Parallel spiking neural network parameter inference using gradient based optimization

Anonymous Authors[1]

Abstract

A common goal in computational neuroscience is to model biologically observed neural activity in order to capture dynamics relating to computation and functional organization. Despite the recent success of gradient based optimization in deep learning, application to spiking models has been limited due to the non-differentiability and complexity of the systems. Using leaky integrate-and-fire neurons, we transform neurons' spike output into continuous, differentiable signals by using soft thresholding, and devise a methodology for spiking neural network (SNN) model parameter inference using gradient based optimization, fitting to spike trains as the target data. To assess goodness of fits, we compare first and second-order statistics, as well as non-negative matrix factorization modules, which are independent to the optimization procedure. The latter is calculated by evaluating the geodesic distance between modules of fitted models and those of the original data sets. Our results show that spike patterns may be captured through gradient based optimization, and further that higher-order spike statistics may be replicated by fitted models. This approach (1) demonstrates a methodology for SNN inference using gradient based optimization and a common data set type, (2) compares two common optimizers for the outlined experiments, and (3) provides a framework for accelerating SNN model inference.

1. Introduction

Fitting single neuron models to patch-clamp recordings has been a common methodology for constructing computational models ever since the seminal work of (Hodgkin & Huxley, 1952), which also happened to capture temporal dynamics relating to the sodium-potassium pump before its discovery later that decade (Skou, 1957). Advancements in recording techniques now make it possible to record from large populations of neurons simultaneously (Jun et al., 2017). These recordings may be used to form spike train data sets indicating which neuron spiked at what time. These signals may be analysed to attain information about the recorded area. However, the use of such data in spiking neural network (SNN) model inference has been limited in part by the non-differentiability of SNN systems, despite the recent success of gradient based methods in deep learning (Huh & Sejnowski, 2017; Tavanaei et al., 2019). Modelling more biologically realistic and thus complex spiking networks, let alone for larger populations of neurons, would thus require methodological advancements over current approaches to make inference tractable.

Several recent studies have tried to address this problem. (Rule et al., 2019) note in their work that recent advances in recording techniques may in conjunction with methodological advances allow for new theoretical connections. Further, they show that intrinsic neuronal states in neural field models may be inferred based on spike train data. (René et al., 2020) demonstrate Bayesian inference for sequential parameter estimation of both leaky integrate-and-fire (LIF) and generalized leaky integrate-and-fire models (Allen Institute for Brain Science, 2017) using spike train data. Nevertheless, these works employ computationally costly methods and do not make use of the work horse for deep learning optimization, which is gradient based.

In this work, we explore the utility of gradient based optimization for parallel parameter inference from spike train data. To this end we implement a modular gradient based optimization framework on top of Pytorch, utilizing its autograd-feature to minimize the loss defined by the outlined distance metrics. We implement leaky integrate-and-fire (LIF) neurons, and crucially define spiking via a soft-threshold function, such that the system is differentiable. For the loss functions, we explore four options: (1) the firing rate distance as a baseline loss function, (2) the van Rossum distance (Van Rossum, 2001) to capture both temporal and spatial spike aspects (relating to both frequency and timing of spiking), and (3) the firing rate distance and the van Rossum distance in conjunction.

We investigate whether the inferred models reproduce spike trains capturing properties of the original data set, and address the questions of tractability and computational cost relating to parallel parameter optimization. In order to assess the performance of our approach, we perform non-negative matrix factorization (NMF) (Seung & Lee, 1999), and assess the geodesic distance between the modules of spike trains produced by the inferred and generative models. NMF has the advantage of naturally resulting in a parts-based representation due to its non-negativity constraint, which has been found to be suitable for naural data and spike train analysis (Seung & Lee, 1999). Further, this functional network characteristic is independent of the optimization-procedure, making it a suitable additional metric to assess the outlined methodology.

Overall, our findings suggest that spike train data may be used for model inference, and that inferred models can reproduce functional network characteristics of the original data. The distance between the NMF modules of the inferred model spike trains and the target spike trains is reduced substantially throughout optimization, with the closest fits falling very close to the ground-truth. While convergence toward true model parameter values cannot be guaranteed, we also observe that the most prominent parameters fall within close proximity of the ground-truth values for the closest model fits. We also observe that fitted model spike trains become more correlated with the target spike trains in all experiment configurations.

2. Methods

The goal in this work is to demonstrate that successful machine learning methods of gradient based optimization may also be applied in spiking network model inference using leaky integrate-and-fire neuron models. Further, we aim to assess quantitatively (1) to what extent ground-truth parameters are inferred, (2) to what extent dependent functional network characteristics of the spike trains are captured, and (3) to what extent independent functional ensembles are captured by and emerge from the inferred models. The neuron model we employed in this work is the leaky integrate-and-fire (LIF) model, as described in among other works (Rolls & Treves, 1998),

$$\frac{dv}{dt} = \frac{E_L - v_t + R_I I_t}{\tau_m}, \tag{1}$$

where E_L is the rest potential, v_t is the membrane potential at time t, R_I is the membrane resistance, and I_t is the synaptic current. The modelled networks are non-transitively fully connected, with the weights normally distributed between $w \in [-1, 1]$. The synapses are modelled as exponentially decaying post-synaptic currents, where a conductance variable g models the conductance for the neurons,

$$\frac{dg}{dt} = -\frac{g}{\tau_g}, \tag{2}$$

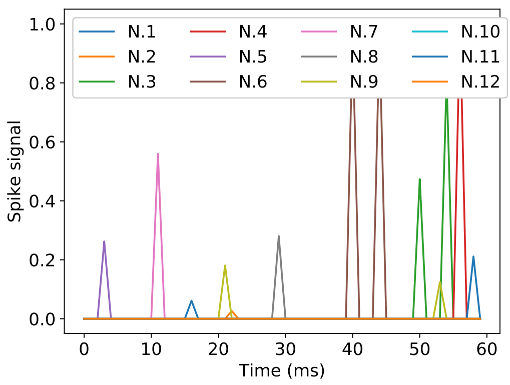

Figure 1. The thresholded spike signal $s_t(v)$ for 60ms of model simulation for 12 neurons. Note that neurons may provide continuous sub-threshold signals.

where the synaptic input current to a neuron j is modelled as $I_{syn,j} = \sum_i w_{i,j} I_{i,j}$.

Key to enabling backpropagation is defining spiking using a differentiable function, for which we used the sigmoid function,

$$s_t(v) = \frac{1}{1 + e^{-(v_t - (\theta_v + \theta_s))}}, \tag{3}$$

where s_t denotes whether a neuron spikes at time t with a membrane potential v_t. This results in that optimization of defined loss metrics always back-propagates error gradients through the soft-thresholded membrane potential, in effect treating this as a continuous spike value. In other words, the distance metrics calculate the distance between a continuous spike train to a binary target spike train. Note that this also results in that sub-threshold values result in a spike signal during optimization.

All parameters were held as free parameters during optimization, resulting in 4 N-dimensional free parameters, and one N^2-dimensional parameter, where N is the number of neurons in the network.

In addition to the previously mentioned parameters, a Poisson input perturbation rate was also optimized as a free scalar variable, r_p. This is used to generate a Poisson input perturbation to the model for each time-step, drawn from a Poisson distribution with a rate r_p. As such, the input current model may be written as,

$$I(t) = I_S + I_{\text{ext}}, \tag{4}$$

where $I_S^j = \sum_i^N w_{ij} g_i$, and $I_{\text{ext}}^j \sim P\{k; r, t\} = \frac{(r_p t)^k e^{-rt}}{k!}$, where t is the time interval, and k is the number of spikes, $P\{k; r, t\}$ denoting drawing k spikes (pseudo-)randomly from the Poisson distribution.

2.1. Gradient based optimization

We implement a LIF SNN optimization framework on top of Pytorch (Paszke et al., 2019) and utilize Pytorch's autograd-feature for backpropagation of error gradients.

To perform optimization, an set of initial model parameter values is drawn uniformly from pre-defined intervals. These intervals are constrained both to meaningful parameter ranges, and also to ensure non-silent models upon initialization. These models are then perturbed with Poisson input, drawn pseudo-randomly from a Poisson distribution given by the current rate parameter (initially $r_p = 10\text{Hz}$). See table 1 under section 4for more details. The model output is then compared with the output of a target data, and the loss is calculated between the model and target output spike trains by using the loss functions defined in the following sub-section. Training is then done for a static number of 20 training iterations. Note, however, that this could easily be modified to depend upon some convergence criterion. However, for comparability, we chose to keep this number constant.

2.1.1. Loss metrics

We performed optimization using (1) the firing rate distance, (2) the van Rossum distance, (3) an additive combination of the van Rossum and firing rate distance, and (4) by an adaptive combination of the two loss functions, emphasising mainly the firing rate in earlier training iterations, and conversely mainly the van Rossum distance in later training iterations. We define the firing rate distance as the Euclidean distance between each neuron's average firing rate for a given spike interval, which may be written as,

$$d_r(S_1, S2) = \sum_i^N \frac{\sqrt{(s_1^i - s_2^i)^2}}{\Delta t}, \tag{5}$$

where s^i is the number of spikes for neuron i in a given interval Δt.

The van Rossum distance (Van Rossum, 2001) may be defined as the Euclidean distance between two spike trains where each spike is convolved with an exponential kernel forward in time,

$$f_{conv}(t, t_{i,spike}) = e^{\frac{-\Delta t_i}{\tau_{\text{vr}}}} \tag{6}$$

where $\Delta t_i = (t - t_{i,spike}),\ t \geq t_{i,spike}$, with t being the current time, $t_{i,spike}$ the most recent time of spiking for neuron i, τ_{vr} a time constant (set to $\tau_{\text{vr}} \in \{10.0, 20.0, 100.0\}\text{ms}$ in the experiments, and Δt_i the time since the neuron's last spike, $\Delta t_i = (t - t_{i,spike})$, the distance between two spike trains is then given by the Euclidean distance between two convolved spike trains,

$$d_v = E(S_1, S_2) = \sum_{t=0}^{t=N} \sqrt{(f_{conv}(S_1) - f_{conv}(S_2))^2} \tag{7}$$

and lastly, we also define two linear combinations of these distance metrics as loss functions, either simply as a static combination, $d_{\text{linear}} = 0.9d_r + 0.1d_v$, or as linearly decreasing or increasing for d_r and d_v, respectively,

$$d_A = (1 - 0.9T)d_r + (0.1 + 0.9T)d_v, \tag{8}$$

where $T = \frac{t}{t_{\max}} \in [0, 1]$.

2.2. Assessment of functional characteristics

In order to assess functional organization in the neural data we employ non-negative matrix factorization (NMF) (Seung & Lee, 1999) to infer ensembles of coactive neurons. This factorization may be written as,

$$V \approx WH, \tag{9}$$

where the original data set V of dimension $n \times t$ is factorised into two matrices of dimensionality $m \times n$ and $m \times t$. What makes this dimensionality reduction method particularly suited for spike train data is its non-negativity constraint, which naturally leads to a parts-based representation.

While NMF may be used to discover spatial and temporal firing patterns in spike data, the geodesic similarity measure between the modules may be used to assess how well these features were recovered in a fitted model. The geodesic distance o between two matrices A and B of dimensionality $m \times n$ may be written as,

$$o(A, B) = (1 - \frac{2}{\pi}\cos^{-1}(A^T B)), \tag{10}$$

where m is the number of modules, and n is the number of nodes in the data set. We also calculate the spike correlation between the generative (target) model and the fitted model by binning spikes into 20ms bins and then calculating the cross-correlation between the binned spike counts for in total $4\ s$ of data from each model, generated by using the same random seed as in the experiment for Poisson input generation. The results for the different optimization setups are shown in section 4.

Lastly, we plot inferred Gaussian kernel density estimates (KDEs) for parameter marginals (Rosenblatt, 1956; Parzen, 1962), and show a sample plot for this in figure 9. Largely parameters fall outside of the kernel-borders - however, some parameters of converged models fall closer to the true values. This cannot, however, be guaranteed, as can be seen in figures 8, and 9.

2.3. Synthetic data set

We simulate data sets with known ground truth to test the (SNN) inference algorithms on. The generative model was designed to have three distinct populations of neurons, with each population exhibiting fairly stable, yet distinct spike patterns. To generate the target data, we perturbed the hand-engineered LIF model with Poisson input with rates of 10Hz, producing target spike train data. To make our results more robust, however, we initialized four different models by normally distributing the parameter values around the pre-defined values, setting a distinct random seed for each initialization.

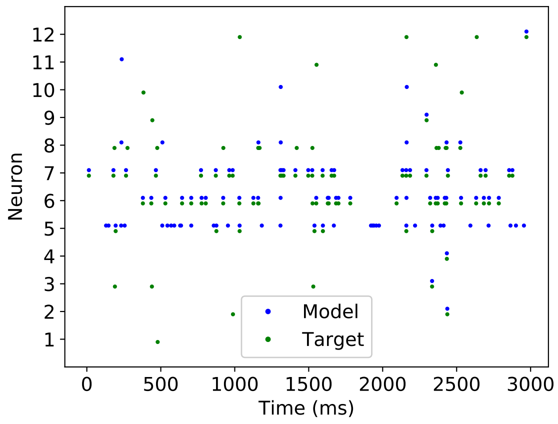

Figure 2. Spike trains after optimization for 20 training iterations, each over 10 batches of 400ms. Here the Poisson input to the generative and fitted models are drawn pseudo-randomly from Poisson distributions their respective rates, but with the same random seed.

2.4. Computational complexity

This research explores parallel parameter inference in order to reduce computational complexity to the extent that it is possible. The upper bound to the parallelizability of the algorithm follows from the computations of highest cost, as as algorithmic bottlenecks such as the one imposed by the sequentiality associated with the temporal nature of spiking models. I.e. the temporal dependence of the membrane potential on the previous, which mean that they have to be simulated in time. One may, however, relax simulation to time-intervals that may be run in parallel. However, parameter updates throughout training still poses a checkpoint where synchronization has to occur, and parallel runs collapse into one. Due to this characteristic of SNNs, sequential parameter optimization quickly becomes intractable due to the sequential nature of model simulation. In our implementation we divide each training interval into a set of batches B that we simulate in parallel, effectively performing batch norm over the batches. This should result in more stable gradients, and may reduce the number of training iterations required for convergence. This makes the complexity of simulation scale with the time frame simulated Δt_B for each batch, and the model complexity itself, which is $N^2 + PN$, where P is the number of N-dimensional parameters, and the weights ω are N^2-dimensional. This makes simulation $O(C_t \Delta t_B (N^2 + NP))$ where C_t is the number of training iterations, which has a lower bound of the reduced form $\Omega(N^2)$. Optimization additionally requires back-propagation through the system, including calculation of the parameter updates for each parameter. This may either be back-propagated using the chain rule, or more efficiently approximated using local traces, which have been shown to successfully perform gradient-descent and back-propagation of error-gradients, and scale linearly with the complexity of the model system. Yet, the lower bound is that of the model dimensionality, which is $\Theta(N^2 + NP)$ per time-step, and due to the sequential constraint of SNN simulation thus $\Theta(2C_t t_B(N^2 + NP)$, where C_t is the number of training iterations, and t_B is the length of each training batch. The temporal constraints relating to spiking neural network simulations make exploring parallel parameter optimization the least costly approach, but yet quadratic in complexity, whereas sequential optimisation would be at least quadratic in the parameter term in the denoted complexity. While the sequentiality remains a constraint for SNNs, optimization may be done over batch-intervals in parallel. Such a type of batch normalization has been shown to reduce the internal covariate shift and reduce the number of training iterations needed for convergence in deep learning models (Ioffe & Szegedy, 2015), and may be useful both in reducing the number of training iterations, as well as in increasing the parallelizability of SNN model optimization.

3. Related work

(Grün, 2010) discuss defining spike metrics to quantify the distance between two spike trains with regards to data analysis. More recently, (Huh & Sejnowski, 2017) applied gradient descent on a network of single-state neurons by reformulating the gate function into a continuous gate function, thus introducing a novel model that may be successfully used in gradient based optimization. In this work, however, we transform the more commonly studied leaky integrate-and-fire model which in contrast also has non-linear reset rules, into a differentiable system. This allows employing successful machine learning approaches for LIF SNN model inference.

Another related piece of work looking at parameter inference in SNNs is the work of (René et al., 2020), where the authors study using Bayesian inference for parameter estimation. This has the advantage of providing a confidence intervals for the estimated parameters. However, the outlined approach requires estimation of a population-level model for Markov Chain Monte Carlo sampling for sequential neuron-level model parameter estimation, resulting in a computationally costly procedure. In this work, the aim is to address employing ML techniques for direct, parallel parameter inference of neuron-level models.

(Rule et al., 2019) perform intrinsic state inference of classical neural field models using spike train data by developing a moment closure based method. In doing so they connect single neuron dynamics with population level dynamics by using spike train data. In our work, we wish to encompass functional level behavior by more direct inference based on spike train data by optimizing spiking network models.

4. Experiments

To assess the performance of the different inference schemes outlined in the methodology section, we fit SNN models to the synthetic data sets. To minimize the loss, we use Adam (Kingma & Ba, 2015), and stochastic gradient descent (SGD) for comparison, over batches consisting of intervals of 400 ms of spike data. For SGD we use learning rates $\alpha = \{1, 5\}$ %, and for Adam only $\alpha = 5\%$ (as it is adaptive). The spike metrics we employ are as defined in section 2, namely the van Rossum distance, the firing rate distance, and two linear combinations of the metrics, after observing empirically that the firing rate distance was more stable than the van Rossum distance. We then fit five different pseudo-randomly initialized models using gradient descent and Adam for each combination of the loss functions, i.e. for the four combinations. These were fitted over spike train intervals of 4s, divided into batches of 400ms, thus effectively normalising over 400ms time windows. The window of 400ms was chosen after empirically determining that it resulted in relatively good convergence for less complex models, more specifically by fitting LIF SNNs to spike trains and attaining similar NMF modules, spike patterns, and firing rates when using one scalar value for each network-parameter (thus reducing the dimensionality by N, where N is the number of network neurons). For the slightly different generative model initializations, we randomly initialize them by using four different random seeds, normally distributing each model parameter around its pre-defined mean with a set standard deviation. This is to make the model more robust to random seed initialization, as well as to test inference for slightly different variants of a similar configuration. Further, we proceed to fit five models that we initialize by uniformly distributing the initial parameter values in the interval given in table 1. Initialization is done using random seeds 0 through 4 for each uniform pseudo-random model initialization, along with $\{\overline{\omega}, \omega_{\text{std}}\} = \{0.3,\ 0.2\}$ for weights initialization, and

Table 1. LIF network parameter intervals for pseudo-random uniform model initialization.

E_L	τ_m	τ_g	R_I
[-55, -45]	[1.2, 2.3]	[2.0, 3.5]	[135, 140]

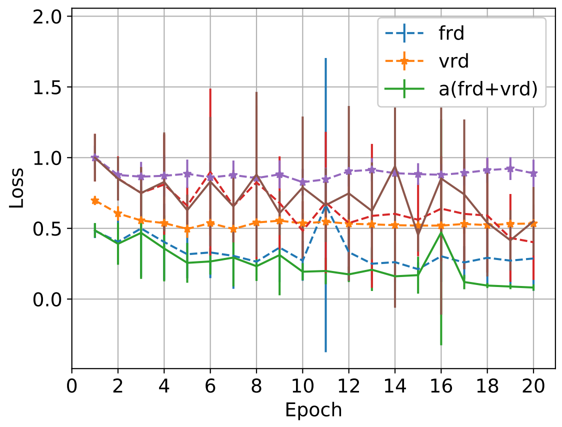

Figure 3. The average normalised training (lower) and test (upper) loss for the three loss metrics, normalised by the maximum test loss, where frd, vrd, and a(frd+vrd) denotes the firing rate distance, van Rossum distance, and the adaptive linear combination of the two, respectively.

$r_p = 10\text{Hz}$, for the Poisson rate.

In sum this resulted in a total of 20 fitted models per experiment configuration for a total of 10 configurations, consisting of 4 different pseudo-random generative model initializations, with 5 fitted models that were pseudo-randomly initialized and fitted for each loss function.

Following optimization we evaluate functional network characteristics, including comparing the modules obtained by non-negative matrix factorization, and also look at second order statistics for the average firing rate, spike correlations between the fitted and generative models, and the spike variance and covariance. On average the geodesic distance is lowest for minimizing the firing rate distance using Adam (figure 6). This also holds for the best model, i.e. the model with the lowest geodesic distance. However, when considering the best fits, minimizing the adaptive combination of the firing rate and van Rossum distance using Adam, and the firing rate distance using SGD attains a model fit with the same distance. Note however that these two have higher average distances, with a higher deviation.

For all configurations, meaning combinations of optimization using either Adam or SGD, we also plot the average Euclidean distance between the inferred and target model parameters (figure 7). Interestingly, these were slightly further away from their true values when minimizing with Adam, but in all cases closer to the true parameter values. When inspecting the single parameter averages, the mem-

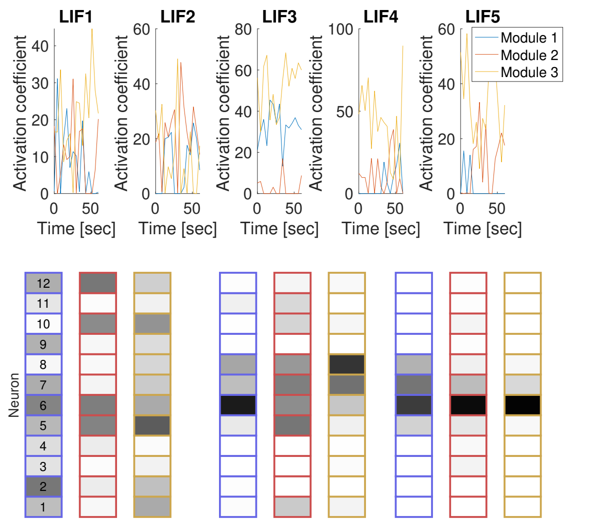

Figure 4. Non-negative matrix factorization activation coefficients (top panels) and modules (bottom panels); for the generative model with random seed 1 (left), and the fitted models in one experiment for random seeds 3 and 5 using the firing rate distance and Adam (middle, right). Note that the spatial modules are ordered and color coded after the activation coefficients, but that their spatial order may vary.

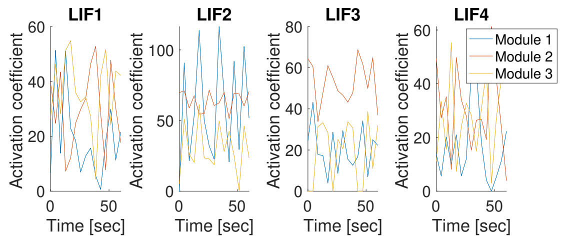

Figure 5. Activation coefficients of the three factorised modules for the generative model and each of its four pseudo-random initializations around the pre-defined parameter means.

brane resistance R_I was almost perfectly recovered when optimizing the firing rate distance using Adam, as illustrated in figure 9. The rest potential E_L and membrane time constant τ_m were also very close to the true values, whereas the weights and synaptic time constant τ_g were somewhat further away in terms of their Euclidean distance from the true parameter values.

Lastly, to verify the implementation of the outlined methodology, we fitted models by using the exact same input for both the model being fitted and the generative model, effectively removing the noise in the fitting procedure. Whilst this results in a fully unrealistic scenario biologically speaking, as we access the ground truth input, this allowed us to verify that perfect convergence did indeed occur using all of the outlined loss metrics.

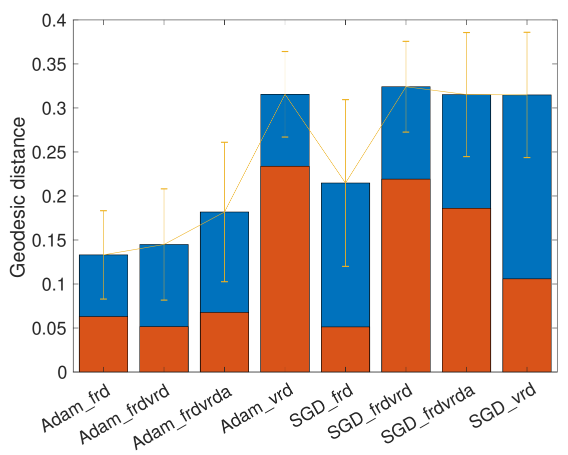

Figure 6. Average geodesic distances (blue, with error bars) between the non-negative matrix factorization modules of spike trains attained through optimization using the listed setups, and the factorized spike trains of the corresponding (target) generative model. The distances for the best model fits are shown in red (lower bars).

5. Discussion

In this paper, we explored parallel parameter optimization for spiking neural network models with LIF neurons and exponentially decaying synapse models by using gradient based optimization for a set of different loss metrics. Our results indicate that the methodology can be used to replicate spiking behavior, and introduces a parallel methodology for spiking model inference.

Gradient based optimization methods are not guaranteed to find the global optimum. In line with this shortcoming, we found that convergence to local minima occurred. While perfect recovery of the true model parameter values cannot be guaranteed when minimizing the defined loss metrics due to the multitude of local minima in the parameter hyperspace arising from the high dimensionality of the model, we still observed that the more prominent model parameters fell close to their ground truth values when the rate based metric was minimized. Further, we observed that similar functional characteristics emerged in the inferred models, i.e high geodesic similarity between the fitted model spike trains and the target spike trains. This suggests that parallel optimization may be successful for the high-dimensional problem of spiking model inference by using spike trains as target data, and that inferred model can capture functional network characteristics of the modelled data. Furthermore, we observed that the inferred models when using the firing rate had a temporal sensitivity. This may be seen when considering that we fitted to sub-second intervals of spike data, which thus introduced local information. It is worth noting, however, that optimization using a metric that is commonly referred to as insensitive to temporal information may both

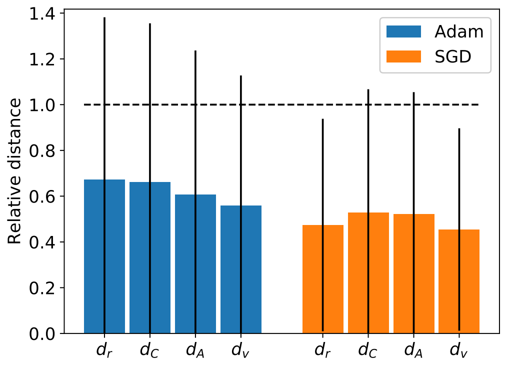

Figure 7. The average relative Euclidean distance across all experiments using the same loss metric for Adam (blue, left group) and SGD (orange, right group), relative to the the pseudo-randomly initialized model parameters' normalized distance to the true parameter values.

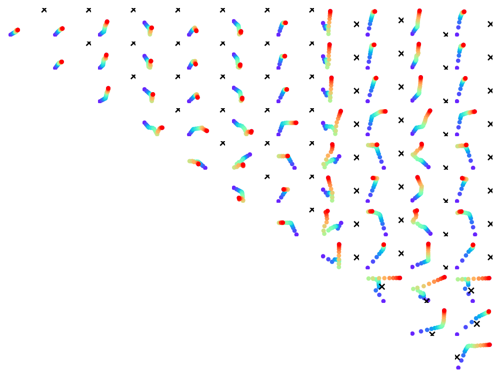

Figure 8. Parameter trajectories for R_I between each pair of neurons in the network model throughout optimization for one particular fit. Red denotes the first training iteration, moving throughout the color scale towards blue and violet.

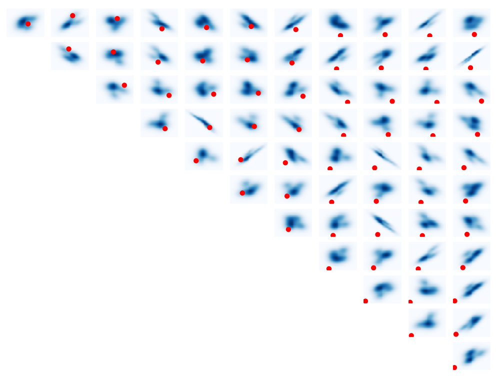

Figure 9. Gaussian kernel density estimates for R_I, inferred using the parameter values of all models fitted to the same generative model. Here using the adaptive combination of the firing rate distance d_r and the van Rossum distance d_v.

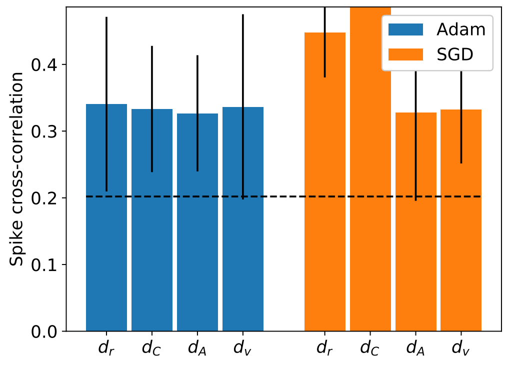

Figure 10. Average spike cross-correlation between fitted and target model spike trains for each loss function, and the two classes of optimizers.

reconstruct similar patterns in time, as well as functional network characteristics.

In some cases of model inference using spike trains as target data, properties about the input might be assumed, but in others little to no assumptions may be made. Therefore, we chose to model the input using a Poisson process, as this resembles the nature of neural activity to some extent (Grün, 2010). While perfect convergence may occur when using both the ground-truth input and outputs, it is possible that the input noise factor may have obscured of the gradient signal stemming from the van Rossum distance metric due to its temporal sensitivity. The firing rate metric is also more robust to local spike variability, since the expected rate over intervals of medium to longer length may be assumed to be roughly the same. This leads to the question of whether

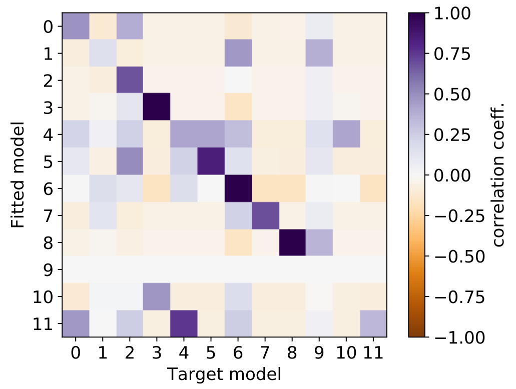

Figure 11. Spike cross-correlations between each pair of neurons for one particular fitted and generative model pair. In this run the adaptive combination of the firing rate distance d_r and the van Rossum distance d_v was minimized.

the van Rossum distance obscures the signal only when the model output is far from that of the target data. It turns out that a combination of the two metrics performs better than the van Rossum distance alone, but on average not as good as when only considering the firing rate. As such, our results suggest that the van Rossum distance should only be used when the signal to noise ratio is lower than in our experimental setup. Note that the above discussion is based on the results for the particular SNN models of this study, where the model is in part driven by noise. In scenarios where the signal-to-noise ratio is greater, or the model in itself is more strongly self-organizing, it might very well be that the loss metrics may have different effects.

Another aspect that may have a large effect on optimization is the continuity and stability of the model system. An SNN has non-linearities in its reset rules, and also may exhibit different types of spiking. Non-linearities may obscure the gradient signal during optimization in itself. When viewing this in conjunction with that we attempt to optimize all parameters in parallel, it might be expected that a signal that is more sensitive to local changes may be too chaotic for optimization to converge. As such, it might be fruitful to further constrain the search space such as by introducing linear constraints, or by performing sequential parameter optimization. We note that the average Euclidean distance between the fitted model parameters and the true model parameters always decreased throughout optimization, and was within the Gaussian kernel density estimates for most of the values of the parameters E_L, τ_m, and particularly R_I. From empirical observation, it seems that the implemented model is the most sensitive to the membrane resistance variable, R_I. Thus, if performing sequential optimization, this could be one variable to fix. Further, it may be shown that this value may be normalised relative to network size in order to generalize the outlined methodology to arbitrary network sizes. This would, however result in a slightly different shape of the term including the membrane resistance, potentially allowing for a wider interval of values for which similar model behavior may occur, ultimately changing the parameter landscape by the normalising of this parameter. In any case, when functional network characteristics are replicated in an artificial model, the model itself may be used for hypothesis testing and generation relating to the modelled area. Future work could address the generalizability of the approach to (1) biological spike train data, and (2) using more complex neuron models, such as the generalized leaky integrate-and-fire model (Allen Institute for Brain Science, 2017).

References

Allen Institute for Brain Science. Allen Cell Types Database, Technical White Paper: GLIF Models, 2017. URL `http://help.brain-map.org/download/attachments/8323525/GLIFModels.pdf`.

Grün, Sonja; Rotter, S. *Analysis of Parallel Spike Trains*. Springer, 2010. ISBN 9781441956743. doi: 10.1007/978-1-4419-5675-0.

Hodgkin, A. L. and Huxley, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117 (4):500–544, 8 1952. ISSN 0022-3751 (Print). doi: 10.1080/00062278.1939.10600645.

Huh, D. and Sejnowski, T. J. Gradient Descent for Spiking Neural Networks. Technical report, Salk Institute, 2017.

Ioffe, S. and Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proceedings of the International Conference on Machine Learning (JMLR)*, 37, 2015.

Jun, J. J., Steinmetz, N. A., Siegle, J. H., Denman, D. J., Bauza, M., Barbarits, B., Lee, A. K., Anastassiou, C. A., Andrei, A., Aydin, , Barbic, M., Blanche, T. J., Bonin, V., Couto, J., Dutta, B., Gratiy, S. L., Gutnisky, D. A., Häusser, M., Karsh, B., Ledochowitsch, P., Lopez, C. M., Mitelut, C., Musa, S., Okun, M., Pachitariu, M., Putzeys, J., Rich, P. D., Rossant, C., Sun, W. L., Svoboda, K., Carandini, M., Harris, K. D., Koch, C., O'Keefe, J., and Harris, T. D. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551(7679):232–236, 11 2017. ISSN 14764687. doi: 10.1038/nature24636. URL `https://www.nature.com/articles/nature24636`.

Kingma, D. P. and Ba, J. L. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations*, pp. 1–13, 2015.

Parzen, E. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, 33(3): 1065–1076, 9 1962. ISSN 0003-4851. doi: 10.1214/aoms/1177704472. URL https://projecteuclid.org/euclid.aoms/1177704472.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., Facebook, Z. D., Research, A. I., Lin, Z., Desmaison, A., Antiga, L., Srl, O., and Lerer, A. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems*, pp. 8024–8035. University of Warsaw, 10 2019.

René, A., Longtin, A., and Macke, J. H. Inference of a Mesoscopic Population Model from Population Spike Trains. *Neural Computation*, pp. 1–51, 6 2020. ISSN 0899-7667. doi: 10.1162/neco{_}a{_}01292. URL https://www.mitpressjournals.org/doi/abs/10.1162/neco_a_01292.

Rolls, E. T. and Treves, A. Introduction. In *Neural Networks and Brain Function*, chapter 1, pp. 418. Oxford University Press, Oxford, UK, 1998. ISBN 0198524323.

Rosenblatt, M. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832–837, 9 1956. ISSN 0003-4851. doi: 10.1214/aoms/1177728190. URL https://projecteuclid.org/euclid.aoms/1177728190.

Rule, M. E., Schnoerr, D., Hennig, M. H., and Sanguinetti, G. Neural Field Models for Latent State Inference: Application to Large-Scale Neuronal Recordings. *bioRxiv*, pp. 543769, 2 2019. doi: 10.1101/543769. URL https://www.biorxiv.org/content/10.1101/543769v1.

Seung, Sebastian, H. and Lee, D. D. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999. ISSN 00280836. doi: 10.1038/44565. URL http://www.nature.com/doifinder/10.1038/44565.

Skou, J. C. The influence of some cations on an adenosine triphosphatase from peripheral nerves. *BBA - Biochimica et Biophysica Acta*, 23(C):394–401, 1 1957. ISSN 00063002. doi: 10.1016/0006-3002(57)90343-8.

Tavanaei, A., Ghodrati, M., Kheradpisheh, S. R., Masquelier, T., and Maida, A. Deep learning in spiking neural networks. *Neural Networks*, 111:47–63, 3 2019. ISSN 18792782. doi: 10.1016/j.neunet.2018.12.002. URL https://doi.org/10.1016/j.neunet.2018.12.002https://pubmed.ncbi.nlm.nih.gov/30682710/.

Van Rossum, M. C. A novel spike distance. *Neural Computation*, 13(4):751–763, 4 2001. ISSN 08997667. doi: 10.1162/089976601300014321. URL https://www.mitpressjournals.org/doix/abs/10.1162/089976601300014321.