Supplemental material to "Probabilistic skeletons endow brain-like neural networks with innate computing capabilities"

Christoph Stöckl [*1], Dominik Lang [*1], and Wolfgang Maass [1]

[1] *Institute of Theoretical Computer Science, Graz University of Technology, Austria*

July 1, 2021

List of Figures

S1 Spiking activity on the computations on spike times task. The model correctly assigns the input to class 1. 2
S2 Spiking activity on the computations on spike times. The model correctly assigns the input to class 2. 3
S3 Spiking activity on the computations on spike times. The model correctly assigns the input to class 4. 4
S4 The temporal dynamics $I_{syn}(t)$ of the four different synapse types, as used in [1]. 5
S5 Full connection probabilities of the probabilistic skeleton on the ant task. Note, that input types were restricted to only connect to one exclusive recurrent type. 6
S6 Illustration of population coding. 6

Contents

1 Supplementary figures **2**

2 Supplementary notes **7**
2.1 Continuous neuron model 7

*These authors contributed equally.

2.2 Synapse model 7
2.3 Computation of the neuron prevalence 8
2.4 One-hot encoding 9
2.5 Derivation of the expected number of synaptic connections of a single neuron 9
2.6 Expected wire length 10
2.7 Population coding 11
2.8 Hardware specifications 12

1 Supplementary figures

Additional spike rasters for the computations on spike times

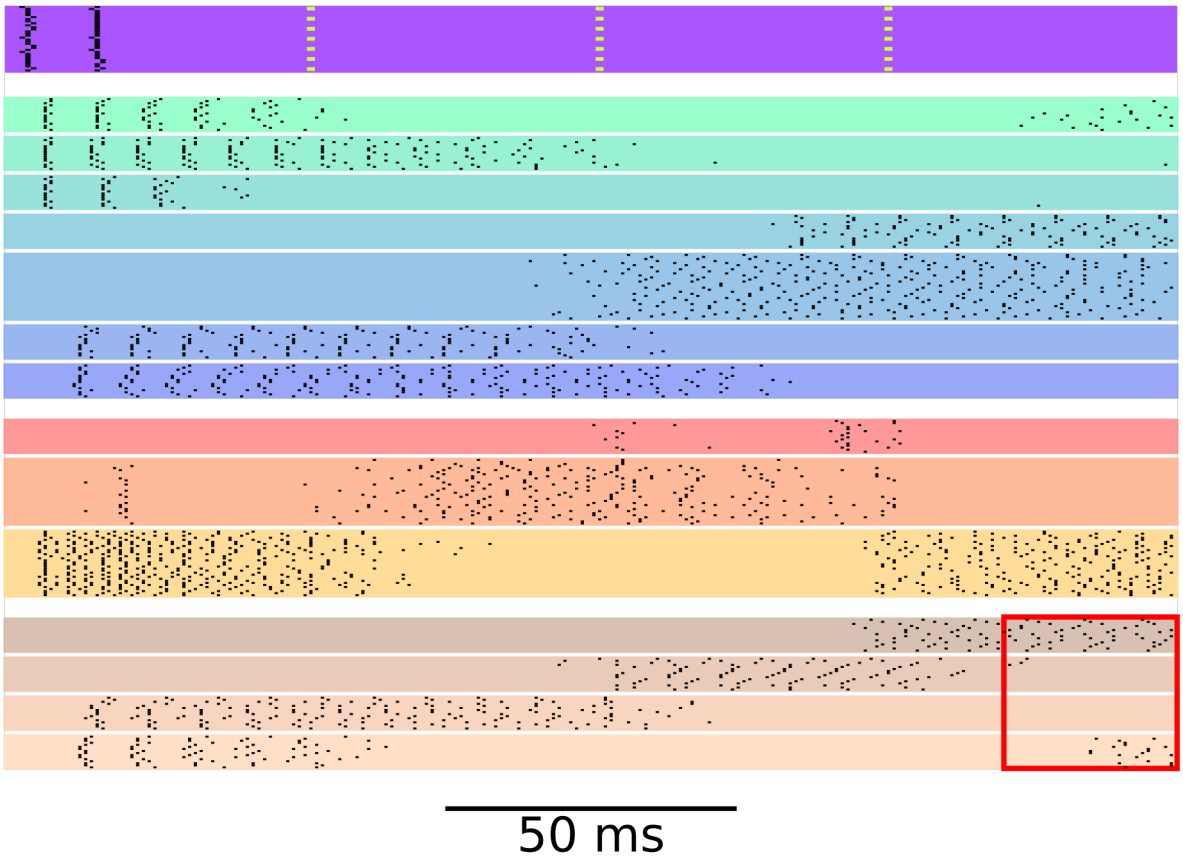

Figure S1: Spiking activity on the computations on spike times task. The model correctly assigns the input to class 1.

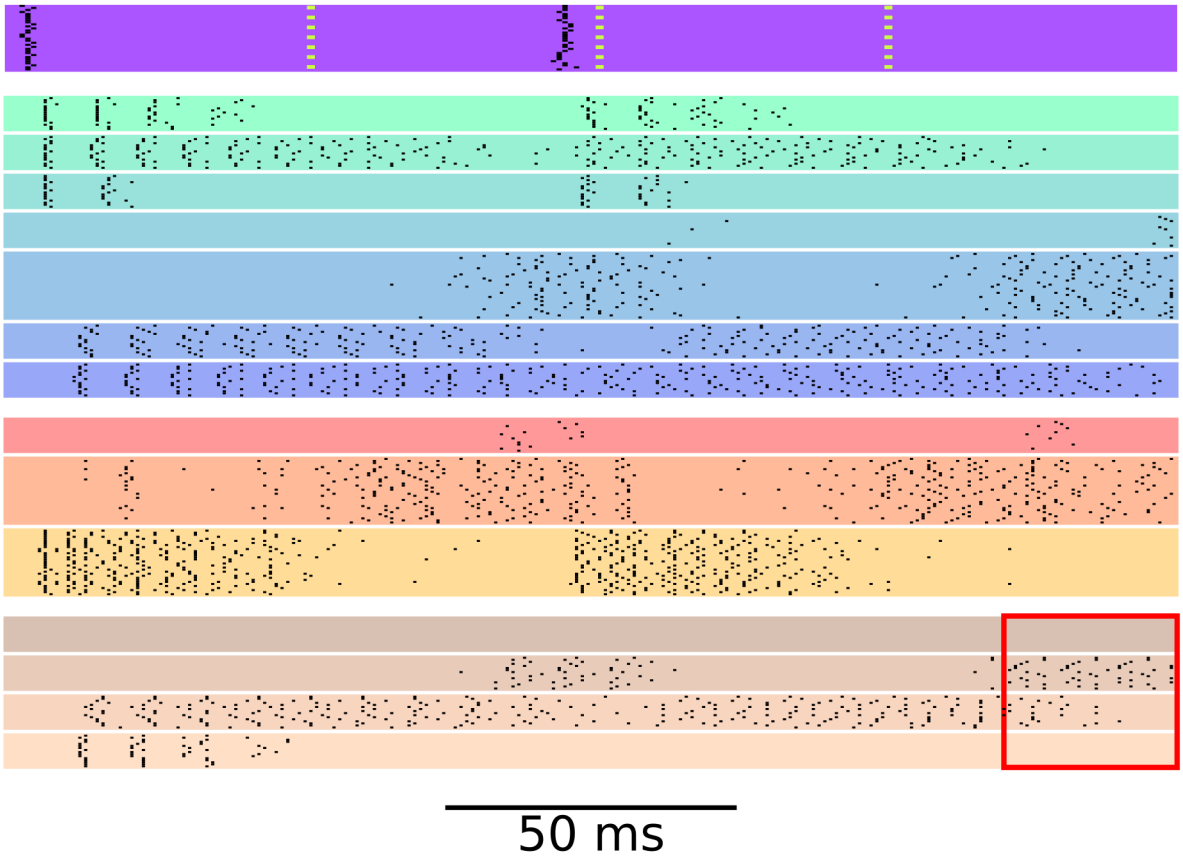

Figure S2: Spiking activity on the computations on spike times. The model correctly assigns the input to class 2.

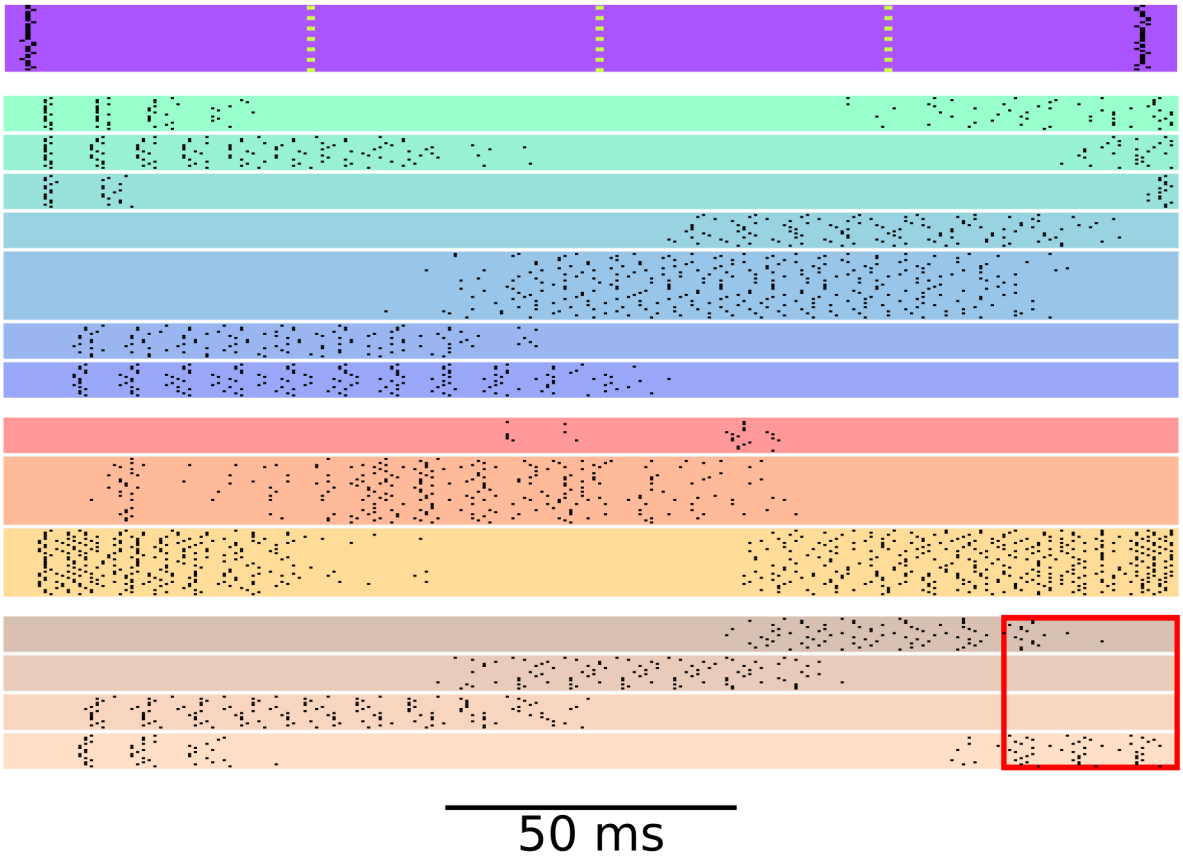

Figure S3: Spiking activity on the computations on spike times. The model correctly assigns the input to class 4.

Responses of the four different synapse types

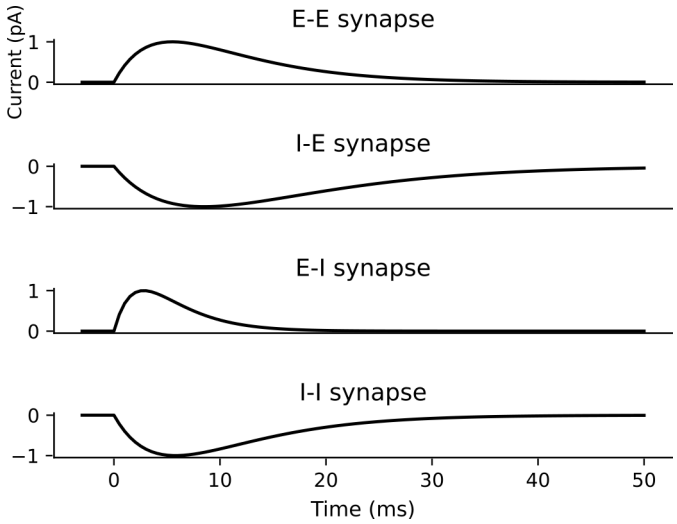

Figure S4: The temporal dynamics $I_{syn}(t)$ of the four different synapse types, as used in [1].

Connection probabilities for ant, including connections of input and output neuron types

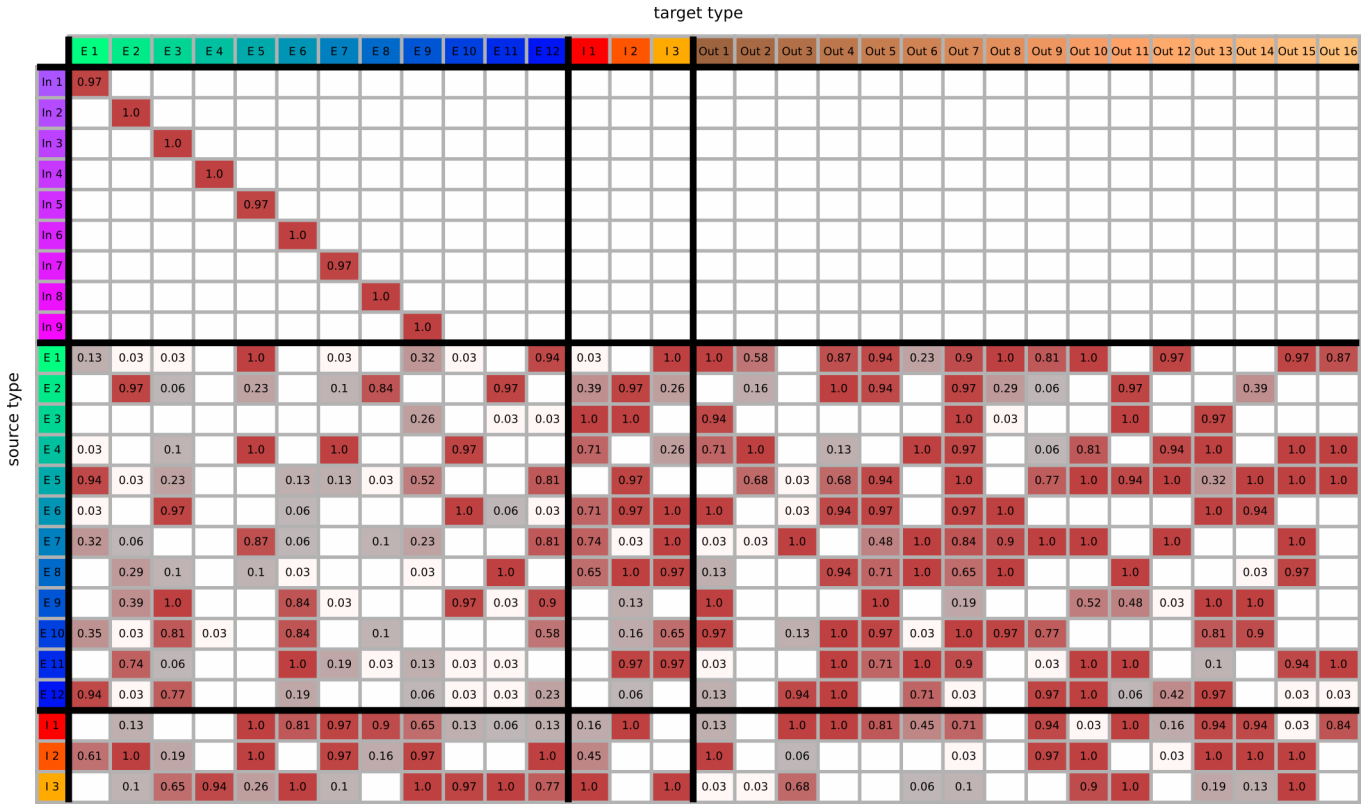

Figure S5: Full connection probabilities of the probabilistic skeleton on the ant task. Note, that input types were restricted to only connect to one exclusive recurrent type.

Illustration of population coding.

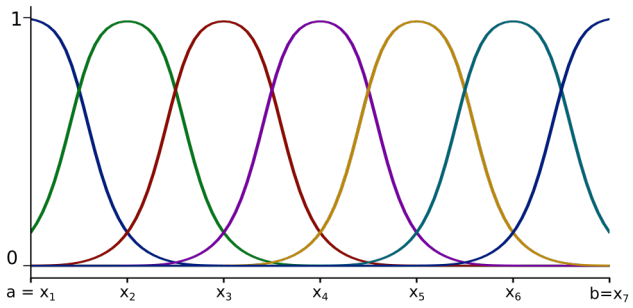

Figure S6: Illustration of population coding.

2 Supplementary notes

2.1 Continuous neuron model

For a given neuron $j \in \{1, \ldots, N\}$ of type J we denote by $V_j : \mathbb{R} \to \mathbb{R}$ the membrane potential and by $I_j : \mathbb{R} \to \mathbb{R}$ the input current. We used different parameters for excitatory and inhibitory neurons that were based on experimental data, see below. Hence these parameters depend for a neuron of type J on whether this is an excitatory or inhibitory neuron type. But for simplicity we will drop in our notation the dependence of the neuron model on the neuron type J until the section below on parameters for neuron and synapse models.

For example, for the membrane conductance we could write $C_m^J \in \mathbb{R}$. Since we will assume for the whole section that we use the same type J we will drop this superscript to simplify the notation, hence

$$C_m = C_m^J.$$

We denote by $\tau \in \mathbb{R}$ the membrane time constant and by $E_L \in \mathbb{R}$ the resting potential. The classic leaky integrate-and-fire linear differential equations reads

$$\tau \frac{d\, V_j(t)}{dt} = -(V_j(t) - E_L) + \frac{1}{C_m} I_j(t). \tag{1}$$

If the voltage is above the threshold v_{th} the parameters get updated by

$$V_j(t+) \leftarrow V_r;\ j = 1, \cdots, N, \tag{2}$$

where $V_r \in \mathbb{R}$ is the reset voltage.

2.2 Synapse model

The time course of a postsynaptic current is modeled like in [1] by a linear increase followed by an exponential decay:

$$I_{syn}(t) = \frac{e}{\tau_{syn}} t\, \delta t\, e^{-\frac{t \delta t}{\tau_{syn}}}, \quad t \in \mathbb{N}. \tag{3}$$

τ_{syn} is the synaptic time constant after which the current amplitude will be at its maximum. Note that τ_{syn} depends on the types of the pre- and postsynaptic neurons. The exact values of τ_{syn} have been set to 5.5 ms for excitatory-to-excitatory synapses, 8.5 ms for inhibitory-to-excitatory synapses, 2.8 ms for excitatory-to-inhibitory synapses and 5.8 ms for inhibitory-to-inhibitory synapses, according to [1]. The current will be negative for synapses from inhibitory neurons. The resulting postsynaptic currents can be seen in Figure S4.

These currents $I_{syn}(t)$ are scaled for each synapse by a general scaling factor w equal to w_{in}, w_E, or w_I, depending on whether the presynaptic neuron is an input neuron, or some other excitatory or inhibitory neuron. Furthermore, they are multiplied for a

synaptic connection from neuron i to neuron j by the number m_{ij} of synaptic connections from i to j. Hence, the current from neuron i to neuron j at time t can be written in terms of the spike train $z_i(s)$ of the presynaptic neuron as

$$I_{ij}(t) = \sum_{k=0}^{t-1} \frac{w\ m_{ij}\ z_i(k)}{\tau_{syn}} e(t-k)\,\delta t\, e^{-\frac{(t-k)\delta t}{\tau_{syn}}}. \tag{4}$$

The input current $I_j(t)$ for neuron j at time t is defined as the sum of these currents over all presynaptic neurons i.

Note, that there is a transmission delay from the creation of a spike until the arrival at a synapse. For excitatory neurons this transmission delay is 3ms and for inhibitory neurons it is 2ms.

2.3 Computation of the neuron prevalence

For the optimization of probabilistic skeletons it is convenient to define neuron prevalences as real values: p_I of type I is a real value that correlates to the fraction of neurons which will belong to type I in a generic minicolumn. In order to compute a corresponding assignment of the neurons in a minicolumn to the neuron types we procede as follows: We apply the softmax function, followed by an adjustment procedure which makes sure that every type has at least one neuron per column and that the total number of neurons stays constant.

The exact algorithm used to obtain the number of neurons per type can be found in algorithm 2.

Algorithm 2 Algorithm for computing the number of neurons per type using the prevalence p, the number of minicolumns m_{ncol} and the total number of recurrent neurons $N_{rec} = (K+3) \cdot n_{mcol}$ (For the contrast enhancement task it is $N_{rec} = (K+2) \cdot n_{mcol}$. The floor function rounds its argument down to the next integer, the maximum function returns the higher valued argument, argsort returns a list of sorted indices, pointing to the highest elements in the argument array first, and the $+=$ operator increments a variable by the amount on the right side.

Require: p, N_{rec}, n_{mcol},
  frac = softmax(p)
  nnpt = frac $\cdot$ N_{rec}
  nnpt = floor(nnpt/n_{mcol})
  nnpt = maximum(nnpt, n_{mcol})
  diff = frac $\cdot$ N_{rec} - nnpt
  max_idx = argsort(diff)
  overlap = floor((N_{rec} - sum(nnpt)) / n_{mcol})
  for $k = 1, \ldots,$ overlap **do**
    nnpt[max_idx[k]] += n_{mcol}
  end for
  return nnpt

2.4 One-hot encoding

To convert the target label y to a one-hot encoded vector one first creates a vector with a length equal to the number of classes consisting only of zeros. Then at the position $y-1$ the value 1 is entered.

For example, if one considers a setting with four classes and the target class is $y=3$ this would yield a vector of:

$$\begin{bmatrix}0\\0\\1\\0\end{bmatrix}. \tag{5}$$

2.5 Derivation of the expected number of synaptic connections of a single neuron

To derive the expected number of outgoing synapses $ST(i)$ a neuron i can form, we consider an infinitely large grid of minicolums. To obtain the desired estimate one can consider neighboring minicolumns to be located on concentric squares around the source neuron i.

Let the function $f(k)$ denote the number of columns in the kth concentric square:

$$f(k)=\begin{cases}1 & \text{if } k=0\\ 8k & \text{else}\end{cases}. \tag{6}$$

Then the expected number of outgoing synapses from neuron i can be written as:

$$ST(i)=\sum_{k=0}^{\infty}\sum_{J} f(k)M_J Sp_{I\to J}e^{-\left(\frac{\alpha k}{\sigma}\right)^2}. \tag{7}$$

, where the first k will sum over all concentric squares, M_J is the number of neurons of type J in a minicolumn, $S=8$ is the maximum number of synapses between two neurons, $\alpha=60\mu m$, and σ accounts for the distance decay. Note, using $\alpha=60\mu m$ will yield an upper bound, as the distance to most minicolumns on the concentric square k will be larger than αk.

An upper bound of this equation is of interest and can be derived:

$$ST(i)=\sum_{k=0}^{\infty}\sum_{J} f(k)M_J Sp_{I\to J}e^{-\left(\frac{\alpha k}{\sigma}\right)^2} \tag{8}$$

$$=S\sum_{J} M_J p_{I\to J}\sum_{k=0}^{\infty} f(k)\gamma^{k^2} \tag{9}$$

, where

$$\gamma=e^{\left(\frac{\alpha}{\sigma}\right)^2}. \tag{10}$$

Breaking up the function $f(k)$ yields:

$$ST(i) = S \sum_J M_J p_{I \to J} \left(1 + 8 \sum_{k=1}^{\infty} k \gamma^{k^2} \right). \tag{11}$$

We get an upper bound by using a standard estimate for the geometric series:

$$S \sum_J M_J p_{I \to J} \left(1 + 8 \sum_{k=1}^{\infty} k \gamma^{k^2} \right) < S \sum_J M_J p_{I \to J} \left(1 + 8 \sum_{k=1}^{\infty} k \gamma^{k} \right) \tag{12}$$

$$= S \sum_J M_J p_{I \to J} \left(1 + 8 \frac{\gamma}{(1-\gamma)^2} \right). \tag{13}$$

Note that this upper bound does not contain anymore an infinite sum.

2.6 Expected wire length

An upper bound for the expected wire length can be computed in a similar fashion. The expected wire length per neuron can be written as:

$$WL(i) = \sum_{k=1}^{\infty} f(k) \sum_J \alpha k M_J p_{I \to J} \left(1 - \left(1 - e^{-\left(\frac{\alpha k}{\sigma}\right)^2} \right)^S \right) \tag{14}$$

Note, that although there can be up to S synapses between two given neurons, but the wire distance is only counted once, based on the assumption that multiple synaptic connections to the same neuron are implemented by axonal connections that branch only locally, so that the additional wire length caused by the branching can be neglected.

A similar upper bound can be computed for the wire length, when one assumes $S = 1$:

$$WL(i) = \sum_{k=1}^{\infty} \sum_J \alpha k f(k) M_J p_{I \to J} e^{-\left(\frac{\alpha k}{\sigma}\right)^2} \tag{15}$$

$$= \alpha \sum_J M_J p_{I \to J} \sum_{k=1}^{\infty} k f(k) \gamma^{k^2} \tag{16}$$

, where

$$\gamma = e^{\left(\frac{\alpha}{\sigma}\right)^2}. \tag{17}$$

Replacing the function $f(k)$ yields:

$$WL(i) = 8\alpha \sum_J M_J p_{I \to J} \left(\sum_{k=1}^{\infty} k^2 \gamma^{k^2} \right). \tag{18}$$

We get an upper bound by using a standard estimate for the geometric series:

$$8\alpha \sum_J M_J p_{I\to J} \left(\sum_{k=1}^{\infty} k^2 \gamma^{k^2} \right) < 8\alpha \sum_J M_J p_{I\to J} \left(\sum_{k=1}^{\infty} k^2 \gamma^{k} \right) \tag{19}$$

$$= 8\alpha \sum_J M_J p_{I\to J} \left(\frac{\gamma(\gamma+1)}{(1-\gamma)^3} \right). \tag{20}$$

. Note that this upper bound does not contain anymore an infinite sum.

For a more general upper bound where $S > 1$ one can consider S RSNNs superimposed, resulting in:

$$WL(i) \leq 8\alpha S \sum_J M_J p_{I\to J} \left(\frac{\gamma(\gamma+1)}{(1-\gamma)^3} \right). \tag{21}$$

Sparsity and wire length of RSNN samples: Table 1 shows the average values of connection sparsity and the total wire length from RSNNs generated by the probabilistic skeletons for each tasks. The sparsity refers to the fraction of synaptic connections between neurons i and j that are realized, i.e. $m_{ij} > 0$..

Task	# of neurons	Sparsity	Avg. # synapses	Wire length
Spike interval classification	304	24.6%	47,700	1.27m
Spike pattern classification	148	37.8%	30,500	0.595m
Contrast enhancement	200	16.03%	4,810	0.505m
Coincidence detection	375	15.44%	54,000	1.47m
Ant	458	11.24%	52,700	0.79m

Table 1: Number of neurons, average connection sparsity, average number of synapses and wire length of RSNNs emerging from the probabilistic skeletons on the different tasks. On the contrast enhancement and the coincidence detection task the RSNNs consisting of 25 minicolumns were considered. Note that the number of synapses in an RSNN will vary between different samples from the same probabilistic skeleton. Hence, the average over 5 RSNN samples has been considered for this value.

2.7 Population coding

Each continuous-valued input value is encoded by the spiking activity of an excitatory population of N_{pop} input neurons. An input variable x can only take values in the bounded interval $(a, b] \subset \mathbb{R}$. We define points $a = x_1, \ldots, x_M = b$ such that the subintervals $(x_k, x_{k+1}]$, $k = 0, \ldots, N_{\text{pop}}$ are of equal lengths $\frac{b-a}{N_{\text{pop}}-1}$ and disjointly overlap the interval. The neurons are chosen such that there is a positive linear dependency between the input values and the spatial position of the neurons. On these subintervals gaussian

density functions are used to model the probability that a neuron i spikes for a given input $x \in (a, b]$ by first defining

$$h_k(x) = \frac{1}{\sigma_{pop}\sqrt{2\pi}} \exp\left\{-\frac{1}{2}\left(\frac{x - x_k}{\sigma_{pop}}\right)^2\right\} \tag{22}$$

for $k = 1, \ldots, N_{\text{pop}}$, where $\sigma_{pop} \in \mathbb{R}$. In the experiments it is chosen to be $\sigma_{pop} = \frac{x_{k+1} - x_{k-1}}{2}$, such that $\sim 68.27\%$ of spiking activity for this type happens in the interval $(x_{k-1}, x_{k+1}]$ and $\sim 95.45\%$ happens in the interval $(x_{k-2}, x_{k+2}]$ for suitable k. Since this is modeled by a density function it is necessary to normalize the values of h_k to obtain spike probabilities for each neuron. A schematic plot of the population coding is given in Figure S4.

The spike train $z_i : \mathbb{R} \to \mathbb{R}$ for an input neuron i and an input value x is therefore given as

$$z_i(t) = \begin{cases} 1 & \text{with probability } \frac{h_i(x)}{\max\limits_{x \in [a,b]} h_i(x)} \\ 0 & \text{else.} \end{cases} \tag{23}$$

2.8 Hardware specifications

Most of our experiments were conducted using 2 × AMD EPYC Rome 7402 CPUs with 2×24 cores and 2.8 GHz, 4 × NVIDIA A100 GPU and 4 × 40 GB HBM2e (Juwels Booster)

References

[1] Yazan N Billeh et al. “Systematic integration of structural and functional data into multi-scale models of mouse primary visual cortex”. In: *Neuron* (2020).