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*

May 18, 2021

Genetically encoded structure endows neural networks of the brain with innate computational capabilities that enable odor classification and basic motor control right after birth. It is also conjectured that the stereotypical laminar organization of neocortical microcircuits provides basic computing capabilities on which subsequent learning can build. However, it has remained unknown how nature achieves this. Insight from artificial neural networks does not help to solve this problem, since their computational capabilities result from learning. We show that genetically encoded control over connection probabilities between different types of neurons suffices for programming substantial computing capabilities into neural networks. This insight also provides a method for enhancing computing and learning capabilities of artificial neural networks and neuromorphic hardware through clever initialization.

1 Introduction

There exists a fundamental difference between the way artificial neural networks receive their functional capabilities and the way biological neural networks acquire them: Artificial neural networks receive their computational capabilities through adaptation of a very large set of parameters, synaptic weights, through training -usually on a very large numbers of examples- starting from a tabula rasa initial state. In contrast, neural networks in brains are already at birth highly structured, and they have innate computing capabilities. [1] has recently reviewed the substantial experimental evidence for that,

*These authors contributed equally.

see e.g. [2], [3], [4], [5], [6], [7], [8]. These data suggest that a substantial fraction of behaviour and sensory representations in brains is innate. In fact, it is necessary for the survival of many species that an individual does not have to learn through trial and error which sources of food are nutritious and which are poisonous. It has also been conjectured that the rather stereotypical local structure of cortical microcircuits, out of which the neocortex is composed as a continuous 2D sheet, endows these neural networks with basic capabilities for processing spike inputs to the microcircuit ([9], [10], [11]). In fact, no method is known for learning these basic functional capabilities from scratch. Rather, innate functional capabilities of cortical microcircuits are likely to provide the platform on which efficient learning through synaptic plasticity takes place.

Substantial knowledge has been assembled during the last two decades about the structure of generic cortical microcircuits, see e.g. [12] and [13] for recent summaries. But it has remained open to what extent the known features of genetically encoded structure is able to program basic computational capabilities into these microcircuits. Network structure is expressed in these data-based models through connection probabilities between genetically different types of neurons, i.e., classes of neurons with specific gene and protein expression profiles. We extract salient aspects of this genetically encoded network structure into a mathematical model, a probabilistic skeleton. We then show that these genetically encoded features suffice for programming substantial computational capabilities into brain-like neural networks. We demonstrate this for a range of concrete computational capabilities for recurrent networks of spiking neurons, such as basic computations on spike times and spike patterns, computations on 2D patters, and basic motor control capabilities. Furthermore, we demonstrate that this method for bringing computational function into neural networks provides network properties that do not arise if function is brought in through training, i.e. through stochastic optimization in an extremely large parameter space. Instead, a probabilistic skeleton determines network structure only on the statistical level, in a much lower dimensional parameter space. This process removes some of the brittleness of trained deep neural networks, for example their sensitivity to weight perturbations. In addition, the resulting number of synapses and the total wire length, i.e., the total length of dendritic and axonal fibers, grow just linearly with the number of neurons. Thus, probabilistic skeletons introduce a new range of more brain-like neural networks with essential structural and functional differences to the networks typically used in machine learning and neuromorphic computing.

We will first present the precise definition of a probabilistic skeleton. We then demonstrate that probabilistic skeletons can induce a number of generic computing capabilities that have been conjectured to be innate. Finally we analyze general properties of the resulting new paradigm for bringing function into neural networks.

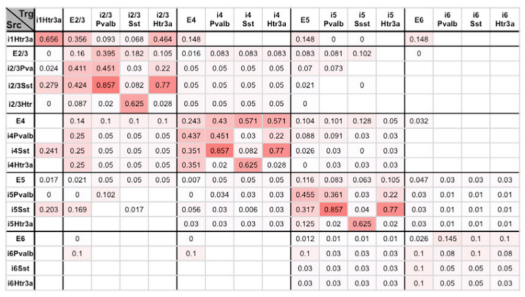
2 Results

2.1 A mathematical model for the way how genetically encoded connection probabilities shape the structure of neural networks

According to [14] ”synaptic junctions are likely organized by trans-synaptic cell-adhesion molecules that bidirectionally orchestrate synapse formation...”. Experimental data on the statistics of synaptic connections in neocortical microcircuits are consistent with these results on the molecular level, since they suggest that the probability of a synaptic connection depends on the genetic type of the pre- and postsynaptic neuron. Existing knowledge about the statistical structure of synaptic connections in area V1 of mouse has recently been compiled by the Allen Institute. These data are shown in the probability table of Fig. 4 in [13], which we have duplicated in Fig. 1a. According to this table the probability of a synaptic connection between two neurons strongly depends on the genetic type of the source and target neuron, for 17 different neuron types. These neuron types are also distinguished through morphological properties and through their location on specific layers of generic cortical microcircuits. Note that some of the entries in this table are empty because of lack of data. In addition, it is known that these 17 types of neurons consist of a substantially larger set of genetically distinguished subtypes, [13] mentions altogether 112 types, for which connection probabilities are currently not yet available.

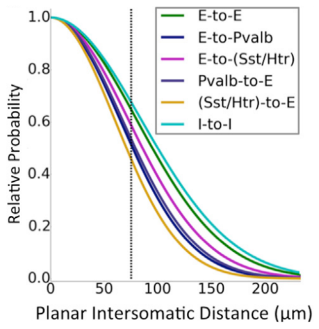
According to [9], [15], [16] the basic spatial unit of the neocortex is a minicolumn, a narrow chain of neurons with a diameter of just 50-60 µm that extends vertically across the neocortical layers. Minicolumns emerge during ontogeny by the iterative division of progenitor cells, that migrate vertically along guiding fibers of glial cells. A minicolumn typically contains 80-120 neurons from all major neuron types, and is hypothesized to be the ”smallest processing unit of the neocortex” [9]. The table in Fig. 1a provides just base connection probabilities for the case that the horizontal distance between the somata of two neurons is very small. We will assume in our model that such probability tables provides the connection probabilities for neurons that are located within the same minicolumn. For neurons that are located in different minicolumns, we multiply this base connection probability, that only depends on the types of the neurons, with an exponentially decaying function of the horizontal distance between their somata; see Fig. 4c of [13], which is reproduced here as Fig. 1b. We approximate this horizontal distance between the somata of neurons by the distance between the centers of the minicolumns in which they are located. A rationale for not taking the vertical distance into account is that most neurons have dendrites that stretch over several layers of the neorcortical microcircuit. Therefore, apart from bundles of vertically running axons within a minicolumn, axons reach target neurons primarily via very fine horizontally running axonal fibers. For computational simplicity we assume that minicolumns form an orthogonal grid, although a hexagonal grid would provide a better approximation of anatomy. We assume that the distance between the centers of two adjacent minicolumns is 60 µm. Furthermore we assume that each minicolumn contains the same number of neurons of each type. Note that according to standard terminology in neuroscience, a cortical microcircuit is substantially larger, and consists of a large number of stereotypical minicolumns.

a

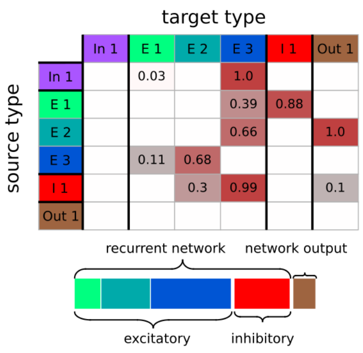


Trg / Src	i1Htr3a	E2/3	i2/3 Pvalb	i2/3 Sst	i2/3 Htr3a	E4	i4 Pvalb	i4 Sst	i4 Htr3a	E5	i5 Pvalb	i5 Ssst	i5 Htr3a	E6	i6 Pvalb	i6 Sst	i6 Htr3a
i1Htr3a	0.656	0.356	0.093	0.068	0.464	0.148				0.148	0	0		0.148			
E2/3	0	0.16	0.395	0.182	0.105	0.016	0.083	0.083	0.083	0.083	0.081	0.102		0			
i2/3Pva	0.024	0.411	0.451	0.03	0.22	0.05	0.05	0.05	0.05	0.07	0.073						
i2/3Sst	0.279	0.424	0.857	0.082	0.77	0.05	0.05	0.05	0.05	0.021		0					
i2/3Htr	0	0.087	0.02	0.625	0.028	0.05	0.05	0.05	0.05	0							
E4		0.14	0.1	0.1	0.1	0.243	0.43	0.571	0.571	0.104	0.101	0.128	0.05	0.032			
i4Pvalb		0.25	0.05	0.05	0.05	0.437	0.451	0.03	0.22	0.088	0.091	0.03	0.03				
i4Sst	0.241	0.25	0.05	0.05	0.05	0.351	0.857	0.082	0.77	0.026	0.03	0	0.03				
i4Htr3a		0.25	0.05	0.05	0.05	0.351	0.02	0.625	0.028	0	0.03	0.03	0.03				
E5	0.017	0.021	0.05	0.05	0.05	0.007	0.05	0.05	0.05	0.116	0.083	0.063	0.105	0.047	0.03	0.03	0.03
i5Pvalb	0	0	0.102			0	0.034	0.03	0.03	0.455	0.361	0.03	0.22	0.03	0.01	0.01	0.01
i5Sst	0.203	0.169		0.017		0.056	0.03	0.006	0.03	0.317	0.857	0.04	0.77	0.03	0.01	0.01	0.01
i5Htr3a						0.03	0.03	0.03	0.03	0.125	0.02	0.625	0.02	0.03	0.01	0.01	0.01
E6		0				0				0.012	0.01	0.01	0.01	0.026	0.145	0.1	0.1
i6Pvalb		0.1				0.1				0.1	0.03	0.03	0.03	0.1	0.08	0.1	0.08
i6Sst										0.03	0.03	0.03	0.03	0.1	0.05	0.05	0.05
i6Htr3a										0.03	0.03	0.03	0.03	0.1	0.05	0.05	0.03

b



c

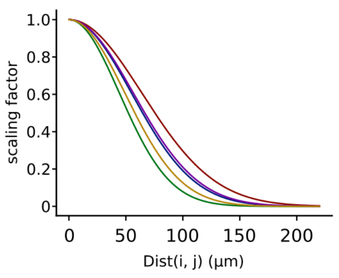


target type

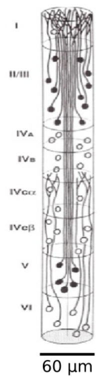
source type	In 1	E 1	E 2	E 3	I 1	Out 1
In 1		0.03		1.0		
E 1				0.39	0.88	
E 2				0.66		1.0
E 3		0.11	0.68			
I 1			0.3	0.99		0.1
Out 1						

recurrent network
network output
excitatory
inhibitory

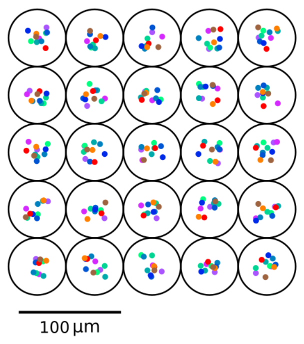
d



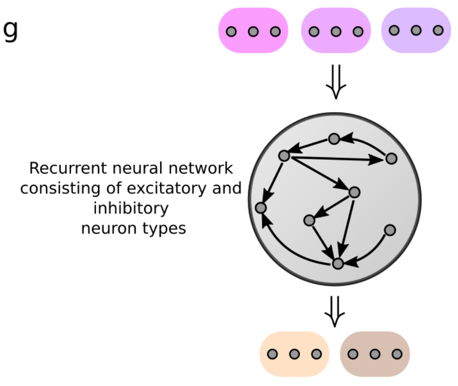
e



f



g


Recurrent neural network consisting of excitatory and inhibitory neuron types

h

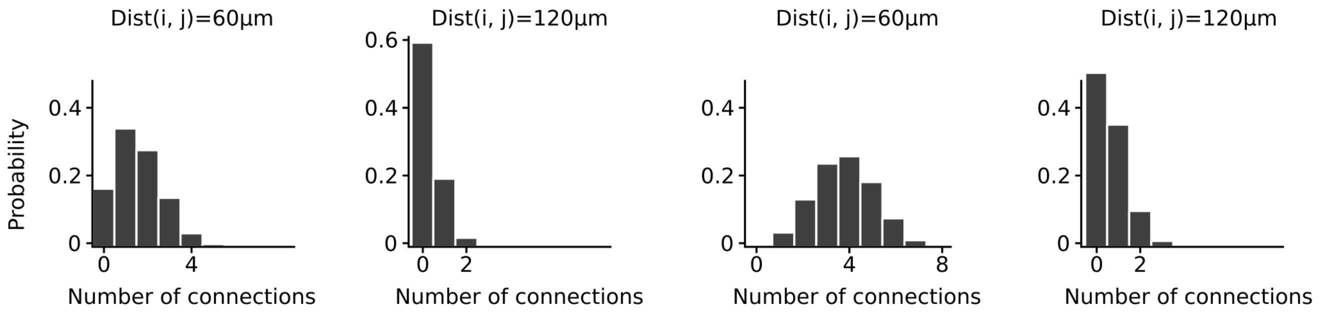


Figure 1: Illustration of the concept of a probabilistic skeleton as generative model for recurrent neural networks. a Connection probabilities between different types of neurons in mouse V1, assuming that the distance between somata can be neglected (copied from [13]). **b** Scaling of connection probabilities with the horizontal distance between the somata for mouse V1 (copied from [13]). **c** Sample base connection probability table and prevalences of neuron types of a probabilistic skeleton for the case of $K = 6$ neuron types (the length of the prevalence-bar encodes the number of neurons in a minicolumn). White grid cells indicate a connection probability 0. **d** Functional forms of the distance dependent scaling functions, with different values of σ^2 in equation (1), that turned out to work well for the induction of innate computing capabilities. **e** Sketch of a cortical minicolumn according [9], showing somata of neurons on different layers and vertically running dendrites. **f** Topdown view of our assumed simple arrangement of minicolumns, each containing the same combination of neurons of different types. Horizontal displacements of neurons within a minicolumn were made here only to facilitate the illustration, our model assumes that they are all located at the center of their minicolumn. **g** Cartoon of the resulting architecture of a neural network sample from a probabilistic skeleton, showing unidirectional connectivity from three input neuron types, to two output neuron types, as well as unidirectional or reciprocal connections between neurons from excitatory or inhibitory recurrent neuron types. **h** Examples for resulting binomial distributions from which the number of synaptic connections are drawn for concrete neurons i, j of types I, J for the case $p_{I\to J} = 0.35$ (two panels on the left) and $p_{I\to J} = 0.85$ (two panels on the right), each for two different values of $\mathrm{Dist}(i, j)$.

In order to answer the question whether these genetically encoded features of the structure of neural networks in the neocortex is sufficient for endowing them with innate computing capabilities, we abstracted these statistical features of neural networks into a mathematical model, which we call a probabilistic skeleton. A probabilistic skeleton consists of

(i) A natural number K (the number of neuron types in the model; we set $K = 6$ in the illustrations of the model in Fig. 1c - f)

(ii) Base connection probabilities $p_{I\to J}$ for neurons of type I to neurons of type J, for the case that they are located within the same minicolumn (see upper part of Fig. 1c for a sample table of such base connection probabilities).

(iii) The prevalence of each neuron type, i.e., the number of neurons of each type in a generic minicolumn, see the bottom plot of Fig. 1c.

(iv) The common weight w_{in} of all synapses from input neurons, as well as the common weight w_E of all synapses from excitatory and the common weight w_I of all synapses from inhibitory neurons in the recurrent network.

(v) A scaling parameter σ^2 that controls the decay of connection probabilities with the horizontal distance between somata.

A probabilistic skeleton is a generative model, which produces a distribution over neural networks that share common structural features, rather than a specific neural network. It neither specifies the number N of neurons in the network, nor which synaptic connections should be present. It just specifies base connection probabilities between neurons according to their types, not between individual neurons. One samples a neural network from a probabilistic skeleton according to the following rules:

1. Pick a number N, the total number of neurons in the network. It should be a multiple of the desired number of minicolumns in the network (since each minicolumn has the same number of neurons). .

2. Draw S times for any pair (i, j) of neurons with i of type I and j of type J from the binomial distribution with probability

$$\mathbb{P}[\text{Synapse from i to j}] = p_{I\to J}\, e^{-\frac{\text{Dist}(i,j)^2}{\sigma^2}} \quad . \tag{1}$$

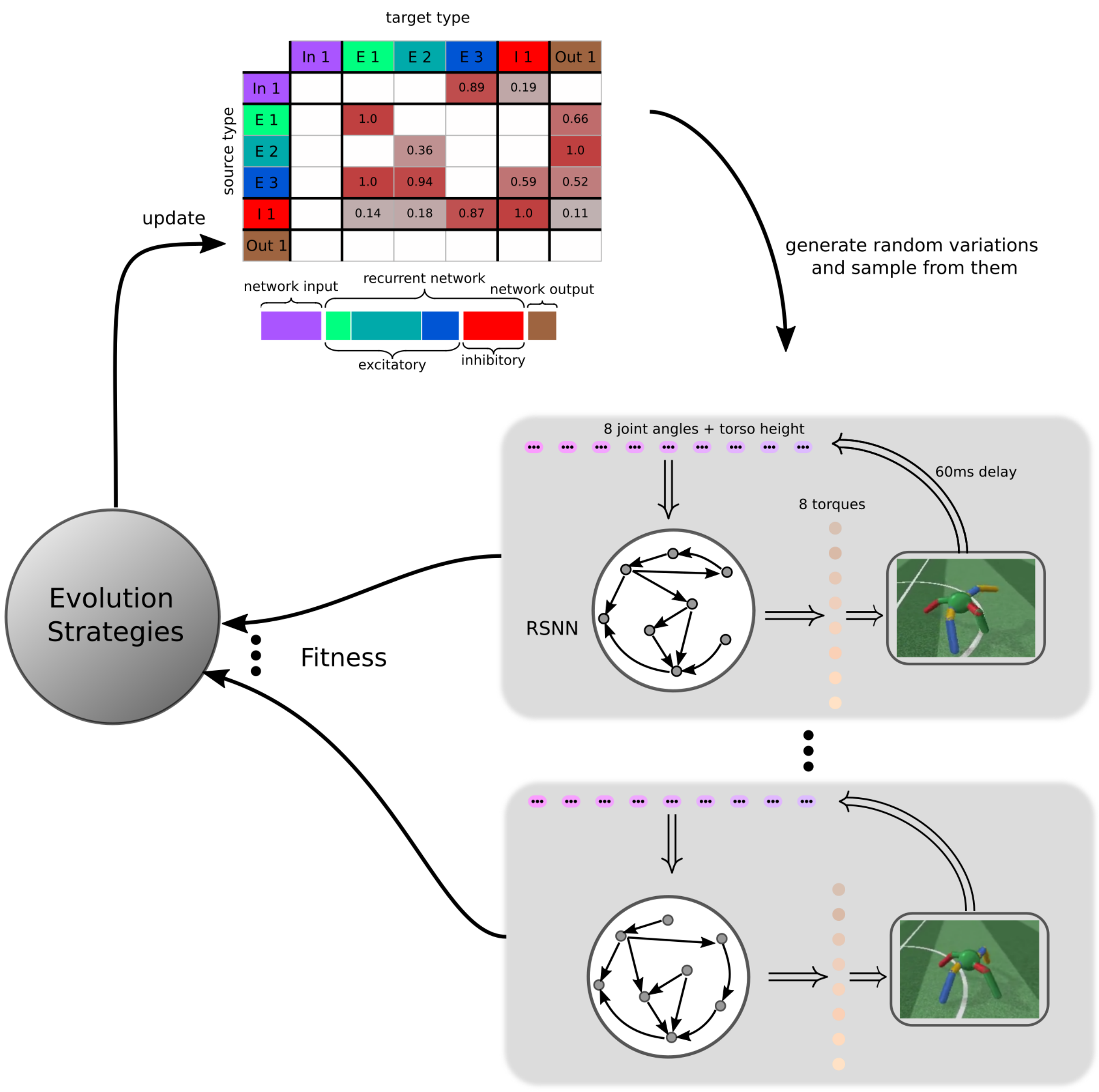
The functional form of the dependence of connection probabilities on $\text{Dist}(i, j)$ approximates the corresponding data from [13], see panels b and d in Fig. 1. The exponential decay of this function entails that the expected total wire length of axons from a neuron is bounded by a constant that does not depend on the network size.

We have set $S = 8$ in all our experiments, thereby allowing up to 8 synaptic connections between any pair of neurons. According to Fig. 7A in [12] most synaptically connected neurons do in fact have multiple synaptic connections. An example for concrete distributions of the number of synaptic connections between two neurons in dependence on the base connection probabilities of their neuron types and the distance between their somata is shown in Fig. 1h. The resulting weight $w_{ij} \in \mathbb{R}$ of a synaptic connection from neuron i to neuron j is then the product of the general scaling parameter w_{in}, w_E, or w_I, that depends on the type of neuron i, and the number of synaptic connections from i to j that results from drawing $S = 8$ times from the distribution given in equ. (1). A sketch of the overall architecture of recurrent neural network samples from a probabilistic skeleton is given in Fig. 1g.

We refer in the following to a computing capability of neural networks as being "innate" if a probabilistic skeleton can endow all, or at least most of its neural network samples (for a given number N of neurons) with this computing capability.

2.2 Algorithmic approach for optimizing probabilistic skeletons

We use the iterative method indicated in Fig. 2 in order to test whether a desired computing capability can be encoded on the statistical level of neural networks with probabilistic skeletons. We use evolution strategies [17] as optimization method, rather than backpropagation through time (BPTT), because the salient fitness function of a probabilistic skeleton is not differentiable. Evolution strategies combine elements of stochastic search with empirical estimates of gradients of the fitness function.



	In 1	E 1	E 2	E 3	I 1	Out 1
In 1				0.89	0.19	
E 1		1.0				0.66
E 2			0.36			1.0
E 3		1.0	0.94		0.59	0.52
I 1		0.14	0.18	0.87	1.0	0.11
Out 1						

Figure 2: Illustration of our algorithmic approach for optimizing a probabilistic skeleton for a computational task. The motor control task of Figure 6 is used as an example. RSNNs are sampled from the current probabilistic skeleton, and their innate computing capability on a specific task, i.e., their fitness, is measured. Evolution strategies modify the probabilistic skeleton on the basis of these fitness values. Then the loop is iterated.

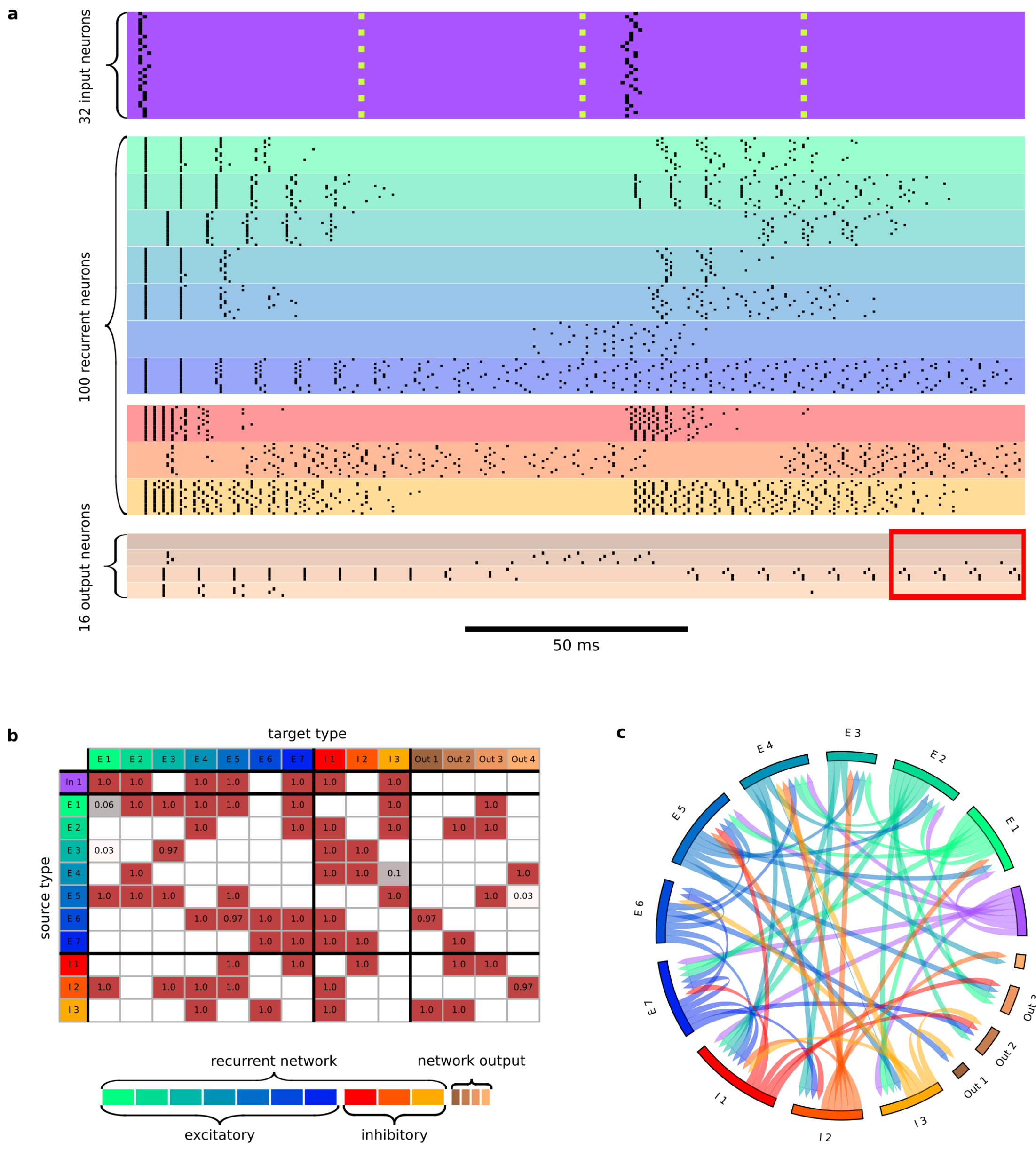
We explore in this article only whether probabilistic skeletons are able to induce non-trivial computing capabilities in recurrent networks of spiking neurons (RSNNs). Our rationale for focusing on recurrent neural networks is that virtually all neural networks in brains are recurrently connected. Additionally, structural properties of artificial recurrent neural networks are less understood than feedforward networks. Furthermore, we focus on networks of spiking neurons because they are better suited than non-spiking neuron models to carry out brain-like computations with spikes. In contrast to computations in recurrent networks of artificial neurons, these computations are not clocked but event-based: A spike represents an event in space and time, and time can be used as additional resource for representing and computational processing of salient information. Besides spikes, also a partition of neurons into excitatory and inhibitory neuron types is essential for relating activity in resulting neural network models to recordings from neurons in the brain.

2.3 Generic computing capabilities of cortical microcircuits on spike times.

One of the most fascinating open problems in neuroscience is how the structure of rather stereotypical, largely translation- and rotation- invariant connectivity templates of laminar cortical microcircuits are able to provide the basis for the astounding computing capabilities of the neocortex [9], [10]. [11] proposed that the laminar spatial arrangement of different types of neurons is less essential for network function than the genetically encoded organization of synaptic connections between them, as well as the relations of neuron types to inputs from other brain areas and outputs to other areas. These structural features are captured by a probabilistic skeleton: It includes connection probabilities from and to external populations of neurons, and supports topographic maps from and to external spatially structured populations of neurons, as well as topographic maps between different types of neurons within the recurrent network.

The primary question that we want to answer is whether the abstract features of a probabilistic skeleton suffice for endowing RSNN samples with generic computing capabilities that are likely to be essential for the function of cortical microcircuits in many different areas of the neocortex. We start with the capability to extract a salient latent variable for computations on spike times: the temporal distance between two waves of spike inputs, see the top row of Fig. 3a. More precisely, we want to install in RSNNs the capability to classify this time difference into four segments of length 50 ms. A particular type of output neurons is supposed to indicate through firing during the last 30 ms of the total 200 ms time span, the class to which the time difference belongs, see bottom right part of Fig. 3a. This relatively large time span of 200 ms is relevant for controlling behaviour, and for regulating bottom-up and top-down processes in cortical networks. It also represents a significant computational challenge for RSNNs because spikes and postsynaptic potentials take place on the shorter time scale of ms and tens of ms. Hence it is rather difficult to induce the computing capability that we are considering here through traditional training of the synaptic weights of a RSNN.

We found that the PS with just 10 types of neurons in the recurrent network, shown in Fig. 3b, can solve this task very well, with a classification accuracy of 96%. The probabilistic skeleton encoded this computing capability with just 164 parameters that we optimized for this task. These represent a tiny fraction of the 33.280 parameters (= number of potential synaptic connections) of the RSNN samples that were considered during this optimization process. Furthermore, the same PS can generate RSNNs of many different sizes, which all can solve this task very well, see Fig. 7e. A closer look at the organization of spiking activity in these RSNNs shows that they rely on persistent firing activity of particular types of neurons. In this way they maintain in their working memory the information whether the second wave of input spikes has already arrived, and if so, within which time interval. A closer look at the network activity for network inputs from all four classes shows that the logic of interactions between the different assemblies of neurons formed by the different neuron types is quite complex. However, the 4 rightmost columns of the plot of the PS in Fig. 3b show that each of the four types of output neurons has a different set of presynaptic neuron types, and the recurrent network manages to activate for each network input the right collection of neuronal assemblies during the trial, which then activate the right type of output neurons during the last 30 ms. Details to all our experiments can be found in the Methods section.



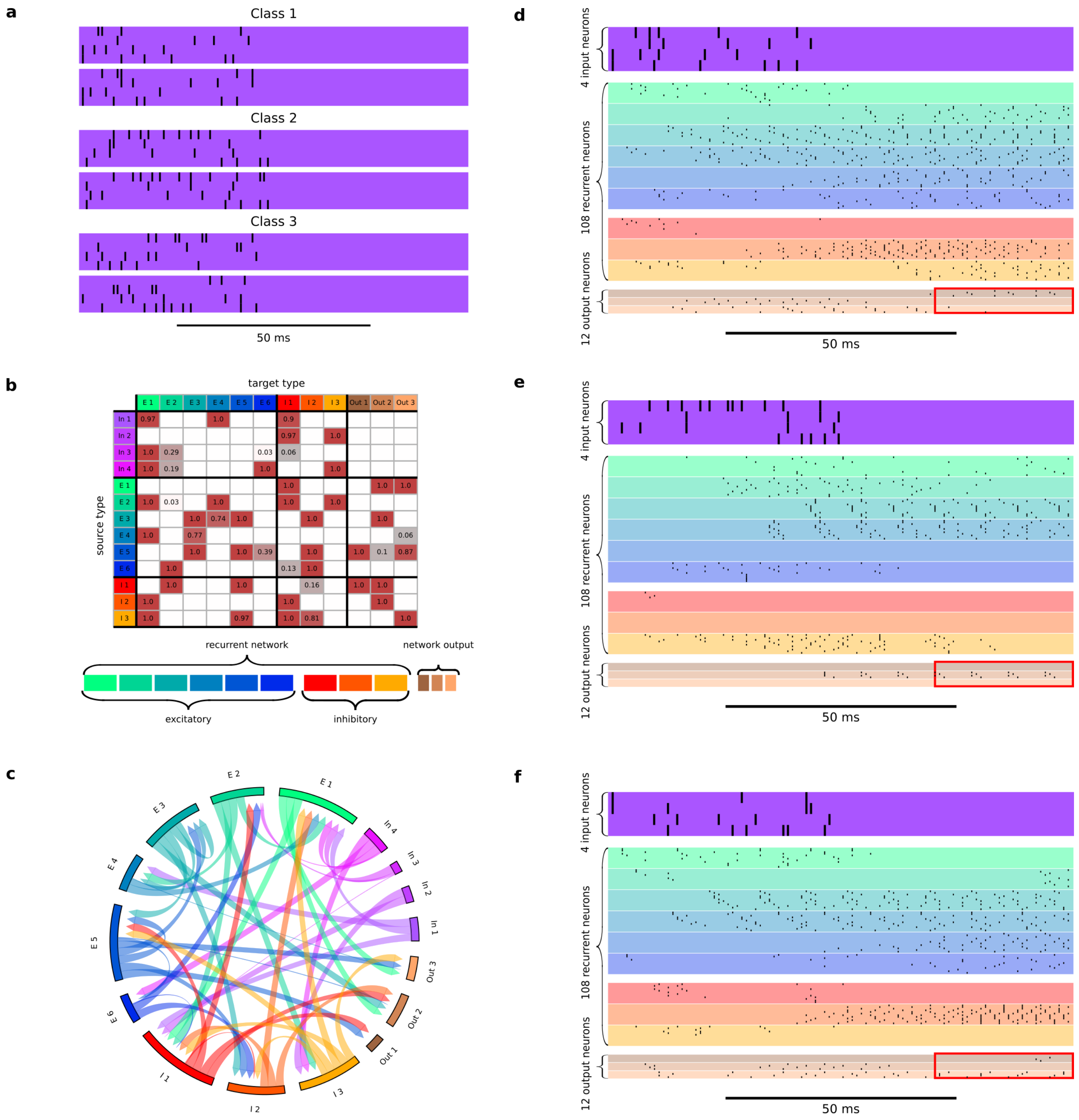
	E 1	E 2	E 3	E 4	E 5	E 6	E 7	I 1	I 2	I 3	Out 1	Out 2	Out 3	Out 4
In 1	1.0	1.0		1.0	1.0		1.0	1.0		1.0				
E 1	0.06	1.0	1.0	1.0	1.0		1.0			1.0			1.0	
E 2				1.0			1.0	1.0		1.0		1.0	1.0	
E 3	0.03		0.97					1.0	1.0					
E 4		1.0						1.0	1.0	0.1				1.0
E 5	1.0	1.0	1.0		1.0					1.0			1.0	0.03
E 6				1.0	0.97	1.0	1.0	1.0			0.97			
E 7						1.0	1.0	1.0	1.0			1.0		
I 1					1.0		1.0		1.0			1.0	1.0	
I 2	1.0		1.0	1.0	1.0			1.0						0.97
I 3				1.0		1.0		1.0			1.0	1.0		

Figure 3: Induction of innate computing capability on spike times through a probabilistic skeleton. a The task is to classify the temporal distance between two waves of spike inputs into 4 classes of inter-spike intervals, each of length 50 ms, that are indicated by dotted lines in the top row. The network is supposed to produce the classification of the inter-spike distance through higher firing of a corresponding type of output neurons, in this sample trial for class 3. Firing activity is shown for all neuron types of the probabilistic skeleton indicated in (**b**). Emergent sequential activation of neuron assemblies is visible for recurrently connected neuron types. **b** Optimized probabilistic skeleton for this task. **c** The connectivity graph induced by the probabilistic skeleton shown in (**b**), plotted in the same style as corresponding experimental data for cortical microcircuits in Fig. 7C of [12]. Thickness of ribbons is proportional to the number of synaptic connections; the ends of incoming ribbons to a neuron type indicate distribution over source types

2.4 Innate pattern classification capability

Distinguishing poisonous from harmless and nutritious food sources is a computing capabilities that needs to be largely innate, because learning this classification through trial and error is too costly for the survival of an individual. But there also exists experimental evidence that innate computing capabilities are involved in visual perception, e.g. in face perception [8]. Inputs from olfactory sensory neurons (see e.g., Fig. 5 of [18]), as well as inputs from peripheral neurons for other sensory modalities, arrive in the form of spatio-temporal spike patterns. Hence we wondered whether a PS can endow RSNNs with the innate capability to distinguish variations of specific spatio-temporal spike patterns, such as the ones shown for classes 1 and 2 in Fig. 4a, from generic spike input streams with the same firing rates, such as the samples of class 3 in Fig. 4a. The spike pattern templates that we used for classes 1 and 2 were frozen Poisson spike trains. Samples from the corresponding classes were generated by adding, deleting, and shifting spikes in time in these spike pattern templates. Spike patterns of class 3 were freshly generated with the same Poisson firing rates as the spike templates for classes 1 and 2.

We found that a PS with 9 neuron types in the recurrent network can solve this task with 91% accuracy. This shows that the capability to distinguish particular spike input patterns from generic spike trains with the same firing rates can be genetically encoded through connection probabilities between neuron types. Furthermore Fig. 4c, d, f show again an emergent orchestration of assembly activations in order to carry out this classification, and to bridge the delay to the decision time during the last 30 ms. The probabilistic skeleton for this task had 157 parameters. In contrast, a RSNN of the size of the RSNN samples that were considered during the optimization of the probalistic skeleton had 13,392 potential synaptic connections. Furthermore Fig. 7e shows that the probabilistic skeleton endows also much larger RSNNs, with many more parameters, with the same computing capability.



	E 1	E 2	E 3	E 4	E 5	E 6	I 1	I 2	I 3	Out 1	Out 2	Out 3
In 1	0.97			1.0			0.9					
In 2							0.97		1.0			
In 3	1.0	0.29				0.03	0.06					
In 4	1.0	0.19				1.0			1.0			
E 1							1.0				1.0	1.0
E 2	1.0	0.03		1.0			1.0		1.0			
E 3			1.0	0.74	1.0			1.0			1.0	
E 4	1.0		0.77									0.06
E 5			1.0		1.0	0.39		1.0		1.0	0.1	0.87
E 6		1.0					0.13	1.0				
I 1		1.0			1.0			0.16		1.0	1.0	
I 2	1.0						1.0				1.0	
I 3	1.0				0.97		1.0	0.81				1.0

Figure 4: Innate spike pattern classification. **a** Two samples from each of the three classes of spike input patterns. **b** Optimized probabilistic skeleton for this task. **c** The connectivity graph induced by this probabilistic skeleton, plotted in the same style as in Fig. 3c. **d-f** Firing activity is shown for all neuron types for samples of spike input patterns from classes 1-3. The 30 ms time window during which the network output is extracted is indicated by the red frame at the bottom of each spike raster.

2.5 Generic 2D computing capabilities.

The neocortex forms a 2D sheet, with topographic maps between cortical areas and from peripheral sensory neurons. Therefore we wondered whether basic 2D computational operations could be programmed into this 2D sheet through probabilistic skeletons. Since noise surpression and contrast enhancement are among the most basic ones, we first tested whether a probabilistic skeleton can endow RSNNs with the capability to extract just the local maxima from a 2D input patterns, thereby filtering out background noise[9] , see Fig. 5c, d for two examples of such computations. It turns out that this computational capability can already be provided by a quite simple probabilistic skeletons with just 5 recurrent types of neurons, see Fig. 5a for the spike raster of an RSNN sample that receives the 2D pattern shown at the top of panel c as input, and produces in its output neurons the output shown at the bottom of panel c. Panel d depicts the output of this RSNN for another 2D input pattern. The grey values of these 12 x 12 arrays were encoded through Poisson firing rates. A similar input/output mapping can be produced by Mexican hat filters or Winner-Take-All circuits. But these are hard to execute by RSNNs in a stable manner, in particular in a way where it applies equally well to all parts of a larger 2D input array. One problem is that strong lateral inhibition among excitatory neurons tends to drive down also the firing rate of the ”winner”. The phasic firing pattern of the output neurons in Fig. 5a suggests that the network exploits the option to implement the local maximum extraction through an iterated temporal competition, where the strongest activated neurons fire first, thereby inhibiting weaker competitors -but also temporary themselves. The probabilistic skeleton that programmed this computation into the network had 44 parameters, whereas the shown RSNN sample with 1008 neurons had 725,760 potential synaptic connections.

A further important 2D computational operation that is carried out by cortical microcircuits is the detection of coincidences between different 2D input streams. These could come from different sensory areas, or from higher and lower cortical areas for the same sensory modality [9], see the cartoon in Fig. 5e and f. In particular, common models for the interaction of higher and lower visual areas propose that topdown predictions of sensory inputs are compared with bottom-up sensory inputs in generic cortical microcircuits. Fig. 5 g-i shows that also this fundamental 2D computation can be induced through a probabilistic skeleton. Fig. 5g shows the computation of an RSNN sample on the two 2D input patterns shown at the top of panel h into the network output shown below them. The target output used for optimizing the probabilistic skeleton. Panel i shows another example of a pair of 2D input patterns and the 2D output of this RSNN. This probabilistic skeleton had 121 parameters, and had been optimized, like the one for local maximum extraction, just for 5x5 input patterns.

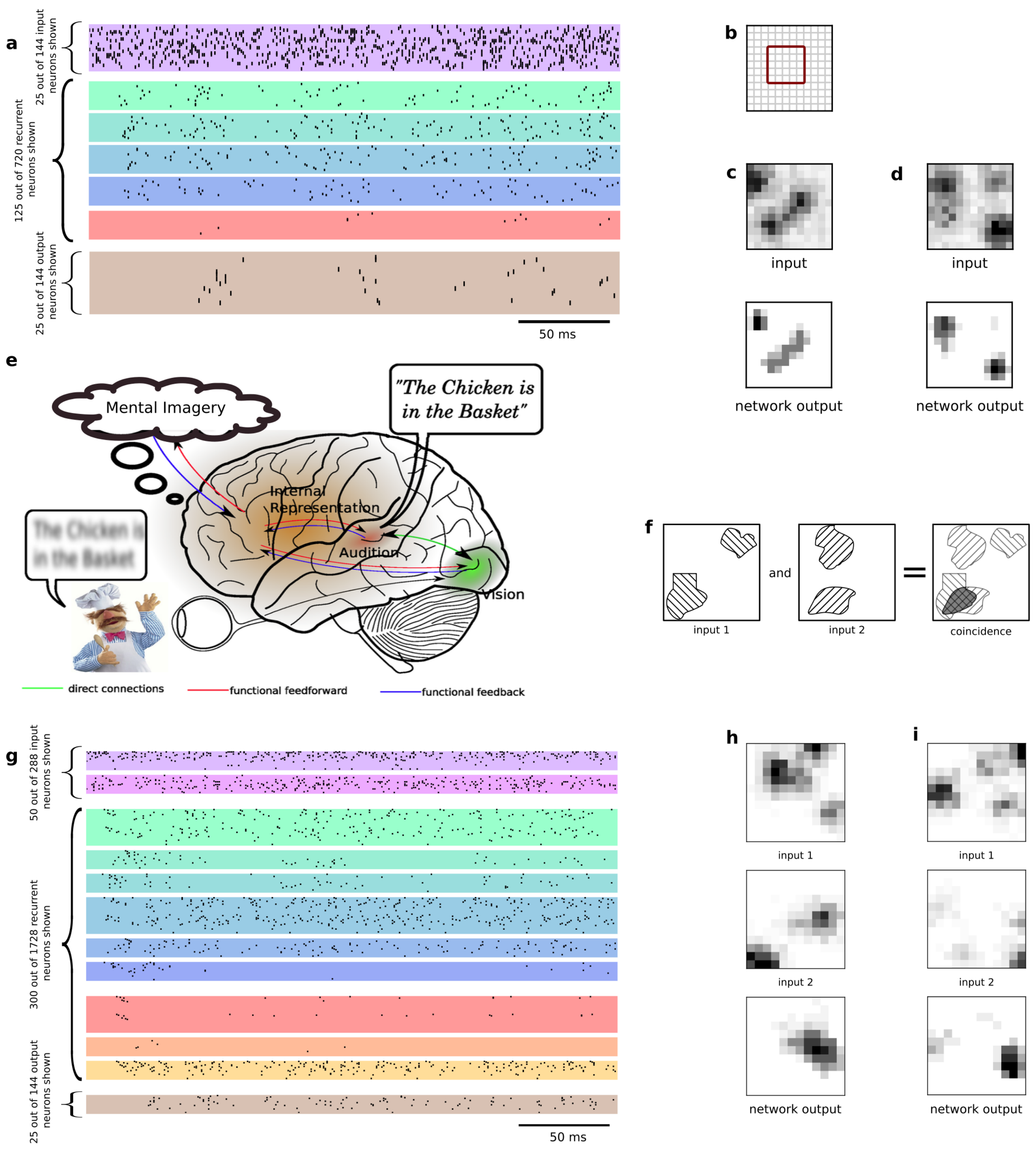

a
25 out of 144 input neurons shown
125 out of 720 recurrent neurons shown
25 out of 144 output neurons shown
50 ms
b
c
input
network output
d
input
network output
e
Mental Imagery
"The Chicken is in the Basket"
Internal Representation
Audition
Vision
The Chicken is in the Basket
direct connections
functional feedforward
functional feedback
f
input 1
and
input 2
=
coincidence
g
50 out of 288 input neurons shown
300 out of 1728 recurrent neurons shown
25 out of 144 output neurons shown
50 ms
h
input 1
input 2
network output
i
input 1
input 2
network output

Figure 5: 2D computing capabilities: Maximum extraction and coincidence detection. **a, c** Spike raster plot of an RSNN sample solving the local maximum extraction task, for the 2D input pattern shown at the top of panel c, producing the 2D output shown at the bottom of panel c. Grey values were encoded through Poisson firing rates. **b** The red box in the 12x12 grid indicates the 5x5 size of the smaller grid for which the probabilistic skeleton had been optimized. **d** A different 2D input and output pattern of the same RSNN. **e** Motivation for the coincidence detection task in the context of a proposed enhancement of perception through interaction of inputs from different sensory modalities and topdown predictions; figure adapted from [19]. **f** Illustration of the calculation of the target 2D output pattern, shown as dark area that consist of the intersection of the preceding two 2D input patterns. **g** Spike raster of an RSNN sample of the probabilistic skeleton for solving the 2D coincidence detection task. **h** Pairs of 2D input patterns, for which this RSNN produced the 2D output pattern shown below. **i** Another sample of pairs of 2D input patterns and the 2D network output of the same RSNN.

2.6 A probabilistic skeleton can endow neural networks with innate motor control capability.

Basic motor control capability right after birth, for example the capability to stand up and walk on 4 legs, is essential for survival in many species. We tested the capability of probabilistic skeletons to endow neural networks with such capability on a common benchmark task for motor control: Enabling a simple quadruped model ("ant") to walk by controlling the the 8 joints of its 4 legs through suitable torques. The neural network controller received 9 spike input streams that encoded -with a delay of 60 ms- through population coding 9 dynamically varying variables: The angles of the 8 joints as well as the torso height, see Fig. 6a. Hence a correspondingly large network is needed to extract salient information from these 9 input time series and to produce the 8 output time series. Nevertheless we found that a probabilistic skeleton with 15 types of neurons in the recurrent network, specified by just 635 parameters, see Fig. 6b, contains already the control strategy that this quadruped needs for locomotion. Compared with an RSNN sample from this PS with 250 neurons in the recurrent part of the network, which has $114,500$ potential synaptic connections, the PS could compress its control strategy into 0.55% of the parameters that are tuned if one train this RSNN, consisting of 458 neurons, in the traditional manner for this task. Fig. 5c shows that the time varying spatial patterns of the 8 input variables cause diverse time varying activity patterns within many of the recurrent neuron types. Fig. 6 d shows a sample of the 9 time-varying network inputs and the resulting 8 time-varying network outputs on a larger time scale. Here and in (Movie of ant) one can see that this task requires the RSNN controller to produce in its 8 output neuron types quite complex time-varying outputs.

After randomly deleting 30 percent of the recurrent and output neurons of this RSNN, the network can still enable the ant model to walk, although somewhat slower, see (Movie of ant after 30% deletion).

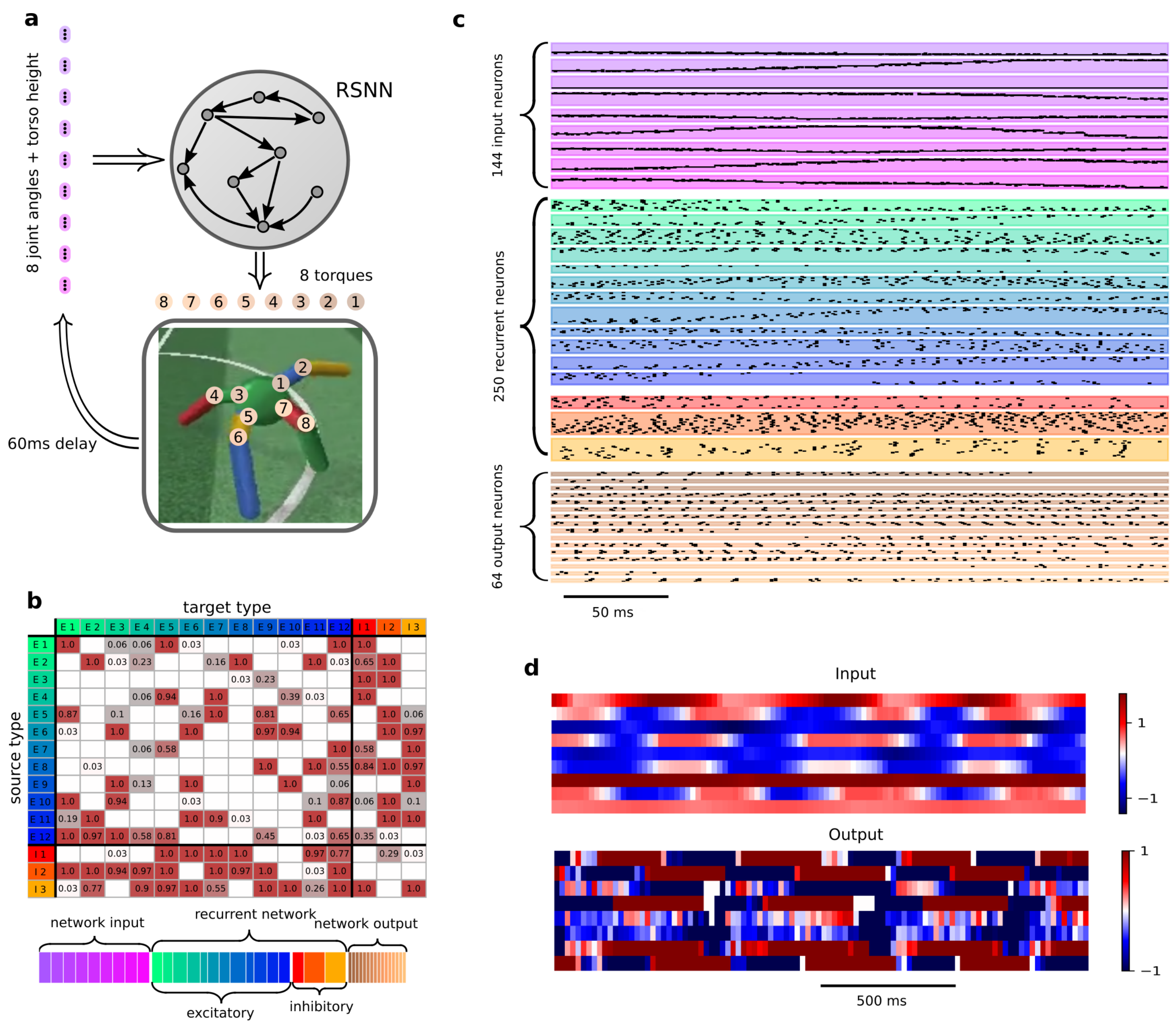


Figure 6: Example for innate motor control capability through a probabilistic skeleton. **a** System architecture, indicating network inputs and outputs, as well as the 8 joints that are controlled by the network outputs. **b** Probabilistic skeleton for solving this motor control task. **c** Spike raster of an RSNN sample with 458 neurons drawn from this probabilistic skeleton. The required spatial organization of network outputs emerges through population coding of 9 input variables. **d** Sample dynamics of input and output variables of the RSNN controller on a larger time scale.

2.7 General results and principles for designing neural networks through probabilistic skeletons.

The number K of neuron types is a new complexity measure for RSNNs that emerges from our generative approach. We show in Fig. 7a that the mean performance of RSNNs samples from probabilistic skeletons increases only slightly with K for the tasks that we have considered (we did not carry out the control experiments for the motor control task because this was computationally substantially more demanding). But we also found that the performance reached via Evolution Strategies by probabilistic skeletons varied widely, even for repeated runs with the same K.

A larger number of neuron types, such as the 112 types considered in [13], could also have the purpose to implement different innate computing capabilities through different sets of neuron types. Multiplexing of computations in cortical microcircuits, possibly using different output types for transmitting results of different computations to diverse downstream networks [20], appears to be a characteristic feature of cortical microcircuits.

A key property of RSNNs that are generated from a probabilistic skeleton is that the connection probability between neurons i and j decays exponentially as function of their horizontal distance $\text{Dist}(i, j)$. Hence, since the density of neurons is bounded through the diameter of a minicolumn, the expected number of synaptic connections from a neuron, as well as the total length of axons from a neuron, are bounded by constants that do not grow with the number of neurons in the network. This feature induces a substantially more rigorous type of sparsity of connections than usually considered, where sparsity is expressed as fraction of the the quadratic number of all synapses in a fully connected recurrent network. In addition, we show in Fig. 7b, c that the performance of RSNN samples from a probabilistic skeleton decays gracefully when synaptic connections with the smallest connection probability are successively removed. Hence this process provides a principled method for further decreasing the number of synapses and total wire length if this becomes relevant. One possible application of this is to fine-tune the performance of a RSNN by replacing its weakest synaptic connections by other connections, a process that is observed in biology in the form of spine motility on the time-scale of hours and days. This occurs spontaneously [21] and even more during learning [22].

We show in Fig. 7d that the performance of RSNN samples from probabilistic skeletons is quite robust with regard to perturbations of the strength of synaptic connections. This feature appears to be important for biological neural networks because underlying molecular processes appear to modify the strengths of synaptic connections even in the absence of other learning processes [21]. Note that robustness against fluctuations of synaptic strengths is also desirable for neuromorphic implementations of RSNNs where synaptic weights are implemented through somewhat volatile but highly energy-efficient memristors or other new devices [23] On a more general level, robustness against weight perturbations can be viewed as a generic implication of the genomic bottleneck [1], where computational properties are encoded in a much lower dimensional parameter space.

Finally, Fig. 7e shows that the computing capabilities that probabilistic skeletons provide generalize very well to RSNN samples that are substantially larger than the ones considered during the optimization of the probabilistic skeleton. This feature highlights another fundamental difference to the traditional method to generate functional neural networks through training, where transition to larger input dimensions usually requires retraining of the network from scratch.

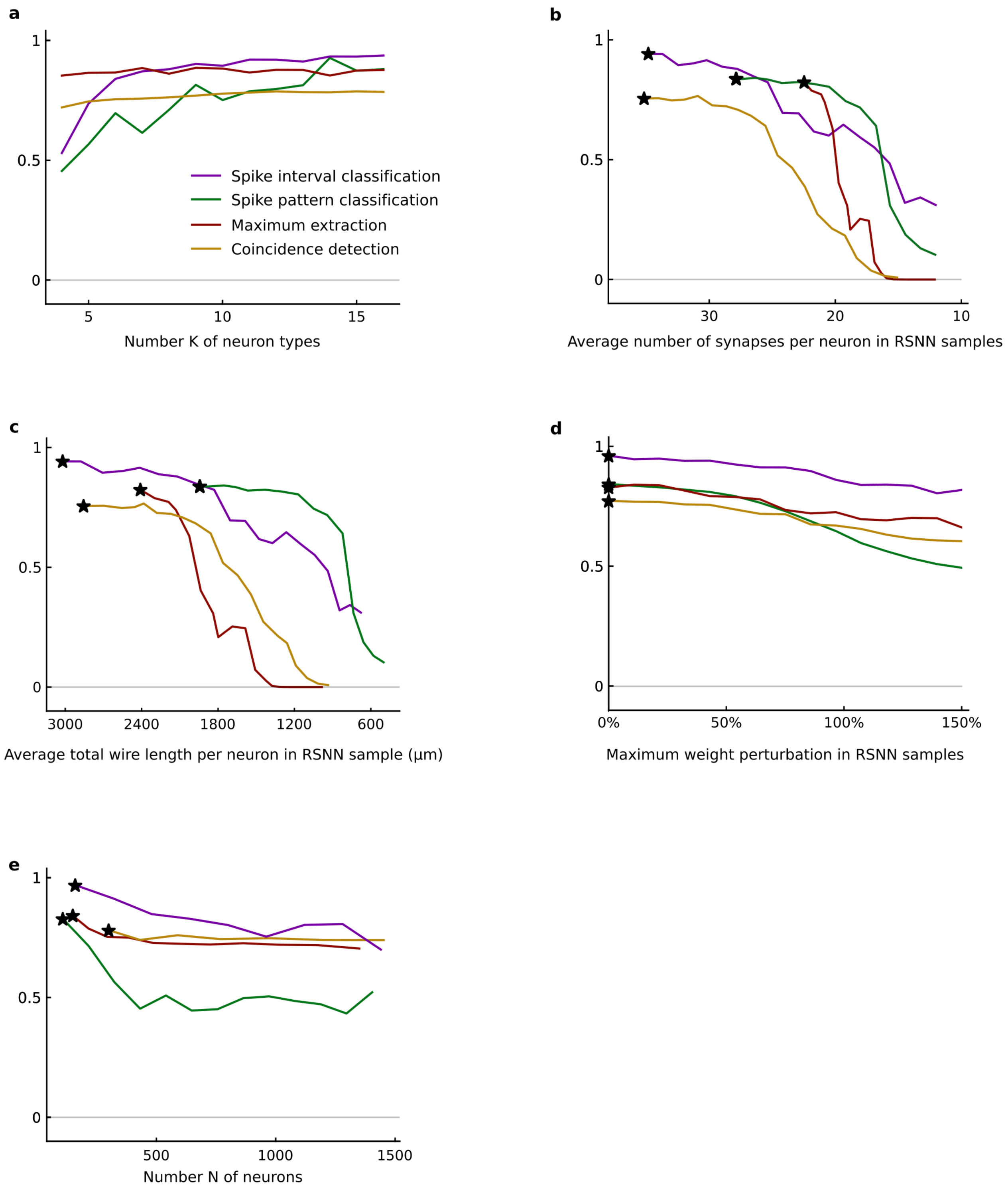


Figure 7: General functional properties of RSNN samples from probabilistic skeletons. Performance of RSNN is measured relative to random guessing as a common baseline for all tasks, see 4 for details. **a** Mean performance values achieved by optimizing probabilistic skeletons with different numbers K of neuron types. **b** Degradation of innate computing capabilities through pruning of synapses (elimination of synapses with the smallest connection probabilities). The x-axis depicts the average number of synapses per neuron. The star corresponds to the performance without pruning. **c** Same data as in b, but plotted here as function of the average total wire length of axons from a neuron. The star corresponds to the performance without pruning. **d** Performance after random perturbation of synaptic connections, plotted as function of the maximal amount of change in the number of synaptic connections for each neuron pair, expressed as fraction of the current number of synaptic connections. **e** Generalization capability of probabilistic skeletons to RSNN samples with different neuron numbers N. The value of N that was considered during the optimization of the probabilistic skeleton is marked by a star.

3 Discussion

Neural networks have become a central tool and concept both for understanding brain function, and for reproducing intelligence in artificial devices. But it has long been ignored that our standard procedure for bringing function into neural networks is fundamentally different from the way how computational capabilities emerge in biological neural networks [1]. Rather than being trained from a tabula rasa initial state with large number of data, often resorting to supervising learning methods that rely on a teacher, it is well known that computational function emerges in neural networks through a combination of nature and nurture, with the contribution of each depending on the species. Even the mammalian neocortex, which exhibits astounding learning capabilities, uses highly structured canonical neural microcircuits for learning, that are known to provide substantial innate computing capabilities [2], [4], [5], [6], [7], [8].

We have examined one essential aspect of the genetically encoded structure of cortical microcircuits, connection probabilities between genetically different types of neurons, and formulated a simple mathematical model for that, probabilistic skeletons. One may view probabilistic skeletons as a fragment of the programming language that nature uses for endowing neural networks of the brain with innate computing capabilities, since they represent an abstraction of a very large body of experimental data on the structure of cortical microcircuits, see [13] for a recent summary. We have shown that this fragment of the genetic programming language is surprisingly powerful: It is able to install innate computing capabilities on spike times (Fig. 3), temporal spike patterns (Fig. 4), 2D spike patterns (Fig. 5), and multiple dynamically changing spike input streams (Fig. 6) in recurrent networks of spiking neurons. Interestingly, the resulting functional recurrent neural networks differ in essential aspects from those that are commonly studied: Their number of neurons and total wire length scales linearly with the number of neurons. Total wire length is obviously an important constraint for the design of cortical microcircuits, where already a single cubic mm of grey matter is estimated to contain 4km of axonal fibers [24].

Such linear scaling of the number of synapses and the total wire length is enforced by a generic exponential decay of connection probabilities with distance in cortical microcircuits [13], which is reproduced by probabilistic skeletons. The resulting connectivity structure encourages topographic maps between 2D populations of neurons, a well-known construction principle of neural circuits in the brain. It induces a computational machinery that is exquisitely suited for computing on 2D input streams from visual and somatosensory peripheral neuron arrays, as well as from other cortical areas, including higher cortical areas that control for example spatial attention. A further new feature of resulting neural networks is their robustness to random changes of synaptic connections between neurons, see Fig. 7. This feature appears to be essential for neural networks of the brain, where a substantial fraction of synaptic connections on spines come and go on the time scale of hours [25]. It also points to an intrinsic difference to the way how computational function is commonly installed in artificial neural networks, whose brittleness with regard to minor network perturbations has frequently been criticized [26].

Probabilistic skeletons can be viewed as a way of meeting the challenge of [1] to explore the functional impact of the "genomic bottleneck", i.e., the fact that the number of bits which the genome uses for encoding brains is quite small compared with the number of synaptic connections in the brain, and actually also if compared with the number of synaptic weights in state-of-the-art deep neural networks. Alternative approaches for meeting this challenge have recently been proposed [27], [28]. [28] models the impact of genomic information compression on functional capabilites of neural networks on a more abstract level. On the other hand, their model is less general insofar as it assumes that connection probabilities, or in this case deterministic connections, can be computed from binary codes for neurons through linear operations. The model of [27] is less abstract than ours and that of [28]. It is less general because each genetic neuron type is assumed to consist of just one neuron, and synapse formation is assumed to be deterministic, and controlled by linear operations. Benefits of these two alternative approaches have been demonstrated for feedforward artificial neural network computations on static inputs, exhibiting intriguing consequences of the underlying parameter compression. [29] considered already previously evolution of network architectures with uniform weight values. But they did not aim at modelling the impact of the "genomic bottleneck" since the existence of a synaptic connection was optimized individually for each pair of neurons. Several other related methods have been considered in the context of neuroevolution [30].

One advantage of the probabilistic skeleton approach is that this model can be related directly to experimental data on the structure of cortical microcircuits, and to recordings from recurrent networks of spiking neurons in the brain. In fact, this approach suggests that for understanding innate computing capabilities of cortical microcircuits it will be fruitful to investigate the computational interaction between genetically different neuron types, for example through Ca-imaging. It also suggests that it will be important for understanding the organization of computations in cortical microcircuits to determine connection probabilities between substantially more types of neurons than the 17 types that were addressed in Fig. 4 of [13]), reproduced in Fig. 1a. Our scaling analysis from Fig. 7a suggests that the number and precision of innate computations can be enhanced by using more neuron types with genetically programmed connection probabilities. It will be very interesting to see whether evolution has pursued this direction.

Another interesting next step will be to elucidate additional benefits of designing neural network via probabilistic skeletons if one takes into account that neurons of different types have according to [12] and [13] different electrophysiological and morphological properties. It turned out that offering probabilistic skeletons the opportunity to use for different neuron types different GLIF$_3$ neuron models from [13] did not provide functional advantages for the computational tasks that we have considered. A further interesting next step will be to take into account that the neocortex also has a substantial number of long range connections. An investigation of the trade off between the resulting increase of total wire length and the benefits of having network modules with different innate competences is likely to improve our understanding of global brain architectures from the computational perspective.

Further facets of the genetic programming language for the generation of neural networks with innate computing and learning capabilities will become apparent if one also takes into account that the short term [31] and long term dynamics [32], [33]) of synapses is highly diverse. Hence it may be fruitful to add diverse synapse types to the fragment of the genetic programming language that is formalized through probabilistic skeletons.

Both the number of synapses and total wire length are critical factors in the cost and efficiency of neuromorphic hardware. Using probabilistic skeletons for the design of neuromorphic hardware, where both measures grow just linearly with the number of neurons in resulting RSNNs, is therefore likely to provide new methods for efficient circuit design. In particular, 2D arrays of neuromorphic sensors can in this way be combined with hardware for computational processing so that not only time, but also space is represented by itself, through topographic maps between neuron types. Additionally, the resulting inherent robustness against weight perturbations supports implemention of synaptic connections by extremely energy-efficient but imprecise memristors [34]. Another possible technological application of the methods presented in this article lies in the area of organoids [35]. They imply that, as soon as one will be able to modify those sections of the genetic code of neurons that control their connection probabilities to other neuron types, one will be able to grow cerebral organoids with powerful innate computing capabilities.

Convolutional neural networks (CNNs) represent one of the practically most successful feedforward artificial neural network architecture in terms of training capability. Like probabilistic skeletons, these architectures have a reduced number of parameters and also employ a simple spatially iterated connection pattern. Hence it is tempting to ask whether probabilistic skeletons could provide analogous benefits for the design of recurrent networks of artificial neurons.

4 Methods

Details to recurrent network of spiking neurons (RSNNs) models

Neuron types

Neuron types fall into three categories: input types, recurrent types, and output types. The neurons from these three categories are referred to as input neurons, recurrent neurons, and output neurons.

Input neurons provide external inputs in the form of spike trains. They have no internal states, and there are no recurrent connections from recurrent neurons or output neurons back to the input neurons. The output neurons receive their input from the recurrent neurons (see Fig. 1g).

Recurrent neurons can have connections from input neurons and other recurrent neurons. All neurons from the same type can be either excitatory or inhibitory. Note that input or output types will always consist of excitatory neurons.

Discrete time neuron model

Recurrent and output neurons are modelled as discrete-time versions of standard Leaky-Integrate-and-Fire (LIF) neuron models. More precisely the GLIF$_1$ model from [36] has been used. Control experiments with the GLIF$_3$ model from [13] produced qualitatively similar results.

For a given neuron $j \in \{1, \dots, N\}$ of type J the membrane potential is denoted by V_j and the input current by I_j. We assume that currents are constant on small intervals $[t, t + \delta t]$, which have been set to a length of 1 ms. The neural dynamics of the model in discrete time can then be given as

$$V_j(t + \delta t) = \begin{cases} \alpha V_j(t) + (1 - \alpha)(E_L + \frac{1}{C_m} I_j(t)) & \text{if } z_j(t) = 0 \\ V_r & \text{else} \end{cases} \tag{2}$$

$$\tag{3}$$

where $\alpha = \exp\left(-\frac{\delta t}{\tau}\right)$ and

$$z_j(t) = H\left(\frac{V_j(t) - v_{th}(t)}{v_{th}(t)}\right) \tag{4}$$

with the Heaviside function $H(x) = \begin{cases} 0 & x < 0 \\ 1 & \text{else} \end{cases}$. Here $\tau \in \mathbb{R}$ is the membrane time constant, $E_L \in \mathbb{R}$ is the resting potential, $C_m \in \mathbb{R}$ is the membrane conductance and v_{th} is the threshold voltage. After spiking the neuron enters a refractory period, lasting $t_{ref} > 0$, in which $z_j(t)$ is fixed to zero.

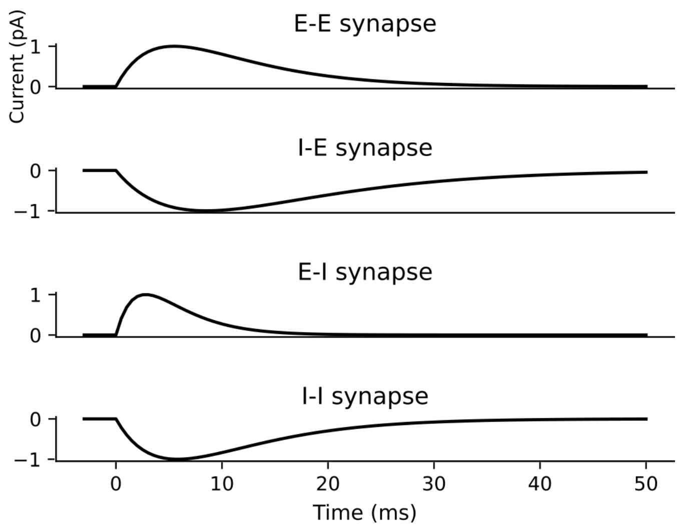


Figure 8: The temporal dynamics of the four different synapse types for $W_{GLIF} = 1$.

Synapse model

The postsynaptic current for a single spike and a single neuron is modeled by the dynamics given in [13] as

$$I_{syn}(t) = \frac{eW_{GLIF}}{\tau_{syn}} t e^{-\frac{t}{\tau_{syn}}}, \quad t \in \mathbb{R}_+, \tag{5}$$

where I_{syn} is the postsynaptic current, τ_{syn} is the synaptic time constant and W_{GLIF} scales the weight of the synaptic connection such that the postsynaptic current has an amplitude of W_{GLIF} after τ_{syn} ms. Note that τ_{syn} depends on the type 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 [13].

The resulting postsynaptic currents can be seen in Figure 8. Using discrete time-steps the equation (5) transforms to

$$I_{syn}(t) = \frac{eW_{GLIF}}{\tau_{syn}} t\, \delta t\, e^{-\frac{t\delta t}{\tau_{syn}}}, \quad t \in \mathbb{N}. \tag{6}$$

Since multiple spikes are possible up to time-step t the current from neuron i to neuron j at timestep t can be written in terms of the synaptic weight w_{ij} defined in section 2.1 as

$$I_{ij}(t) = \sum_{s=0}^{t-1} \frac{ew_{ij} z_i(s)}{\tau_{syn}} (t-s)\, \delta t\, e^{-\frac{(t-s)\delta t}{\tau_{syn}}}. \tag{7}$$

The input current $I_j(t)$ for neuron j at time t is defined as the sum of currents from all neurons:

$$I_j(t) = \sum_i \frac{e w_{ij}}{\tau_{syn}} \sum_{s=0}^{t-1} z_i(s)(t-s)\,\delta t\, e^{-\frac{(t-s)\delta t}{\tau_{syn}}} \tag{8}$$

$$= \sum_{s=0}^{t-1} \frac{e}{\tau_{syn}}(t-s)\,\delta t\, e^{-\frac{(t-s)\delta t}{\tau_{syn}}} \sum_i w_{ij} z_i(s). \tag{9}$$

Parameters of neuron and synapse models

The previously defined neuron- and synapse models use the following set of additional parameters:

$$\mathcal{H} = \{C_m^J, \tau^J, V_r^J, v_{th}^J, t_{ref}^J \mid J = 1, \ldots, K\}$$

. The values for $\{C_m^J, \tau^J, V_r^J, v_{th}^J, t_{ref}^J \mid J = 1, \ldots, K\}$ are taken from [13], and the raw data is available in [37]. A good overview of these neuron types has been made available online in the database of the Allen institute. Detailed biological and modelling data for the prototype of the excitatory neuron can be found at Excitatory neuron and the prototype for the inhibitory neuron at Inhibitory neuron. We have seen no evidence that the exact values of the GLIF$_1$ parameters are essential for the results reported in this paper.

Synapse population and spatial structure

In the first four tasks the neurons were arranged in a 2D array of minicolumns as indicated in Fig. 1f. The minimum distance between neurons in different minicolumns is given by the diameter of a minicolumn, which is 60 µm. Every recurrent neuron type has at least one neuron in each minicolumn, but a type can have more than one neuron in a single minicolumn if its prevalence is high.

Optimization method

The probabilistic skeletons were optimized using the Separable Natural Evolution Strategy (Separable NES), which was first introduced in [17]. The algorithm is available in pseudo code 1. Considering the optimization of a d-dimensional vector of network parameters $\boldsymbol{\theta}$, the algorithm uses a Gaussian distribution in every dimension with means $\boldsymbol{\mu} \in \mathbb{R}^d$ and variances $\boldsymbol{\sigma} \in \mathbb{R}^d$. The basic idea is that one samples λ times from this distributions, then evaluates the fitness values of the so-called offsprings, i.e. the vectors $\boldsymbol{\theta}_j \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{I}\boldsymbol{\sigma})$, and finally updates the distributions to capture more of the parameter space, where the fitness of the offsprings is higher. The fitness function F depends on the task, that the probabilistic skeleton should perform. The mean values of the parameters are initialized by truncated normal random variables with mean zero and variance 1.0 and the variance values are initialized as ones. We found that choosing the learning rate for $\boldsymbol{\mu}$ as $\eta_\mu = 1.0$ yields good results, which is consistent with the suggested value in

[38] and [39]. The learning rate for $\boldsymbol{\sigma}$ was chosen as $\eta_\sigma = 0.01$. As suggested in [39] mirrored sampling has been employed, see, e.g., [40]. That is, for every Gaussian noise vector $\mathbf{s} \in \mathbb{R}^d$ also the offspring, which results from using $-\mathbf{s}$, will be evaluated.

Algorithm 1 Separable NES

```
Require: $\lambda \in \mathbb{N}, \boldsymbol{\mu} \in \mathbb{R}^d, \boldsymbol{\sigma} \in \mathbb{R}^d, \eta_{\boldsymbol{\mu}}, \eta_{\boldsymbol{\sigma}}, F$
Ensure: $\lambda \equiv 0 \bmod 2,\ \eta_{\boldsymbol{\mu}} > 0,\ \eta_{\boldsymbol{\sigma}} > 0$
  for epoch=1,...,N do
    for j=1,...,$\lambda/2$ do
      Init $\mathbf{s} \in \mathbb{R}^{(\lambda,d)}$ as $\mathbf{s}_j \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{s}_{(j+\lambda/2)} = -\mathbf{s}_j$
      $\boldsymbol{\theta}_j \leftarrow \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \mathbf{s}_j$
      Compute Fitness $F(\boldsymbol{\theta}_j)$
    end for
    Compute gradients   $\nabla_{\boldsymbol{\mu}} J \leftarrow \sum_{j=1}^{\lambda} F(\boldsymbol{\theta}_j)\mathbf{s}_j$
                        $\nabla_{\boldsymbol{\sigma}} J \leftarrow \sum_{j=1}^{\lambda} F(\boldsymbol{\theta}_j)(\mathbf{s}_j^{\mathrm{T}}\mathbf{s}_j - 1)$
    Update parameters   $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \eta_{\boldsymbol{\mu}}\boldsymbol{\sigma}\nabla_{\boldsymbol{\mu}} J$
                        $\boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma} \exp\{\frac{\eta_{\boldsymbol{\sigma}}}{2}\nabla_{\boldsymbol{\sigma}} J\}$
  end for
```

Note, that the value of the scaling parameter σ^2 from equation (1) was optimized using a large-scale hyperparameter search and not using evolutionary strategies. For the optimization of the base probabilites, there is the additional constraint that $p_{I\to J} \in [0,1]$. Hence real values $\kappa_{IJ} \in \mathbb{R}$ are optimized and the connection probabilities are obtained by using the sigmoid function:

$$p_{I\to J} = \frac{1}{1 + \mathrm{e}^{-\kappa_{IJ}}}. \tag{10}$$

In the following generic parameter vectors will be denoted by $\boldsymbol{\theta}$.

Experiments

Details to generic computing capabilities of cortical microcircuits on spike times.

Task description: In this task the goal is to classify the time lengths between two spikes. There is a fixed time interval of 200 ms, which is divided into four bins of 50 ms. For each class the first spike occurs at the beginning $t = 0$ and the second spike is uniformly drawn from the four bins, which results in four classes of input spike trains. The precise timing of the second spike is again uniformly sampled within the time interval of the chosen bin.

Input: The network receives as input a spike train from one of the four classes with equal probability. Additionally, for each neuron Gaussian noise with mean zero and variance 0.5 ms has been added to the spike times to avoid that all input neurons spike at the exact same time.

Output: Every class is represented by a corresponding output neuron type ($J_i, i = 1, \ldots, 4$), consisting of four neurons. For each of these populations at every time step t the average spike count is given by

$$s^J(t) := \frac{1}{|J|} \sum_{j \text{ of type } J} z_j(t). \tag{11}$$

These spike counts can be used to derive a spike rate of the population, where just the last 30 ms of the simulation are taken into account:

$$r_i := \sum_{t=T-30}^{T} s^{J_i}(t), \quad i = 1, \ldots, 3, \tag{12}$$

with $T = 200$ ms. Then the softmax function is used on the vector $(r_1, r_2, r_3, r_4)^T$ to obtain the class probabilities $(p_1, p_2, p_3, p_4)^T$. The output class C is then given by the class for which the class probability p_C is maximal.

Fitness function: The fitness function is given by the negative cross entropy loss. For a single example with one-hot-encoded true class label $\mathbf{y}$ the fitness is defined as:

$$F(\boldsymbol{\theta}) = \sum_{c=1}^{4} y_c \log(p_c). \tag{13}$$

Another measure that is commonly used is the accuracy, which is given by the fraction of correct samples.

Constant value of synapses: In the spike interval classification task the weight values of a single synapse are: $w_{in} = 40.82$, $w_E = 11.19$, $w_I = 6.11$.

Task-specific model details: Neurons are arranged in a 4x4 grid of minicolumns with 60 µm between them. In every minicolumn there are two input neurons and one output neuron. Ten recurrent neuron types and 160 recurrent neurons are used, resulting in 164 free parameters, whereas the full RSNN would have 33,280 parameters. Hence only 0.4928% of the potential synaptic connections are used. A decay constant of $\sigma = 77.7$ was used for this task. The weight sparsity on the spike interval classification task is 25.6% and the total wire length is 0.798 meters.

Details to innate pattern classification capability

Task description: The goal of this task is to classify between three different classes of spike patterns. Two of them correspond to spike pattern templates and the third class contains random Poisson spike patterns with a Poisson rate of 50 Hz. Additionally a delay of 50 ms gets added after the spike patterns are completed, hence the network has to memorize the prediction during this time.

Class generation: First for all four input neurons one Poisson spike train with a rate of 50 Hz is created, which yields a binary matrix $\mathbf{z} \in \{0,1\}^{4\times 50}$. In the next step random Poisson spike patterns with the same rate of 50 Hz are sampled until the L^2 difference of the postsynaptic currents caused by the two spike patterns is greater than 0.19. This L^2 difference is for spike patterns $\mathbf{z}$ and $\tilde{\mathbf{z}}$ given by

$$\sum_{t=0}^{\infty} \left(I_{\mathbf{z}}(t) - I_{\tilde{\mathbf{z}}}(t)\right)^2 ,$$

where $I_{\mathbf{z}}$ and $I_{\tilde{\mathbf{z}}}$ are the corresponding postsynaptic currents. The idea of measuring the distance of spike trains by using the L^2 difference of the postsynaptic currents was first introduced in [41].

For the generation of the first two classes, two spike patterns are used as templates. To create samples of these classes the templates are varied by flipping for every neuron at two timesteps from spike to no spike or vice-versa, followed by a shift in time for every spike. The shift is normal distributed with mean zero and variance 0.5 ms. The third class includes random Poisson spike patterns with rate 50 Hz, which have a neglectable probability to belong to one of the two classes.

Input: The network receives as input a spike pattern from one of the three classes with the same probability followed by a 50 ms delay, hence altogether $T = 100$ timesteps.

Output: The output convention used in this task is the same as in the spike interval classification task.

Fitness function: The same fitness function as in the spike interval classification task has been used. As for the spike interval task also in this task the accuracy is a useful measure, that was used to have a better intuition of the performance.

Constant value of synapses: For this task the weight values of a single synapse are: $w_{in} = 19.18$, $w_E = 9.69$, $w_I = 10.90$.

Task-specific model details: Neurons are arranged in a 3x4 grid of minicolumns. There is one input neuron in every corner of the grid. In every minicolumn there is one output neuron. Nine recurrent neuron types and 108 recurrent neurons are used, resulting in 157 free parameters, whereas the full RSNN has 13,392 parameters (compression to 1.172%). A decay constant of $\sigma = 62.7$ was used for this task. The connection sparsity on the spike pattern classification task is 25.8% and the total wire length is 0.241 meters.

Details to local maximum extraction

Task description: In this task 2D patterns $\mathbf{x} \in [0,1]^{\sqrt{n_{col}}\times\sqrt{n_{col}}}$ are presented to the network, where n_{col} is the number of minicolumns, which are arranged on a squared grid. For optimizing the probabilistic skeltons, $n_{col} = 25$ has been selected, resulting in a 5x5 grid. The goal is to produce 2D patterns, where the locations of local maxima in the input signal should be extracted.

Pattern generation To generate a pattern $\mathbf{x}$ first a fixed number N_{points} of coordinate pairs $(m,n) \in \{2,\ldots,\sqrt{n_{col}}-1\}^2$ get drawn randomly. In a second step, the value 1.0 gets assigned to these points, all other points are set to zero. Finally a discrete 2D Gaussian filter with variance 1.25 is applied to this binary matrix and the result is normalized to $[c,1]$, where c is a baseline intensity, which is drawn uniformly from $[0.05, 0.15]$. For the 12x12 patterns that can be seen in Fig. 5 $N_{points} = 6$ was used.

Input: The 2D input signals are encoded for the network by using Poisson spike trains between zero and 200 Hz, where the firing rate of the neuron is proportional to the value at the corresponding index. The length of the simulation is 300 ms.

Computing the targets: The targets of the maximum extraction task are computed by applying the Mexican hat filter with symmetric padding to the original input, followed by a clipping at zero and a normalization to $[0,1]$. The Mexican hat filter is defined as:

$$\text{filter} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{14}$$

Output: There are n_{col} output neurons, which are arranged on a squared grid. The output of the model can be computed by summing up all spikes over time for all mini-columns and normalizing the result:

$$\hat{s}^i(t) = \sum_t z^i(t), \tag{15}$$

$$s^i(t) = \frac{\hat{s}^i(t)}{\max(\hat{s}^i(t))} \tag{16}$$

Fitness function: To calculate the fitness the normalized cross correlation between target signal $\mathbf{t}$ and output signal $\mathbf{y}$ is used:

$$F(\boldsymbol{\theta}) = NCC = \frac{\sum_{i=1}^{n_{col}} t_i y_i}{\sqrt{\sum_{i=1}^{n_{col}} t_i^2 \sum_{i=1}^{n_{col}} y_i^2}}$$

The value of the normalized cross correlation can easily be interpreted. A value of 0.0 would indicate that the two patterns are totally different while a value of 1.0 would show that the two patterns are the same.

Constant value of synapses: The constant values of a single synapse on the local maximum extraction task are: $w_{in} = 1.845$, $w_E = 1.458$, $w_I = 3.1$.

Task-specific model details: A decay constant of $\sigma = 90.6$ is used for this task. The weight sparsity is 15.27% for $n_{col} = 25$ and the total wire length is 0.495 meters. The sparsity decreases to 3.27% for $n_{col} = 144$.

The PS optimized on the local maximum extraction task has five recurrent types and uses seven neurons per column. This enables the PS to generate a RSNN with 1008 neurons when using a 12x12 grid. Note that the PS only has 44 free parameters, where the full RSNN has 21,875 parameters, yielding a compression of 0.201% The average normalized correlation on this task was is 0.81 for a 5x5 grid and 0.76 for a 12x12 grid.

Details to the coincidence detection

Task description: Two 2D input patterns are presented to the network. The goal is to locate the positions where both signals display a high value. The patterns are generated using the same algorithm described for the maximum extraction task.

Input: The same input convention and input patterns as for the maximum extraction task are used, with the difference that two patterns are presented simultaneously.

Computing the targets: The targets of the coincidence detection task can be computed by utilizing the following formula:

$$\mathbf{t} = \frac{\phi(\mathbf{x}_1 \odot \mathbf{x}_2)}{\max(\phi(\mathbf{x}_1 \odot \mathbf{x}_2))} \tag{17}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are the two inputs and ϕ is a function that returns element wise the identity of the input if it is higher than the average of the two inputs, else it returns zero.

Output and Fitness function The output convention and the fitness function are the same as for the maximum extraction task.

Constant value of synapses: The constant values of a single synapse on the coincidence detection task are: $w_{in} = 2.21$, $w_E = 2.97$, $w_I = 2.8$.

Task-specific model details: A decay constant of $\sigma = 69.5$ is used for the coincidence detection task. The weight sparsity is 14.38% for $n_{col} = 25$ and the total wire length is 1.47 meters. Note, that the sparsity drops as the size of the RSNN increases. For example, this value decreases to 3.08% for $n_{col} = 144$.

The PS consists of nine recurrent types and 15 neurons per column. Hence it is possible to construct RSNNs with 375 neurons on a 5x5 grid and also with 2160 neurons on a 12x12 grid. This PS has a total of 121 free parameters, whereas the full RSNN would have 112.500 (5x5) or 3,732,480 (12x12) parameters, yielding a compression of

0.108% and 0.003% respectively. The average normalized correlation on the coincidence detection task is 0.779 for 5x5 grids.

Note, that in the coincidence detection task, the objective is not to compute the sum of the two inputs but much rather to compute the product of the inputs. This is an important consideration, since simply computing a sum with a certain threshold is a trivial task for an RSNN, however, obtaining a product is a lot more challenging. In Fig. 9 it becomes apparent that simply computing the sum would not be a good strategy, as it would yield a low fitness value.

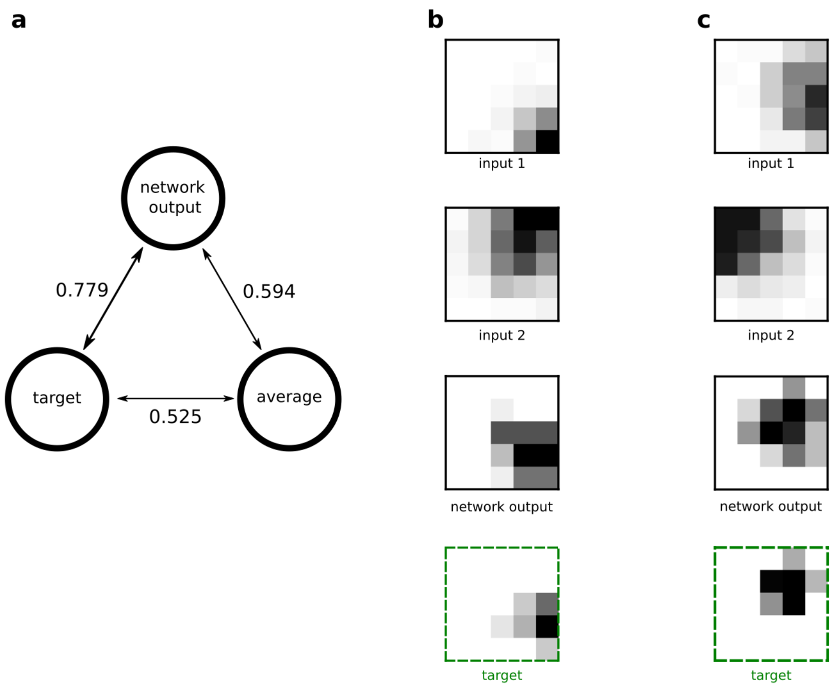


Figure 9: a Normalized cross entropy between the output of the RSNN, the target and the average of the two input patterns. Approximating the target with a sum of the two input patterns does not yield a high correlation. Panel **b** and **c** show 2D input patterns along with the network output and the target pattern. A 5x5 grid was used in this figure.

Details to innate motor control capabilities through probabilistic skeletons

Task description: For the simulation of the environment (AntMuJoCoEnv-v0) the PyBullet physics engine [42] is used. The agent is a quadruped walker and is usually referred to as 'ant' in the literature. It consists of four legs having four joints, which are attached by another four joints to a torso. The goal of this task is to achieve a high forward velocity over the whole simulation period. The episode is terminated if the center of mass of the actor falls below a height of 0.2 m or if the maximum number of time steps has been reached.

Spatial structure: The spatial structure of this task differs from the remaining tasks in the sense that the neurons are not arranged in minicolumns, but the neurons are distributed evenly over a 1D line in the interval $[0, 660]$ µm. This 1D circuit structure turns out to be more effective if network inputs have a pronounced 1D organization, which is induced in this case through the encoding of analog variables in 1D populations of input neurons. For every neuron type, of the recurrent network the neurons are placed such that they are evenly spaced. The distance measure $\mathrm{Dist}(i, j)$ simply computes the distance between two neurons i and j by subtracting the one-dimensional coordinate value of i from the coordinate value of j. Note, that the output locations are not evenly spaced on the 1D line, but their locations are optimized alongside the other parameters of the PS.

Input: In the environment the time is discretized to time steps of 20 ms. For this reason the network receives for 20 ms the corresponding real-valued input, which is encoded to spikes using population coding with nine neuron types, each having 16 neurons. Population coding is used which is commonly employed in the brain to encode analog variables [43]. The input is given by the current state of the environment, where usually the state space has 111 dimensions, but most of them are excluded, for example the angular velocities, to have a more biologically realistic input. In total the observation space consists of the eight joint angles and the height of the torso. Additionally a delay of 60 ms is introduced between the environment and the model.

Each analog input value is encoded by the spiking activity of an excitatory population of N_{pop} input neurons. An input variable x can only takes values in the bounded interval $(a, b] \subset \mathbb{R}$. We define points $a = x_1, \ldots, x_M = b$ such that the subintervals $(x_i, x_{i+1}]$, $i = 0, \ldots, N_{\mathrm{pop}}$ are of equal lengths $\frac{b-a}{N_{\mathrm{pop}}-1}$ 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

$$h_i(x) = \frac{1}{\sigma_{pop}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-x_i}{\sigma_{pop}}\right)^2} \tag{18}$$

for $i = 1, \ldots, N_{\mathrm{pop}}$, where $\sigma_{pop} \in \mathbb{R}$. In the experiments it is chosen to be $\sigma_{pop} = \frac{x_{i+1}-x_{i-1}}{2}$, such that $\sim 68.27\%$ of spiking activity for this type happens in the interval $(x_{i-1}, x_{i+1}]$ and $\sim 95.45\%$ happens in the interval $(x_{i-2}, x_{i+2}]$ for suitable i. Since this is modeled by a density function it is necessary to normalize the values of h_i to obtain spike probabilities for each neuron. A schematic plot of the population coding is given in Figure 10.

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

$$z_i(t) = \begin{cases} 1 & \text{with probability } \frac{h_i(x)}{\max\limits_{x\in[a,b]} h_i(x)} \\ 0 & \text{else} \end{cases} \tag{19}$$

. For an input vector $\mathbf{x} \in R^d$ d types and $d \cdot N_{\mathrm{pop}}$ neurons are used.

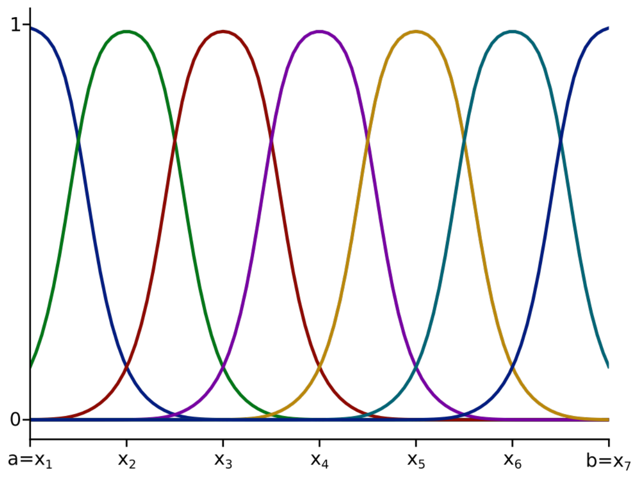


Figure 10: Illustration of population coding.

Output: The action space of the environment is given by $\mathcal{A} = [-1, 1]^8$, which corresponds to torques that get applied to the eight joints of the ant. An output torque $y \in [-1, 1]$ of the model is computed by using two output neuron types representing negative and positive torques denoted by J_- and J_+ and each having four output neurons. This idea is inspired from how animals move their joints by using two opposing muscles, which contract depending on the firing rate that they receive. The output scalar values are computed by the normalized linear combination of the spike rates over the 20 ms , where the model receives the same input, and output neurons of the given type. The output is then computed by

$$y = \frac{\sum_{t=1}^{20} e^{-\frac{20-t}{\tau_{out}}} \left(s^{J_-}(t) - s^{J_+}(t)\right)}{\sum_{t=1}^{20} e^{-\frac{20-t}{\tau_{out}}} \max\left(s^{J_-}(t), s^{J_+}(t)\right)}, \tag{20}$$

where $\tau_{out} = 10$. For computing the outputs 16 output types and 64 neurons are used.

Fitness function: The fitness is given by the total reward received from the environment summed up over time. Every timestep the agent receives a reward

$$v_{\text{fwd}} - 0.1 j_l + 1, \tag{21}$$

where v_{fwd} := forward velocity into the x direction, j_l := number of joints which are at the limit and there is a constant reward of 1 for each time step.

Constant value of synapses: The constant values of a single synapse on this task are: $w_{in} = 4.75$, $w_E = 4.5$, $w_I = 2.3$.

Task-specific model details: A decay constant of $\sigma = 80.0$ is used for this task. The sparsity of the model is 11.24% and the total wire length is 0.79 meters.

The experiment where 30% of the neurons get deleted is conducted by sampling a RSNN from the PS and randomly removing 30% of the recurrent and output neurons. Note, that input neurons do not get deleted.

Our version of the ant task poses a much more challenging task than the original ant which has been considered previously in the literature [44]. The reason for this is the limited observation space, which makes it very hard for the model to know in which direction it is facing, especially at a later point in the simulation. As the environment only considers speed along the x-axis for the fitness it is important to run in a straight line, which is more difficult if the feedback regarding the orientation is missing. Furthermore, to increase biological realism, there is a delay of 60 ms between the environment and the RSNN, making the task even more demanding. Our model manages to achieve an average fitness of 517 using 250 steps in the environment, where the average is computed over 100 trials. The version of the model where 30% of the recurrent and output neurons are randomly deleted achieves a fitness of 421.

Details to general results and principles for designing neural networks through probabilistic skeletons

To compare the different tasks it is necessary to use a common scale. This can be achieved by defining baselines for every task, which correspond to the performance of a random strategy. For the spike interval classification task there are four classes, hence picking random classes would give an expected accuracy of 25 %. Analogously the baseline accuracy for the spike pattern classification, which involves three classes, is 33.33%. For the maximum extraction task and the coincidence detection task it is possible that a PS has a fitness of 0.0, hence this was chosen to be the baseline for these tasks.

The performances on these different tasks can be compared by calculating the difference of the actual performance (either accuracy or normalized cross correlation) to the baseline performance and normalizing this difference to $[0, 1]$.

For each of the numbers of neuron types in panel a 80 probabilitic skeletons were optimized for every task and the best performing ones were used.

The pruning experiments in panel b and c were done by deleting synapses that have a connection probability below a certain threshold, where the threshold was raised until the average number of synaptic connections was at the desired levels needed for plotting panel b. Simultaneously the average wire length per neuron was calculated, which gave the results that can be seen in panel c.

The weight perturbation experiment in panel d was performed by sampling an RSNN from the probabilistic skeleton and then perturbing for every neuron pair the current number of synapses N_{syn} by

$$N_{syn} \leftarrow N_{syn} + N_{pert}.$$

The perturbation number N_{pert} is sampled uniformly from

$$\{\max(-N_{syn}, -\lfloor cN_{syn} + \frac{1}{2}\rfloor), \ldots, \lfloor cN_{syn} + \frac{1}{2}\rfloor\}.$$

Hence the weight is perturbed, such that the maximal perturbation is as close as possible to c and the number of synapses has to stay above zero. The x-axis in panel d corresponds to different values for c.

For panel e the number of neurons was varied by increasing the number of minicolumns.

References

[1] Anthony Zador. "A critique of pure learning and what artificial neural networks can learn from animal brains". In: *Nature Communications* 10 (Dec. 2019). DOI: 10.1038/s41467-019-11786-6.

[2] Raimund Apfelbach et al. "The Effects of Predator Odors in Mammalian Prey Species: A Review of Field and Laboratory Studies". In: *Neuroscience and biobehavioral reviews* 29 (Feb. 2005), pp. 1123–44. DOI: 10.1016/j.neubiorev.2005.05.005.

[3] Melis Yilmaz and Markus Meister. "Rapid Innate Defensive Responses of Mice to Looming Visual Stimuli". In: *Current biology : CB* 76 (Oct. 2013). DOI: 10.1016/j.cub.2013.08.015.

[4] Nikolaas Tinbergen. *The study of instinct.* Pygmalion Press, an imprint of Plunkett Lake Press, 2020.

[5] Jesse Weber and Hopi Hoekstra. "The evolution of burrowing behavior in deer mice (genus Peromyscus)". In: *Animal Behaviour* 77 (Mar. 2009), pp. 603–609. DOI: 10.1016/j.anbehav.2008.10.031.

[6] Hillery Metz et al. "Evolution and Genetics of Precocious Burrowing Behavior in Peromyscus Mice". In: *Current Biology* 27 (Nov. 2017). DOI: 10.1016/j.cub.2017.10.061.

[7] Rosamund Langston et al. "Development of the Spatial Representation System in the Rat". In: *Science (New York, N.Y.)* 328 (June 2010), pp. 1576–80. DOI: 10.1126/science.1188210.

[8] Elinor Mckone, Kate Crookes, and Nancy Kanwisher. "The Cognitive and Neural Development of Face Recognition in Humans". In: *The Cognitive Neurosciences* Vol. 4 (Jan. 2009).

[9] Vernon B Mountcastle. *Perceptual neuroscience: the cerebral cortex.* Harvard University Press, 1998.

[10] R. Douglas and K. Martin. "Neuronal circuits of the neocortex." In: *Annual review of neuroscience* 27 (2004), pp. 419–51.

[11] Kenneth Harris and Gordon Shepherd. "The neocortical circuit: Themes and variations". In: *Nature neuroscience* 18 (Feb. 2015), pp. 170–181. DOI: 10.1038/nn.3917.

[12] Henry Markram et al. "Reconstruction and Simulation of Neocortical Microcircuitry". In: *Cell* 163 (Oct. 2015), pp. 456–492. DOI: 10.1016/j.cell.2015.09.029.

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

[14] Thomas C Südhof. "Towards an understanding of synapse formation". In: *Neuron* 100.2 (2018), pp. 276–293.

[15] Luis Cruz et al. "A statistically based density map method for identification and quantification of regional differences in microcolumnarity in the monkey brain". In: *Journal of Neuroscience Methods* 141.2 (2005), pp. 321–332.

[16] Javier DeFelipe. "The anatomical problem posed by brain complexity and size: a potential solution". In: *Frontiers in neuroanatomy* 9 (2015), p. 104.

[17] Tom Schaul, Tobias Glasmachers, and Jürgen Schmidhuber. "High Dimensions and Heavy Tails for Natural Evolution Strategies". In: Jan. 2011, pp. 845–852. DOI: 10.1145/2001576.2001692.

[18] John S Kauer and Joel White. "Imaging and coding in the olfactory system". In: *Annual review of neuroscience* 24.1 (2001), pp. 963–979.

[19] Durk Talsma. "Predictive coding and multisensory integration: an attentional account of the multisensory mind". In: *Frontiers in Integrative Neuroscience* 9 (2015), p. 19.

[20] Jerry L Chen et al. "Behaviour-dependent recruitment of long-range projection neurons in somatosensory cortex". In: *Nature* 499.7458 (2013), pp. 336–340.

[21] Nobuaki Yasumatsu et al. "Principles of long-term dynamics of dendritic spines". In: *Journal of Neuroscience* 28.50 (2008), pp. 13592–13608.

[22] Sho Yagishita et al. "A critical time window for dopamine actions on the structural plasticity of dendritic spines". In: *Science* 345.6204 (2014), pp. 1616–1620.

[23] Tianda Fu et al. "Bioinspired bio-voltage memristors". In: *Nature communications* 11.1 (2020), pp. 1–10.

[24] Valentino Braitenberg and Almut Schüz. *Cortex: statistics and geometry of neuronal connectivity*. Springer Science & Business Media, 2013.

[25] Anthony Holtmaat and Karel Svoboda. "Holtmaat A, Svoboda K. Experience-dependent structural synaptic plasticity in the mammalian brain. Nat Rev Neurosci 10: 647-658". In: *Nature reviews. Neuroscience* 10 (Oct. 2009), pp. 647–58. DOI: 10.1038/nrn2699.

[26] Douglas Heaven. "Why deep-learning AIs are so easy to fool". In: *Nature* 574 (Oct. 2019), pp. 163–166. DOI: 10.1038/d41586-019-03013-5.

[27] Dániel L. Barabási and Taliesin Beynon. "Complex Computation from Developmental Priors". In: *bioRxiv* (2021). DOI: 10.1101/2021.03.29.437584. eprint: https://www.biorxiv.org/content/early/2021/04/12/2021.03.29.437584.full.pdf. URL: https://www.biorxiv.org/content/early/2021/04/12/2021.03.29.437584.

[28] Alexei Koulakov, Sergey Shuvaev, and Anthony Zador. "Encoding innate ability through a genomic bottleneck". In: *bioRxiv* (2021). DOI: 10.1101/2021.03.16.435261. eprint: https://www.biorxiv.org/content/early/2021/03/16/2021.03.16.435261.full.pdf. URL: https://www.biorxiv.org/content/early/2021/03/16/2021.03.16.435261.

[29] Adam Gaier and David Ha. "Weight agnostic neural networks". In: *arXiv preprint arXiv:1906.04358* (2019).

[30] Kenneth O Stanley et al. "Designing neural networks through neuroevolution". In: *Nature Machine Intelligence* 1.1 (2019), pp. 24–35.

[31] Henry Markram et al. "Interneurons of the neocortical inhibitory system". In: *Nature reviews. Neuroscience* 5 (Nov. 2004), pp. 793–807. DOI: 10.1038/nrn1519.

[32] Seth Grant. "The molecular evolution of the vertebrate behavioural repertoire". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 371 (Jan. 2016), p. 20150051. DOI: 10.1098/rstb.2015.0051.

[33] Fei Zhu et al. "Architecture of the Mouse Brain Synaptome". In: *Neuron* 99.4 (2018), 781–799.e10. ISSN: 0896-6273. DOI: https://doi.org/10.1016/j.neuron.2018.07.007. URL: https://www.sciencedirect.com/science/article/pii/S0896627318305816.

[34] Wei Wang et al. "Integration and co-design of memristive devices and algorithms for artificial intelligence". In: *Iscience* (2020), p. 101809.

[35] Aparna Bhaduri et al. "Are Organoids Ready for Prime Time?" In: *Cell stem cell* 27.3 (2020), pp. 361–365.

[36] Corinne Teeter et al. "Generalized leaky integrate-and-fire models classify multiple neuron types". In: *Nature Communications* 9 (Feb. 2018). DOI: 10.1038/s41467-017-02717-4.

[37] *V1 Network Models from the Allen Institute*. URL: https://www.dropbox.com/sh/w5u31m3hq6u2x5m/AACpYpeWnm6s_qJDpmgrYgP7a?dl=0.

[38] Daan Wierstra et al. "Natural Evolution Strategies". In: June 2008, pp. 3381–3387. DOI: 10.1109/CEC.2008.4631255.

[39] Tim Salimans et al. "Evolution strategies as a scalable alternative to reinforcement learning". In: *arXiv preprint arXiv:1703.03864* (2017).

[40] Dimo Brockhoff et al. "Mirrored Sampling and Sequential Selection for Evolution Strategies". In: Apr. 2010. ISBN: 978-3-642-15843-8. DOI: 10.1007/978-3-642-15844-5_2.

[41] Mark van Rossum. "A Novel Spike Distance". In: *Neural Computation* 13 (Apr. 2001), pp. 751–763. DOI: 10.1162/089976601300014321.

[42] Erwin Coumans and Yunfei Bai. *PyBullet, a Python module for physics simulation for games, robotics and machine learning*. http://pybullet.org. 2016–2021.

[43] Apostolos P Georgopoulos, Andrew B Schwartz, and Ronald E Kettner. "Neuronal population coding of movement direction". In: *Science* 233.4771 (1986), pp. 1416–1419.

[44] John Schulman et al. "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In: (June 2015).