S1 Appendix

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

We present all the elements needed in addition to the main text to reproduce the results reported in this paper. We also provide some additional experiments that we judge interesting but that are not central to get the main message of the paper.

A The stomatogastric ganglion and the dopaminergic neuron models

This section details the equations of the conductance-based models (CBMs) used in this work. The stomatogastric ganglion (STG) model was adapted from [S1], and the dopaminergic (DA) model was adapted from [S2].

Both models follow a similar structure where the membrane potential V dynamic is described by an ODE dependent on ionic and leak currents:

$$C\frac{dV}{dt} + g_{\text{leak}}(V - E_{\text{leak}}) = -\sum_{i\in\mathcal{I}} \bar{g}_i m_i^{p_i}(V,t) h_i^{q_i}(V,t)(V - E_i) + I_{\text{ext}} \quad . \tag{S1}$$

Here, C is the membrane capacitance. The set $\mathcal{I}$ contains all ionic conductances considered for the model and detailed for both models below. Each ionic current is characterized by its maximal conductance $\bar{g}_i$ and by gating variables, m_i (activation variable) and h_i (inactivation variable), each raised to integer powers p_i and q_i, respectively. E_i is the Nernst reversal potential of the associated ion. The leak current is modeled as $I_{\text{leak}} = g_{\text{leak}}(V - E_{\text{leak}})$. External input current I_{ext} was set to zero in all simulations ($I_{\text{ext}} = 0$). We denote $\bar{g} = [\bar{g}_1;\ \bar{g}_2;\ \ldots;\ \bar{g}_{|\mathcal{I}|}, g_{\text{leak}}] \in \mathbb{R}^{N_{\text{model}}}$ as the vector of maximal conductances (extended by the leak one) defining different instances of a model. The maximal conductances were the only parameters that we varied in this work.

Gating variables $X \in \{m_i, h_i\}$ are dimensionless quantities constrained between 0 and 1, and representing the percentage of ion channels being activated (m_i) and not inactivated (h_i). They follow first-order voltage-dependent dynamics of the form:

$$\tau_X(V)\frac{dX}{dt} = X_\infty(V) - X \quad , \tag{S2}$$

where $\tau_X(V)$ is the voltage-dependent time constant, and $X_\infty(V)$ is the steady-state activation or inactivation value.

A.1 The stomatogastric ganglion neuron model

The STG model [S1] includes eight ionic currents: the fast transient sodium current (Na), the delayed rectifier potassium current (Kd), the calcium-activated potassium current (KCa), the A-type potassium current (A), the slow calcium current (CaS), the transient calcium current (CaT), the hyperpolarization-activated current (H), and the leak current. The membrane capacitance was set to $1\,\mu\text{F}\,\text{cm}^{-2}$. 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 S1 provides the values of the fixed model parameters.

Simulations were performed as described in the Materials and Methods section of the main text. The transients were discarded. As initial conditions used, we used $V_0 = -70\,\text{mV}$ and $\text{Ca}_0 = 0.5\,\mu\text{M}$,

Table S1. 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}
$-50\,\text{mV}$	$50\,\text{mV}$	$-80\,\text{mV}$	$-20\,\text{mV}$	$80\,\text{mV}$	$20\,\text{ms}$	$0.94\,\text{mM}\,\text{nF}\,\text{nA}^{-1}$	$0.05\,\mu\text{M}$

with initial gating variables evaluated at steady-state $X_0 = X_\infty(V_0; \text{Ca}_0)$. Table S3 lists the mathematical expressions for the steady-state gating functions, the time constant functions, and the corresponding gating exponents.

A.2 The dopaminergic neuron model

The DA model [S2] includes seven ionic currents: the fast sodium current (Na), the delayed rectifier potassium current (Kd), the ERG potassium current (ERG), the L-type calcium current (CaL), the N-type calcium current (CaN), the NMDA receptor-mediated current (NMDA), and the leak current. The capacitance was set to $1\,\mu\text{F}\,\text{cm}^{-2}$. Fixed model parameters are provided in Table S2.

In this model, 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.

Specifically for the ERG channel, the gating variables o_{ERG} and i_{ERG} evolved according to the differential equations:

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

$$\frac{di_{\text{ERG}}}{dt} = a_i(V) \cdot o_{\text{ERG}} - b_i(V) \cdot i_{\text{ERG}} \quad .$$

The steady-state expressions for the ERG gating variables under constant voltage V were:

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

$$i_{\mathrm{ERG},\infty}(V) = \frac{a_0(V) \cdot a_i(V)}{a_0(V)(a_i(V) + b_i(V)) + b_0(V) \cdot b_i(V)} \quad .$$

The corresponding ERG current was given by:

$$I_{\mathrm{ERG}} = \bar{g}_{\mathrm{ERG}} \cdot o_{\mathrm{ERG}} \cdot (V - E_K) \quad .$$

Simulations were performed as described in the Materials and Methods section of the main text. The transients were discarded. As initial conditions, we used $V_0 = -90\,\mathrm{mV}$, with initial gating variables evaluated at steady-state $X_0 = X_\infty(V_0)$. Table S4 lists the mathematical expressions for the steady-state gating functions, the time constant functions, and the corresponding gating exponents.

Table S2. 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
$60\,\mathrm{mV}$	$-85\,\mathrm{mV}$	$60\,\mathrm{mV}$	$-50\,\mathrm{mV}$	$0\,\mathrm{mV}$	1.4

Table S3. 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. S3).

Current I_i	p_i	q_i	$m_{i,\infty}(V)$ or $m_{i,\infty}(V,\text{Ca})$	$h_{i,\infty}(V)$	$\tau_{m_i}(V)$	$\tau_{h_i}(V)$
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	$\frac{\text{Ca}}{\text{Ca}+3}\cdot f(V,0,1,-12.6,28.3)$	—	$f(V,90.3,-75.1,-22.7,46)$	—
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 S4. Gating functions and kinetics for the DA model. All f functions refer to the generalized sigmoid function (Eq S3). 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: $a_0(V) = 0.0036\exp(0.0759V)$, $b_0(V) = 1.2523\times10^{-5}\exp(-0.0671V)$, $a_i(V) = 0.1\exp(0.1189V)$, and $b_i(V) = 0.003\exp(-0.0733V)$.

Current	p_i	q_i	$m_{i,\infty}(V)$ or $m_{i,\infty}(V,\text{Mg})$	$h_{i,\infty}(V)$	$\tau_{m_i}(V)$	$\tau_{h_i}(V)$
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	$\frac{1}{1+\frac{\text{Mg}\cdot\exp(-0.08V)}{10}}$	—	—	—
I_{leak}	0	0	—	—	—	—

$$f(V,A,B,C,D) = A + \frac{B}{1+\exp\left(\frac{V+D}{C}\right)} \tag{S3}$$

B Dynamic input conductances (DICs)

This article uses the concept of Dynamic Input Conductances (DICs) [S3]. DICs consist of voltage-dependent conductances that separate according to timescales. In this work, we choose three timescales: fast, slow, and ultra-slow. These three DICs have been shown to be sufficient to qualitatively determine excitability in neuron models. Specifically, based on threshold values of the DICs, it becomes possible to predict the firing pattern of the neuron. The computation of the DICs in this article follows recent work on the subject [S4, 5].

We denote $g_{\mathrm{f}}(V)$, $g_{\mathrm{s}}(V)$, and $g_{\mathrm{u}}(V)$ the fast, slow, and ultra-slow DIC components, respectively. The total DIC is $g_{\mathrm{t}}(V) = g_{\mathrm{f}}(V) + g_{\mathrm{s}}(V) + g_{\mathrm{u}}(V)$. In this work, we focus on a particular value of voltage, the threshold voltage, denoted V_{th}. As an approximation, we consider the threshold voltage to be a fixed value shared by all instances across a given CBM. We made this approximation mainly because computing V_{th} requires knowing $\bar{g}$, which is unknown in advance.

B.1 DICs computation details

In CBMs, these timescale-specific conductances can be computed analytically using:

$$\begin{cases} g_{\mathrm{f}}(V) &= -\dfrac{\partial I_{\mathrm{f}}}{\partial V}(V)\dfrac{1}{g_{\mathrm{leak}}} = \left[-\dfrac{\partial \dot{V}}{\partial V} - \sum_i w_{\mathrm{fs},X_i}(V) \left(\dfrac{\dot{V}}{\partial X_i} \dfrac{\partial X_{i,\infty}}{\partial V} \right) \right]\Bigg|_V \dfrac{1}{g_{\mathrm{leak}}} \quad , \\ g_{\mathrm{s}}(V) &= -\dfrac{\partial I_{\mathrm{s}}}{\partial V}(V)\dfrac{1}{g_{\mathrm{leak}}} = \left[- \sum_i \left(w_{\mathrm{su},X_i}(V) - w_{\mathrm{fs},X_i}(V) \right) \left(\dfrac{\dot{V}}{\partial X_i} \dfrac{\partial X_{i,\infty}}{\partial V} \right) \right]\Bigg|_V \dfrac{1}{g_{\mathrm{leak}}} \quad , \\ g_{\mathrm{u}}(V) &= -\dfrac{\partial I_{\mathrm{u}}}{\partial V}(V)\dfrac{1}{g_{\mathrm{leak}}} = \left[- \sum_i \left(1 - w_{\mathrm{su},X_i}(V) \right) \left(\dfrac{\dot{V}}{\partial X_i} \dfrac{\partial X_{i,\infty}}{\partial V} \right) \right]\Bigg|_V \dfrac{1}{g_{\mathrm{leak}}} \quad , \end{cases} \tag{S4}$$

where:

- the terms X_i correspond to gating variables (m_i or h_i) that control the activation and inactivation of ion channels;
- the terms $w_{\mathrm{fs},X_i}(V)$ and $w_{\mathrm{su},X_i}(V)$ are voltage-dependent weighting factors.

These weights determine how each variable contributes to different timescales. Their values are computed based on a logarithmic scaling between 0 and 1, using reference timescales:

$$w_{\mathrm{fs},X_i}(V) = \begin{cases} 1 & , \quad \tau_{X_i}(V) \leq \tau_{\mathrm{f}}(V) \quad , \\ \dfrac{\log(\tau_{\mathrm{s}}(V)) - \log(\tau_{X_i}(V))}{\log(\tau_{\mathrm{s}}(V)) - \log(\tau_{\mathrm{f}}(V))} & , \quad \tau_{\mathrm{f}}(V) < \tau_{X_i}(V) \leq \tau_{\mathrm{s}}(V) \quad , \\ 0 & , \quad \tau_{X_i}(V) > \tau_{\mathrm{s}}(V) \quad , \end{cases}$$

$$w_{\mathrm{su},X_i}(V) = \begin{cases} 1 & , \quad \tau_{X_i}(V) \leq \tau_{\mathrm{s}}(V) \quad , \\ \dfrac{\log(\tau_{\mathrm{u}}(V)) - \log(\tau_{X_i}(V))}{\log(\tau_{\mathrm{u}}(V)) - \log(\tau_{\mathrm{s}}(V))} & , \quad \tau_{\mathrm{s}}(V) < \tau_{X_i}(V) \leq \tau_{\mathrm{u}}(V) \quad , \\ 0 & , \quad \tau_{X_i}(V) > \tau_{\mathrm{u}}(V) \quad . \end{cases} \tag{S5}$$

The reference timescales τ_{f}, τ_{s}, and τ_{u} are chosen according to the characteristic timescales of bursting and spiking dynamics [S3]. τ_{f} corresponds to the activation time constant of the fastest depolarizing current, while τ_{s} represents the activation time constant of the fastest repolarizing current. Finally, τ_{u} is associated with the slowest variable in the system, which typically governs burst adaptation.

For the STG model, we used $\tau_{m_{\mathrm{Na}}}(V)$, $\tau_{m_{\mathrm{Kd}}}(V)$, and $\tau_{\mathrm{H}}(V)$, respectively. In the DA neuron model, we used $\tau_{m_{\mathrm{Na}}}(V)$, $\tau_{m_{\mathrm{Kd}}}(V)$, and a constant-value function $\tau_{\mathrm{u,DA}} = 100\,\mathrm{ms}$, respectively.

In the standard case of purely voltage-dependent ion channels, the factors $\frac{\partial \dot{V}}{\partial X_i}$ can be expressed as:

$$\frac{\partial \dot{V}}{\partial m_i} = \bar{g}_i p_i m_{i,\infty}^{p_i-1} h_{i,\infty}^{q_i} \quad ; \qquad \frac{\partial \dot{V}}{\partial h_i} = \bar{g}_i q_i m_{i,\infty}^{p_i} h_{i,\infty}^{q_i-1} \quad . \tag{S6}$$

A more convenient way of writing equations S4 is through the sensitivity matrix $S(V;\bar{g})$:

$$g_{\mathrm{DICs}}(V) = \begin{bmatrix} g_{\mathrm{f}}(V) \\ g_{\mathrm{s}}(V) \\ g_{\mathrm{u}}(V) \end{bmatrix} = S(V;\bar{g}) \cdot \bar{g} \quad , \tag{S7}$$

and it is the formulation that we use in the generation procedure. One can construct the sensitivity matrix element by element by computing:

$$S_{i,j} = \frac{g_j^{(i)}}{\bar{g}_i}, \quad (i,j) \in \{1,2,\ldots,N_{\text{model}}\} \times \{\text{f},\text{s},\text{u}\} \quad , \tag{S8}$$

where $g_j^{(i)}$ corresponds to the partial sum of definition (S4) involving the current i (so we only keep the terms in which $\bar{g}_i$ is involved, e.g. the elements from equation (S6)) from the DIC associated with the j timescale.

C Population generation procedure

In this paper, we improved the procedure described in [S5] to generate degenerate populations of CBMs. The improved method is called the iterative compensation algorithm. We detail below the exact steps we follow to generate the population of STG models and DA models. The procedure is used both during dataset generation (Fig 3 from the main text) and at inference once the deep learning architecture has been trained (Fig 6-9 from the main text). The main idea behind the procedure is to impose the value of DICs at threshold voltage, as they are known to shape the firing pattern [S3, 5].

C.1 Two different conductances distributions

This section describes the choices related to the distributions of maximal conductances, denoted as $\bar{g} \sim \mathcal{D}$. In practice, we introduce two distinct distributions:

- $\mathcal{D}_{\text{analysis}}$: a distribution that covers a wide region of the conductance space, used for preliminary analyses prior to the generation of degenerate populations.
- $\mathcal{D}_{\text{generation}}$: a distribution whose density is concentrated on a smaller region of the conductances space, specifically employed during the generation of degenerate populations.

The use of these two different distributions is justified by their respective objectives. The distribution $\mathcal{D}_{\text{analysis}}$ is designed to explore a wide range of instances to observe various properties of the CBM under study. It is advised to use a distribution that is broad enough to avoid losing information by limiting it to a smaller region of the space.

The distribution $\mathcal{D}_{\text{generation}}$ is crucial in defining a parameter space that leads to degenerate behaviors. It should be spread out enough to allow for significant degeneracy, while minimizing the impact on firing variability. The variability in firing patterns when enforcing DIC values at threshold depends on the distribution used for the uncompensated conductances. Thus, generated instances should exhibit similar activity to be considered degenerate ($\mathcal{D}_{\text{generation}}$ concentrated), while maintaining large variability in conductance values ($\mathcal{D}_{\text{generation}}$ spread). Essentially, $\mathcal{D}_{\text{generation}}$ should maximize spread to facilitate degeneracy, yet remain concentrated enough to ensure similar activity within the population when DIC values are enforced.

C.1.1 A distribution for preliminary analyses

We adopted the method of [S5] using independent uniform distributions as specified in Table S5a to explore a wide conductance space. Unlike [S5], the leak conductance is first drawn from a Gamma distribution, chosen for its properties and the role of leak conductance. The Gamma distribution models variables with positive support and does not put a prior on the maximum value. The gamma distribution, with suitable parameters, stabilizes normalization by minimizing density near zero, unlike the uniform distribution used in [S5]. The shape and scale parameters match the first and second moments with the uniform distribution from [S5]. The Gamma distribution is characterized by the following probability density function:

$$X \sim \text{Gamma}(k, \theta) \implies p_X(x) = \frac{1}{\Gamma(k)\theta^k} x^{k-1} e^{-\frac{x}{\theta}} \quad , \tag{S9}$$

where k is the shape parameter, and θ is the scale parameter.

C.1.2 A distribution for the generation procedure

Following [S5], we used a more concentrated distribution for the generation procedure. Beyond concentrating the probability mass, this distribution also controls the effect of *homogeneous scaling* [S5]. Here, conductances are no longer independent: the leak conductance is sampled first, and the remaining conductances are scaled by the factor $\frac{g_{\mathrm{leak}}}{g_{\mathrm{leak,mean}}}$. The distributions follow [S5], i.e., uniform laws with bounds given in Table S5b. As before, we adopt a Gamma distribution for the leak conductance, fitted to match the uniform bounds of [S5]. No explicit bounds are set for $\bar{g}_{\mathrm{Na}}, \bar{g}_{\mathrm{Kd}}$, and $\bar{g}_{\mathrm{H}}$, since these are compensated during Step 1 of the generation procedure (Algorithm S1), i.e., $\bar{g}_{\text{comp., spont.}} = (\bar{g}_{\mathrm{Na}}, \bar{g}_{\mathrm{Kd}}, \bar{g}_{\mathrm{H}})$.

Table S5. Distributions of maximal conductances in the STG model. Two distinct distributions $\bar{g} \sim \mathcal{D}$ are used at different stages of the study: (a) $\mathcal{D}_{\mathrm{analysis}}$, a broad distribution for preliminary analyses, and (b) $\mathcal{D}_{\mathrm{generation}}$, at inference and generation. Values are given such that the conductances are in $\mathrm{mS\,cm^{-2}}$.

(a) Preliminary conductance distribution $\mathcal{D}_{\mathrm{analysis}}$, used for broad exploratory analyses. Conductances are sampled independently from uniform distributions with predefined upper bounds, except for the leak conductance, which follows a gamma distribution.

Conductance $\bar{g}_i \sim \mathcal{U}(0; \bar{g}_{\max})$	$\bar{g}_{\mathrm{Na}}$	$\bar{g}_{\mathrm{Kd}}$	$\bar{g}_{\mathrm{CaT}}$	$\bar{g}_{\mathrm{CaS}}$	$\bar{g}_{\mathrm{KCa}}$	$\bar{g}_{\mathrm{A}}$	$\bar{g}_{\mathrm{H}}$
Maximum value $\bar{g}_{\max}$	8000	350	12	50	250	600	0.7

Conductance $\bar{g}_i \sim \mathrm{Gamma}(k, \theta)$	k	θ
g_{leak}	3	$\frac{1}{300}$

(b) Generation conductance distribution $\mathcal{D}_{\mathrm{generation}}$, used to produce degenerate populations. This distribution is more concentrated and incorporates homogeneous scaling control by normalizing conductances relative to the leak conductance.

Conductance $\bar{g}_i \sim \mathcal{U}(\bar{g}_{\min}; \bar{g}_{\max}) \frac{g_{\mathrm{leak}}}{g_{\mathrm{leak,mean}}}$	$\bar{g}_{\mathrm{Na}}$	$\bar{g}_{\mathrm{Kd}}$	$\bar{g}_{\mathrm{CaT}}$	$\bar{g}_{\mathrm{CaS}}$	$\bar{g}_{\mathrm{KCa}}$	$\bar{g}_{\mathrm{A}}$	$\bar{g}_{\mathrm{H}}$
Minimum value $\bar{g}_{\min}$	—	70	2	6	140	—	—
Maximum value $\bar{g}_{\max}$	—	140	7	22	180	—	—

Conductance $\bar{g}_i \sim \mathrm{Gamma}(k, \theta)$	k	θ
g_{leak}	27	$\frac{1}{2570}$

A similar approach was followed for the DA model and we report the distribution in Tables S6a and S6b. The only distinction is that $\bar{g}_{\mathrm{NMDA}}$ was computed to be directly proportional to the leak conductance, corresponding to a fixed biological network around the cell across instances as this current is mainly involved in synaptic transmission.

C.2 Approximation of the threshold voltage

We based our choice of a shared fixed threshold voltage approximation on statistical estimation built from extensive sampling of conductance distributions. The results were $V_{\mathrm{th}} \approx -55.5\,\mathrm{mV}$ and $V_{\mathrm{th}} \approx -51\,\mathrm{mV}$ for the DA and the STG model, respectively. These values correspond to the median

Table S6. Distributions of maximal conductances in the DA model. Two distinct distributions $\bar{g} \sim \mathcal{D}$ are used: (a) $\mathcal{D}_{\text{DA-analysis}}$ for preliminary analyses, and (b) $\mathcal{D}_{\text{DA-generation}}$ for inference and generation. Values are in $\mathrm{mS\,cm^{-2}}$.

(a) Preliminary conductance distribution $\mathcal{D}_{\text{DA-analysis}}$ for exploratory analyses. Conductances are sampled independently from uniform distributions with predefined upper bounds, except for the leak conductance, which follows a gamma distribution.

Conductance $\bar{g}_i \sim \mathcal{U}(0; \bar{g}_{\max})$	$\bar{g}_{\text{Na}}$	$\bar{g}_{\text{Kd}}$	$\bar{g}_{\text{CaL}}$	$\bar{g}_{\text{CaN}}$	$\bar{g}_{\text{ERG}}$	$\bar{g}_{\text{NMDA}}$
Maximum value $\bar{g}_{\max}$	60	20	0.1	0.12	0.25	0.012

Conductance $\bar{g}_i \sim \text{Gamma}(k, \theta)$	k	θ
$\bar{g}_{\text{leak}}$	3	$\frac{1}{300}$

(b) Generation conductance distribution $\mathcal{D}_{\text{DA-generation}}$ for producing degenerate populations. Distribution is more concentrated and normalized relative to the leak conductance.

Conductance $\bar{g}_i \sim \mathcal{U}(\bar{g}_{\min}; \bar{g}_{\max}) \frac{g_{\text{leak}}}{g_{\text{leak, mean}}}$	$\bar{g}_{\text{Na}}$	$\bar{g}_{\text{Kd}}$	$\bar{g}_{\text{CaL}}$	$\bar{g}_{\text{CaN}}$	$\bar{g}_{\text{ERG}}$	$\bar{g}_{\text{NMDA}}$
Minimum value $\bar{g}_{\min}$	—	6	0.015	—	—	0.012
Maximum value $\bar{g}_{\max}$	—	10	0.075	—	—	0.012

Conductance $\bar{g}_i \sim \text{Gamma}(k, \theta)$	k	θ
g_{leak}	28.76	$\frac{1}{2238}$

of a set of 4000-instance populations sampled from $\mathcal{D}_{\text{analysis}}$, for which we computed an approximation of V_{th} as the first decreasing root of the total DIC g_{t}. That is, using a bisection method, we computed the first value of V such that:

$$g_{\text{t}}(V_{\text{th}}) = 0 \ \text{ with } \ g_{\text{t}}(V_{\text{th}} - \delta V) > 0 > g_{\text{t}}(V_{\text{th}} + \delta V), \quad \text{for all sufficiently small } \delta V > 0 \quad , \tag{S10}$$

for each $\bar{g} \sim \mathcal{D}_{\text{analysis}}$.

The estimates were robust to the set size (we checked 8000-instance populations and found close values). Figure S1 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.

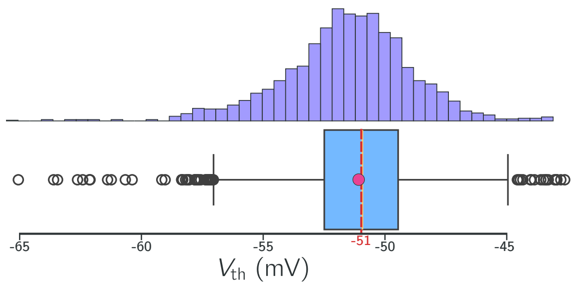

Fig S1. Histogram of threshold voltage estimates ($V_{\mathbf{th}}$) for 8000 STG model instances. The distribution remains nearly identical to that obtained with 4000 instances, indicating convergence of the estimation around $V_{\mathrm{th}} \approx -51$ mV. The mean (pink dot) and median (yellow line) are close to the estimated value (red line).

C.3 The algorithmic procedure

We summarize the procedure used to generate populations of conductance-based models (CBMs) targeting specific Dynamic Input Conductances (DICs) as an algorithmic workflow. The method applies to a generic N-channel neuron model and consists of two main steps: generating spontaneously active populations and modulating them to reach a target point in the DIC space.

The first step produces a spontaneously active (spiking) population, while the second step iteratively adjusts the conductances to reach the target DIC values $(g_{\mathrm{s}}, g_{\mathrm{u}})$. We acknowledge that some negative conductances may remain in the DA model for $g_{\mathrm{u}} < 1.5$, but the selection of conductances for compensation reduces most invalid instances.

C.4 Target DIC values

In this section, we report the exact DIC values we targeted to produce the different results of the paper, as well as the bounds on the g_{s} - g_{u} spaces we used when generating the datasets.

The considered bounds of the DIC spaces were derived by extensively sampling the $\mathcal{D}_{\mathrm{analysis}}$ conductance distributions. Based on the same 4000 instances (then 8000 to ensure convergence) used to estimate a share value of threshold voltage, we extracted the observed bounds of DICs at threshold. The results were largely enclosed in $(g_{\mathrm{s}}(V_{\mathrm{th}}); g_{\mathrm{u}}(V_{\mathrm{th}})) \in [-20; 20] \times [-2; 20]$ for the STG and $[-15; 15] \times [0; 20]$ for the DA. We restricted the ultra-slow DIC to be positive (effectively using

Algorithm S1 Iterative compensation procedure for CBM population generation

```
1: **Input:** Number of channels $N$, DIC target $g_{\mathrm{DICs}} = (g_{\mathrm{s}}, g_{\mathrm{u}})$, threshold value $V_{\mathrm{th}}$, conductance
   distributions $\mathcal{D}_{\mathrm{generation}}$ and the leak distribution Gamma.
2: **Output:** Population of CBM instances with compensated conductances.
3: **procedure** GENERATESPONTANEOUSPOPULATION
4:     Draw leak conductance $g_{\mathrm{leak}} \sim$ Gamma.
5:     Draw $N-3$ maximal conductances from $\mathcal{D}_{\mathrm{generation}}$ scaled proportionally to $g_{\mathrm{leak}}$.
6:     Compute the remaining 3 compensated conductances to impose a sufficiently negative $g_{\mathrm{f}}(V_{\mathrm{th}})$:
         • STG: $\bar{g}_{\mathrm{comp.,\ spont.}} = (\bar{g}_{\mathrm{Na}}, \bar{g}_{\mathrm{Kd}}, \bar{g}_{\mathrm{H}})$
         • DA: $\bar{g}_{\mathrm{comp.,\ spont.}} = (\bar{g}_{\mathrm{Na}}, \bar{g}_{\mathrm{CaN}}, \bar{g}_{\mathrm{ERG}})$
7:     Target the DIC values in Table S7a.
8: **end procedure**
9: **procedure** MODULATEPOPULATIONTOTARGETDIC(population, $g_{\mathrm{DICs}}$)
10:    Initialize iteration counter $k = 0$
11:    **while** $k < K_{\max}$ **do**                                        ▷ e.g., $K_{\max} = 5$
12:        Update compensated conductances using the iterative solver based on sensitivity $S(\bar{g}_{\mathrm{comp.}})$
13:        to approach $g_{\mathrm{DICs}}$.
14:        Increment $k \leftarrow k + 1$
15:    **end while**
16:    Remove any instances with negative compensated conductances.
17:    To reduce negative values, select conductances to compensate as:
```

$$\bar{g}_{\mathrm{comp.}} = \begin{cases} (\bar{g}_{\mathrm{CaS}}, \bar{g}_{\mathrm{H}}), & g_{\mathrm{s}} < 0 \\ (\bar{g}_{\mathrm{A}}, \bar{g}_{\mathrm{H}}), & g_{\mathrm{s}} > 0 \end{cases} , \quad \bar{g}_{\text{DA-comp.}} = \begin{cases} (\bar{g}_{\mathrm{ERG}}, \bar{g}_{\mathrm{CaL}}), & g_{\mathrm{s}} < 0 \\ (\bar{g}_{\mathrm{ERG}}, \bar{g}_{\mathrm{Kd}}), & g_{\mathrm{s}} > 0 \end{cases} \tag{S11}$$

```
18: **end procedure**
19: **Execute:**
20: *population* $\leftarrow$ GenerateSpontaneousPopulation()
21: *population* $\leftarrow$ ModulatePopulationToTargetDIC(*population*, $g_{\mathrm{DICs}}$)
22: **return** *population*
```

$(g_\mathrm{s}(V_\mathrm{th}); g_\mathrm{u}(V_\mathrm{th})) \in [-20; 20] \times [0; 20]$ for the STG) because we observed generation issues (populations were not degenerate) for negative values. We did not explore much the origin of the failure, but we noticed that such negative ultra-slow threshold DIC corresponded to the negative outliers from Fig S1. We hypothesize that our choice of threshold voltage value was not adapted for such instances. For the DA, we focused during inference on $g_\mathrm{s} \in [-10; 15]$, as all instances generated from $g_\mathrm{s} < -10$ were silent.

The residual analysis of Fig 2 from the main text is based on sampling within the above-reported range for the STG. We generated 5000 populations of 250 instances from uniform sampling of the DIC targets and compensated $\bar{g}_\mathrm{CaS}$ and $\bar{g}_\mathrm{A}$. We discarded any populations with negative conductance values and used the remaining 1633 ones for the analysis. The random set of values is shared for all methods used, so the analysis directly compared the results of the compensation step.

The spiking and bursting populations given as examples in Fig 2 from the main text were generated by targeting the DICs reported in Table S7b.

Table S7. Summary of the target DIC values used in this work.
(a) Target DIC values at threshold used in the first step of the generation procedure. Those correspond to spontaneously active spiking populations.

Target DIC values	g_f	g_s	g_u
STG	-6.2	4	5
DA	-12.95	0.5	5

(b) Target DIC values for the example populations of Fig 2 from the main text. Values are targeted at threshold.

Target DIC values	g_s	g_u
Spiking	5	4
Bursting	-2.71	5.63

C.5 Definition of the firing activity descriptors and spike definition

Throughout the paper, we rely on descriptors of the neuronal activity. Those features are **not** inputs of the deep learning architecture; however, they provide useful intermediates to describe the firing pattern of instances (see Fig 2 from the main text, panels D and E). When simulating CBMs, we obtained the full voltage trace $V(t)$. To simulate experimental conditions where only the spike timing is recorded, we extracted the spike times from the voltage trace $V(t)$ using a two-threshold method. A spike was considered to occur at the midpoint between the voltage crossing an upper threshold $V_{\mathrm{up}} = 10\,\mathrm{mV}$ from below and subsequently crossing a lower threshold $V_{\mathrm{down}} = 0\,\mathrm{mV}$ from above. Formally, for each spike i, we defined its time t_i as:

$$t_i = \frac{t_{\mathrm{up}}^{(i)} + t_{\mathrm{down}}^{(i)}}{2} \quad ,$$

where $t_{\mathrm{up}}^{(i)}$ is the time when $V(t)$ first exceeds V_{up} and $t_{\mathrm{down}}^{(i)}$ is the time when it next falls below V_{down}. Applying this procedure to all spikes in the trace produces the sequence:

$$V(t) \xrightarrow{\text{spike extraction}} x = [t_1, t_2, \ldots, t_{N_{\mathrm{spikes}}}] \quad .$$

Given a sequence of spike times $x = [t_1, \ldots, t_{N_{\mathrm{spikes}}}]$ and the corresponding sequence of inter-spike intervals $x_{\mathrm{ISI}} = \Delta x = [t_2 - t_1, t_3 - t_2, \ldots, t_{N_{\mathrm{spikes}}} - t_{N_{\mathrm{spikes}}-1}]$ where N_{spikes} is the number of recorded spikes, we used the following features:

- **For spiking activities**, we reported *the mean firing frequency* computed as:

$$f_{\mathrm{spk}} = \left(\frac{1}{N_{\mathrm{spikes}} - 1} \sum_{i=1}^{N_{\mathrm{spikes}}-1} \Delta x_i \right)^{-1} \quad . \tag{S12}$$

- **For bursting activities**, we computed metrics based on the burst extraction. That is, from the spike times sequence, we built a sequence of bursts $\mathcal{B} = [B_1, \ldots, B_{N_{\mathrm{burst}}}]$, where the B_i are sequences of $N_{i,\mathrm{spikes}}$ consecutive spikes split based on the mid-range value of x_{ISI}. The first spike of each burst was such that the duration since the previous spike was longer than $\frac{\min(x_{\mathrm{ISI}})+\max(x_{\mathrm{ISI}})}{2}$. When building $\mathcal{B}$, we discarded the first and last bursts, as they could be incomplete due to the start and end points of the recordings (or here, the simulations). We then reported:

  1. *the mean intra-burst frequency* computed as:

$$f_{\mathrm{intra}} = \left(\frac{1}{N_{\mathrm{burst}}} \sum_{i=1}^{N_{i,\mathrm{spikes}}-1} \Delta B_i \right)^{-1} \quad . \tag{S13}$$

2. *the mean inter-burst frequency* computed as:

$$f_{\text{inter}} = \left(\frac{1}{N_{\text{burst}} - 1} \sum_{i=1}^{N_{\text{burst}}-1} (B_{i+1})_{N_{i+1,\text{spikes}}} - (B_i)_1 \right)^{-1} \quad , \tag{S14}$$

   that is, the inverse of the mean duration *between* bursts.

3. *the mean burst duration* computed as:

$$\text{Duration} = \frac{1}{N_{\text{burst}}} \sum_{i=1}^{N_{\text{burst}}} \sum_{j=1}^{N_{i,\text{spikes}}-1} (\Delta B_i)_j \quad . \tag{S15}$$

4. *the mean number of spikes per burst* computed as:

$$\# = \frac{1}{N_{\text{burst}}} \sum_{i=1}^{N_{\text{burst}}} N_{i,\text{spikes}} \quad . \tag{S16}$$

   We note that some bursters were *slightly* irregular; for example, some produced alternating bursts with 3 spikes and bursts with 4 spikes, and we kept the mean as a *non-integer* value for those instances. This is why this metric can take non-integer values despite the quantized nature of the number of spikes per burst.

D Heterogeneous populations at the spiking-bursting transition

Throughout the whole paper, there is, in the figures (e.g., Fig 4 and Fig 7 from the main text), an overlap between the DIC region associated with bursting and the one associated with spiking. We can roughly consider that $g_{\text{s}} \approx 0$ marks the transition between spiking ($g_{\text{s}} > 0$) and bursting ($g_{\text{s}} < 0$), with almost no dependency on the value of g_{u}. However, in practice, this transition is not sharp, and there is a whole region that marks it. In this region, generated populations are heterogeneous, with a mix of spiking and bursting instances; the bursting instances produce doublets (a burst with only 2 spikes) or alternations of single spikes and doublets. Such regions can be highlighted by generating populations and computing the class entropy of each population P:

$$\mathcal{H}(P) = - \sum_{t \in \{\text{silent},\text{spiking},\text{bursting}\}} p_t \log_3(p_t) \in [0,1] \quad , \tag{S17}$$

where homogeneous populations have zero entropy, and highly heterogeneous populations have an entropy of one. Figure S2 highlights the bursting-spiking transitions based on the entropy. We also observe a small region of mixed spiking-silent instances in the bottom right of the DIC space.

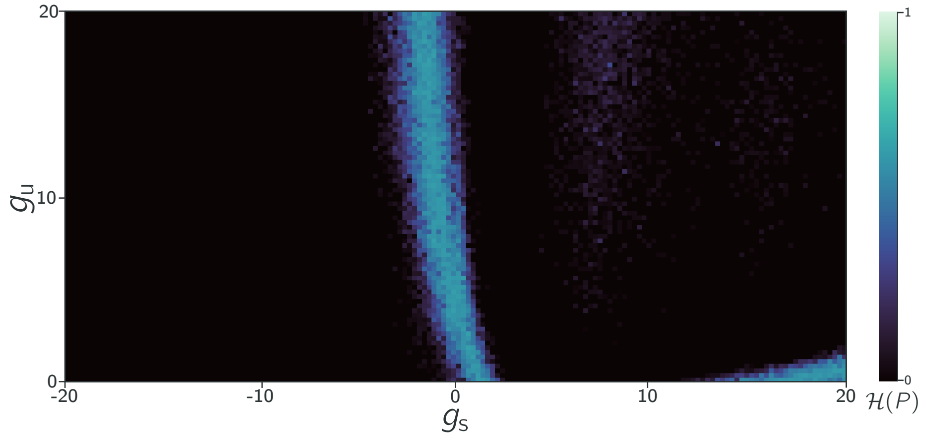

Fig S2. Population heterogeneity across the STG DIC space. The spiking–bursting transition ($g_s \approx 0$) correspond to a highly heterogeneous region where populations are a mix of spiking and bursting instances.

E Generating stochastic firing patterns using Poisson processes

To study the resilience of our pipeline to noisy input, we used spike time sequences generated from Poisson processes (Fig 8 from the main text). The exact definition is provided in the Materials and Methods section of the main text. Table S8 reports the range used when uniformly sampling the process parameters.

Table S8. 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

F The deep learning architecture

F.1 Training details

The deep learning implementation was done using PyTorch. We used the AdamW optimizer [S6] and a cosine annealing schedule with warm restarts. The final architecture was trained for 200 epochs, and we saved the parameter values that achieved the best performance on the validation set based on the value of $\mathcal{L}_{\text{DICs}}$. 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:

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

where:

- η is the initial learning rate. This value was tuned through hyperparameter tuning.
- $\eta_{\min}$ is the minimum learning rate. We used $\eta_{\min} = \frac{\eta}{10}$.

- 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.

F.2 Hyperparameter tuning of the backbone architecture

To achieve optimal performance from our pipeline, we explored various hyperparameters to identify a suitable architecture and training algorithm. We employed an extended random search procedure over $N_{\text{search}} = 100$ sets of hyperparameters [S7]. Table S9 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* [S8, 9], `tanh` to the hyperbolic tangent, and `gelu` to the *gaussian error linear units* [S8]. 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 S3 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 S10.

Table S9. Hyperparameters and distributions explored in random search. $\tilde{\lambda}_{\text{c}}$ and $\tilde{\lambda}_{\text{m}}$ are relative weights of the auxiliary losses to the primary DICs loss (e.g., $\tilde{\lambda}_{\text{c}} = 10$ makes the classification loss 10× 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 ($\tilde{\lambda}_{\text{m}}$)	Uniform	$[0.01, 10.0]$
Classification weighting factor ($\tilde{\lambda}_{\text{c}}$)	Uniform	$[0.01, 10.0]$
Batch size ($\lvert B \rvert$)	Discrete	$\{16, 32, 64, 128\}$

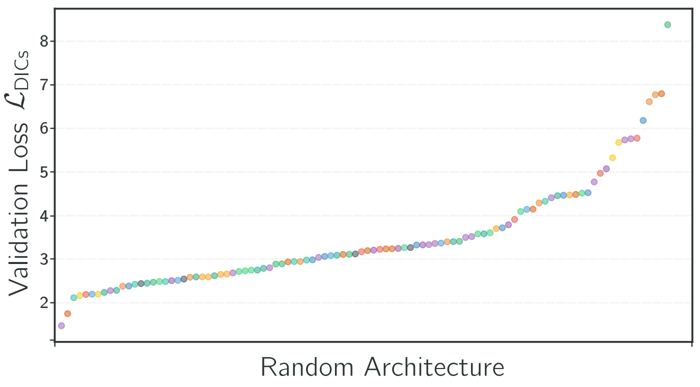

Fig S3. Performance of random architectures. Scatter plot of best validation loss $\mathcal{L}_{\mathrm{DICs}}$ across randomly sampled architectures. Each point is a unique configuration, illustrating how design choices impact performance.

Table S10. 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 ($B_{\mathcal{R}_{\mathrm{DICs}}}$)	2
Activation function ($\sigma(\cdot)$)	`gelu`
Apply log transform to input?	`true`
Regression weighting factor ($\tilde{\lambda}_{\mathrm{m}}$)	0.0919
Classification weighting factor ($\tilde{\lambda}_{\mathrm{c}}$)	5.44
Batch size ($\|B\|$)	32

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

Number of Parameters	115,627

F.3 Hyperparameter tuning of the LoRA-adapted architecture

The trained backbone architecture was used to transfer from the STG model to the DA model. We introduced LoRA adapters [S10] in the linear layers of the architecture. The Fig S4 illustrates how we introduced the LoRA adapter within the backbone architecture.

Low-Rank Adaptation Adapters LoRA adapters [S10] are a type of adapter characterized by a low number of additional parameters. To achieve this, LoRA relies on parameterized matrices with low rank: $A \in \mathbb{R}^{d \times r}$, $B \in \mathbb{R}^{r \times k}$ (Fig S4B). The r value is an hyperparameter that we tuned through grid-search. The adapter can be introduced in fully connected layers or self-attention layers by modifying each matrix multiplication by $W \in \mathbb{R}^{d \times k}$ as follows:

$$Wx \xrightarrow[\text{(plug icon)}]{\text{LoRA adapter}} \underbrace{Wx}_{\text{frozen (freeze icon)}} + \underbrace{\frac{1}{r} xAB}_{\text{learnable (fire icon)}} . \tag{S18}$$

In this work, we only introduced them into fully-connected layers as it was sufficient to get good transfer performances. It allows us to not increase the number of parameters too much.

LoRA adapters were controlled by a single hyperparameter, r, which defined the intermediate (low-rank) dimension. Table S11 reports the best validation loss for each r, along with the number of additional parameters introduced. The number of LoRA parameters grows linearly with r:

$$\#\theta_{\text{LoRA}} = \underbrace{1245r}_{\text{LoRA } A \text{ and } B \text{ matrices}} + \underbrace{4}_{\mu_{\text{train}} \text{ and } \sigma_{\text{train}}} . \tag{S19}$$

The best results were obtained with $r = 32$; we used this value in the final training step.

Table S11. LoRA performance and parameter count for different r. Validation loss $\mathcal{L}_{\text{DICs}}$ and number of additional parameters for various ranks r.

r	2	4	8	16	**32**	48	64
$\mathcal{L}_{\text{DICs}}$	3.926	2.964	2.596	1.979	1.633	1.935	1.721
$\#\theta_{\text{LoRA}}$	2,494	4,984	9,964	19,924	39,844	59,764	79,684

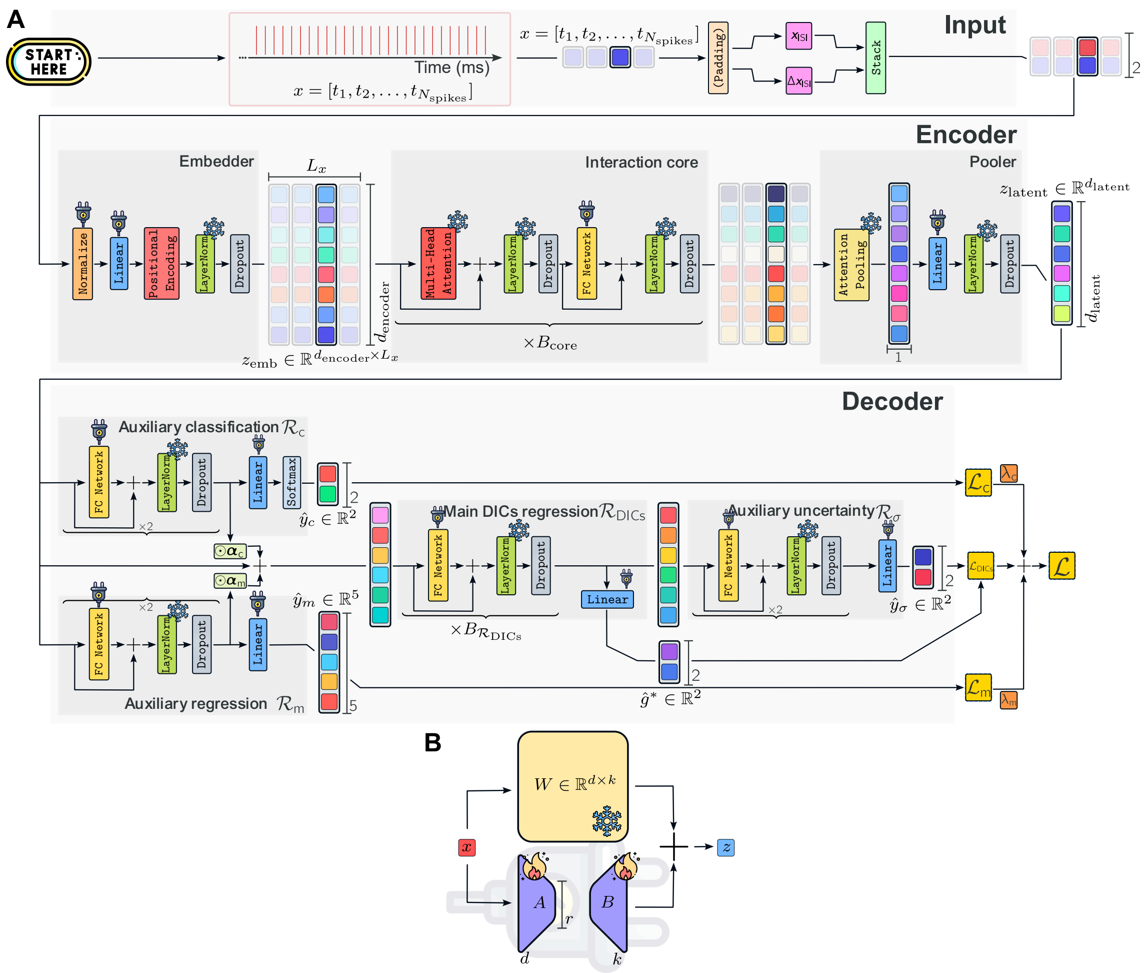
A
START HERE
Time (ms)
$x = [t_1, t_2, \dots, t_{N_{\text{spikes}}}]$
(Padding)
x_{ISI}
Δx_{ISI}
Stack
Input
2
Encoder
Embedder
Normalize
Linear
Positional Encoding
LayerNorm
Dropout
L_x
d_{encoder}
$z_{\text{emb}} \in \mathbb{R}^{d_{\text{encoder}} \times L_x}$
Interaction core
Multi-Head Attention
FC Network
$\times B_{\text{core}}$
Pooler
Attention Pooling
$z_{\text{latent}} \in \mathbb{R}^{d_{\text{latent}}}$
d_{latent}
1
Decoder
Auxiliary classification $\mathcal{R}_c$
Softmax
$\hat{y}_c \in \mathbb{R}^2$
Main DICs regression $\mathcal{R}_{\text{DICs}}$
$\times B_{\mathcal{R}_{\text{DICs}}}$
Auxiliary uncertainty $\mathcal{R}_\sigma$
$\hat{y}_\sigma \in \mathbb{R}^2$
Auxiliary regression $\mathcal{R}_m$
$\hat{y}_m \in \mathbb{R}^5$
$\odot \alpha_c$
$\odot \alpha_m$
×2
5
$\hat{g}^* \in \mathbb{R}^2$
$\mathcal{L}_c$
λ_c
$\mathcal{L}_{\text{DICs}}$
$\mathcal{L}_m$
λ_m
$\mathcal{L}$
B
$W \in \mathbb{R}^{d \times k}$
x
z
A
B
r
d
k

Fig S4. 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 $W \in \mathbb{R}^{d\times k}$.

F.4 Additional experiment: comparison between training from scratch with using the LoRA

This section aims to demonstrate that using adapters is an effective method with a favorable trade-off between the number of parameters and performance. To do this, we compared the performance of the adapters with that of a model trained from scratch.

The architecture trained from scratch is no longer able to take STG neurons as input and to generate accurate predictions, whereas the adapted architecture can once we deactivate the adapters. Table S12 reports the complete results and standard deviations of various metrics on the DA model dataset. Storing both architectures (one for the STG trained from scratch and one for the DA trained from scratch) would require $231,254$ parameters, whereas storing the backbone STG architecture along with the adapters for the DA requires $155,471$; this is a significant saving of memory when considering scaling to many CBMs.

Table S12. Performance summary of the architecture for the DA neuron. 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 number of spikes per bursts (#). Comparison is shown between training from scratch and using LoRA adapters.

Descriptor	f_{spk}	f_{intra}	f_{inter}	Duration	#
MAE (trained from scratch)	1.57 Hz	2.43 Hz	0.60 Hz	39.64 ms	2.65
MAE (LoRA adapters)	1.67 Hz	2.94 Hz	0.44 Hz	30.13 ms	2.63
Dataset std	19.85 Hz	14.50 Hz	1.18 Hz	296 ms	15.64

(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, comparing training from scratch versus using LoRA adapters.

DIC	g_{s}	g_{u}
MAE (trained from scratch)	2.36	1.01
MAE (LoRA adapters)	2.41	1.10

DIC	g_{s} (spiking)	g_{s} (bursting)	g_{u} (spiking)	g_{u} (bursting)
MAE (trained from scratch)	2.57	1.25	1.08	0.61
MAE (LoRA adapters)	2.62	1.28	1.18	0.68

The first observation is that our method can be easily generalized to new CBMs. The performances were not due to chance on the STG model, since we can also obtain them on the DA model. We argue that our method is general and could be applied without issue to any type of model that is interpretable and generable through the theory of DICs. The only constraint is the availability of data, but since the data are synthetic, they can be easily generated.

Next, we observed that although the performance of the model trained from scratch was slightly better on the primary DICs regression task, the adapted model presented close results or even better for the other tasks. Classification accuracies were 99.55% for the model from scratch, and 99.75% with the LoRA. We conclude that this technique is effective for extending the pipeline at a lower cost. The major difference seemed to be in the results of the auxiliary regression task, and we hypothesize that fine-tuning the encoder could potentially yield equivalent or even better performance with the adapters. Our primary interest is in the main task of regressing DIC values, and this task is effectively accomplished by the model with its adapters.

References

S1. Liu Z, Golowasch J, Marder E, Abbott LF. A Model Neuron with Activity-Dependent Conductances Regulated by Multiple Calcium Sensors. Journal of Neuroscience. 1998;18(7):2309-20. arXiv:9502792. doi:10.1523/JNEUROSCI.18-07-02309.1998.

S2. Qian K, Yu N, Tucker KR, Levitan ES, Canavier CC. Mathematical Analysis of Depolarization Block Mediated by Slow Inactivation of Fast Sodium Channels in Midbrain Dopamine Neurons. Journal of Neurophysiology. 2014;112(11):2779-90. doi:10.1152/jn.00578.2014.

S3. Drion G, Franci A, Dethier J, Sepulchre R. Dynamic Input Conductances Shape Neuronal Spiking. eNeuro. 2015;2(1). doi:10.1523/ENEURO.0031-14.2015.

S4. Fyon A, Sacré P, Franci A, Drion G. Reliable Neuromodulation from Adaptive Control of Ion Channel Expression. IFAC-PapersOnLine. 2023;56(2):458-63. doi:10.1016/j.ifacol.2023.10.1610.

S5. Fyon A, Franci A, Sacré P, Drion G. Dimensionality Reduction of Neuronal Degeneracy Reveals Two Interfering Physiological Mechanisms. PNAS Nexus. 2024 Oct;3(10):pgae415. doi:10.1093/pnasnexus/pgae415.

S6. Loshchilov I, Hutter F. Decoupled Weight Decay Regularization. arXiv; 2019. doi:10.48550/arXiv.1711.05101.

S7. Bergstra J, Bengio Y. Random Search for Hyper-Parameter Optimization. J Mach Learn Res. 2012 Feb;13(null):281-305.

S8. Hendrycks D, Gimpel K. Gaussian Error Linear Units (GELUs). arXiv; 2023. arXiv:1606.08415.

S9. Ramachandran P, Zoph B, Le QV. Searching for Activation Functions. arXiv; 2017. arXiv:1710.05941.

S10. Hu EJ, Shen Y, Wallis P, Allen-Zhu Z, Li Y, Wang S, et al.. LoRA: Low-Rank Adaptation of Large Language Models. arXiv; 2021. arXiv:2106.09685.