{L}_{SM}$. For this purpose, we consider that the recorded spike trains $\boldsymbol{z}^{\mathcal{D}}$ are sampled from an unknown distribution $\mathcal{P}_{\mathcal{D}}$ and we formalize that our goal is to bring the distribution $\mathcal{P}_{\phi}$ as close as possible to $\mathcal{P}_{\mathcal{D}}$. In this view, we re-write $\mathcal{L}_{SM} = d(\mathbb{E}_{\mathcal{P}_{\mathcal{D}}}[\mathcal{T}(\boldsymbol{z})], \mathbb{E}_{\mathcal{P}_{\phi}}[\mathcal{T}(\boldsymbol{z})])$ and we re-interpret $\mathcal{L}_{MLE}$ as the Kullback-Leibler divergence (D_{KL}) from $\mathcal{P}_{\phi}$ to $\mathcal{P}_{\mathcal{D}}$. This is equivalent because the divergence $D_{KL}(\mathcal{P}_{\mathcal{D}}, \mathcal{P}_{\phi})$ is equal to $-\mathbb{E}_{\mathcal{P}_{\mathcal{D}}}\left[\log \mathcal{P}_{\phi}(\boldsymbol{z}^{\mathcal{D}})\right]$ up to a constant.

In Figure 1 we represent the losses $\mathcal{L}_{MLE}$ and $\mathcal{L}_{PSTH}$ in the space of distributions and we can represent $\mathcal{L}_{MLE} = D_{KL}(\mathcal{P}_{\mathcal{D}}, \mathcal{P}_{\phi})$ as the distance between $\mathcal{P}_{\mathcal{D}}$ and $\mathcal{P}_{\phi}$. To represent the sample-and-measure loss function $\mathcal{L}_{SM}$ (or specifically $\mathcal{L}_{PSTH}$ in Figure 1), we project the two distributions onto the space of summary statistics $\mathcal{T}$ represented in light green. Hence, these projections represent the expected statistics $\mathbb{E}_{\mathcal{P}_{\phi}}[\mathcal{T}(\boldsymbol{z})]$ and $\mathbb{E}_{\mathcal{P}_{\mathcal{D}}}\left[\mathcal{T}(\boldsymbol{z}^{\mathcal{D}})\right]$ and $\mathcal{L}_{SM}$ can be represented as the distance between the two projected statistics.

Although minimizing $\mathcal{L}_{MLE}$ should recover the true distributions (i.e. the biological network) if the recorded data is sufficient and the model is well specified, these ideal conditions do not seem to apply in practice because the MLE solution often generates unrealistic activity (see Figure 1D-E). Panels B and C in Figure 1 use the geometrical representation of panel A to summarize the two main scenarios where minimizing $\mathcal{L}_{MLE+SM}$ is better than $\mathcal{L}_{MLE}$ alone. In panel B, we describe a first scenario in which the model is misspecified meaning that it is not possible to find ϕ^* so that $\mathcal{P}_{\mathcal{D}} = \mathcal{P}_{\phi^*}$. In this misspecified setting, there is no guarantee that the MLE solution yields truthful activity statistics and it can explain why the MLE solution generates unrealistic activity (Figure 1E). In this case, adding a sample-and-measure loss function can penalize unrealistic solutions to solve this problem (Figure 1B and F). Another possible scenario is sketched in panel C. It describes the case where the model is well specified but $\mathcal{L}_{MLE}$ is flat around $\mathcal{P}_{\mathcal{D}}$ for instance because too few trials are recorded or some neurons are not recorded at all. In that case we suggest to minimize $\mathcal{L}_{MLE+SM}$ to nudge the solution towards another optimum where $\mathcal{L}_{MLE}$ is similarly low but the statistics $\mathcal{T}$ match precisely. In this sense, $\mathcal{L}_{SM}$ can act similarly as a Bayesian log-prior to prefer solutions producing truthful activity statistics.

Theoretical analysis of the sample-and-measure loss function To describe formal properties of the sample-and-measure loss function $\mathcal{L}_{SM}$, we say that two distributions are indistinguishable according to the statistics $\mathcal{T}$ if the expectation $\mathbb{E}[\mathcal{T}(\boldsymbol{z})]$ is the same for both distributions. We assume that the dissimilarity function $d(\mathcal{T}, \mathcal{T}')$ reaches a minimum if and only if $\mathcal{T} = \mathcal{T}'$ (this is true for the mean-squared error and the cross-entropy). Then for any statistics $\mathcal{T}$ and associated dissimilarity function d we have:

Property 1. *If the RSNN model is expressive enough so that there exists parameters ϕ° for which $\mathcal{P}_{\phi}$ and $\mathcal{P}^{\mathcal{D}}$ are indistinguishable according to the statistics $\mathcal{T}$, then ϕ° is a global minimum of $\mathcal{L}_{SM}$. Reciprocally, if this minimum is reached then $\mathcal{P}_{\phi}$ and $\mathcal{P}^{\mathcal{D}}$ are indistinguishable according to $\mathcal{T}$.*

This property is a direct consequence of our assumption on the function d. If $\mathcal{T}$ measures the PSTH it means that the optimized simulator produces the same PSTH as measured in the data. This can be true even if the model is misspecified which is why we represented in Figure 1B that the RSNN minimizing $\mathcal{L}_{MLE+PSTH}$ lands in the dark green region where the PSTH of the data is matched accurately. As one may also want to target other statistics like the noise correlation (NC), it is tempting to consider different statistics $\mathcal{T}_1$ and $\mathcal{T}_2$ with corresponding dissimilarity functions d_1 and d_2 and to minimize the sum of the two losses $\mathcal{L}_{SM_1+SM_2} = \mu_1\mathcal{L}_{SM_1} + \mu_2\mathcal{L}_{SM_2}$ where $\mu_1, \mu_2 > 0$ are constant scalars. Indeed if d_1 and d_2 follow the same assumption as previously, we have (see Figure 2 for an illustration):

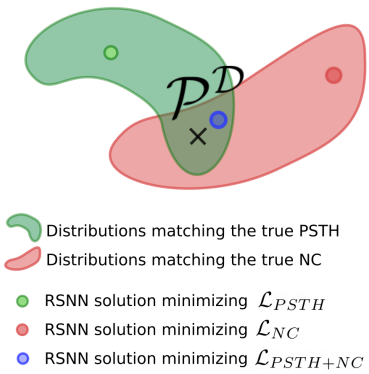

Figure 2: Combining $\mathcal{L}_{PSTH}$ and $\mathcal{L}_{NC}$.

Property 2. *If the RSNN model is expressive enough so that there exists ϕ° for which $\mathcal{P}_\mathcal{D}$ and $\mathcal{P}_\phi$ are indistinguishable according to both statistics $\mathcal{T}_1$ and $\mathcal{T}_2$, then ϕ° is a global minimum for $\mathcal{L}_{SM_1+SM_2}$. Reciprocally, if this minimum is reached, $\mathcal{P}_\mathcal{D}$ and $\mathcal{P}_\phi$ are indistinguishable according to $\mathcal{T}_1$ and $\mathcal{T}_2$.*

This is again a direct consequence of the assumptions on d_1 and d_2. Additionally Figure 1C conveys the idea that $\mathcal{L}_{SM}$ and $\mathcal{L}_{MLE}$ are complementary and $\mathcal{L}_{SM}$ can be interpreted as a log-prior. This interpretation is justified by the following Property which is similar to an essential Property of Bayesian log-priors. It shows that when it is guaranteed to recover the true model by minimizing $\mathcal{L}_{MLE}$, minimizing the regularized likelihood $\mathcal{L}_{MLE+SM}$ will also recover the true model.

Property 3. *If the RSNN is well specified and identifiable so that $\mathcal{P}_\mathcal{D} = \mathcal{P}_{\phi^*}$ and in the limit of infinite data, then the global minimum of $\mathcal{L}_{MLE+SM}$ exists, it is unique and equal to ϕ^*.*

To prove this, we first note that all the conditions are met for the consistency of MLE so ϕ^* is the unique global minimum of $\mathcal{L}_{MLE}$. Also the assumption $\mathcal{P}_\mathcal{D} = \mathcal{P}_{\phi^*}$ is stronger than the assumption required in Properties 1 and 2 (previously ϕ° only needed to match summary statistics: with parameters ϕ^* the model is indeed matching any statistics) so it is also a global minimum of $\mathcal{L}_{SM}$. As a consequence it provides a global minimum for the summed loss $\mathcal{L}_{MLE+SM}$. This solution is also unique because it has to minimize $\mathcal{L}_{MLE}$ which has a unique global minimum.

It may seem that those properties are weaker than the classical properties of GLMs: in particular $\mathcal{L}_{SM+MLE}$ is not convex anymore because of $\mathcal{L}_{SM}$ and the optimization process is not guaranteed to find the global minimum. This could be a disadvantage for $\mathcal{L}_{MLE+SM}$ but we never seemed to encounter this issue in practice. In fact, as we argue later when analyzing our simulation results, the optimum of $\mathcal{L}_{MLE+SM}$ found empirically always seem to be closer to the biological network than the global minimum of $\mathcal{L}_{MLE}$. We think that it happens because the conditions for the consistency of the MLE and Property 3 (identifiability and infinite data) are not fully met in practice. On the contrary, the Properties 1 and 2 hold with a misspecified model or a limited amount of recorded data which may explain the success of the sample-and-measure loss functions in practice. See Figure 1 for a geometrical interpretation.

4 Numerical simulations without hidden neurons

For our first quantitative results we consider a single session of in-vivo recordings from the primary visual cortex of a monkey watching repetitions of the same natural movie [39]. We refer to this dataset as the V1-dataset. It contains the spike trains of 69 simultaneously recorded neurons for 120 repetitions lasting 30 seconds. We only keep the first 80 repetitions in our training set and 10 repetitions are used for early-stopping. Performances are tested on the remaining 30. In our first numerical results we do not include hidden neurons.

To illustrate that minimizing $\mathcal{L}_{MLE}$ alone does not fit well the statistics of interest, we show in Figure 3A-C the learning curves obtained when minimizing $\mathcal{L}_{MLE}$, $\mathcal{L}_{PSTH}$ and $\mathcal{L}_{MLE+PSTH}$. We evaluate the PSTH correlation between simulated and recorded activity every time the training loss function reaches a new minimum. With MLE in Figure 3B, the PSTH correlation saturates at a sub-optimal level and drops unexpectedly when $\mathcal{L}_{MLE}$ decreases. In contrast, with the sample-and-measure loss function, the PSTH correlation improves monotonously and steadily (see Figure 3A).

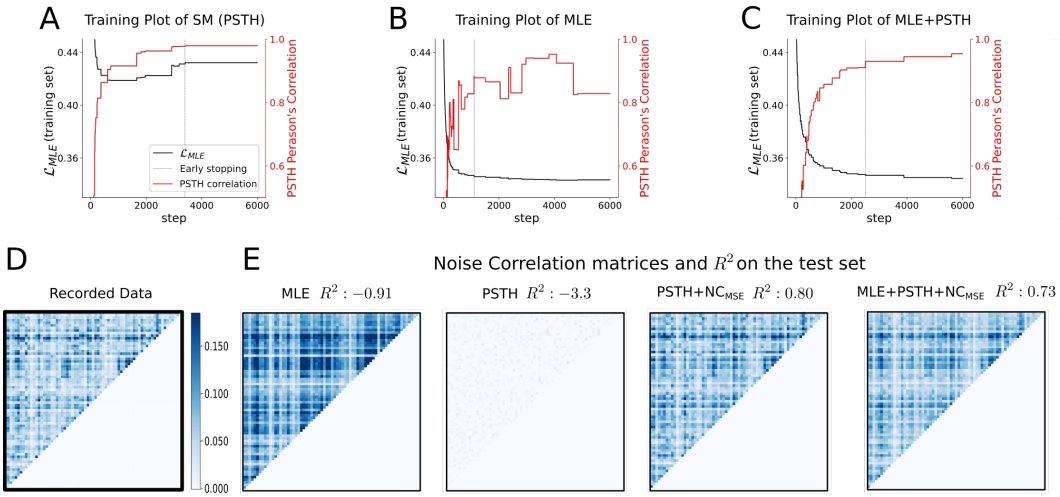

Figure 3: Learning curves and NC performance summary on the V1-dataset. **A-C)** Negative log-likelihood (i.e. $\mathcal{L}_{MLE}$) and PSTH correlation on the training set for three networks trained with $\mathcal{L}_{PSTH}$, $\mathcal{L}_{MLE}$ and $\mathcal{L}_{MLE+PSTH}$. The red curves represent the correlation between the PSTH of the recorded and simulated data. To show how the training loss influences the resulting PSTH correlation we plot a new point of the red curve only when the training loss reaches a new minimum. The vertical blue line represents the best network achieving the lowest validation losses (i.e. $\mathcal{L}_{PSTH}$, $\mathcal{L}_{MLE}$ and $\mathcal{L}_{MLE+PSTH}$ for the three corresponding plots). **D)** Noise-correlation (NC) matrix as recorded in the data. x- and y-axis represent the neuron identities. **E)** NC matrices when the spikes are simulated from the model, it uses the same colorbar as in **D)**. R^2 values are reported to compare the recorded NC and the simulated ones, more results are provided in Table 1.

In Figure 3C, one sees that minimizing $\mathcal{L}_{MLE+SM}$ produces low values of $\mathcal{L}_{MLE}$ and maximizes efficiently the PSTH correlation.

We then fit simultaneously the PSTH and the noise-correlation (NC) on the V1-dataset. The NC matrix is complementary to the PSTH and it is used regularly to measure the fit performance [21, 37, 14]. Its entries can be viewed as a measure of functional connectivity, and each coefficient is defined for the neuron pair i and j as the correlation of their activity. Concretely it is proportional to the statistics $\mathcal{T}(\boldsymbol{z})_{i,j} = \frac{1}{KT}\sum_{k,t}(z^k_{t,i} - \bar{z}_{t,i})(z^k_{t,j} - \bar{z}_{t,j})$ where $\bar{z}_{t,i}$ is the PSTH (see appendix C for details). Therefore the natural sample-and-measure loss function for NC is the mean-squared error between the coefficients $\mathcal{T}(\boldsymbol{z}^{\mathcal{D}})_{i,j}$ and the Monte-carlo estimates $\frac{1}{KT}\sum_{k,t}(\sigma(u^k_{t,i}) - \bar{\sigma}_{t,i})(\sigma(u^k_{t,j}) - \bar{\sigma}_{t,j})$. We denote the resulting loss function as $\mathcal{L}_{NC_{MSE}}$. We also tested an alternative loss $\mathcal{L}_{NC}$ which uses the cross entropy instead of mean-squared error and compares: $\mathcal{T}(\boldsymbol{z}^{\mathcal{D}})_{i,j} = \frac{1}{KT}\sum_{k,t} z^{k,\mathcal{D}}_{t,i} z^{k,\mathcal{D}}_{t,j}$ with the Monte-Carlo estimate $\frac{1}{KT}\sum_{k,t}\sigma(u^k_{t,i})\sigma(u^k_{t,j})$.

We compare quantitatively the effects of the loss functions $\mathcal{L}_{MLE}$, $\mathcal{L}_{PSTH}$, $\mathcal{L}_{NC}$ and $\mathcal{L}_{NC_{MSE}}$ and their combinations on the V1-dataset. The results are summarized in Table 1 and NC matrices are shown in Figure 3E. The network fitted solely with $\mathcal{L}_{PSTH}$ shows the highest PSTH correlation while its noise correlation is almost zero everywhere (see Figure 3E), but this is corrected when adding $\mathcal{L}_{NC}$ or $\mathcal{L}_{NC_{MSE}}$. In fact a network minimizing $\mathcal{L}_{MLE}$ alone yields lower performance than minimizing $\mathcal{L}_{PSTH+NC}$ for both metrics. When combining all losses into $\mathcal{L}_{MLE+PSTH+NC}$ or $\mathcal{L}_{MLE+PSTH+NC_{MSE}}$ the log-likelihood on the test set is not compromised and it fits better the PSTH and the NC: the coefficient of determination R^2 of the NC matrix improves by a large margin in comparison with the MLE solution. Analyzing the failure of MLE we observe in Figure 3 that the NC coefficients are overestimated. We wondered if the fit was mainly impaired by trials with unrealistically high activity as in Figure 1E. But that does not seem to be the case, because the fit remains low with MLE ($R^2 = -0.78$) even when we discard trials where the firing probability of a neuron is higher than 0.85 for 10 consecutive time steps.

We report in Figure S2 the PSTH correlation and noise-correlation in a different format to enable a qualitative comparison with the results obtained with a spike-GAN on the same dataset (see

Table 1: Performance summary on the test set when fitting RSNN models to the V1-dataset. The precise definition of the performance metrics are given in Appendix C. The standard deviation across neurons is provided for the PSTH correlation. The variability across different network initialization is relatively low in comparison with the difference across algorithms, for instance we computed the standard deviation of $\mathcal{L}_{MLE}$ over 3 seeds for MLE and MLE+PSTH+NC$_{MSE}$ and found respectively $2 \cdot 10^{-5}$ and $3 \cdot 10^{-4}$. For the noise correlation R^2, the standard deviation was $9 \cdot 10^{-3}$ for MLE+PSTH+NC$_{MSE}$.

Method	PSTH correlation	Noise Correlation (R^2)	Negative log-likelihood $\mathcal{L}_{MLE}$ (on test set)
MLE	0.67 ± 0.16	-0.91	0.370
PSTH	0.72 ± 0.15	-3.3	0.44
PSTH+NC$_{MSE}$	0.69 ± 0.15	0.80	0.50
MLE+PSTH+NC$_{MSE}$	0.69 ± 0.15	0.73	0.370

Figure 3B,C from [21]). The fit is qualitatively similar even if we do not need a separate discriminator network. Also our RSNN model is better suited to make interpretations about the underlying circuitry because it models explicitly the neural dynamics and the recurrent interactions between the neurons (whereas a generic stochastic binary CNN without recurrent connections was used with the spike-GAN).

We also compare our approach with the 2-step method which is a contemporary alternative to MLE for fitting RSNNs [14]. The PSTH and noise correlation obtained with the 2-step method were measured on another dataset of 25 neurons recorded in-vitro in the retina of the Rat [9]. We trained our method on the same dataset under the two stimulus conditions and a quantitative comparison is summarized in Table S6. Under a moving bar stimulus condition we achieve a higher noise correlation (3% increase) and a higher PSTH correlation (19% increase). But this difference might be explained by the use of a linear-simulus filter [14] instead of a CNN. Under a checkerboard stimulus condition, the 2-step method was tested with a CNN but we still achieve a better noise-correlation (5% improvement) with a slightly worse PSTH correlation (2% decrease). Another difference is that it is not clear how the 2-step method can be extended to model the activity of hidden neurons as done in the following section.

In summary, this section shows that using a differentiable simulator and simple sample-and-measure loss functions leads to a competitive generative model of neural activity. The approach can also be generalized to fit single-trial statistics as explained in the Appendix D and Figure S3.

5 Model identification

Beyond simulating realistic activity statistics, we want the RSNN parameters to reflect a truthful anatomical circuitry or realistic neural properties. To test this, we consider a synthetic dataset generated by a *target network* for which we know all the parameters. We build this target network by fitting it to the V1-dataset and sample from this model a synthetic dataset of similar size as the V1-dataset (80 training trials of approximately 30 seconds). Since our target network can generate as much data as we want, we simulate a larger test set of 480 trials and a larger validation set of 40 trials. We then fit *student networks* on this synthetic dataset and compare the parameters ϕ of the *student* networks with the ground-truth parameters ϕ^* of the *target* networks.

Well specified model without hidden neurons As a first comparison we consider the simplest case where the target network is fully observed: the target network consist of 69 visible neurons and each student network is of the same size. This is in fact the ideal setting where the log-likelihood is tractable and the MLE enjoys strong theoretical guarantees. In particular if the CNN weights are not trained and are copied from the target-network, the loss function $\mathcal{L}_{MLE}$ is convex with respect to the remaining RSNN parameters ϕ and the target network is identifiable [45]. The resulting fitting performance is summarized in Figure 4A where we show the NC matrix and the connectivity matrix $(\sum_d W^d_{i,j})$ for the target network and two students networks. We do not show the PSTH because all methods already fit it well on the V1-dataset (see Table 1). In this setting, combining $\mathcal{L}_{NC}$ and

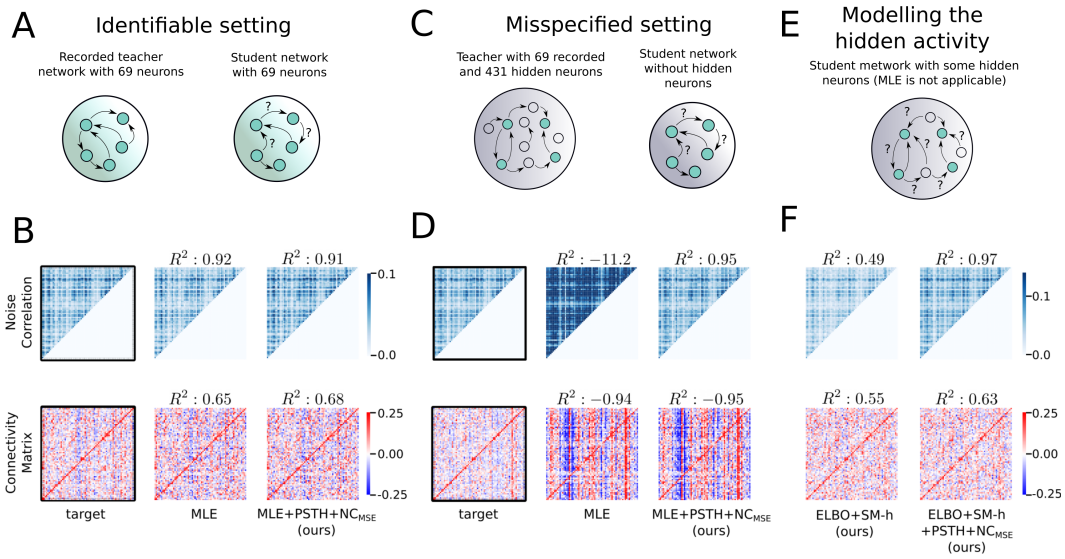

Figure 4: **A, B**) Results of the model identification experiment in the fully identifiable setting. **A** summarizes the simulation setup. The results are shown in **B**: in the first row, we show the noise correlation matrix. In the second row, we show the connectivity matrix $\sum_d W^d_{ji}$ where the x- axis indicates pre-synaptic neuron i and y-axis indicates post-synaptic neuron j. **C, D**) Same as **A, B** but in a misspecified setting: the target network has 500 neurons and student networks have 69 neurons. **E, F**) Same as before with the same target network as in **C, D**, but the student has some hidden neurons. When averaging R^2 of the connectivity matrices over 5 different student network initialization, we find $R^2 = 0.56 \pm 0.0069$ for ELBO+SM-h and 0.63 ± 0.013 for ELBO+SM-h+PSTH+NC$_{MSE}$. For the noise correlation, we find 0.66 ± 0.1 for ELBO+SM-h and 0.95 ± 0.2 for ELBO+SM-h+PSTH+NC$_{MSE}$. Here the number of hidden neurons in the student network is the same as in the target network, but this is not necessary as seen in Table S4. The connectivity matrices are displayed with higher resolution in Figure S5 and S6.

$\mathcal{L}_{PSTH}$ with $\mathcal{L}_{MLE}$ brings almost no advantage: the MLE already provides a good reconstruction of the NC and connectivity matrices.

Model misspecification when ignoring hidden neurons From these results we hypothesize that this fully identifiable setting does not capture the failure of MLE observed with real data because the recorded neurons are embedded in a much larger biological network that we cannot record from. To model this, we construct another synthetic dataset based on a larger target network of 500 neurons where the first 69 neurons are fitted to the neurons recorded in the V1-dataset and the remaining 431 are only regularized to produce a realistic mean firing rate (see appendix for simulation details). As in the standard setting where one ignores the presence of hidden neurons, we first consider that the student networks model only the first 69 visible neurons. This model is therefore misspecified because the number of neurons are different in the target and student networks, hence this setting is well described by the scenario sketched in Figure 1B.

The results are shown in Figure 4B. We found that MLE is much worse than the sample-and-measure method in this misspecified setting and the results resemble better what has been observed with real data. With MLE the noise-correlation coefficient are over estimated and the overall fit is rather poor (negative R^2), but it significantly improves after adding the sample-and-measure loss functions ($R^2 = 0.95$). This suggest that ignoring the impact of hidden neurons can explain the failure of MLE experienced in the real V1-dataset. We find little relationship between the student and teacher connectivity matrices (only the connectivity between visible neurons are compared, see Figure 3). This suggests that the standard strategy, where the hidden neurons are ignored, is unlikely to be informative about true cortical connectivity.

Well specified model with hidden neurons To investigate whether including hidden neurons leads to more truthful network models, we take the same target network of 500 neurons and fit now student

networks of the same size (500 neurons) but where only the first 69 are considered visible (Figure 4C). Since the model is well specified but data about the hidden neurons is missing, this experiment is well summarized by the scenario of Figure 1C. We use $\mathcal{L}_{ELBO}$ for the visible units and we add a sample-and-measure loss function $\mathcal{L}_{SM-h}$ to constrain the average firing rate of the hidden neurons which are completely unconstrained otherwise (see appendix). As seen in Figure 4C, it yields more accurate NC matrix ($R^2 = 0.49$) and connectivity matrix ($R^2 = 0.55$) compared to the previous misspecified models which did not include the hidden neurons. When we add sample-and-measure loss functions to fit the PSTH and NC of the visible neurons, the noise-correlation matrix and connectivity matrix are fitted even better ($R^2 = 0.97$ and $R^2 = 0.63$). Quantitatively, the R^2 for the connectivity matrices are almost as high as in the easy setting of panel A where the target network is fully-visible and identifiable. Although the student network had exactly the same number of hidden neurons in Figure 4 E-F, the success is not dependent on the exact number of hidden neurons as shown in Table S4. Rather, assuming a small hidden population size of only 10 neurons was enough to alleviate the failure observed in the absence of hidden neurons in Figure 4 C-D. Table S4 also shows however that the accuracy of the reconstruction improves substantially if the hidden population is made larger in the student network.

6 Discussion

We have introduced the sample-and-measure method for fitting an RSNN to spike train recordings. This method leverages deep learning software and back-propagation for stochastic RSNNs to minimize sample-and-measure loss functions. A decisive feature of this method is to model simply and efficiently the activity of hidden neurons. We have shown that this is important to reconstruct trustworthy connectivity matrices in cortical areas. We believe that our approach paves the way towards better models with neuroscientifically informed biases to reproduce accurately the recorded activity and functional connectivity. Although we have focused here on GLMs, PSTH and NC, the method is applicable to other spiking neuron models and statistics.

Perspective One of the promising aspects of our method is to fit models which are much larger. One way to do this, is to combine neurons from separate sessions in a single larger network by considering them alternatively visible or hidden. This problem was tackled partially in [46, 47]. It is natural to implement this with our method and we believe that it is a decisive step to produce models with a dense coverage of the recorded areas.

To investigate if our method is viable in this regime we carried out a prospective scaling experiment on a dataset from the Mouse brain recorded with multiple Neuropixels probes across 58 sessions [48]. The goal of this scaling experiment is only to evaluate the amount of computing resources required to fit large networks. We ran three fitting experiments with 2, 10 and 20 sessions respectively. Focusing on neurons from the visual cortices, it yielded models with 527, 2219 and 4995 neurons respectively. Each simulation could be run on a single NVIDIA V100 GPU and running 100 training epochs took approximately 4, 12 and 36 hours respectively. We conclude that this large simulation paradigm is approachable with methods like ours and we leave the fine-tuning of these experiments and the analysis of the results for future work.

Acknowledgments and Disclosure of Funding

This research was supported by Swiss National Science Foundation (no. 200020_184615) and the Intel Neuromorphic Research Lab. Many thanks to Christos Sourmpis, Gabriel Mahuas, Ulisse Ferrari, Franz Scherr and Wolfgang Maass for helpful discussions. Special thanks to Stéphane Deny, Olivier Marre and Ulisse Ferrari for sharing with us the Retina dataset and to Matthew Smith and Adam Kohn for making their dataset publicly available.

Authors contributions GB and SW conceived the project initially. SW did most of the simulations under the supervision of GB. All authors contributed significantly to the theory and the writing.

References

[1] Jonathan W Pillow, Jonathon Shlens, Liam Paninski, Alexander Sher, Alan M Litke, EJ Chichilnisky, and Eero P Simoncelli. Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, 454(7207):995–999, 2008.

[2] Caroline A Runyan, Eugenio Piasini, Stefano Panzeri, and Christopher D Harvey. Distinct timescales of population coding across cortex. *Nature*, 548(7665):92–96, 2017.

[3] Rajeev V Rikhye, Aditya Gilra, and Michael M Halassa. Thalamic regulation of switching between cortical representations enables cognitive flexibility. *Nature neuroscience*, 21(12):1753–1763, 2018.

[4] Felipe Gerhard, Tilman Kispersky, Gabrielle J Gutierrez, Eve Marder, Mark Kramer, and Uri Eden. Successful reconstruction of a physiological circuit with known connectivity from spiking activity alone. *PLoS Comput Biol*, 9(7):e1003138, 2013.

[5] Ryota Kobayashi, Shuhei Kurita, Anno Kurth, Katsunori Kitano, Kenji Mizuseki, Markus Diesmann, Barry J Richmond, and Shigeru Shinomoto. Reconstructing neuronal circuitry from parallel spike trains. *Nature communications*, 10(1):1–13, 2019.

[6] Skander Mensi, Richard Naud, Christian Pozzorini, Michael Avermann, Carl CH Petersen, and Wulfram Gerstner. Parameter extraction and classification of three cortical neuron types reveals two distinct adaptation mechanisms. *Journal of neurophysiology*, 107(6):1756–1775, 2012.

[7] Christian Pozzorini, Richard Naud, Skander Mensi, and Wulfram Gerstner. Temporal whitening by power-law adaptation in neocortical neurons. *Nature neuroscience*, 16(7):942–948, 2013.

[8] Corinne Teeter, Ramakrishnan Iyer, Vilas Menon, Nathan Gouwens, David Feng, Jim Berg, Aaron Szafer, Nicholas Cain, Hongkui Zeng, Michael Hawrylycz, et al. Generalized leaky integrate-and-fire models classify multiple neuron types. *Nature communications*, 9(1):1–15, 2018.

[9] Stephane Deny, Ulisse Ferrari, Emilie Mace, Pierre Yger, Romain Caplette, Serge Picaud, Gašper Tkačik, and Olivier Marre. Multiplexed computations in retinal ganglion cells of a single type. *Nature communications*, 8(1):1–17, 2017.

[10] Abhranil Das and Ila R Fiete. Systematic errors in connectivity inferred from activity in strongly recurrent networks. *Nature Neuroscience*, 23(10):1286–1296, 2020.

[11] Jonathan W Pillow and Peter Latham. Neural characterization in partially observed populations of spiking neurons. *Adv Neural Information Processing Systems*, 20(3.5), 2008.

[12] Wulfram Gerstner. Time structure of the activity in neural network models. *Physical review E*, 51(1):738, 1995.

[13] Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

[14] Gabriel Mahuas, Giulio Isacchini, Olivier Marre, Ulisse Ferrari, and Thierry Mora. A new inference approach for training shallow and deep generalized linear models of noisy interacting neurons. *Advances in neural information processing systems*, 2020.

[15] David Hocker and Il Memming Park. Multistep inference for generalized linear spiking models curbs runaway excitation. In *2017 8th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 613–616. IEEE, 2017.

[16] Felipe Gerhard, Moritz Deger, and Wilson Truccolo. On the stability and dynamics of stochastic spiking neuron models: Nonlinear hawkes process and point process glms. *PLoS computational biology*, 13(2):e1005390, 2017.

[17] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[18] Tapani Raiko, Mathias Berglund, Guillaume Alain, and Laurent Dinh. Techniques for learning binary stochastic feedforward neural networks. *arXiv preprint arXiv:1406.2989*, 2014.

[19] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Advances in Neural Information Processing Systems*, pages 787–797, 2018.

[20] David Sussillo, Rafal Jozefowicz, LF Abbott, and Chethan Pandarinath. Lfads-latent factor analysis via dynamical systems. *arXiv preprint arXiv:1608.06315*, 2016.

[21] Poornima Ramesh, Mohamad Atayi, and Jakob H Macke. Adversarial training of neural encoding models on population spike trains. *NeurIPS workshop 2019: Real Neurons and Hidden Units*, 2019.

[22] Vernon Lawhern, Wei Wu, Nicholas Hatsopoulos, and Liam Paninski. Population decoding of motor cortical activity using a generalized linear model with hidden states. *Journal of neuroscience methods*, 189(2):267–280, 2010.

[23] Johanni Brea, Walter Senn, and Jean-Pascal Pfister. Sequence learning with hidden units in spiking neural networks. *Advances in neural information processing systems*, 24:1422–1430, 2011.

[24] Johanni Brea, Walter Senn, and Jean-Pascal Pfister. Matching recall and storage in sequence learning with spiking neural networks. *Journal of neuroscience*, 33(23):9565–9575, 2013.

[25] Danilo Jimenez Rezende and Wulfram Gerstner. Stochastic variational learning in recurrent spiking networks. *Frontiers in computational neuroscience*, 8:38, 2014.

[26] Diego M Arribas, Yuan Zhao, and Il Memming Park. Rescuing neural spike train models from bad mle. *NeurIPS 2020*, 2020.

[27] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[28] George Papamakarios and Iain Murray. Fast ε-free inference of simulation models with bayesian conditional density estimation. In *Advances in Neural Information Processing Systems*, pages 1028–1036, 2016.

[29] Jan-Matthis Lueckmann, Pedro J Goncalves, Giacomo Bassetto, Kaan Öcal, Marcel Nonnenmacher, and Jakob H Macke. Flexible statistical inference for mechanistic models of neural dynamics. In *Advances in neural information processing systems*, pages 1289–1299, 2017.

[30] Pedro J Gonçalves, Jan-Matthis Lueckmann, Michael Deistler, Marcel Nonnenmacher, Kaan Öcal, Giacomo Bassetto, Chaitanya Chintaluri, William F Podlaski, Sara A Haddad, Tim P Vogels, et al. Training deep neural density estimators to identify mechanistic models of neural dynamics. *bioRxiv*, page 838383, 2020.

[31] Sean R Bittner, Agostina Palmigiano, Alex T Piet, Chunyu A Duan, Carlos D Brody, Kenneth D Miller, and John P Cunningham. Interrogating theoretical models of neural computation with deep inference. *bioRxiv*, page 837567, 2019.

[32] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[33] Manuel Molano-Mazon, Arno Onken, Eugenio Piasini, and Stefano Panzeri. Synthesizing realistic neural population activity patterns using generative adversarial networks. *arXiv preprint arXiv:1803.00338*, 2018.

[34] Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727. PMLR, 2015.

[35] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. Ddsp: Differentiable digital signal processing. *arXiv preprint arXiv:2001.04643*, 2020.

[36] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

[37] Jakob H Macke, Lars Buesing, John P Cunningham, Byron M Yu, Krishna V Shenoy, and Maneesh Sahani. Empirical models of spiking in neural populations. *Advances in neural information processing systems*, 24:1350–1358, 2011.

[38] Lane T McIntosh, Niru Maheswaranathan, Aran Nayebi, Surya Ganguli, and Stephen A Baccus. Deep learning models of the retinal response to natural scenes. *Advances in neural information processing systems*, 29:1369, 2016.

[39] Matthew A Smith and Adam Kohn. Spatial and temporal scales of neuronal correlation in primary visual cortex. *Journal of Neuroscience*, 28(48):12591–12603, 2008.

[40] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[41] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[42] George Tucker, Andriy Mnih, Chris J Maddison, Dieterich Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. *arXiv preprint arXiv:1703.07370*, 2017.

[43] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks. *IEEE Signal Processing Magazine*, 36:61–63, 2019.

[44] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[45] Liam Paninski. Maximum likelihood estimation of cascade point-process neural encoding models. *Network: Computation in Neural Systems*, 15(4):243–262, 2004.

[46] Oleksandr Sorochynskyi, Stéphane Deny, Olivier Marre, and Ulisse Ferrari. Predicting synchronous firing of large neural populations from sequential recordings. *PLoS computational biology*, 17(1):e1008501, 2021.

[47] Srini Turag, Lars Buesing, Adam M Packer, Henry Dalgleish, Noah Pettit, Michael Hausser, and Jakob H Macke. Inferring neural population dynamics from multiple partial recordings of the same neural circuit. *NIPS*, 2014.

[48] Joshua H Siegle, Xiaoxuan Jia, Séverine Durand, Sam Gale, Corbett Bennett, Nile Graddis, Greggory Heller, Tamina K Ramirez, Hannah Choi, Jennifer A Luviano, et al. Survey of spiking in the mouse visual system reveals functional hierarchy. *Nature*, pages 1–7, 2021.

[49] Adam Kohn and Matthew A Smith. Utah array extracellular recordings of spontaneous and visually evoked activity from anesthetized macaque primary visual cortex (v1). *CRCNS. org*, 2016.

[50] William F Kindel, Elijah D Christensen, and Joel Zylberberg. Using deep learning to probe the neural code for images in primary visual cortex. *Journal of vision*, 19(4):29–29, 2019.

Appendices of:

Fitting summary statistics of neural data with a differentiable spiking network simulator

A Datasets

V1-dataset The dataset we used was collected by Smith and Kohn [49] and is publicly available at: http://crcns.org/data-sets/vc/pvc-11. In summary, macaque monkeys were anesthetized with Utah arrays placed in the primary visual cortex (V1). In our analysis, we considered population spiking activity of monkey-I in response to a gray-scale natural movie. The movie is about a monkey wading through water. It lasts for 30 seconds (with sampling rate 25Hz) and was played repeatedly for 120 times. Similarly as in [21], we used the last 26 seconds of the movies and recordings. Each frame of the movie has 320×320 pixels and we downsampled them to 27×27 pixels. We used the recording from the 69 neurons with time bins 40ms and considered that there cannot be more than one spike per bin (5% of the time bins had more than one spike).

Synthetic dataset Two target networks are trained using the V1-dataset: one with no hidden neuron and one with 431 hidden neurons which makes 500 neurons in total. To build the target network without hidden neurons, we fitted a network with the loss function $\mathcal{L}_{MLE+PSTH+NC}$. For the target network with hidden neurons, we train a network using $\mathcal{L}_{MLE+SM-h+PSTH+NC}$.

Retina dataset The data we used were the same as [14] and was initially published in [9]. It was generously shared with us privately. It contained recorded spike trains for 25 OFF Alpha retinal ganglion cells' in the form of binarized spike counts in 1.667ms bins. There were two stimulus conditions. For the checkerboard, the unrepeated movie (1080s) plus one repeated movie (600s in total for 120 repetitions) were used for training and the other repeated movie (480s in total for 120 repetitions) were used for testing. For the moving bar, the unrepeated movie (1800s) plus one repeated movie (166s in total for 50 repetitions) were used for training and the other repeated movie (322s in total for 50 repetitions) were used for testing.

B Simulation details

For the V1- and synthetic datasets The model combines a spatio-temporal CNN and an RSNN. Input to the CNN consists of 10 consecutive movie frames. The CNN has 2 hidden layers and its output is fed into the RSNN. To feed the images to the CNN the 10 gray-scaled images are concatenated on the channel dimension. The two hidden layers include convolution with 16 and 32 filters, size 7 by 7 (with padding) followed by a ReLU activation function and then a MaxPool layer with kernel size 3 and stride 2 as in [50]. The weights from the CNN to the RSNN are initialized with a truncated normal distribution with standard deviation $\frac{1}{\sqrt{n_{in}}}$ where n_{in} in the number of inputs in the weight matrix. The tensor of recurrent weights $\boldsymbol{W}$ consider spike history of last 9 frames ($d_{\max}$) and the weight distribution is initialized as a truncated normal distribution with standard deviation $\frac{1}{\sqrt{d_{\max} n_{\mathcal{H}+\mathcal{V}}}}$. The bias $\boldsymbol{b}$ is initialized with zero. The voltage threshold v_{thr} is set to 0.4 and the dampening factor γ is 0.3. We used an Adam optimizer. More hyper-parameters like learning rates are given in Table S2 and Table S3. To implement the loss $\mathcal{L}_{MLE+PSTH+NC}$, we process the CNN once and simulate the RSNN twice. Once the RSNN is clamped to the recorded spikes to compute $\mathcal{L}_{MLE}$ or $\mathcal{L}_{ELBO}$, the second time the sample and generated "freely" to compute $\mathcal{L}_{PSTH}$ and $\mathcal{L}_{NC}$.

For the retina dataset experiment Since the time step is much smaller for the Retina dataset than for the V1-dataset (1.67ms rather than 40ms) the temporal filters have to be larger to take into account the full temporal context. For both the receptive fields of the CNN and the tensor $\boldsymbol{W}$ we chose to cover time scales that are consistent with [14]. Hence we adapted the model architecture from the previous paragraph and added as a first layer of the CNN a causal temporal convolution (Conv1D with appropriate padding). The temporal convolution has a receptive filed of 300 time bins and outputs 16 filters. In the RSNN we choose $d_{\max} = 24$ so that the spike history filter covers around 40ms. Two fitting algorithms were tested, one with $\mathcal{L}_{MLE}$ and the other one with $\mathcal{L}_{MLE+single-trial+NC}$. The loss function $\mathcal{L}_{single-trial}$ is used to fit single-trial statistics as defined in Appendix D and we used it here to replace $\mathcal{L}_{PSTH}$ because some movies of training dataset are unrepeated and we saw in Figure S3 that it fits the PSTH almost as well as $\mathcal{L}_{PSTH}$. To implement the

loss $\mathcal{L}_{MLE+single-trial+NC}$, we process the CNN once and simulate the RSNN twice for T time steps. The first time the RSNN is clamped to the recorded spikes for T_{gt} time steps and then clamping is terminated and the RSNN generates samples "freely" for the next $T - T_{gt}$ time steps. For the first T_{gt} time steps, $\mathcal{L}_{MLE}$ is computed. And for the rest $T - T_{gt}$ time steps where the activity is not clamped, $\mathcal{L}_{single-trial}$ is computed as the cross entropy between $\boldsymbol{z}^{\mathcal{D}}$ and the spike probabilities. We also run the RSNN a second time with the same CNN input and without any clamping to compute $\mathcal{L}_{NC}$. For each gradient descent step, we sample uniformly from the dataset a batch of size $K^{\mathcal{D}} = K_m \times K_t$ gathering truncated movie clips and corresponding spikes with K_m different starting time points and from K_t different movies. The hyper-parameters can be found in Table S5.

C Performance metrics

For the definition of our performance metrics we use the following notations. The trial averaged firing probability of neuron i in the time bin t is denoted $\bar{z}_{t,i} = \frac{1}{K}\sum_k z^k_{t,i}$ where $z^k_{t,i} \in \{0,1\}$ is the spike and K is the number of trials. Neuron i's mean firing rate is further computed as $\bar{z}_i = \frac{1}{T}\sum_t \bar{z}_{t,i}$ where T is the number of time steps.

Peristimulus time histogram (PSTH) correlation The fit performance of the PSTH is measured by the Pearson's correlation between the simulated PSTH and the recorded PSTH. Hence for each neuron the PSTH correlation is defined by:

$$\rho_i^{\text{PSTH}} = \frac{\sum_t \left(\bar{z}_{t,i} - \bar{z}_i\right)\left(\bar{z}^{\mathcal{D}}_{t,i} - \bar{z}^{\mathcal{D}}_i\right)}{\sqrt{\sum_t \left(\bar{z}_{t,i} - \bar{z}_i\right)^2}\sqrt{\sum_t \left(\bar{z}^{\mathcal{D}}_{t,i} - \bar{z}^{\mathcal{D}}_i\right)^2}}\,, \tag{7}$$

and a slightly better estimator of the asymptotical Pearson correlation which is less noisy can be estimated by replacing $\bar{z}_{t,i}$ and $\bar{z}_i$ with $\bar{\sigma}_{t,i}$ and $\bar{\sigma}_i$.

Noise-correlation matrix Pairwise noise correlations are computed as in [46]. We first define total covariance $M^{\text{total}}_{i,j}$ and noise covariance $M^{\text{noise}}_{i,j}$ between neuron i and j.

$$M^{\text{total}}_{i,j} = \frac{1}{TK}\sum_{t,k}\left(z^k_{t,i} - \bar{z}_i\right)\left(z^k_{t,j} - \bar{z}_j\right) \tag{8}$$

$$M^{\text{noise}}_{i,j} = \frac{1}{TK}\sum_{t,k}\left(z^k_{t,i} - \bar{z}_{t,i}\right)\left(z^k_{t,j} - \bar{z}_{t,j}\right) \tag{9}$$

Then in the performance tables we report the normalized noise correlation $\mathcal{M}^{\text{noise}}_{i,j}$ for $i \neq j$:

$$\mathcal{M}^{\text{noise}}_{i,j} = \frac{M^{\text{noise}}_{i,j}}{\sqrt{M^{\text{total}}_{i,i} M^{\text{total}}_{j,j}}}\,. \tag{10}$$

We then define the coefficient of determination of the NC matrix R^2 as in [14]. Given the NC matrices computed from the data $\mathcal{M}^{\text{noise},\mathcal{D}}_{i,j}$ and the NC matrix obtained from the simulation $\mathcal{M}^{\text{noise},\phi}_{i,j}$ we define $\mathcal{M}^{\text{noise},\mathcal{D}} = \frac{1}{n_{\mathcal{V}}{}^2}\sum_{i,j}\mathcal{M}^{\text{noise},\mathcal{D}}_{i,j}$ and:

$$R^2 = 1 - \frac{\sum_{i,j}\left(\mathcal{M}^{\text{noise},\mathcal{D}}_{i,j} - \mathcal{M}^{\text{noise},\phi}_{i,j}\right)^2}{\sum_{i,j}\left(\mathcal{M}^{\text{noise},\mathcal{D}}_{i,j} - \mathcal{M}^{\text{noise},\mathcal{D}}\right)^2}\,. \tag{11}$$

D Derivations of the loss functions

Normalization of the sample-and-measure functions Most sample-and-measure may be defined one multiplicative constant away from their formal definition. For instance when computing $\mathcal{L}_{MLE}$ we compute the binary cross entropy between the relevant probabilities aggregate them by taking the mean and not the sum. We find the resulting number to be easier to interpret because is it independent from the number of trials and the number of time steps.

Noise correlation We tested two sample-and-measure loss function for the noise correlation. We explain here why the Monte-Carlo estimate of the simulated statistics is unbiased for $\mathcal{L}_{NC}$ but the same argument applies to $\mathcal{L}_{NC_{MSE}}$.

We consider the statistics $\mathcal{T}(\boldsymbol{z})_{ij} = \frac{1}{KT}\sum_{t,k} z^k_{t,i} z^k_{t,j}$ which measure the frequency of coincident spikes between neurons i and j. Since $z^k_{t,i}$ and $z^k_{t,j}$ are independent given the past, we have $\mathbb{E}_{\mathcal{P}_\phi}\left[z^k_{t,i} z^k_{t,j}\right] =$

$\mathbb{E}_{\mathcal{P}_\phi}\left[\sigma(u^k_{t,i})\sigma(u^k_{t,j})\right]$ so we use the following Monte-Carlo estimate $\pi^\phi_{i,j} = \frac{1}{KT}\sum_{t,k}\sigma(u^k_{t,i})\sigma(u^k_{t,j})$ to evaluate the expected simulated statistics in equation (4). Choosing the dissimilarity d to be the cross entropy and denoting $\pi^\mathcal{D}_{i,j} = \mathcal{T}(\boldsymbol{z}^\mathcal{D})_{ij}$ we define:

$$\mathcal{L}_{NC} = \sum_{i,j} CE(\pi^\mathcal{D}_{i,j}, \pi^\phi_{i,j}) \tag{12}$$

As an attempt to replace the terms in $\mathcal{L}_{NC_{MSE}}$ which take into account the correlation from the PSTH, we tried to add a related correction term in $\mathcal{L}_{NC}$. To do do we considerd another loss $\mathcal{L}_{NC\text{-shuffled}}$ which is computed like $\mathcal{L}_{NC}$ but where we shuffle the trial identities in $\boldsymbol{z}_i$ and not in $\boldsymbol{z}_j$. It seems that it was not as efficient as $\mathcal{L}_{NC_{MSE}}$.

Single-trial statistics Since both PSTH and NC are trial-averaged statistics we wondered whether another simple measuring model could account for single-trial statistics. We therefore considered the following problem which is notoriously challenging for the MLE [15]: we clamp the network to the recorded data until time t and generate a simulated spike train for $t' > t$. With MLE the network activity quickly diverges away from the real data. To measure this quantitatively we estimate the multi-step log-likelihood $\mathcal{P}_\phi(\boldsymbol{z}^\mathcal{D}_{t+\Delta t}|\boldsymbol{z}^\mathcal{D}_0 \cdots \boldsymbol{z}^\mathcal{D}_t)$. It is intractable but an unbiased Monte-Carlo estimate can be computed. The multi-step log-likelihood drops quickly as Δt increases as expected for MLE in Figure S3.

To resolve that issue, we first suggest an extension of the definition of $\mathcal{L}_{SM}$ in equation 4 which formalizes the clamping condition:

$$\mathcal{L}_{SM} = d\big(\ \mathbb{E}_{\mathcal{P}_\mathcal{D}}\left[\ \mathcal{T}(\boldsymbol{z})\mid \boldsymbol{c}\ \right],\ \mathbb{E}_{\mathcal{P}_\phi}\left[\ \mathcal{T}(\boldsymbol{z})\mid \boldsymbol{c}\ \right]\ \big)\ , \tag{13}$$

where we have introduced a condition $\boldsymbol{c}$ into the expectations. All the theory and the geometrical interpretations can be extended with this conditioning, but this allows to formalize that the visible units can be clamped to the recorded data. For instance if we choose $\boldsymbol{c}$ such that $\boldsymbol{z}^\mathcal{V}_{1:t} = \boldsymbol{z}^\mathcal{D}_{1:t}$ we formalize a sample-and-measure loss function for which the visible units are clamped until time t.

Back to the problem of fitting the multi-step log-likelihood, we consider the sample-and-measure loss function where $\mathcal{T}$ is identity, $\sigma(\boldsymbol{u})$ is the Monte-Carlo estimator and d is the cross-entropy. It yields:

$$\mathcal{L}_{single\text{-}trial} = CE(\boldsymbol{z}^\mathcal{D}, \sigma(\boldsymbol{u}^\mathcal{V}))\ , \tag{14}$$

which is pretty much computed like $\mathcal{L}_{MLE}$ but where the data is only clamped until time t. Note that since the statistics $\mathcal{T}$ do not involve a trial average, the computation of the expectation is not very precise but it may be improved for the expectation $\mathbb{E}_{\mathcal{P}_\phi}$ by averaging over multiple batches clamped to the same data. Although this is an interesting direction we did not try it and always sample a single batch per clamping condition. When using this loss function, we see in Figure S3B that MLE only better just at the first time step after the clamping terminates and optimizing $\mathcal{L}_{single\text{-}trial}$ makes better prediction after that. To provide a meaningful baseline we show the m-step likelihood obtained with a theoretical model fitting perfectly the PSTH without being aware of the clamping history. The multi-step likelihood obtained with $\mathcal{L}_{single\text{-}trial}$ is above this baseline for 5 time-steps (200ms) on the training set proving that the model tries to make a clever usage of the trial specific firing history up to this duration.

Derivation of the ELBO Like for capturing single trial statistics, the most natural way to fit neural activity in the presence of hidden neurons is to minimize the cross-entropy between the visible spikes and their probability while sampling from the hidden neurons. Here we want to show that this is actually the negative of a variational lower bound of the maximum likelihood. Following [40], for any distribution $q(\boldsymbol{z}^\mathcal{H})$ of the hidden neural activity we have:

$$\begin{aligned}
\log \mathcal{P}_\phi(\boldsymbol{z}^\mathcal{D}) &= \log \sum_{\boldsymbol{z}^\mathcal{H}} \mathcal{P}_\phi(\boldsymbol{z}^\mathcal{D}, \boldsymbol{z}^\mathcal{H}) && (15)\\
&= \log \sum_{\boldsymbol{z}^\mathcal{H}} q(\boldsymbol{z}^\mathcal{H}) \frac{\mathcal{P}_\phi(\boldsymbol{z}^\mathcal{D}, \boldsymbol{z}^\mathcal{H})}{q(\boldsymbol{z}^\mathcal{H})} && (16)\\
&\geq \sum_{\boldsymbol{z}^\mathcal{H}} q(\boldsymbol{z}^\mathcal{H}) \log \frac{\mathcal{P}_\phi(\boldsymbol{z}^\mathcal{D}, \boldsymbol{z}^\mathcal{H})}{q(\boldsymbol{z}^\mathcal{H})} && (17)
\end{aligned}$$

Writing $\boldsymbol{z}_t$ as the concatenation of $\boldsymbol{z}^\mathcal{D}_t$ and $\boldsymbol{z}^\mathcal{H}_t$, we now choose specifically q so that for all t: $q(\boldsymbol{z}^\mathcal{H}_t) = \mathcal{P}_\phi(\boldsymbol{z}^\mathcal{H}_t|\boldsymbol{z}_{1:t-1})$, using the factorization and seeing that the probability factorizes as follows: $\mathcal{P}_\phi(\boldsymbol{z}^\mathcal{D}, \boldsymbol{z}^\mathcal{H}) = \prod_t \mathcal{P}_\phi(\boldsymbol{z}^\mathcal{D}_t, \boldsymbol{z}^\mathcal{H}_t|\boldsymbol{z}_{1:t-1}) = \prod_t \mathcal{P}_\phi(\boldsymbol{z}^\mathcal{D}_t|\boldsymbol{z}_{1:t-1}) \cdot \prod_t \mathcal{P}_\phi(\boldsymbol{z}^\mathcal{H}_t|\boldsymbol{z}_{1:t-1})$, some products inside the log are cancelling out and we found the lower bound:

$$\begin{aligned}
\log \mathcal{P}_\phi(\boldsymbol{z}^\mathcal{D}) &\geq \mathbb{E}_q\left[\sum_t \log \mathcal{P}_\phi(\boldsymbol{z}^\mathcal{D}_t|\boldsymbol{z}_{1:t-1})\right] && (18)\\
&= -\mathbb{E}_q\left[CE(\boldsymbol{z}^\mathcal{D}, \sigma(\boldsymbol{u}^\mathcal{D}))\right], && (19)
\end{aligned}$$

Method	learning rate	batch size	μ_{PSTH}	μ_{NC}	μ_{MLE}
MLE	1e-3	20	0	0	1
PSTH			1	0	0
MLE+PSTH			0.5	0.5	0
PSTH+NC			0.11	0.89	0
MLE+PSTH+NC			0.1	0.5	0.4
MLE+PSTH+NC$_{MSE}$			0.1	50	0.4

Table S2: Hyper-parameter table used when fitting the V1-dataset (Figure 3).

Method	learning rate	batch size	μ_{PSTH}	μ_{NC}	μ_{MLE}	μ_{SM-h}
MLE	1.5e-3	20	0	0	1	0
MLE+PSTH+NC			0.1	0.7	0.2	0
MLE+SM-h			0	0	0	1e-3
MLE+SM-h+PSTH+NC			0.1	0.7	0.2	1e-3

Table S3: Hyper-parameter table used when fitting the synthetic dataset (Figure 4). Early-stopping was used on 40 validation trials to prevent over-fitting.

Interestingly, a similar loss function can also be formulated as a sample-and-measure loss function. To do so we consider the definition from equation (13) with the condition $\boldsymbol{c}$ being $\boldsymbol{z}^{\mathcal{V}} = \boldsymbol{z}^{\mathcal{D}}$ meaning that all the visible units are clamped to the data. Choosing otherwise $\mathcal{T}$ to be the identity and d as the cross-entropy, we obtain the following loss function denoted as $\mathcal{L}_{ELBO-SM}$:

$$\mathcal{L}_{ELBO-SM} = CE\left(\boldsymbol{z}^{\mathcal{D}}, \mathbb{E}_{\mathcal{P}_\phi}\left[\sigma(\boldsymbol{u}^{\mathcal{D}})\right]\right). \tag{20}$$

Comparing the two loss functions we see that the essential difference is the placement of the expectation $\mathbb{E}_{\mathcal{P}_\phi}$. In practice our current optimization minimizes (19) rather than (20) because we sample a single trial for each clamping condition and apply stochastic gradient descent with momentum. This implements implicitly the averaging of the gradients which corresponds better to the expectation from equation (19). However it is also possible to minimize (20) by averaging the Monte-Carlo estimates obtained with multiple simulations with the same clamping condition. With enough sample it may provides a better estimate of the expectation $\mathbb{E}_{\mathcal{P}_\phi}\left[\sigma(u^{\mathcal{D}})\right]$. The down side of this alternative is that it requires to sample more RSNN trajectories for each gradient update which may consume compute time inefficiently. On the other hand, this might be relevant in another setting or at the end of training to benefit from the theoretical properties of the sample-and-measure loss function. We leave this to future work.

Regularization of the firing rate of hidden neurons When simulating hidden neurons which are never recorded it is desirable to insert that as much prior knowledge as possible about the hidden activity to keep the network model in a realistic regime. The most basic prior is to assume that every neuron i should have a realistic average firing rate, to implement this we design again a sample-and-measure objective as a variant of $\mathcal{L}_{PSTH}$. This time we consider that the statistics $\mathcal{T}$ are the average firing rate of a neuron $\mathcal{T}(\boldsymbol{z}_i) = \sum_{t,k} z_{t,i}^k$. This results in the objective $\mathcal{L}_{SM\text{-}h}$ which is otherwise similar to $\mathcal{L}_{PSTH}$ as defined in equation (5). Unfortunately the objective cannot be implemented as such because of one missing element: the empirical probability $\pi_i^{\mathcal{D}}$ of a hidden neuron. Instead we simply take another neuron j at random in the visible population and use this average firing rate in place of the probability $\pi_i^{\mathcal{D}}$. In this way, the distribution of average firing rates across neurons of the hidden neurons is realistic at a population level because it becomes the same in the recorded population and in simulated population.

Number of Hidden Neurons	Noise Correlation R^2	Connectivity Matrix R^2
0	0.95	-0.95
10	0.96	0.50
200	0.97	0.59
400	0.95	0.62

Table S4: Performance summary on the test set when fitting RSNN models with variable number of hidden neurons to the synthetic dataset. ELBO+SM-h+PSTH+NC$_{MSE}$ method is used when hidden neurons are included, and MLE+PSTH+NC$_{MSE}$ method is used when no hidden neurons are included.

Dataset	Method	T_{gt}	T	K_m	K_t	learning rate
Moving bars stimulus	MLE + single-trial + NC	45	50	20	8	5e-3
	MLE	50	50	20	8	5e-3
Checkerboard stimulus	MLE + single-trial + NC	75	80	30	1	1e-3
	MLE	75	75	8	4	1e-3

Table S5: Hyper-parameter table for fitting the Retina Dataset, the definition of the hyper-parameter is given in Appendix B. The results are reported in Table S6.

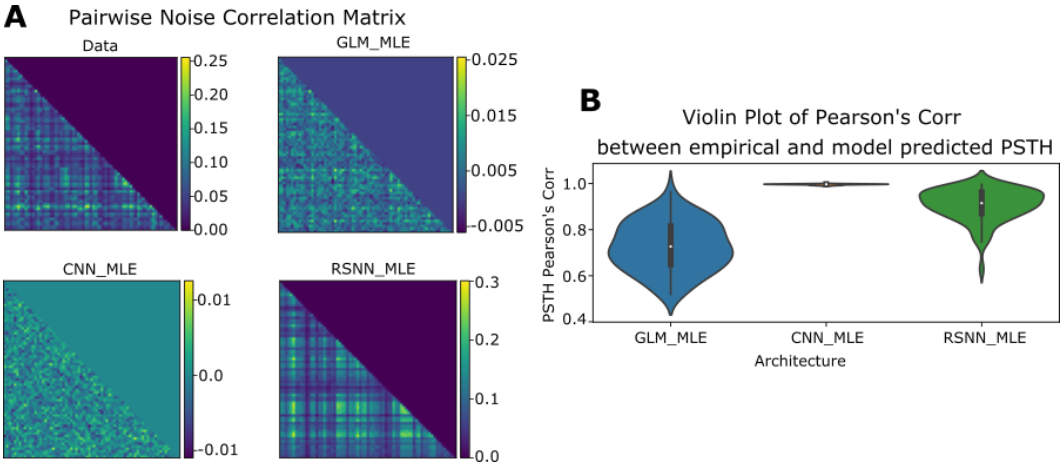

Figure S1: **Comparison of the network architecture** As a preliminary experiment we compared our network architecture against the official GLM code [1]. In our architecture a CNN replaces the spatio-temporal stimulus filter of the CNN, both models are fitted with MLE. **A** Noise correlation matrix of the different models. We included a control architecture where we pruned out the recurrent connection. It is called CNN because only the CNN parameters become relevant. **B** The PSTH correlation computed on the training set. The violon plot represents the distribution of neurons.

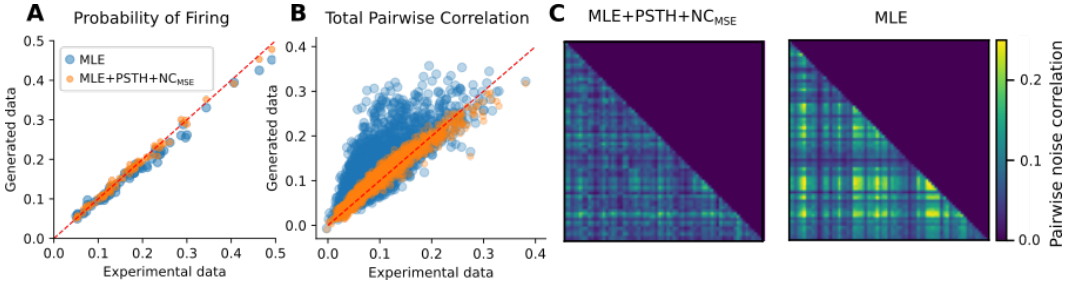

Figure S2: **Comparison with SpikeGAN** This figure is meant to be compared with the Figure 3 from [21]. In this other paper, the authors fitted a spike-GAN to the same dataset. We argue that the PTSH correlation and NC coefficients are as good qualitatively as the results obtained in [21].

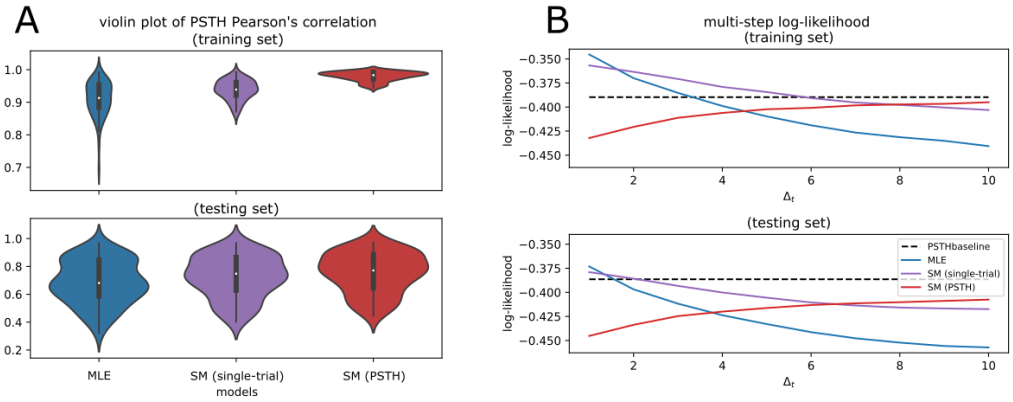

Figure S3: **Improving the mutli-step log-likelihood** We tackle the challenge identified in [15]. We evaluated the multi-step log-likelihood $\log \mathcal{P}_\phi(\boldsymbol{z}^{\mathcal{D}}_{t+\Delta t}|\boldsymbol{z}^{\mathcal{D}}_0 \cdots \boldsymbol{z}^{\mathcal{D}}_t)$ as explained in the main text, and we trained two networks to minimize $\mathcal{L}_{MLE}$ and $\mathcal{L}_{single-trial}$ respectively. **A**) The PSTH correlation of the different models trained in this context. **B**) The multi-step log-likelihood is reported for different models. The dashed baseline represent the ideal model which would always fire a spike with the true PSTH probability. The blue baseline is $\mathcal{L}_{MLE}$ it has never seen self-generated activity during the training, so it's performance drops quickly when the network is not clamped anymore ($\Delta t > 0$). The red-baseline is a model trained with $\mathcal{L}_{PSTH}$ only, it is increasing because the model has never been clamped during training. Therefore it is not trained to be accurate right after the clamping terminates.

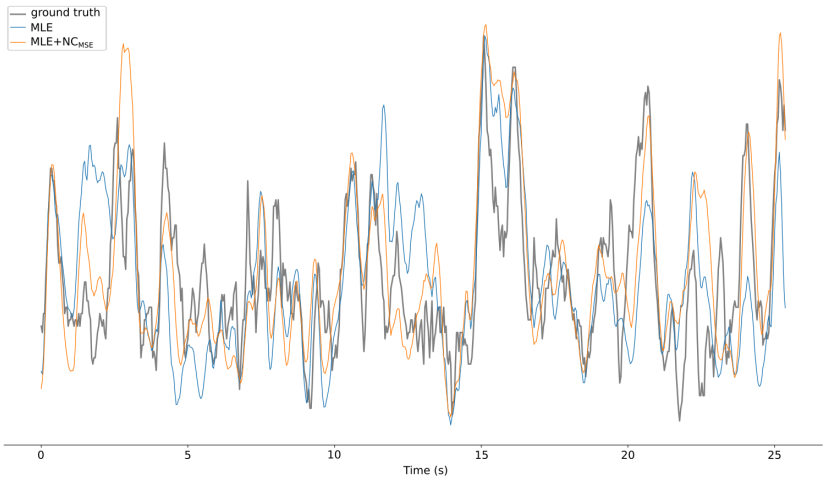

Figure S4: Example of PSTH obtained for a neuron from the V1-dataset. The PSTH are smoothed with a window size of 280ms.

Method	Moving bars stimulus		Checkerboard stimulus	
	PSTH	noise-corr.	PSTH	noise-corr.
MLE + single-trial + NC	0.91 ± 0.003	0.94	0.85 ± 0.004	0.96
MLE	0.90 ± 0.002	0.91	0.84 ± 0.003	0.96
2-step (CNN)	-	-	0.87 ± 0.04	0.91
2-step	0.72 ± 0.10	0.91	0.81 ± 0.05	0.95

Table S6: Performance comparison with the 2-step method [14] on the Retina Dataset. The performance of the 2-step dataset are taken from their paper. For this comparison we used their definition of R^2 which does not penalize a constant offset between prediction and target. For historical reasons, we used here a sample-and-measure NC loss with cross-entropy and not mean-square error. It works but considering later simulations on other datasets we expect higher R^2 with mean-square error.

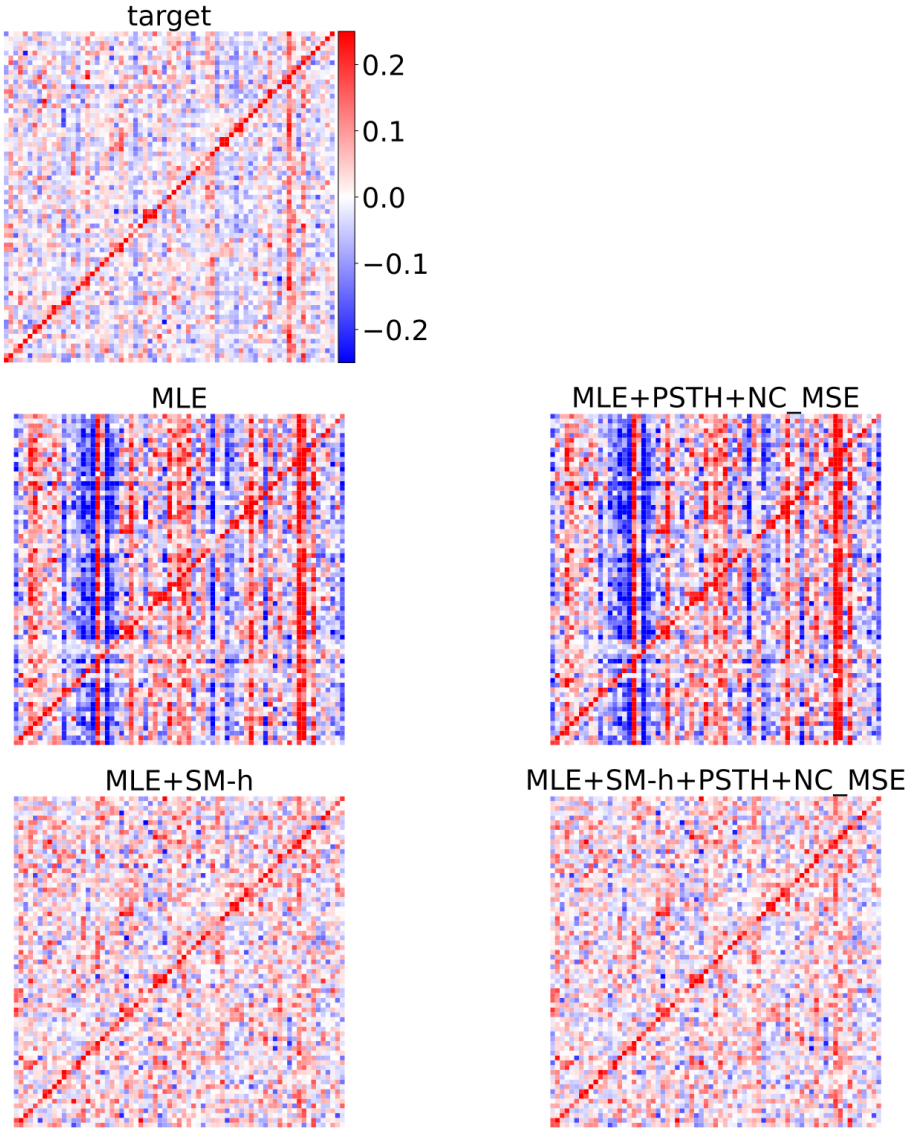

Figure S5: **High resolution plot of connectivity matrices in Figure 4 (large target network)**

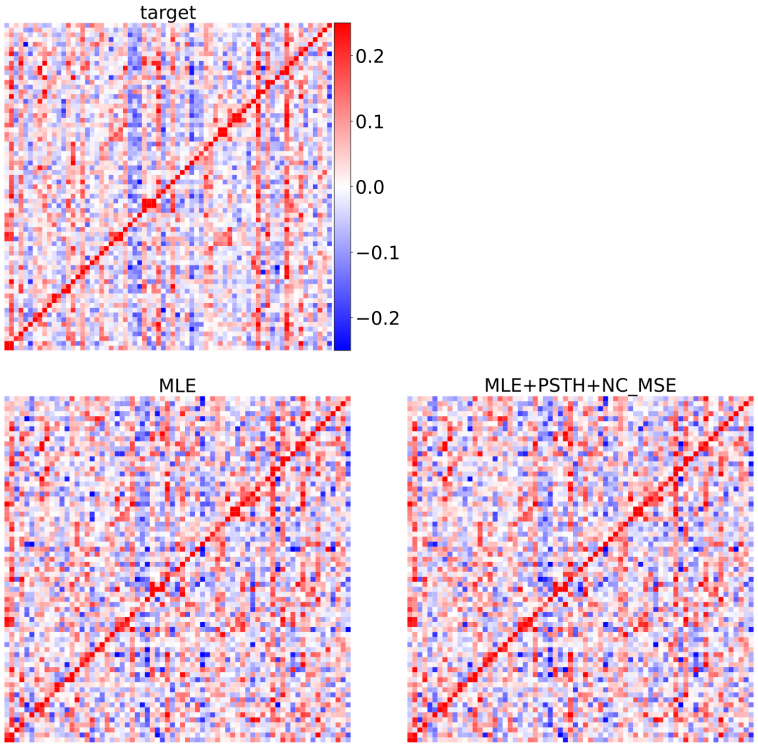

Figure S6: **High resolution plot of connectivity matrices in Figure 4 (small target network)**