Fast reconstruction of degenerate populations of conductance-based neuron models from spike times using deep learning

Julien Brandoit[1] **Damien Ernst**[1] **Guillaume Drion**[1] **Arthur Fyon**[1]
[1]Department of Electrical Engineering and Computer Science, University of Liège, Liège, Belgium
{jbrandoit, dernst, gdrion, afyon}@uliege.be

Abstract

Inferring the biophysical parameters of conductance-based models (CBMs) from experimentally accessible recordings remains a central challenge in computational neuroscience. Spike times are the most widely available data; yet, they reveal little about which combinations of ionic conductances generate the observed activity. This inverse problem is further complicated by neuronal degeneracy, where multiple distinct sets of conductances yield similar spiking patterns. We introduce a method that addresses this challenge by combining deep learning with Dynamic Input Conductances (DICs), a theoretical framework that reduces complex CBMs to three interpretable aggregated conductances that separate according to timescales. DIC values directly relate to excitability and firing patterns. Our approach first maps spike times directly to DIC values at threshold using a lightweight neural network that learns a low-dimensional representation of neuronal activity. The predicted DIC values are then used to generate degenerate CBM populations via an improved state-of-the-art algorithm. Applied to two neuronal models, this algorithmic pipeline reconstructs spiking, bursting, and irregular regimes with high accuracy and robustness to variability, including spike trains generated by Poisson processes. It produces diverse degenerate populations within milliseconds on standard hardware, enabling scalable and efficient inference from spike recordings alone. Beyond methodological advances, we provide an open-source software package with a graphical interface that allows experimentalists to generate and explore CBM populations directly from spike trains without requiring programming expertise.

1 Introduction

One major challenge in neuroscience is to understand how neurons generate and regulate electrical activity through ion channels, linking microscopic mechanisms to macroscopic dynamics. Spike times remain the most widely accessible experimental data [1–3]. They reveal what neurons do and how they respond to perturbations, but not which specific ion channel densities underlie the observed activity. This gap is critical since ion channel densities are the primary determinants of neuronal excitability, shaping single-cell dynamics and, hence, firing patterns. These same biophysical parameters are also the principal targets of neuromodulators and medications [4–6]. Neuronal degeneracy blurs the picture even more, as many distinct combinations of ion channel densities can yield similar spiking behaviors [7].

Conductance-based models (CBMs) capture neuronal dynamics by explicitly modeling ion channel dynamics through experimental fitting. In these models, the effective densities of ion channels on the membrane, modeled by *maximum conductance values*, are the major parameters that dictate neuronal

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Data on the Brain & Mind.

excitability and firing patterns. However, this framework leads to high-dimensional models with dozens of parameters [8–10]. These models also exhibit neuronal degeneracy, where different sets of conductances can produce similar voltage traces. As a result, inference from spike data becomes both complex and intractable [8, 11, 12].

The above observations define a core inverse problem: *given a spike train, how can one identify the potential population of CBMs that could have generated it?* Existing methods often require full voltage traces, are sensitive to noise, are computationally expensive, or fail to account for degeneracy [13–22].

Dynamic Input Conductances (DICs) offer a way to bypass the complexity of degenerate ion channels and represent a promising alternative to classical approaches. They consist of three voltage-dependent conductances that capture aggregated channel effects on fast, slow, and ultra-slow timescales [23, 24]. DICs provide interpretable low-dimensional representations of excitability. Moreover, the complete DIC curves are unnecessary since their values at threshold voltage mostly dictate firing patterns [24]. Any CBM, regardless of complexity, can be reduced *analytically* to three scalar DIC values that mostly explain the associated spontaneous activity, enabling degenerate population generation [24]. Neuronal excitability is mainly shaped by these three threshold DICs. Yet no quantitative mapping exists from spike trains to DIC, and existing DIC-based generation methods lack accuracy.

We address these gaps with two contributions. First, a lightweight deep network maps spike times directly to threshold DICs. Second, an iterative compensation algorithm improves DIC-to-CBM generation by enforcing similar DIC values within a degenerate population, thereby enforcing similar firing patterns. Combined, these form a pipeline that reconstructs degenerate CBM populations from spike times within milliseconds, even on standard lab hardware. Validation across two models [8, 25] shows accurate reproduction of spiking and bursting, with robustness to variability and noise.

By integrating deep learning with DICs theory, our method bridges spike data and mechanistic CBMs, demonstrating that DICs serve as interpretable intermediates that enable scalable inference of degenerate populations. To promote adoption, we released an open-source package with a graphical interface [26], allowing experimentalists to generate and explore CBM populations directly from spike recordings without programming experience.

General problem statement

We consider a known conductance-based model (CBM) with membrane dynamics:

$$C\frac{dV}{dt} + g_{\text{leak}}(V - E_{\text{leak}}) = -\sum_{i\in\mathcal{I}} \bar{g}_i m_i^{p_i} h_i^{q_i}(V - E_i) + I_{\text{ext}},$$

where V is the membrane potential, C the capacitance, g_{leak} and E_{leak} the leak parameters, and each ionic current i is defined by maximal conductance $\bar{g}_i$, gating variables m_i, h_i with powers p_i, q_i, and reversal potential E_i. The unknown parameters are the conductances $\bar{g} = [\bar{g}_1, \ldots, \bar{g}_{|\mathcal{I}|}, g_{\text{leak}}] \in \mathcal{G}$ while other parameters are fixed. Spike times are used as the observed description of the activity:

$$x = [t_1, t_2, \ldots, t_{N_{\text{spikes}}}] \in \mathbb{R}^{1\times *},$$

with variable length depending on recording duration and firing pattern.

Due to degeneracy [4, 27], the mapping:

$$x \longmapsto \bar{g} \in \mathcal{G}^*(x),$$

is not bijective: the solution is a subspace $\mathcal{G}^*(x) \subset \mathcal{G}$ containing many compatible parameter sets [23, 24, 28]. Our goal is to build a set of P models $\mathcal{P} = \left\{[\bar{g}_1, \ldots, \bar{g}_{|\mathcal{I}|}, g_{\text{leak}}]_i\right\}_{i=1}^{P}$ that reproduce the firing pattern x.

Traditional approaches either infer a single solution or attempt to learn the full high-dimensional solution space, which is slow, data-intensive, and hard to train. To overcome these limitations, we leverage low-dimensional *dynamic input conductances* (DICs) [23, 24]. Evaluating DICs at threshold V_{th} mostly explains observed neuronal activity. Formally, they are defined as:

$$g_{\text{DICs}}(V) = S(V; \bar{g}) \cdot \bar{g}, \tag{1}$$

where S is the sensitivity matrix that can be constructed from the CBM equations following the formulation in [23] (see appendix), and $g_{\text{DICs}}(V) : V \mapsto \mathbb{R}^3$ provides a compact representation.

The three components correspond to fast, slow, and ultra-slow timescales $g_{\mathrm{f}}(V_{\mathrm{th}})$, $g_{\mathrm{s}}(V_{\mathrm{th}})$, $g_{\mathrm{u}}(V_{\mathrm{th}})$, reducing the high-dimensional solution space to three scalars [24]. The threshold value can be approximated following the procedure provided in appendix.

Our method proceeds in two steps (Fig 1). First, a deep network maps spike times to threshold DIC values, reducing inference to a low-dimensional space. We focused on the slow and ultra-slow DICs $\hat{g}^* \approx g^* = [g_{\mathrm{s}}(V_{\mathrm{th}}); g_{\mathrm{u}}(V_{\mathrm{th}})]$, because the fast one does not require precise quantitative inference and only needs to be sufficiently negative to support spontaneous activity [24, 29]. Second, an iterative compensation algorithm generates valid conductance vectors compatible with the inferred DICs (that is, compatible with Eq. (1)), recovering diverse biologically plausible models. Together, the pipeline reads:

$$x \xrightarrow{\text{Deep learning}} g_{\mathrm{DICs}}(V_{\mathrm{th}}) \xrightarrow{\text{Iterative compensation}} \bar{g} \in \mathcal{G}^*.$$

This requires a robust method to map back from DICs to $\bar{g}$, a synthetic dataset for training, an architecture that handles variable-length input, robustness to noise as it is intrinsic to experimental reality, and potential for generalization across CBMs.

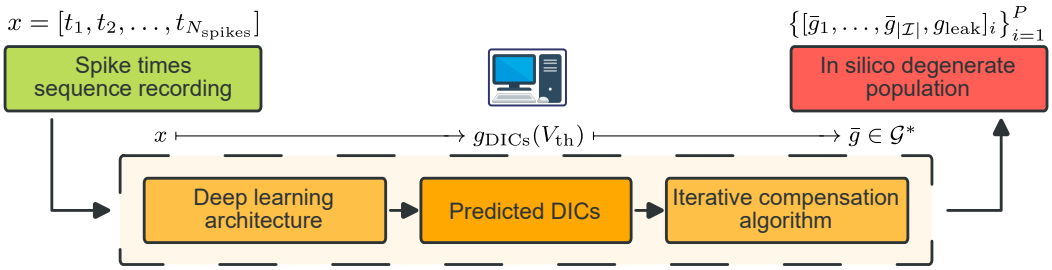

Fig 1: Pipeline for reconstructing degenerate CBM populations from spike times. Spike times x are mapped by a deep neural network to Dynamic Input Conductances (DICs) evaluated at threshold. These three interpretable values form a low-dimensional representation of excitability. An iterative compensation algorithm then maps DICs to conductance vectors $\bar{g}$, generating degenerate CBM populations that reproduce the input firing pattern. The pipeline combines efficient spike-to-DIC inference with population-level CBM reconstruction, providing a scalable solution to the inverse problem.

2 Inferring DICs from spike times

To enable prediction of DIC values from spike times, we constructed large synthetic datasets linking slow and ultra-slow DICs to diverse firing patterns [30]. They span broad DIC ranges, covering typical and edge-case neuronal activities, including spiking and bursting regimes. Spike times were extracted from simulations, classified by firing type, and quantitative firing descriptors were computed (firing frequency, intra- and inter-burst frequencies, burst duration, and number of spikes per burst) providing auxiliary representation that helped during training to structure the latent space.

An initial examination of the dataset reveals a strong relationship between g^* and the corresponding spontaneous activity. We illustrate this connection by highlighting some relations with intermediate descriptors of firing patterns (Fig 2A for bursting, and Fig 2B for spiking). Those descriptors are the mean firing frequency for spiking instances, and the mean intra- and inter-burst frequencies, burst duration, and number of spikes per burst for bursting ones (see illustrations in appendix, Fig 8). Our deep learning architecture is designed to capture and model these underlying nonlinear relationships.

The deep learning architecture directly maps variable-length spike sequences x to slow and ultra-slow DIC targets $\hat{g}^* \approx g^*$. It follows an encoder–decoder design, projecting variable-length spike sequences into a latent space and decoding to predicted DICs (see Fig. 9 and Table 4 in appendix for architecture details):

$$\underbrace{x \in \mathbb{R}^{1\times *} \xrightarrow{\texttt{Encoder } \mathcal{E}} z_{\text{latent}}}_{\text{Projection into latent space}} \in \underbrace{\mathbb{R}^{d_{\text{latent}}} \xrightarrow{\texttt{Decoder } \mathcal{R}} \hat{g}^* \in \mathbb{R}^2}_{\text{Mapping to DICs space}}. \tag{2}$$

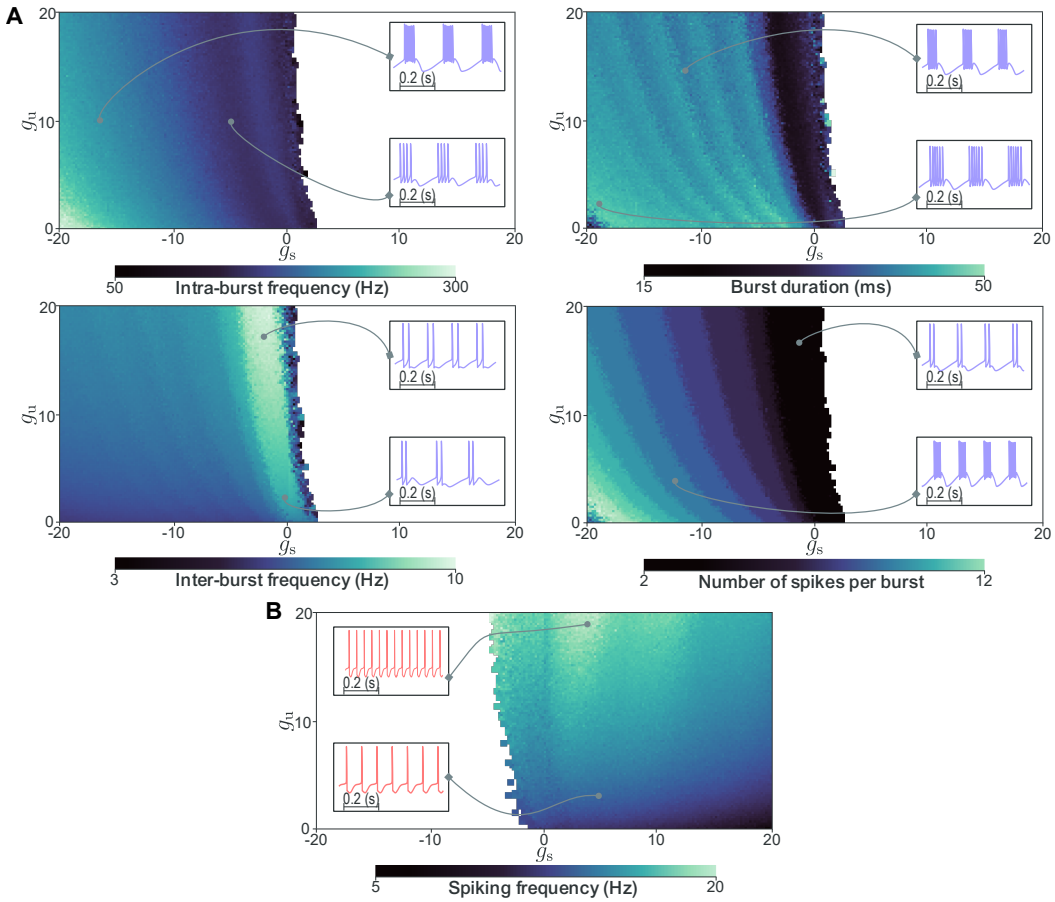

Fig 2: Activity descriptors vary smoothly across the DIC space. Heatmaps show how individual activity metrics vary across the DIC space, revealing clear gradients that reflect a nonlinear relationship between DIC constraints and neuronal firing patterns. **(A)** Bursting metrics including intra-burst frequency, inter-burst frequency, burst duration, and spikes per burst. **(B)** Spiking metrics summarized by mean firing rate. These patterns highlight the learnability of the inverse mapping from spike trains to DICs and support DICs as meaningful low-dimensional intermediates. Representative example traces illustrate diverse spiking and bursting activity within the dataset.

Auxiliary tasks, including firing class prediction and descriptors regression, are integrated to improve representation learning and generalization. Biologically inspired data augmentations, such as spike jittering and deletions, further enhance robustness. Inference is extremely fast, supporting real-time application even on standard lab hardware.

Evaluation shows that the model accurately captures the main DIC components across firing regimes (Fig 3A). Bursting activity is well represented, with predictions covering the full range of observed bursting dynamics (Fig 3B). In the bursting regime, slow and ultra-slow DICs define distinct features, since bursting requires a slow positive feedback together with an ultra-slow negative feedback [23]. In contrast, during spiking the effects of slow and ultra-slow components overlap near threshold, producing a global negative feedback that governs excitability [23, 24]. This overlap reduces the dimensionality of their combined effect, producing a one-dimensional manifold of DIC predictions and the observed degeneracy. As a result, errors in the DIC space should be interpreted with caution: due to DIC degeneracy, multiple DIC configurations can correspond to indistinguishable firing patterns, at least in spiking (Fig 3C). Moreover, DICs serve only as low-dimensional intermediates, and the ultimate objective is to generate populations of CBMs whose activity matches the input spike sequences, rather than perfectly predicting individual DIC values. Despite these considerations, the predictions span all spike-time based descriptors, ensuring coverage of the full descriptor space and

thereby the full range of spike times (Fig. 3B,C). This provides a reliable foundation for downstream population generation.

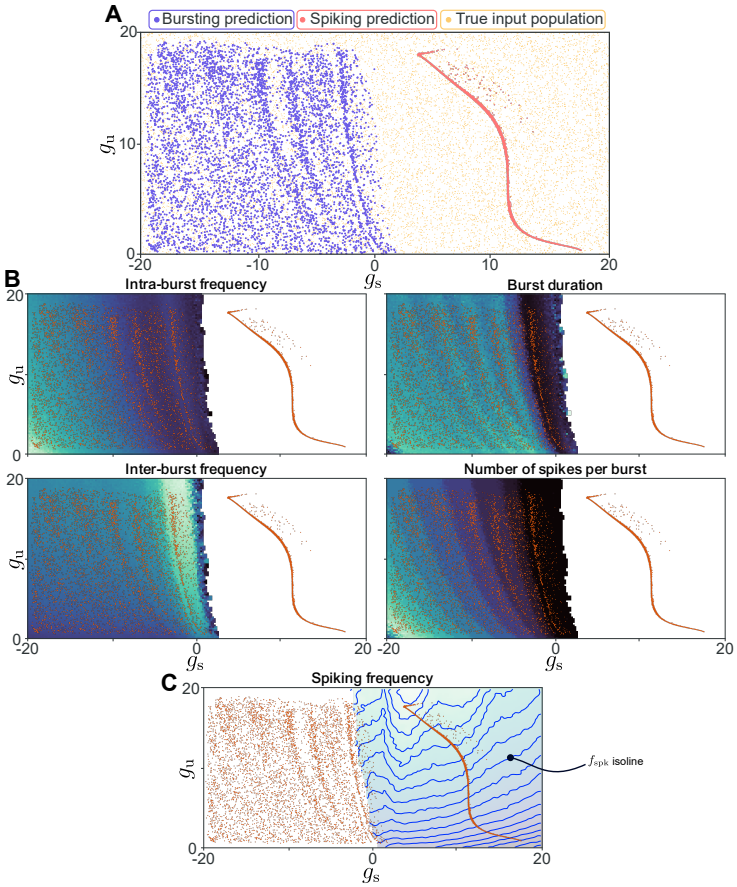

Fig 3: Architecture predictions and test set distributions in the DIC space for the STG neuron model. (A) Comparison of predictions (purple for bursting, red for spiking) with the true test set distribution (yellow) across the DIC space. Bursting inputs ($g_s < 0$) and spiking inputs ($g_s > 0$) are clearly separated, with bursting predictions spread broadly and spiking predictions confined along a one-dimensional manifold. **(B)** Predicted bursting DICs span the full range of bursting activity present in the dataset. **(C)** Predicted spiking DICs highlight the one-dimensional manifold across spiking frequencies. Blue isolines indicate spike frequency, showing how predictions traverse frequency levels transversally along the manifold. Together, these panels show that the architecture captures the geometrical organization of bursting and spiking regimes in DIC space, supporting the use of DICs as meaningful low-dimensional intermediates for neuronal activity prediction.

These results validate DICs as low-dimensional, interpretable intermediates linking spike timing to neuronal dynamics. The model captures structured relationships between DIC targets and firing patterns, providing a reliable mapping from experimentally accessible spike times to the underlying biophysical parameters that shape activity. This mapping forms the first block of our pipeline (Fig 1).

3 Building models from DICs

To generate degenerate populations of CBMs, we aim to find conductance vectors $\bar{g}$ that satisfy target DIC constraints $g_{\text{DICs}}(V_{\text{th}}) = [g_{\text{f}}(V_{\text{th}});\ g_{\text{s}}(V_{\text{th}});\ g_{\text{u}}(V_{\text{th}})]$ (second block of our pipeline, Fig 1). These targets come from the deep learning pipeline when inferring from spike times. Multiple solutions exist due to degeneracy, making Eq. (6) an underdetermined problem [23, 24].

A previous approach [24] used a linear compensation step. The conductance vector was split into a random subset $\bar{g}_{\text{random}}$ and a compensable subset $\bar{g}_{\text{comp.}}$, with the latter adjusted to satisfy the

DIC constraints (Eq. (1)). The random subset was sampled first and ensured degeneracy while making the compensation problem fully determined with respect to the remaining conductances $\bar{g}_{\text{comp.}}$. Because the sensitivity structure can be nonlinear, some of the unknowns may appear in $S(V;\bar{g})$. In [24], they tackled this problem by approximating $S(V;\bar{g}) \approx \hat{S}(V;\hat{g})$, effectively approximating the compensation into a linear one with respect to the compensable conductances. They used $\hat{g} = [\bar{g}_{\text{random}}; \hat{g}_{\text{comp.}}]$, with $\hat{g}_{\text{comp.}}$ fixed for a given CBM.

Their approach is exact when the sensitivity matrix is independent of $\bar{g}_{\text{comp.}}$, but fails for nonlinear compensation, producing residual errors and outliers in activity distributions. To address this, we introduce the **iterative compensation algorithm**. We perform several linear compensation steps, using the solution of the previous one to approximate the sensitivity with a better estimate. Starting from an initial guess $\bar{g}^{(0)}_{\text{comp.}} = \hat{g}_{\text{comp.}}$, $\bar{g}^{(k)}_{\text{comp.}}$ is updated iteratively, recomputing an approximation of the sensitivity matrix at each step until the DIC residual norm is sufficiently small:

$$S(V_{\text{th}}; [\bar{g}_{\text{random}}; \bar{g}^{(k)}_{\text{comp.}}]) \cdot \begin{bmatrix} \bar{g}_{\text{random}} \\ \bar{g}^{(k+1)}_{\text{comp.}} \end{bmatrix} = \bar{g}_{\text{DICs}}(V_{\text{th}}), \quad k = 0, \ldots, K+1 \quad . \tag{3}$$

This procedure reduces deviations between target and actual DICs while preserving variability in $\bar{g}_{\text{random}}$, ensuring heterogeneous yet functionally consistent populations. Fig 4A shows that after five iterations ($K = 5$), residuals decrease by over an order of magnitude compared to the linear method. The resulting populations also exhibit tighter, more consistent activity statistics across spiking (Fig 4B) and bursting neurons (Fig 4C), while maximal conductances remain heterogeneous.

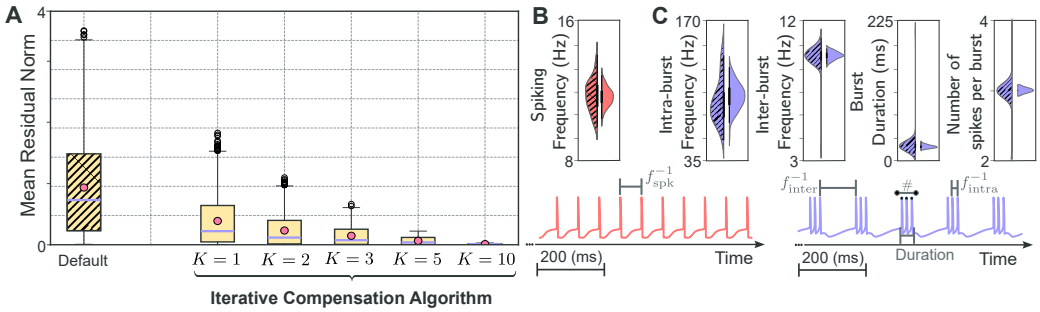

Fig 4: Iterative compensation improves constraint satisfaction and preserves degeneracy in CBMs with nonlinear dynamics. (A) Mean residuals of DIC constraints across 1,633 targets and 250 instances per population, comparing linear compensation (hatched, 0 iterations from [24]) to iterative compensation (plain), showing a rapid reduction in residuals. **(B)** Distribution of mean firing frequency f_{spk} in spiking neurons (red), demonstrating tighter and more consistent activity with iterative compensation (plain) compared to linear (hatched). **(C)** Distributions of bursting features (intra-burst frequency f_{intra}, inter-burst frequency f_{inter}, burst duration, spikes per burst $\#$) in bursting neurons (purple), showing more compact activity profiles with iterative compensation.

By iteratively enforcing DIC constraints, this algorithm provides a robust and accurate method to generate diverse degenerate CBM populations from low-dimensional DIC representations. It forms the foundation both for creating the large synthetic datasets used to train the deep learning model and for generating populations at inference in the full pipeline (Fig 1).

4 Pipeline evaluation and performance

Reconstructing accurate and diverse degenerate populations from spike times. Our generative pipeline converts spike time sequences into degenerate CBM populations that reproduce the input activity while spanning diverse conductance configurations (Fig 1). Using DICs at threshold as intermediates, the method combines deep learning-based DIC predictions with the iterative compensation algorithm, generating populations rather than single solutions.

Generated spike trains closely match target activity in both spiking and bursting regimes (Fig 5A), with variability consistent with biological degeneracy. Maximal conductances remain heterogeneous

across populations (Fig 5B), confirming that multiple distinct solutions produce equivalent functional outputs.

Quantitative evaluation across 500 input populations, each of 16 instances, shows that reconstructed distributions of spiking frequency, intra- and inter-burst frequencies, burst duration, and spikes per burst closely match input statistics (see appendix for reconstruction quantitative analysis).

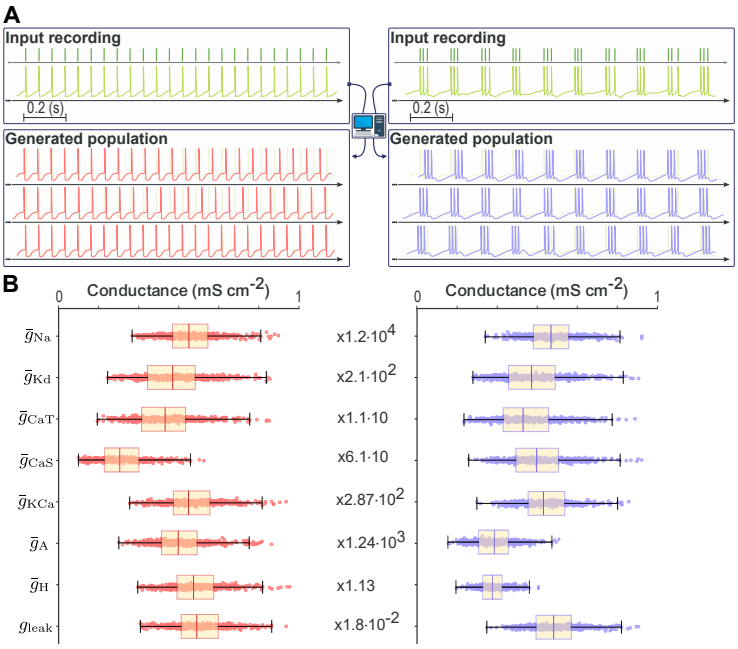

Fig 5: Backbone pipeline output for the stomatogastric ganglion (STG) neuron model. (A) Target spike trains (dark green, top) compared with generated populations for spiking (red, left) and bursting (purple, right) regimes, showing accurate reproduction of activity patterns. **(B)** Distributions of maximal conductances across 500 generated neurons in spiking (red, left) and bursting (purple, right) regimes, demonstrating broad parameter variability despite similar dynamics.

Robustness to stochastic inputs. To test generalization under realistic variability, we applied the pipeline to spike trains generated from homogeneous Poisson processes and Poisson bursting generators. Such processes closely mimic biological data and are a common model of biological variability [31–33]. Despite training on regular patterns, the pipeline successfully reconstructs spiking and bursting populations with accurate statistics (Fig 6). Mean absolute error (MAE) values for spiking frequency and bursting metrics remain low across input ranges, and classification of spiking versus bursting neurons is consistent with biological priors. The method averages over variability while producing coherent populations, demonstrating resilience to temporal irregularity.

Generalization across CBM types. Using low-rank adaptation (LoRA) [34], the STG-trained backbone was efficiently transferred to a dopaminergic (DA) neuron model with minimal additional parameters. The adapted pipeline reproduces DA firing patterns, including spiking, bursting, and fast-spiking classes, while maintaining broad conductance variability. This confirms that the pipeline generalizes across neuron types with different dynamics, preserving both accuracy and degeneracy.

5 Conclusion and perspective

In summary, the pipeline faithfully reconstructs firing patterns from spike times, preserves population-level degeneracy, is resilient to stochastic inputs, and scales efficiently across distinct CBM models, demonstrating the robustness and versatility of DIC-based intermediate representations.

We demonstrated that our pipeline generalizes across activity regimes, noisy spike trains, and distinct conductance-based models, consistently preserving degeneracy and accuracy. Dynamic Input

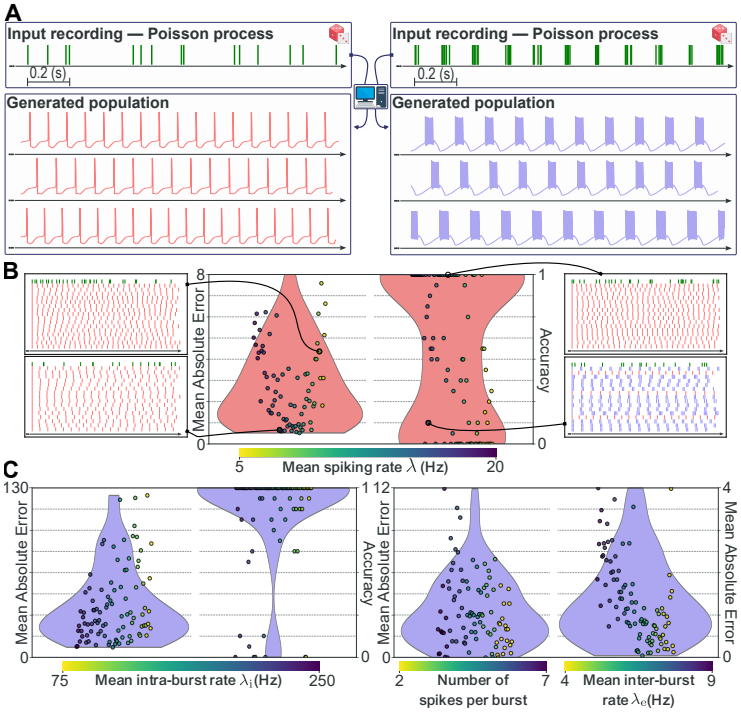

Fig 6: Comparison of input Poisson spike trains with generated populations. (A) Example input spike trains (top, dark green) from spiking (left) and bursting (right) Poisson processes, with corresponding generated population traces (bottom; red for spiking, purple for bursting). (B) Violin plots show mean absolute error (MAE) between input firing rate λ and population-averaged spiking frequency across 16 generated instances (left) and spiking accuracy, i.e., proportion classified as spikers (right). Insets show example input (top) and generated (bottom) spike rasters. (C) Violin plots for bursting inputs display MAE of intra-burst rate λ_i, bursting accuracy, MAE of spikes per burst, and MAE of inter-burst rate λ_e; scatter points show errors for each population.

Conductances (DICs) serve as a robust intermediate representation, linking spike times to underlying conductance variability. By combining a deep learning model predicting DICs from spikes with an iterative compensation algorithm, the approach provides an efficient and interpretable solution to the inverse inference problem.

The iterative compensation algorithm improves enforcement of DIC constraints while maintaining biological heterogeneity. Although illustrated on STG neurons, the method is broadly applicable to other CBMs with nonlinear compensatory structures [10, 35], enabling the generation of degenerate populations consistent with target constraints.

Limitations include reliance on spike times, which excludes subthreshold dynamics. Extensions could incorporate DICs at multiple voltages or responses to external stimuli, enhancing the reconstruction of richer neuronal behaviors. At the network level, the framework could integrate synaptic conductances, paving the way for real-time applications such as closed-loop neuromodulation [29, 36] or online tuning of neuromorphic devices [37].

Beyond inference, the pipeline allows reconstruction of conductance distributions from spike times, providing a principled alternative to averaging conductances across neurons [28]. This enables experimental investigation of how modulators reshape conductance variability and neuronal activity.

The full framework is released as open-source software with a graphical interface [26], facilitating adoption by researchers without programming expertise. The computational resources required are small, and real-time application on standard lab hardware is possible. In conclusion, combining deep learning with DIC theory allows efficient reconstruction of degenerate conductance-based populations, offering a scalable and interpretable tool to study how degeneracy supports robust neuronal computation.

References

[1] Paul Tiesinga, Jean-Marc Fellous, and Terrence J. Sejnowski. Regulation of spike timing in visual cortical circuits. *Nature Reviews. Neuroscience*, 9(2):97–107, 2008. ISSN 1471-0048. doi: 10.1038/nrn2315.

[2] Z. F. Mainen and T. J. Sejnowski. Reliability of spike timing in neocortical neurons. *Science (New York, N.Y.)*, 268(5216):1503–1506, 1995. ISSN 0036-8075. doi: 10.1126/science.7770778.

[3] Alex H. Williams, Ben Poole, Niru Maheswaranathan, Ashesh K. Dhawale, Tucker Fisher, Christopher D. Wilson, David H. Brann, Eric M. Trautmann, Stephen Ryu, Roman Shusterman, Dmitry Rinberg, Bence P. Ölveczky, Krishna V. Shenoy, and Surya Ganguli. Discovering precise temporal patterns in large-scale neural recordings through robust and interpretable time warping. *Neuron*, 105(2):246–259.e8, 2020. ISSN 0896-6273. doi: 10.1016/j.neuron.2019.10.020.

[4] Eve Marder. Neuromodulation of neuronal circuits: Back to the future. *Neuron*, 76(1):1–11, 2012. ISSN 0896-6273. doi: 10.1016/j.neuron.2012.09.010.

[5] Cornelia I. Bargmann and Eve Marder. From the connectome to brain function. *Nature Methods*, 10(6): 483–490, June 2013. ISSN 1548-7105. doi: 10.1038/nmeth.2451.

[6] David A. McCormick, Dennis B. Nestvogel, and Biyu J. He. Neuromodulation of brain state and behavior. *Annual Review of Neuroscience*, 43(Volume 43, 2020):391–415, July 2020. ISSN 0147-006X, 1545-4126. doi: 10.1146/annurev-neuro-100219-105424.

[7] Jean-Marc Goaillard and Eve Marder. Ion channel degeneracy, variability, and covariation in neuron and circuit resilience. *Annual Review of Neuroscience*, 44:335–357, 2021. ISSN 0147-006X, 1545-4126. doi: 10.1146/annurev-neuro-092920-121538.

[8] Zheng Liu, Jorge Golowasch, Eve Marder, and L. F. Abbott. A model neuron with activity-dependent conductances regulated by multiple calcium sensors. *Journal of Neuroscience*, 18(7):2309–2320, 1998. ISSN 0270-6474, 1529-2401. doi: 10.1523/JNEUROSCI.18-07-02309.1998.

[9] Adam L. Taylor, Jean-Marc Goaillard, and Eve Marder. How multiple conductances determine electrophysiological properties in a multicompartment model. *Journal of Neuroscience*, 29(17):5573–5586, 2009. ISSN 0270-6474, 1529-2401. doi: 10.1523/JNEUROSCI.4438-08.2009.

[10] Na Yu and Carmen C. Canavier. A mathematical model of a midbrain dopamine neuron identifies two slow variables likely responsible for bursts evoked by sk channel antagonists and terminated by depolarization block. *The Journal of Mathematical Neuroscience (JMN)*, 5(1):5, 2015. ISSN 2190-8567. doi: 10.1186/s13408-015-0017-6.

[11] Mark S. Goldman, Jorge Golowasch, Eve Marder, and L. F. Abbott. Global structure, robustness, and modulation of neuronal models. *Journal of Neuroscience*, 21(14):5229–5238, 2001. ISSN 0270-6474, 1529-2401. doi: 10.1523/JNEUROSCI.21-14-05229.2001.

[12] Jorge Golowasch, L. F. Abbott, and Eve Marder. Activity-dependent regulation of potassium currents in an identified neuron of the stomatogastric ganglion of the crab cancer borealis. *Journal of Neuroscience*, 19 (20):RC33–RC33, 1999. ISSN 0270-6474, 1529-2401. doi: 10.1523/JNEUROSCI.19-20-j0004.1999.

[13] Shaul Druckmann, Yoav Banitt, Albert A. Gidon, Felix Schürmann, Henry Markram, and Idan Segev. A novel multiple objective optimization framework for constraining conductance-based neuron models by experimental data. *Frontiers in Neuroscience*, 1, October 2007. ISSN 1662-453X. doi: 10.3389/neuro.01.1.1.001.2007.

[14] Astrid A. Prinz, Cyrus P. Billimoria, and Eve Marder. Alternative to hand-tuning conductance-based models: Construction and analysis of databases of model neurons. *Journal of Neurophysiology*, 90(6): 3998–4015, 2003. ISSN 0022-3077. doi: 10.1152/jn.00641.2003.

[15] Dylan Shepardson. *Algorithms for Inverting Hodgkin-Huxley Type Neuron Models*. Ph.d. thesis, Georgia Institute of Technology, 2009. URL `http://hdl.handle.net/1853/31686`.

[16] Quentin J. M. Huys, Misha B. Ahrens, and Liam Paninski. Efficient estimation of detailed single-neuron models. *Journal of Neurophysiology*, 96(2):872–890, August 2006. ISSN 0022-3077. doi: 10.1152/jn.00079.2006.

[17] Thiago B. Burghi, Maarten Schoukens, and Rodolphe Sepulchre. Feedback identification of conductance-based models. *Automatica*, 123:109297, January 2021. ISSN 0005-1098. doi: 10.1016/j.automatica.2020.109297.

[18] Thiago B. Burghi, Maria Ivanova, Ekaterina Morozova, Huaxinyu Wang, Eve Marder, and Timothy O'Leary. Rapid, interpretable data-driven models of neural dynamics using recurrent mechanistic models. *Proceedings of the National Academy of Sciences*, 122(32):e2426916122, 2025. doi: 10.1073/pnas.2426916122.

[19] Liang Meng, Mark A. Kramer, and Uri T. Eden. A sequential monte carlo approach to estimate biophysical neural models from spikes. *Journal of Neural Engineering*, 8(6):065006, 2011. ISSN 1741-2552. doi: 10.1088/1741-2560/8/6/065006.

[20] Liang Meng, Mark A. Kramer, Steven J. Middleton, Miles A. Whittington, and Uri T. Eden. A unified approach to linking experimental, statistical and computational analysis of spike train data. *PLOS ONE*, 9 (1):e85269, 2014. ISSN 1932-6203. doi: 10.1371/journal.pone.0085269.

[21] Susanne Ditlevsen and Adeline Samson. Estimation in the partially observed stochastic morris–lecar neuronal model with particle filter and stochastic approximation methods. *The Annals of Applied Statistics*, 8(2):674–702, June 2014. ISSN 1932-6157, 1941-7330. doi: 10.1214/14-AOAS729.

[22] 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, David S Greenberg, and Jakob H Macke. Training deep neural density estimators to identify mechanistic models of neural dynamics. *eLife*, 9:e56261, 2020. ISSN 2050-084X. doi: 10.7554/eLife.56261.

[23] Guillaume Drion, Alessio Franci, Julie Dethier, and Rodolphe Sepulchre. Dynamic input conductances shape neuronal spiking. *eNeuro*, 2(1), 2015. ISSN 2373-2822. doi: 10.1523/ENEURO.0031-14.2015.

[24] Arthur Fyon, Alessio Franci, Pierre Sacré, and Guillaume Drion. Dimensionality reduction of neuronal degeneracy reveals two interfering physiological mechanisms. *PNAS Nexus*, 3(10):pgae415, October 2024. ISSN 2752-6542. doi: 10.1093/pnasnexus/pgae415.

[25] Kun Qian, Na Yu, Kristal R. Tucker, Edwin S. Levitan, and Carmen C. Canavier. Mathematical analysis of depolarization block mediated by slow inactivation of fast sodium channels in midbrain dopamine neurons. *Journal of Neurophysiology*, 112(11):2779–2790, 2014. ISSN 1522-1598. doi: 10.1152/jn.00578.2014.

[26] Julien Brandoit. Spike2pop: Bridging experimental neuroscience and computational modeling. Software available at `https://github.com/julienbrandoit/Spike2Pop---Bridging-Experimental-Neuroscience-and-Computational-Modeling`, 2025. Version 1.1.1.

[27] Eve Marder and Adam L. Taylor. Multiple models to capture the variability in biological neurons and networks. *Nature Neuroscience*, 14(2):133–138, 2011. ISSN 1546-1726. doi: 10.1038/nn.2735.

[28] Jorge Golowasch, Mark S. Goldman, L. F. Abbott, and Eve Marder. Failure of averaging in the construction of a conductance-based neuron model. *Journal of Neurophysiology*, 87(2):1129–1131, 2002. ISSN 0022-3077. doi: 10.1152/jn.00412.2001.

[29] A. Fyon, P. Sacré, A. Franci, and G. Drion. Reliable neuromodulation from adaptive control of ion channel expression. *IFAC-PapersOnLine*, 56(2):458–463, 2023. ISSN 2405-8963. doi: 10.1016/j.ifacol.2023.10.1610.

[30] Julien Brandoit, Damien Ernst, Guillaume Drion, and Arthur Fyon. Degenerate conductance-based models populations datasets: Stg and da models, Aug 2025.

[31] Charles F. Stevens and Anthony M. Zador. Input synchrony and the irregular firing of cortical neurons. *Nature Neuroscience*, 1(3):210–217, 1998. ISSN 1546-1726. doi: 10.1038/659.

[32] W. R. Softky and C. Koch. The highly irregular firing of cortical cells is inconsistent with temporal integration of random EPSPs. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, 13(1):334–350, 1993. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.13-01-00334.1993.

[33] Mark M. Churchland, Byron M. Yu, John P. Cunningham, Leo P. Sugrue, Marlene R. Cohen, Greg S. Corrado, William T. Newsome, Andrew M. Clark, Paymon Hosseini, Benjamin B. Scott, David C. Bradley, Matthew A. Smith, Adam Kohn, J. Anthony Movshon, Katherine M. Armstrong, Tirin Moore, Steve W. Chang, Lawrence H. Snyder, Stephen G. Lisberger, Nicholas J. Priebe, Ian M. Finn, David Ferster, Stephen I. Ryu, Gopal Santhanam, Maneesh Sahani, and Krishna V. Shenoy. Stimulus onset quenches neural variability: A widespread cortical phenomenon. *Nature Neuroscience*, 13(3):369–378, 2010. ISSN 1546-1726. doi: 10.1038/nn.2501.

[34] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[35] Guillaume Drion, Laurent Massotte, Rodolphe Sepulchre, and Vincent Seutin. How modeling can reconcile apparently discrepant experimental results: The case of pacemaking in dopaminergic neurons. *PLOS Computational Biology*, 7(5):e1002050, 2011. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1002050.

[36] Adam O. Hebb, Jun Jason Zhang, Mohammad H. Mahoor, Christos Tsiokos, Charles Matlack, Howard Jay Chizeck, and Nader Pouratian. Creating the feedback loop: Closed-loop neurostimulation. *Neurosurgery Clinics of North America*, 25(1):187–204, 2014. ISSN 1558-1349. doi: 10.1016/j.nec.2013.08.006.

[37] Loris Mendolia and Alessio Franci. Designing, tuning and building ultra-low-power neuromodulable mixed-feedback silicon neurons. In *44th Benelux Meeting on Systems and Control*, 2025.

[38] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[39] Lawrence F. Shampine and Mark W. Reichelt. The matlab ode suite. *SIAM Journal on Scientific Computing*, 18(1):1–22, 1997. ISSN 1064-8275. doi: 10.1137/S1064827594276424.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[41] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[42] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012.

[43] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023.

[44] Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1): 73–101, 1964. ISSN 0003-4851, 2168-8990. doi: 10.1214/aoms/1177703732.

[45] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(null):281–305, February 2012. ISSN 1532-4435.

[46] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, January 2019.

[47] Prajit Ramachandran, raarret Zoph, and Quoc V. Le. Swish: A self-gated activation function, October 2017.

Appendix

A Conductance-based models details

We use conductance-based models (CBMs) to simulate neuronal dynamics. CBMs are biologically plausible and grounded in biophysical principles, describing the membrane potential V as a function of ionic and leak currents across the membrane. These dynamics are governed by the following equation:

$$C\frac{dV}{dt} + g_{\text{leak}}(V - E_{\text{leak}}) = -\sum_{i\in\mathcal{I}} \bar{g}_i m_i^{p_i}(V,t) h_i^{q_i}(V,t)(V - E_i) + I_{\text{ext}} \quad . \tag{4}$$

Here, C is the membrane capacitance, set to $1\,\mu\text{F}\,\text{cm}^{-2}$. The set $\mathcal{I}$ contains all ionic conductances considered for the model. Each ionic current is characterized by its maximal conductance $\bar{g}_i$, reflecting the maximal effective channel density, and by gating variables m_i and h_i raised to integer powers p_i and q_i, representing activation and inactivation dynamics, respectively. E_i is the Nernst reversal potential of the considered ion. The leak current $I_{\text{leak}} = g_{\text{leak}}(V - E_{\text{leak}})$ models passive ion flow. External input current I_{ext} (e.g., injected current or synaptic input) is set to zero in all simulations

($I_{\text{ext}} = 0$). In this work, we denote $\bar{g} = [\bar{g}_1;\ \bar{g}_2;\ \ldots;\ \bar{g}_{|\mathcal{I}|}, g_{\text{leak}}] \in \mathbb{R}_+^{N_{\text{model}}} = \mathcal{G}$ the vector of maximal conductances (extended with the leak conductance) that distinguishes different instances of a model through its N_{model} components. The maximal conductances vector contains the only parameters that are not treated as fixed constants in this work.

Gating variables $X \in \{m_i, h_i\}$ are dimensionless quantities constrained between 0 and 1. They follow first-order voltage-dependent dynamics of the form:

$$\tau_X(V)\frac{dX}{dt} = X_\infty(V) - X \quad , \tag{5}$$

where $\tau_X(V)$ is the voltage-dependent time constant describing how rapidly the gating variable responds to voltage changes, and $X_\infty(V)$ is the steady-state activation or inactivation value.

As a proof of concept for our pipeline, we use two established neuron models: a slightly modified isolated stomatogastric ganglion (STG) neuron model [8], in which we simplify the calcium reversal potential computation to reduce complexity; and a dopaminergic (DA) neuron model [25], in which we block SK channels to enable bursting in a wider range of conductances. These two models exhibit different firing patterns. The ones considered in this work are spiking and bursting for the STG model and slow pacemaking, bursting, and fast spiking for the DA model. Biologically, these two neurons have different roles and timescales.

The stomatogastric ganglion neuron model

The STG neuron model is adapted from [8] and includes 7 ionic conductances: g_{Na} (fast sodium), g_{Kd} (delayed rectifier potassium), g_{KCa} (calcium-dependent potassium), g_{A} (transient A-type potassium), g_{CaS} (slow calcium), g_{CaT} (transient T-type calcium), and g_{H} (hyperpolarization-activated inward cation).

This model explicitly incorporates intracellular calcium concentration dynamics through an additional ordinary differential equation involving $\frac{d\text{Ca}}{dt}$, which modulates calcium-dependent potassium currents through a voltage- and calcium-dependent steady-state gating function $m_{\infty,\text{KCa}}(V, \text{Ca})$. The calcium dynamic is governed by:

$$\tau_{\text{Ca}}\frac{d\text{Ca}}{dt} = -\alpha_{\text{Ca}}\left(I_{\text{CaS}} + I_{\text{CaT}}\right) - \text{Ca} + \beta_{\text{Ca}} \quad .$$

Table 1 provides the values of the fixed model parameters. Simulations were performed as described in the following appendix about the datasets generation. The transients were discarded; initial conditions used were $V_0 = -70\,\text{mV}$ and $\text{Ca}_0 = 0.5\,\mu\text{M}$, with initial gating variables evaluated at steady-state $X_0 = X_\infty(V_0; \text{Ca}_0)$. Table 7 lists the mathematical expressions for the steady-state gating functions, the time constant functions, and the corresponding gating exponents.

Table 1: **Fixed parameters for the STG model.** Reversal potentials, calcium time constant, and calcium dynamics parameters used in the STG model.

E_{leak}	E_{Na}	E_{K}	E_{H}	E_{Ca}	τ_{Ca}	α_{Ca}	β_{Ca}
$-50\,\text{mV}$	$50\,\text{mV}$	$-80\,\text{mV}$	$-20\,\text{mV}$	$80\,\text{mV}$	$20\,\text{ms}$	$0.94\,\text{mM}\,\text{nF}\,\text{nA}^{-1}$	$0.05\,\mu\text{M}$

The dopaminergic neuron model

The DA neuron model [25] includes 6 ionic conductances: g_{Na} (fast sodium), g_{Kd} (delayed rectifier potassium), g_{CaL} (L-type calcium), g_{CaN} (N-type calcium), g_{ERG} (ether-à-go-go–related gene potassium), and g_{NMDA} (NMDA receptor-mediated). It incorporates a magnesium-sensitive current through NMDA influenced by a fixed magnesium concentration. The NMDA current was considered instantaneous and always evaluated at its steady-state value:

$$I_{\text{NMDA}} = \bar{g}_{\text{NMDA}}(V - E_{\text{NMDA}}) \cdot m_{\text{NMDA},\infty}(V, \text{Mg}) \quad ,$$

where $m_{\text{NMDA},\infty}$ is the voltage- and magnesium-dependent steady-state activation function. For the ERG channel, gating variables o_{ERG} and i_{ERG} follow:

$$\frac{do_{\text{ERG}}}{dt} = a_0(V)(1 - o_{\text{ERG}} - i_{\text{ERG}}) + b_i(V)i_{\text{ERG}} - o_{\text{ERG}}(a_i(V) + b_0(V)),$$
$$\frac{di_{\text{ERG}}}{dt} = a_i(V)o_{\text{ERG}} - b_i(V)i_{\text{ERG}}.$$

At steady state,

$$o_{\text{ERG},\infty}(V) = \frac{a_0(V)b_i(V)}{a_0(V)(a_i(V) + b_i(V)) + b_0(V)b_i(V)},$$
$$i_{\text{ERG},\infty}(V) = \frac{a_0(V)a_i(V)}{a_0(V)(a_i(V) + b_i(V)) + b_0(V)b_i(V)}.$$

The current is:

$$I_{\text{ERG}} = \bar{g}_{\text{ERG}}\, o_{\text{ERG}}(V - E_K).$$

Fixed model parameters are provided in Table 2. Simulations were performed as described in the following appendix about the datasets generation. The transients were discarded; initial condition used was $V_0 = -90\,\text{mV}$, with initial gating variables evaluated at steady-state $X_0 = X_\infty(V_0)$. Table 8 lists the mathematical expressions for the steady-state gating functions, the time constant functions, and the corresponding gating exponents.

Table 2: **Fixed parameters for the DA model.** Reversal potentials and magnesium concentration used in the DA model.

E_{Na}	E_{K}	E_{Ca}	E_{leak}	E_{NMDA}	Mg
$60\,\text{mV}$	$-85\,\text{mV}$	$60\,\text{mV}$	$-50\,\text{mV}$	$0\,\text{mV}$	1.4

B Dynamic input conductances (DICs) details

While CBMs are biologically grounded, they are generally not analytically tractable, as it is not possible to directly infer the membrane potential from $\bar{g}$ without simulation. Dynamic input conductances (DICs), introduced in [23], address this limitation by providing a scalable and mathematically grounded framework for analyzing CBMs. The DIC formalism decomposes the total membrane response into a sum of timescale-specific components that reflect the dynamical influence of different ionic currents.

B.1 Timescale decomposition of membrane dynamics

We partition the influence of membrane currents into three characteristic temporal components. The fast dynamic conductance $g_{\text{f}}(V)$ governs the rapid voltage changes underlying spike upstroke. The slow conductance $g_{\text{s}}(V)$ regulates membrane repolarization and interspike interval (ISI) behavior. Finally, the ultra-slow component $g_{\text{u}}(V)$ accounts for long-term subthreshold integration, adaptation, and shapes the bursting envelope.

In CBMs, these timescale-specific conductances can be computed analytically. For each ionic gating variable or internal state X_i, we evaluate its influence on the membrane current through:

$$\begin{aligned}
g_{\text{f}}(V) &= \frac{1}{g_{\text{leak}}}\left[-\frac{\partial \dot{V}}{\partial V} - \sum_i w_{\text{fs},X_i}\left(\frac{\partial \dot{V}}{\partial X_i}\cdot\frac{\partial X_{i,\infty}}{\partial V}\right)\right]\Bigg|_V, \\
g_{\text{s}}(V) &= \frac{1}{g_{\text{leak}}}\left[-\sum_i (w_{\text{su},X_i} - w_{\text{fs},X_i})\left(\frac{\partial \dot{V}}{\partial X_i}\cdot\frac{\partial X_{i,\infty}}{\partial V}\right)\right]\Bigg|_V, \\
g_{\text{u}}(V) &= \frac{1}{g_{\text{leak}}}\left[-\sum_i (1 - w_{\text{su},X_i})\left(\frac{\partial \dot{V}}{\partial X_i}\cdot\frac{\partial X_{i,\infty}}{\partial V}\right)\right]\Bigg|_V.
\end{aligned} \tag{6}$$

The voltage dependence notation is omitted in the right-hand sides for improved readability. The weighting functions $w_{\text{fs},X_i}(V)$ and $w_{\text{su},X_i}(V)$ distribute the contribution of each variable across timescales based on their voltage-dependent kinetics. These weights are defined as logarithmic distances relative to chosen reference time constants that separate fast, slow, and ultra-slow regimes. As a result, a given ionic current can contribute simultaneously to multiple timescales, depending on the behavior of its gating variables across voltages. The sign convention and the normalization by the leak conductance used here follow the approach in [24], and thus differ from the original formulation in [23]. The exact functional form of these weights and details about DICs are provided in [23].

A more convenient formulation of relations (6) is possible through a sensitivity matrix $S(V)$:

$$g_{\text{DICs}}(V) = \begin{bmatrix} g_{\text{f}}(V) \\ g_{\text{s}}(V) \\ g_{\text{u}}(V) \end{bmatrix} = S(V;\bar{g}) \cdot \bar{g} \quad . \tag{7}$$

The sensitivity matrix summarizes a normalized influence of each conductance on the different timescales that should be scaled by the maximal conductances to get the underlying dynamic conductances of a given instance. When S is independent of $\bar{g}$, we say that the compensation structure is *linear*, whereas the opposite is called a *nonlinear compensation*.

B.2 Dimensionality reduction via DICs

In the DIC framework, neuronal excitability is largely characterized by the values of the conductance components at a critical voltage called the threshold potential V_{th}. This threshold corresponds to the voltage at which the neuron is maximally sensitive to changes in its maximal conductance parameters [23]. Following [24], we approximate V_{th} as the first decreasing zero of the total conductance curve $g_{\text{t}} = g_{\text{f}} + g_{\text{s}} + g_{\text{u}}$:

$$g_{\text{t}}(V_{\text{th}}) = 0 \;\text{ with }\; g_{\text{t}}(V_{\text{th}} - \delta V) > 0 > g_{\text{t}}(V_{\text{th}} + \delta V), \quad \forall \text{ sufficiently small } \delta V > 0. \tag{8}$$

Evaluating the DICs at this voltage yields a compact vector representation of the model:

$$g_{\text{DICs}}(V_{\text{th}}) = \begin{bmatrix} g_{\text{f}}(V_{\text{th}}) \\ g_{\text{s}}(V_{\text{th}}) \\ g_{\text{u}}(V_{\text{th}}) \end{bmatrix} \in \mathbb{R}^3 \quad . \tag{9}$$

This transforms the high-dimensional parameter space $\bar{g} \in \mathcal{G}$ into a low-dimensional space:

$$\bar{g} \in \mathcal{G} \quad \longrightarrow \quad g_{\text{DICs}}(V_{\text{th}}) \in \mathbb{R}^3 \quad . \tag{10}$$

Importantly, this mapping is not injective: many distinct sets of maximal conductances can yield the same DIC vector and therefore similar activity. This property enables controlled exploration of degeneracy in CBM populations.

B.3 Generating maximal conductances from DICs

The method introduced in [24] provides a principled way to generate maximal conductance vectors $\bar{g}$ from $g_{\text{DICs}}(V_{\text{th}})$. This approach samples directly from the solution space defined by the DIC constraints at threshold, allowing the generation of degenerate model realizations that exhibit the same spontaneous activity.

The procedure consists of two compensation steps. Each compensation can be described in two parts and aims to make $\bar{g}$ compatible with DIC constraints. The first part samples a subset of the conductance vector, and the second computes the remaining components such that they are compatible with the constraint we want to enforce on DIC values. We proceed in two compensations performed in series in order to (i) first impose a sufficiently negative fast DIC value $g_{\text{f}}(V_{\text{th}})$, such that the resulting $\bar{g}$ will be spontaneously active, and (ii) to modulate the spontaneous activity toward the targeted one, associated with DIC constraints on the slow and ultra-slow values $g_{\text{s}}(V_{\text{th}})$ and $g_{\text{u}}(V_{\text{th}})$, respectively. Therefore, the first compensation aims to constrain $n = 3$ DIC values (the fast, the slow and the ultra-slow), and the second $n = 2$ DIC values (only the slow and the ultra-slow, the fast one being already enforced to be sufficiently negative). The DIC constraints of the second step are either the one that we infer using deep learning $\hat{g}^*$, or g^* that we get from the uniform sampling when generating

the dataset. The DIC constraints of the first step are fixed and chosen such that the population is spontaneously active [24].

The first step starts by sampling a subset $\bar{g}_{\text{random}} \sim \mathcal{D}_{\text{generation}}$ of the maximal conductance vector $\bar{g} = [\bar{g}_{\text{random}}; \bar{g}_{\text{comp.}}]$, from $\mathcal{D}_{\text{generation}}$ taken to extend beyond the biological range [7, 24]. To impose $n = 3$ DIC values, the subset $\bar{g}_{\text{random}}$ is such that it has $N_{\text{model}} - n$ components. The remaining n components of $\bar{g}$ correspond to the subset $\bar{g}_{\text{comp.}}$, whose values are determined through the linear compensation step. A convenient way of writing this compensation step is by decomposing the sensitivity matrix into $S = [S_{\text{random}}; S_{\text{comp.}}]$ following $\bar{g}$:

$$\underbrace{S_{\text{comp.}}(V_{\text{th}})}_{A} \cdot \bar{g}_{\text{comp.}} = \underbrace{g_{\text{DICs}}(V_{\text{th}}) - S_{\text{random}}(V_{\text{th}}) \cdot \bar{g}_{\text{random}}}_{b} \quad . \tag{11}$$

We introduce the following notations, $A := S_{\text{comp.}}(V_{\text{th}}) \in \mathbb{R}^{n \times (N_{\text{model}} - n)}$ and $b := g_{\text{DICs}}(V_{\text{th}}) - S_{\text{random}}(V_{\text{th}}) \cdot \bar{g}_{\text{random}} \in \mathbb{R}^n$ that will be helpful to describe our iterative compensation algorithm.

The second step follows a similar structure, starting from the $\bar{g}$ resulting from the first compensation, a system similar to (11) is built based on a decomposition into two subsets that can differ from the one used during the initial compensation (see [24]). Since we start from the results of the first step, no sampling is done at the beginning of this compensation step.

The generation procedure described above relies on two approximations. First, the threshold potential V_{th} must be chosen a priori, rather than determined consistently from the total conductance curve of the compensated solution. Second, the generation step is exact only if the compensation structure is linear in the compensated conductances. In [24], when the sensitivity matrix depends on the compensated conductances, it is approximated using a fixed default value $\hat{g}_{\text{comp.}}$, i.e. $S(V_{\text{th}}; \hat{g}_{\text{comp.}}) \approx \hat{S}(V_{\text{th}})$. It is this approximation that is then used to build the system (11). However, it may introduce substantial residuals and compromise the reliability of the generation procedure.

C Iterative Compensation Algorithm

C.1 Iterative solution to the nonlinear compensation problem

While the linear compensation (equation 11) works well for simpler models like the DA neuron model, where sensitivities depend only on voltage, it becomes inaccurate in complex CBMs. In the STG model, intracellular calcium dynamics introduce nonlinear dependencies, and the sensitivity matrix depends on the compensated conductances.

To address this, we introduce the iterative compensation algorithm, which generalizes the linear scheme by applying multiple compensation steps iteratively. At each iteration, the sensitivity matrix is recalculated based on the current conductances:

$$A(\bar{g}^{(k)}_{\text{comp.}}) \cdot \bar{g}^{(k+1)}_{\text{comp.}} = b(\bar{g}^{(k)}_{\text{comp.}}), \quad k = 0, \ldots, K+1 \quad . \tag{12}$$

Starting from an initial guess $\bar{g}^{(0)}_{\text{comp.}}$, the process is repeated until the residual norm between the target and actual DIC values is sufficiently small. During the dataset generation and inference processes, we use $K = 5$ iterations as a default value, and the default value $\hat{g}_{\text{comp.}}$ from [24] as the initial guess.

C.2 Choice of a priori threshold voltage

The generation requires evaluating sensitivities at a fixed voltage V_{th}. However, the actual threshold voltage of a conductance vector $\bar{g}$ is unknown before knowing $\bar{g}$. To circumvent this, we approximate V_{th} by a constant value shared across all instances of a given CBM. This approximation is supported empirically by sampling 4000 conductance vectors from broad conductance distributions $\mathcal{D}_{\text{analysis}}$ and computing their theoretical V_{th} using equation (8). The resulting distribution is narrow and unimodal, allowing selection of the median (close to the mean) as a stable reference for generation. We use $-51\,\text{mV}$ and $-55.5\,\text{mV}$ for the STG and the DA, respectively. The estimates are robust to the set size (we checked 8,000-instance populations and found close values). Fig 7 shows the histogram obtained for the 8000-instance population. We can see that $V = -51$ is close to both the median (yellow line) and the mean (pink dot), and that most of the threshold values are gathered around the estimate.

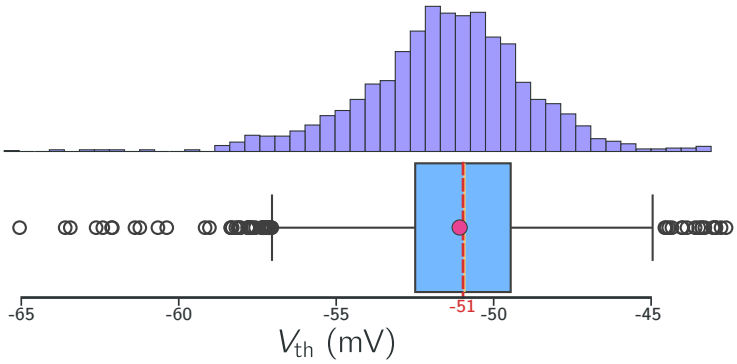

Fig 7: Histogram of threshold voltage estimates (V_{th}) for 8000 STG model instances. The distribution remains nearly identical to that obtained with 4000 instances, indicating convergence of the estimation around $V_{\text{th}} \approx -51$ mV. The mean (pink dot) and median (yellow line) are close to the estimated value.

D Synthetic dataset generation details

We simulate CBMs using Python 3.11 with the SciPy library [38]. The system of differential equations is solved using the `solve_ivp` function with the BDF method [39], well-suited for stiff systems such as models with multiple timescales. Although RK45 was tested, BDF consistently provided superior stability and accuracy. The maximum allowed time step is set to $0.05\,\text{ms}$, matching the spike time sampling period, with an adaptive step size during integration.

Simulations are run for a total duration of $5000\,\text{ms}$ for the STG model and $12\,000\,\text{ms}$ for the DA model. To avoid transient effects, the initial $3000\,\text{ms}$ of each simulation is discarded. Our implementation leverages multi-core CPU architectures to parallelize simulations, accelerating dataset generation.

From each simulated membrane potential trace $V(t)$, we emulate experimental recordings that only capture spike times:

$$V(t) \xrightarrow{\text{transformed into}} x = [t_1, t_2, \ldots, t_{N_{\text{spikes}}}], \quad \text{where } t_1 < t_2 < \cdots < t_{N_{\text{spikes}}} \quad .$$

The synthetic dataset is generated by uniformly sampling values in the g_{s} - g_{u} space to ensure broad coverage (Fig 8). For each sampled DICs vector, we generate 16 instances with distinct $\bar{g}$ using our iterative compensation algorithm. Bounds for DICs sampling are empirically derived from extensive model sampling of $\mathcal{D}_{\text{analysis}}$, and are $[-20; 20] \times [0; 20]$ for the STG and $[-10; 15] \times [0; 20]$ for the DA [24].

Dataset sizes are as follows: for the STG model, 1,000,000 instances are generated for training, 200,000 for validation, and 200,000 for testing. Validation sets are used for hyperparameter tuning, while the test set is reserved for final performance evaluation. Instances sharing the same DICs vector are kept within the same dataset partition to prevent data leakage. For the DA model, a reduced dataset (about 40% of the STG dataset size) is used, this fraction being determined through an ablation study on the STG data size. Transfer learning techniques employing LoRA adapters facilitate model adaptation with this smaller dataset.

Overall, the datasets consist of pairs $(\text{spike times}, \text{target DICs}) = (x, g^*)$, with underlying maximal conductances not used directly by the deep learning pipeline, which focuses on predicting DICs from spike times.

Instances are classified as *silent* if there are less than 3 spikes detected in the simulation; they are classified as *bursting* if the coefficient of variation of the inter-spike intervals (ISIs) is greater than 0.15, and as *spiking* otherwise.

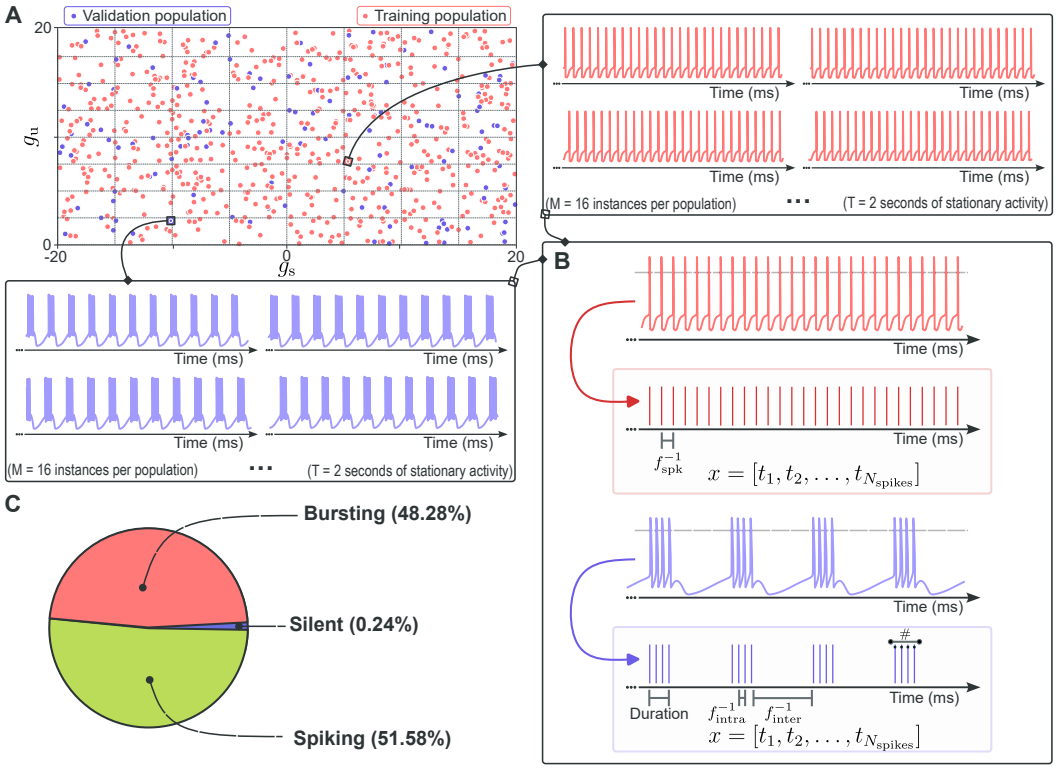

Fig 8: The synthetic dataset generation process from samling in the DICs space. (A) Subset of the sampled DIC space used to generate degenerate CBM populations. Each dot corresponds to one population (16 instances), belonging to the training (red) or validation (purple) set. Sampling is uniform and bounded to ensure broad coverage. **(B)** Schematic of the dataset generation pipeline: each population is simulated, spike times are extracted, and descriptors are computed. Each instance is then classified as spiking, bursting, or silent. **(C)** Class distribution across the full dataset: 51.58% spiking, 48.28% bursting, and 0.24% silent.

E Deep learning architecture and training details

E.1 Problem formulation as supervised learning

We frame the prediction of DIC values from spike trains as a supervised learning task. Specifically, we aim to learn a parametric function f_θ such that:

$$f_\theta : \mathbb{R}_+^{1\times *} \to \mathbb{R}^2, \quad x \mapsto \hat{g}^* \approx g^* \quad , \tag{13}$$

where the input $x = [t_1, t_2, \ldots, t_{N_{\text{spikes}}}]$ is a variable-length sequence of spike times, and the output $\hat{g}^* = (\hat{g}_s^*, \hat{g}_u^*)$ is an estimate of the true target values g^*. The notation $1 \times *$ emphasizes the arbitrary length of the input sequences of a single measure per element. This regression task poses two main challenges: (1) the input sequences are of variable-length, while deep neural networks typically require fixed-size inputs; and (2) learning meaningful latent representations from raw spike data, without relying on manually engineered summary statistics, is inherently difficult.

To address these challenges, we design an encoder-decoder architecture that transforms raw spike trains into fixed-dimensional latent representations, from which the DIC values are regressed.

E.2 Network architecture

Our architecture is represented in Fig 9, and comprises an attention-based encoder [40] and a multi-headed decoder. The encoder (Fig 9B) maps the variable-length input sequence to a fixed-size latent vector $z_{\text{latent}} \in \mathbb{R}^{d_{\text{latent}}}$, addressing the first challenge. The decoder (Fig 9C) then processes this latent

vector via multiple parallel heads to facilitate multi-task learning, thereby improving the structure and expressivity of the latent space.

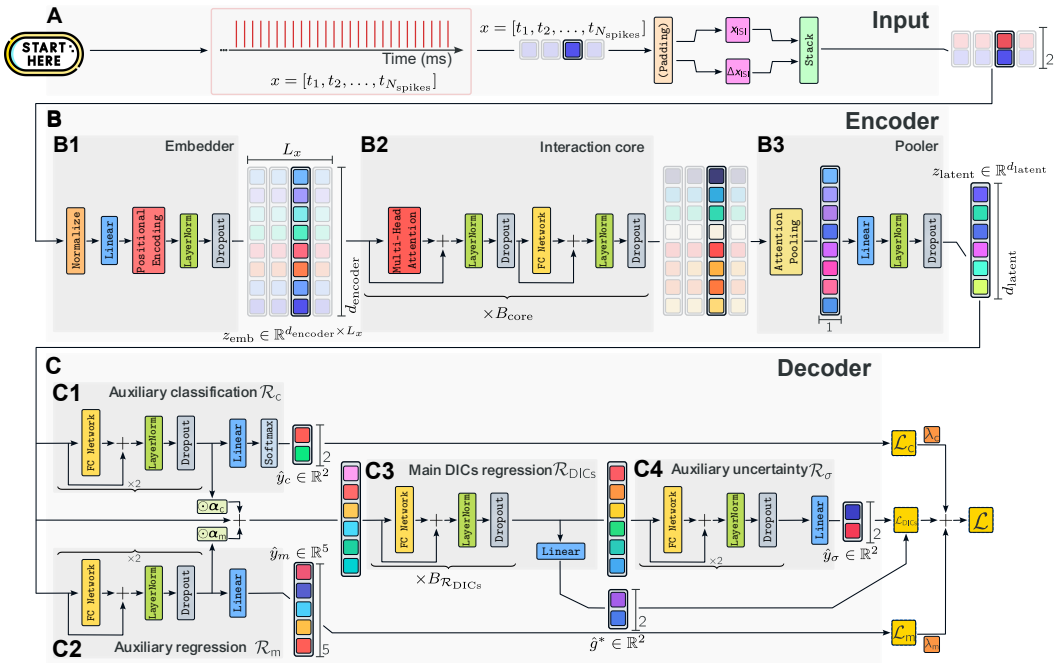

Fig 9: The deep learning architecture. **(A)** The input to the model consists of spike time sequences, from which ISIs and delta ISIs are extracted. These features are then stacked and fed into the encoder. **(B)** The encoder processes the input through three main components: the embedder, the interaction core, and the pooler. The embedder transforms the input sequence into a normalized, higher-dimensional representation. The interaction core processes this representation using multi-head attention mechanisms and fully-connected networks. The pooler aggregates the variable-length representation into a fixed-size latent representation. **(C)** The decoder transforms the fixed-size latent representation into the DICs space. It includes auxiliary tasks for classifying neuronal activity and regressing electrical activity metrics, as well as predicting uncertainty measures.

During training, the decoder includes four distinct output heads operating on the shared latent representation:

- **Main regression head** $\mathcal{R}_{\text{DICs}}$ (Fig 9C3) : $\mathbb{R}^{d_{\text{latent}}} \to \mathbb{R}^2$ predicts the DIC values.
- **Classification head** $\mathcal{R}_{\text{c}}$ (Fig 9C1) : $\mathbb{R}^{d_{\text{latent}}} \to \Delta^1$ classifies the input sequence as spiking or bursting based on inter-spike interval (ISI) variability.
- **Activity metrics head** $\mathcal{R}_{\text{m}}$ (Fig 9C2) : $\mathbb{R}^{d_{\text{latent}}} \to \mathbb{R}^5$ regresses five firing metrics. For spiking sequences, this includes mean firing rate. For bursting sequences, it includes mean intra-burst frequency, inter-burst frequency, burst duration, and number of spikes per burst.
- **Uncertainty head** $\mathcal{R}_{\sigma}$ (Fig 9C4) : $\mathbb{R}^{d_{\text{latent}}} \to \mathbb{R}^2$ estimates $\log \sigma^2_{g^*_{\text{s}}}$ and $\log \sigma^2_{g^*_{\text{u}}}$, which are used for heteroscedastic loss weighting.

At inference time, only the output from $\mathcal{R}_{\text{DICs}}$ is used downstream in the pipeline as the target values for the generation process. The auxiliary heads are used during training to regularize the encoder.

E.3 Encoder design

The encoder consists of three components: the embedder (Fig 9B1), the interaction core (Fig 9B2), and the pooler (Fig 9B3).

Given a spike train x, we first compute the sequence of ISIs, denoted as $x_{\text{ISI}} = \Delta x$. To enhance burst detection, this sequence is augmented with second-order differences Δx_{ISI}. The resulting

two-dimensional feature matrix is:

$$x_{\text{features}} = \begin{bmatrix} \log(1 + x_{\text{ISI},1}) & \log(1 + x_{\text{ISI},2}) & \dots \\ \Delta x_{\text{ISI},1} & \Delta x_{\text{ISI},2} & \dots \end{bmatrix} \in \mathbb{R}_+^{2\times *} \quad . \tag{14}$$

The logarithmic transformation $\log(1 + x_{\text{ISI}})$ stabilizes the ISI distribution.

The embedder converts this into a structured feature representation suitable for downstream processing. The feature matrix is standardized using statistics computed from the training set:

$$x_{\text{norm}} = \frac{x_{\text{features}} - \mu_{\text{train}}}{\sigma_{\text{train}}} \in \mathbb{R}^{2\times *} \quad . \tag{15}$$

The normalized sequence is projected into a higher-dimensional space $\mathbb{R}^{d_{\text{encoder}}\times *}$ and enriched with sinusoidal positional encoding $P_{\sin}$[40]:

$$z_{\text{emb}} = W x_{\text{norm}} + P_{\sin} \in \mathbb{R}^{d_{\text{encoder}}\times *} \quad . \tag{16}$$

Layer normalization [41] and dropout [42] are applied to the embeddings before they are processed by the interaction core.

The interaction core consists of B_{core} stacked transformer blocks. Each block performs multi-head self-attention followed by a position-wise fully connected (FC) network. A fully connected network is defined by two hidden linear layers, with a GELU activation function [43] and a dropout layer in between. Residual connections, layer normalization, and dropout are applied to each sublayer to stabilize training and improve generalization. This structure captures both local and global dependencies in the spike train.

Finally, the output of the interaction core is aggregated using self-attention pooling, which reduces the representation from $\mathbb{R}^{d_{\text{encoder}}\times *}$ to $\mathbb{R}^{d_{\text{encoder}}}$. A linear projection maps this pooled representation into the latent space:

$$z_{\text{latent}} = W_{\text{pool}} z + b_{\text{pool}} \in \mathbb{R}^{d_{\text{latent}}} \quad . \tag{17}$$

E.4 Decoder design

Each decoder head consists of residual fully connected (FC) blocks, each followed by layer normalization and dropout. All auxiliary heads (Fig 9C1-2,4) use two such blocks, concluding with a final linear layer; a softmax activation is applied after the classification head.

The final latent outputs of the auxiliary heads $\mathcal{R}_{\text{c}}$ and $\mathcal{R}_{\text{m}}$ are integrated with the encoder output via learnable element-wise mixing:

$$x_{\mathcal{R}_{\text{DICs}}} = z_{\text{latent}} + \alpha_{\text{m}} \odot z_{\text{m}} + \alpha_{\text{c}} \odot z_{\text{c}} \quad , \tag{18}$$

where $\alpha_{\text{m}}, \alpha_{\text{c}} \in \mathbb{R}^{d_{\text{latent}}}$ are learnable parameters and $z_{\text{m}}, z_{\text{c}}$ are the auxiliary latents.

The main DIC regression head $\mathcal{R}_{\text{DICs}}$ is made of $B_{\mathcal{R}_{\text{DICs}}}$ blocks. The final latent output serves as input to the uncertainty head $\mathcal{R}_{\sigma}$.

E.5 Training procedure and loss functions

The total loss is a weighted sum of the primary and auxiliary objectives:

$$\mathcal{L} = \mathcal{L}_{\text{DICs}} + \lambda_{\text{m}} \mathcal{L}_{\text{m}} + \lambda_{\text{c}} \mathcal{L}_{\text{c}} \quad , \tag{19}$$

where λ_{m} and λ_{c} are hyperparameters that control the importance of auxiliary tasks.

The primary loss $\mathcal{L}_{\text{DICs}}$ is a heteroscedastic Huber loss:

$$\begin{aligned} \mathcal{L}_{\text{DICs}} = \frac{1}{N} \sum_{i=1}^{N} \Big[& \frac{1}{\sigma^2_{g^*_{\text{s}},i}} \cdot \text{Huber}(g^*_{\text{s},i} - \hat{g}^*_{\text{s},i}) + \log(\sigma^2_{g^*_{\text{s}},i}) \\ & + \frac{1}{\sigma^2_{g^*_{\text{u}},i}} \cdot \text{Huber}(g^*_{\text{u},i} - \hat{g}^*_{\text{u},i}) + \log(\sigma^2_{g^*_{\text{u}},i}) \Big] \quad , \end{aligned} \tag{20}$$

where the Huber loss [44] is defined as:

$$\text{Huber}(a) = \begin{cases} \frac{1}{2}a^2 & \text{if } |a| \leq \delta, \\ \delta(|a| - \frac{1}{2}\delta) & \text{otherwise,} \end{cases} \tag{21}$$

with $\delta = 1$. This loss is robust to outliers and incorporates predictive uncertainty that smooth learning especially for spiking predictions where degeneracy leads to large errors.

The auxiliary loss for activity metrics is a masked mean squared error:

$$\mathcal{L}_{\text{m}} = \frac{1}{N} \sum_{i=1}^{N} \|(y_{\text{m},i} - \hat{y}_{\text{m},i}) \odot m_i\|^2 \quad , \tag{22}$$

where the binary mask m_i selects relevant metrics based on the activity type:

$$m_i = \begin{cases} [1, 0, 0, 0, 0] & \text{if } y_{\text{c},i} = 1 \text{ (spiking)}, \\ [0, 1, 1, 1, 1] & \text{if } y_{\text{c},i} = 2 \text{ (bursting)}. \end{cases} \tag{23}$$

The classification loss is standard cross-entropy:

$$\mathcal{L}_{\text{c}} = -\frac{1}{N} \sum_{i=1}^{N} [\mathbb{I}(y_{\text{c},i} = 1) \log(\hat{y}_{\text{c},i,1}) + \mathbb{I}(y_{\text{c},i} = 2) \log(\hat{y}_{\text{c},i,2})] \quad . \tag{24}$$

A balanced version is used during training on the DA model due to dataset imbalance.

To improve generalization, we apply three forms of data augmentation during training: (1) random cropping of spike trains to a window $D \sim \mathcal{U}[N_{\text{spikes}}/2, N_{\text{spikes}}]$, (2) Gaussian noise $\epsilon_i \sim \mathcal{N}(0, (2\,\text{ms})^2)$ added to spike times, and (3) 5% spike dropout. These augmentations simulate experimental variability and are applied independently for each training sample at each update.

Hyperparameters are optimized via extended random search [45] over 100 configurations, with performance evaluated using the primary DIC loss $\mathcal{L}_{\text{DICs}}$ on the validation set. Optimization uses AdamW [46] and a cosine annealing learning rate schedule with warm restarts.

The deep learning implementation was done using PyTorch. The final architecture was trained for 200 epochs, and we saved the parameter values that achieved the best performance on the validation set. Validation was performed each quarter of an epoch.

The optimizer was set up with a learning rate η, which controls the size of the weight updates at each iteration. The parameters β_1 and β_2 controlled the decay rates for the first and second moment estimates used in the calculations of the optimizer. We used $\beta_1 = 0.9$ and $\beta_2 = 0.98$. We used $\lambda_{\text{weight decay}} = 0.04$ for L2 regularization, helping to prevent overfitting by penalizing large weight values.

The scheduler adapted the learning rate η_t at epoch t by:

$$\eta_t = \eta_{\text{min}} + \frac{1}{2}(\eta - \eta_{\text{min}}) \left(1 + \cos\left(\frac{T_{\text{cur}}}{T_i}\pi\right)\right) \quad ,$$

where:

- η is the initial learning rate. This value was tuned through hyperparameter tuning.
- η_{min} is the minimum learning rate. We used $\eta_{\text{min}} = \frac{\eta}{10}$.
- T_{cur} is the number of epochs since the last restart.
- T_i is the number of epochs between two restarts. We used $T_i = 10$ epochs.

Table 3 summarizes the explored hyperparameters and their respective distributions. In this table, `relu` corresponds to the classical *rectified linear unit* ($\text{ReLU}(x) = \max(0, x)$), `silu` to the *sigmoid linear unit* [43, 47], `tanh` to the hyperbolic tangent, and `gelu` to the *gaussian error linear units* [43]. We trained the model for $N_{\text{epoch}} = 50$ epochs – that is, the number of times the model was exposed to the training set $\mathcal{T}_{\text{train}}$. We evaluated the model on the validation set $\mathcal{T}_{\text{val}}$ using the DICs regression loss, $\mathcal{L}_{\text{DICs}}$, every quarter of an epoch. As training was quite time-consuming, only half of the training set was used for this hyperparameter tuning step; the size of the validation set remained unchanged. For each set of hyperparameters, the best performance obtained on the validation set was retained for comparison. We report in Fig 10 the distribution of the best validation results we obtained. The final architecture and training hyperparameters were chosen to be close to the best performance from the random search; those parameter values are reported in Table 4.

Table 3: **Hyperparameters and distributions explored in random search.** $\tilde{\lambda}_c$ and $\tilde{\lambda}_m$ are relative weights of the auxiliary losses to the primary DICs loss (e.g., $\tilde{\lambda}_c = 10$ makes the classification loss $10\times$ the primary loss on the first batch). "*Apply log transform to input?*" indicates whether a logarithmic transform is applied to ISIs.

Hyperparameter	**Type / Distribution**	**Values or Range**
Learning rate (η)	Log-uniform	$[10^{-5}, 5 \cdot 10^{-3}]$
Dropout (p_{dropout})	Uniform	$[0.0, 0.4]$
Latent space dimension (d_{latent})	Discrete	$\{16, 32, 64, 128\}$
Encoder space dimension (d_{encoder})	Discrete	$\{16, 32, 64, 128\}$
Number of heads (H)	Discrete	$\{2, 4, 8\}$
Number of encoder blocks (B_{core})	Discrete	$\{1, 2, 3, 4, 5, 6, 7, 8\}$
Number of decoder blocks ($B_{\mathcal{R}_{\text{DICs}}}$)	Discrete	$\{2, 4, 6, 8, 10\}$
Activation function ($\sigma(\cdot)$)	Categorical	`relu`, `gelu`, `silu`, `tanh`
Apply log transform to input?	Boolean	`true`, `false`
Regression weighting factor ($\tilde{\lambda}_m$)	Uniform	$[0.01, 10.0]$
Classification weighting factor ($\tilde{\lambda}_c$)	Uniform	$[0.01, 10.0]$
Batch size ($\lvert B \rvert$)	Discrete	$\{16, 32, 64, 128\}$

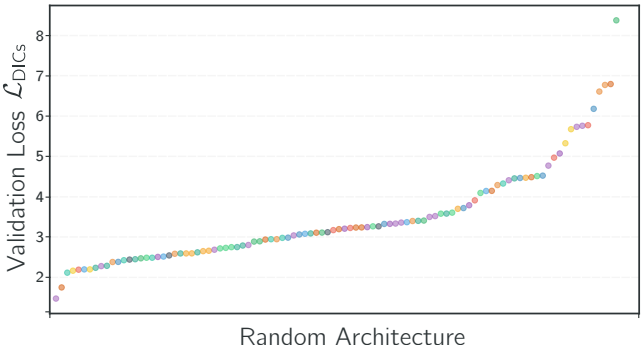

Fig 10: Performance of random architectures. Scatter plot of best validation loss $\mathcal{L}_{\text{DICs}}$ across randomly sampled architectures. Each point is a unique configuration, illustrating how design choices impact performance.

E.6 Evaluation metrics

Model evaluation is carried out at two levels: training/validation and final test performance. During training, we use the heteroscedastic loss $\mathcal{L}_{\text{DICs}}$ as the primary metric to assess whether the model is learning effectively. Hyperparameter optimization is performed based on the value of $\mathcal{L}_{\text{DICs}}$ on the validation set $\mathcal{T}_{\text{val}}$. For test-time evaluation on $\mathcal{T}_{\text{test}}$, we report the mean absolute error (MAE) between predicted and ground truth DIC values. We also evaluate the performance of the auxiliary tasks. The auxiliary regression head is evaluated using MAE. The auxiliary classification head is evaluated using balanced accuracy to account for potential class imbalance. These auxiliary metrics serve to verify that the network learns a meaningful and structured latent representation of the input spike trains.

We evaluated the backbone architecture, trained on the STG neuron dataset, using metrics spanning the primary DIC regression task as well as auxiliary regression and classification tasks. The model achieved high accuracy across these tasks, with 99.83% classification accuracy for firing classes and precise regression of neuronal activity features, which underscores its effectiveness in capturing complex neuronal dynamics.

The architecture accurately predicts key neuronal activity features, including mean spike frequency, intra- and inter-burst frequencies, burst duration, and number of spikes per burst. Mean absolute

Table 4: **Optimized STG neuron architecture.** Summary of the final hyperparameters and model size, optimized for performance and efficiency in real-time applications.

(a) **Final hyperparameters.** Values selected after tuning, giving best validation performance under the uncertainty-weighted DICs loss.

Hyperparameter	**Final value**
Learning rate (η)	2.10×10^{-5}
Dropout (p_{dropout})	0.034
Latent space dimension (d_{latent})	16
Encoder space dimension (d_{encoder})	64
Number of heads (H)	8
Number of encoder blocks (B_{core})	4
Number of decoder blocks ($B_{\mathcal{R}_{\text{DICs}}}$)	2
Activation function ($\sigma(\cdot)$)	`gelu`
Apply log transform to input?	`true`
Regression weighting factor ($\tilde{\lambda}_{\text{m}}$)	0.0919
Classification weighting factor ($\tilde{\lambda}_{\text{c}}$)	5.44
Batch size ($\lvert B \rvert$)	32

(b) **Model size.** The final architecture has 115,627 learnable parameters, enabling real-time use on standard hardware.

Number of Parameters	115,627

errors (MAE) for these auxiliary regressions are substantially lower than the intrinsic variability of the dataset (Table 5a), which confirms that the latent representation meaningfully captures essential firing features.

Table 5: **Performance summary of the backbone architecture on the STG neuron dataset.** The table reports prediction accuracy for auxiliary regression tasks and the primary DICs regression task.

(a) **Auxiliary regression tasks: architecture error vs. dataset variability.** Mean absolute error (MAE) values for auxiliary regression tasks demonstrate substantially lower error than the inherent dataset variability (standard deviation). Descriptors include mean spike frequency (f_{spk}), intra- and inter-burst frequencies (f_{intra}, f_{inter}), burst duration, and mean number of spikes per burst ($\#$).

Descriptor	f_{spk}	f_{intra}	f_{inter}	Duration	$\#$
MAE	0.11 Hz	2.26 Hz	0.15 Hz	0.81 ms	0.16
Dataset std	3.02 Hz	47.91 Hz	1.11 Hz	7.85 ms	2.25

(b) **Dynamic Input Conductances prediction performance.** The first subtable shows the overall MAE for slow (g_{s}) and ultra-slow (g_{u}) DIC components. The second subtable separates MAE values by firing regime reflecting differences in prediction accuracy across distinct neuronal activity modes.

DIC	g_{s}	g_{u}
MAE	2.84	1.75

DIC	g_{s} (spiking)	g_{s} (bursting)	g_{u} (spiking)	g_{u} (bursting)
MAE	4.65	0.80	2.29	1.13

We next provide quantitative comparisons of input and generated firing patterns (Fig 11). We sampled 500 random points in the $(g_{\text{s}}, g_{\text{u}})$ plane and, for each point, generated a population of 16 neurons using the iterative compensation method. These neurons share identical DIC values at threshold and

are therefore degenerate by construction, but still display variability in their firing patterns due to the approximation done through DICs constrained only at threshold, and the iterative procedure. Such variability remains compatible with degeneracy and reflects biological heterogeneity.

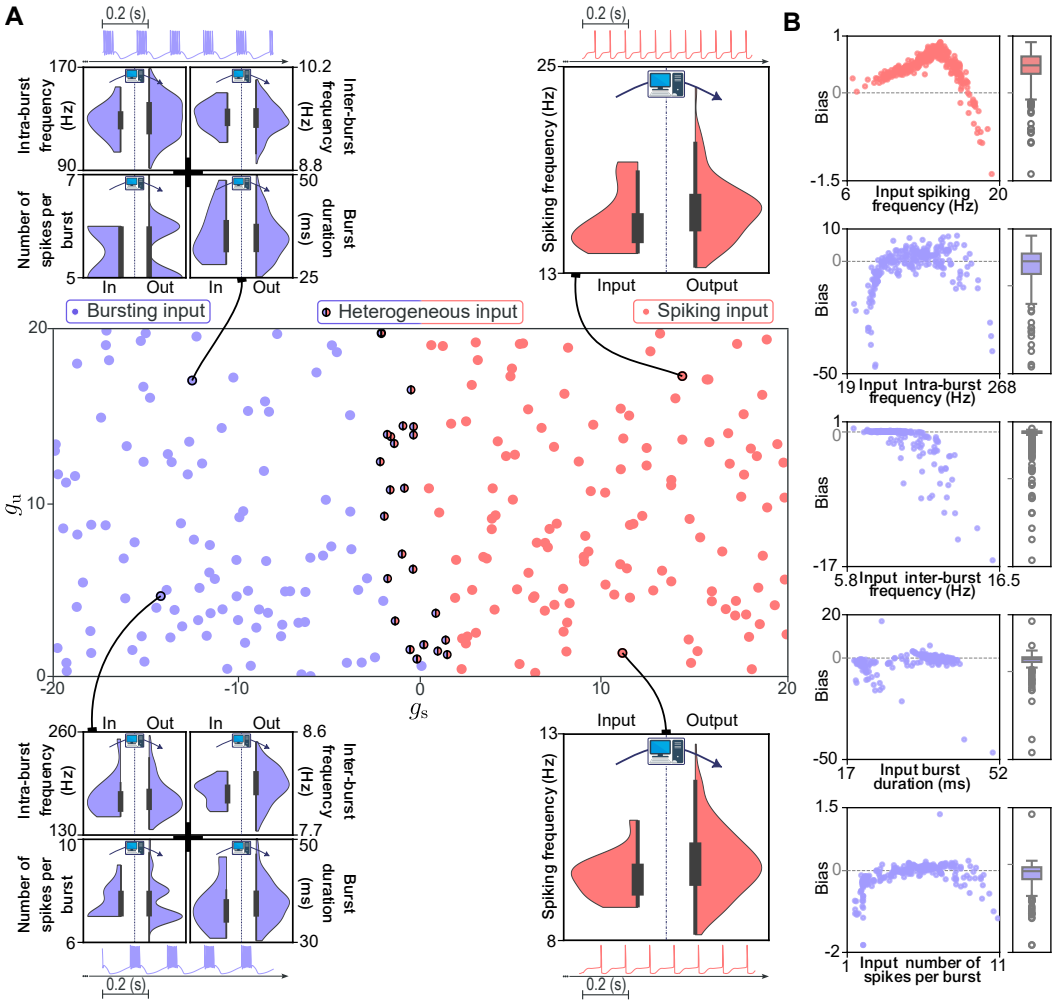

Fig 11: Quantitative comparison of input and generated populations. (A) Sampling of 500 points in (g_s, g_u) space (40% shown for clarity). Purple and red points indicate representative examples; two-colored points denote mixed populations near the spiking–bursting transition ($g_s \approx 0$). Violin plots compare distributions of selected activity metrics between the input population (left) and the corresponding generated population (right) for each example point. **(B)** Bias analysis across all 500 populations for five activity metrics: spiking frequency, intra-burst frequency, inter-burst frequency, burst duration, and number of spikes per burst. Scatter plots show biases for individual populations; box plots summarize the distribution of biases. Generated populations exhibit minimal bias and closely match the input metrics, with slight increases in spread and rare outliers in extreme conditions.

For each of the 500 input populations, we extracted the spike times of the 16 neurons and passed them through the full pipeline, yielding 16 predicted DICs per point. Each prediction was then used to generate a new population of 16 neurons via iterative compensation, which results in 16×16 neurons per point in the DIC space. This enabled direct comparison of input and output distributions for key activity descriptors: spiking frequency, intra-burst frequency, inter-burst frequency, burst duration, and number of spikes per burst.

Fig 11A shows the sampled DIC space, with representative example points highlighted. Points near $g_s \approx 0$ correspond to mixed populations containing both spiking and bursting neurons, reflecting the transition between regimes [24]. For selected examples, violin plots compare input distributions (left) with reconstructed ones (right). Fig 11B summarizes reconstruction accuracy across all sampled

points, reporting the bias (mean difference between generated and input values) for each metric as both scatter plots and box plots.

The generated distributions closely match the inputs, with only slightly increased spread. This is expected since predictions are made from individual spike trains before being aggregated into populations. Spiking metrics are reconstructed with high accuracy: biases are centered near $0.5\,\mathrm{Hz}$, with occasional outliers (about $1\,\mathrm{Hz}$). Bursting metrics are equally well preserved, with biases well below the scale of the features themselves and centered around zero; in particular, the mean number of spikes per burst is reconstructed with high precision.

F Transfer details to the DA model

We perform a transfer from the STG neuron model to the DA neuron model. The same data generation pipeline and neural network architecture are reused, with the exception of the introduction of parameter-efficient fine-tuning using Low-Rank Adaptation (LoRA) [34].

We introduce LoRA adapters in the linear layers of the network, while attention layers remain unmodified. The majority of parameters from the original STG-trained pipeline are frozen during training, and only the newly introduced LoRA parameters are updated. The input normalization layer is recalculated based on the DA training set. Fig 12 illustrates the modified architecture, indicating which components are frozen and which are adapted.

The transfer introduces one additional hyperparameter, r, which controls the rank of the LoRA adapter matrices. To select an appropriate configuration, we perform a grid search over $r \in \{2, 4, 8, 16, 32, 48, 64\}$. Each configuration is trained on the DA dataset using the same optimization strategy and training protocol as for the STG model. The best results are observed for $r = 32$.

The dataset for the DA model is generated using the same procedure as for the STG model, including rejections of silent populations, and extraction of the corresponding DIC values.

During dataset generation, an initial scan of the sampled populations reveals a substantial class imbalance: a large proportion of silent populations, and a relatively small number of bursting cases compared to spiking ones. Silent populations are discarded. The number of sampled configurations is set to yield approximately $25{,}000$ active (non-silent) populations in total.

G Generation of Poisson and bursting spike trains

To assess the robustness of our pipeline to biologically plausible timing variability, we generate synthetic input spike trains using two stochastic models: a homogeneous Poisson process and a burst-extended variant. Table 6 reports the range used when uniformly sampling the process parameters.

Table 6: **Parameter ranges used to generate stochastic spike trains.** Homogeneous Poisson processes were sampled with rate λ, while burst-extended Poisson processes used separate rates for inter-burst (λ_e) and intra-burst (λ_i) spikes. The number of spikes per burst was uniformly sampled within the indicated range. All parameters were drawn from uniform distributions, either continuous $\mathcal{U}(p_{\min}, p_{\max})$ or discrete $\mathcal{U}\{p_{\min}, p_{\max}\}$.

Parameters $p \sim \mathcal{U}(p_{\min}; p_{\max})$	$p_{\min}$	$p_{\max}$
λ	5	20
λ_i	75	250
λ_e	4	9
Parameters $p \sim \mathcal{U}\{p_{\min}; p_{\max}\}$	**$p_{\min}$**	**$p_{\max}$**
Number of spikes per burst #	2	7

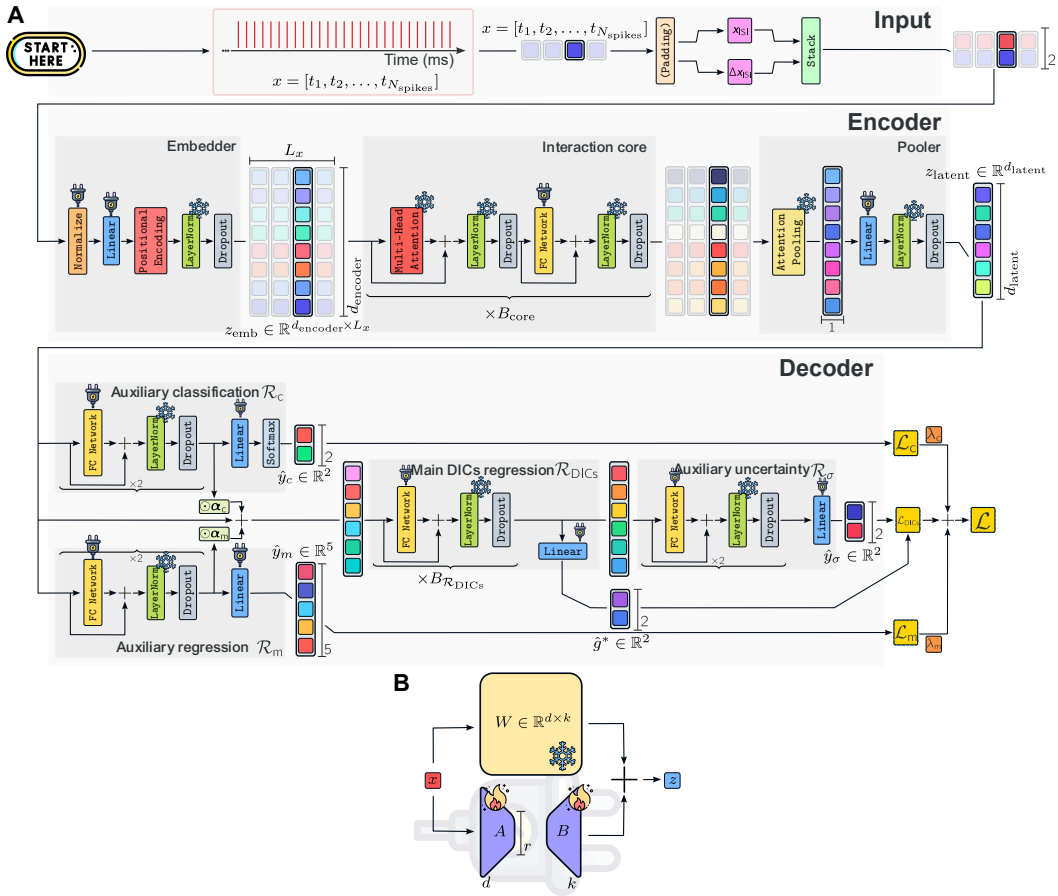

Fig 12: The deep learning architecture with LoRA to transfer from the STG to the DA. (A) The plug icon indicates the insertion of a LoRA adapter, where the original layer parameters are frozen and a new linear projection is added in an additive manner with new learnable parameters (panel (B)). The freeze icon signifies that the parameters of the corresponding layer are not retrained, preserving its original functionality and weights. The only exception is the normalization block (orange) at pipeline input. This one is not adapted by a LoRA, but directly replaced by the new normalization statistics calculated on the DA train set. **(B)** The LoRA adapter introduces a small number of additional parameters through low-rank matrices $A \in \mathbb{R}^{d\times k}$ and $A \in \mathbb{R}^{r\times k}$. These matrices are used to modify the matrix multiplication in fully connected by adding a learnable component to the frozen pre-trained weights $W \in \mathbb{R}^{d\times k}$.

G.1 Homogeneous Poisson process

Spike times are generated from a Poisson process with constant rate λ. Inter-spike intervals (ISIs) are sampled from an exponential distribution:

$$f_{\text{ISI}}(\tau) = \lambda e^{-\lambda\tau}, \quad \tau \geq 0 \quad ,$$

resulting in irregular and memoryless spike trains.

G.2 Poisson burst generator

To simulate bursting behavior, we use a nested Poisson process with two timescales:

1. **Burst onsets** follow a Poisson process with rate λ_{e}, with inter-burst intervals distributed as:
$$f_{\text{inter}}(\Delta B) = \lambda_{\text{e}} e^{-\lambda_{\text{e}}\Delta B} \quad .$$
2. **Intra-burst spikes** are generated from an independent exponential distribution with rate λ_{i}:
$$f_{\text{intra}}(\delta) = \lambda_{\text{i}} e^{-\lambda_{\text{i}}\delta} \quad .$$

Each burst contains a fixed number of spikes, making it the only deterministic component of the process.

$$f(V, A, B, C, D) = A + \frac{B}{1 + \exp\left(\frac{V+D}{C}\right)} \tag{25}$$

Table 7: **Gating functions and kinetics for the STG model.** Steady-state activation/inactivation functions, time constants, and exponents for each ionic current in the STG model. All f functions refer to the generalized sigmoid function (Eq. 25).

Current I_i	p_i	q_i	$m_{i,\infty}(V)$ or $m_{i,\infty}(V, \mathrm{Ca})$	$h_{i,\infty}(V)$	$\tau_{m_i}(V)$	$\tau_{h_i}(V)$
I_{Na}	3	1	$f(V,0,1,-5.29,25.5)$	$f(V,0,1,5.18,48.9)$	$f(V,1.32,-1.26,-25,120)$	$f(V,0,0.67,-10,62.9) \cdot f(V,1.5,1,3.6,34.9)$
I_{Kd}	4	0	$f(V,0,1,-11.8,12.3)$	—	$f(V,7.2,-6.4,-19.2,28.3)$	—
I_{CaT}	3	1	$f(V,0,1,-7.2,27.1)$	$f(V,0,1,5.5,32.1)$	$f(V,21.7,-21.3,-20.5,68.1)$	$f(V,105,-89.8,-16.9,55)$
I_{CaS}	3	1	$f(V,0,1,-8.1,33)$	$f(V,0,1,6.2,60)$	$1.4+\frac{7}{\exp\left(\frac{V+27}{10}\right)+\exp\left(\frac{V+70}{-13}\right)}$	$60+\frac{150}{\exp\left(\frac{V+55}{9}\right)+\exp\left(\frac{V+65}{-16}\right)}$
I_{KCa}	4	0	$\frac{\mathrm{Ca}}{\mathrm{Ca}+3} \cdot f(V,0,1,-12.6,28.3)$	—	$f(V,90.3,-75.1,-22.7,46)$	—
I_{A}	3	1	$f(V,0,1,-8.7,27.2)$	$f(V,0,1,4.9,56.9)$	$f(V,11.6,-10.4,-15.2,32.9)$	$f(V,38.6,-29.2,-26.5,38.9)$
I_{H}	1	0	$f(V,0,1,6,70)$	—	$f(V,272,1499,-8.73,42.2)$	—
I_{leak}	0	0	—	—	—	—

Table 8: **Gating functions and kinetics for the DA model.** All f functions refer to the generalized sigmoid function (Eq 25). Additionally, there is a current I_{ERG} described by the ERG channel, whose rate functions are defined as exponentials of the membrane potential V. The activation and inactivation parameters are as follows: $a_0(V) = 0.0036\exp(0.0759V)$, $b_0(V) = 1.2523 \times 10^{-5}\exp(-0.0671V)$, $a_i(V) = 0.1\exp(0.1189V)$, and $b_i(V) = 0.003\exp(-0.0733V)$.

Current	p_i	q_i	$m_{i,\infty}(V)$ or $m_{i,\infty}(V, \mathrm{Mg})$	$h_{i,\infty}(V)$	$\tau_{m_i}(V)$	$\tau_{h_i}(V)$
I_{Na}	3	1	$f(V,0,1,-9.7264,30.0907)$	$f(V,0,1,10.7665,54.0289)$	$0.01+\frac{1.0}{(-\frac{15.6504+0.4043V}{\exp(-19.565-0.5052V)-1.0})+3.0212\exp(-0.007463V)}$	$0.4+\frac{1.0}{(0.00050754\exp(-0.063213V))+9.7529\exp(0.13442V)}$
I_{Kd}	3	0	$f(V,0,1,-12,25)$	—	$f(V,20,-18,-10,38)$	—
I_{CaL}	2	0	$f(V,0,1,-2,50)$	—	$f(V,30,-28,-3,45)$	—
I_{CaN}	1	0	$f(V,0,1,-7,30)$	—	$f(V,30,-25,-6,55)$	—
I_{NMDA}	1	0	$\frac{1}{1+\frac{\mathrm{Mg}\cdot\exp(-0.08V)}{10}}$	—	—	—
I_{leak}	0	0	—	—	—	—