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

The genetic code endows neural networks of the brain with innate computing capabilities. But it has remained unknown how it achieves this. Experimental data show that the genome encodes the architecture of neocortical circuits through pairwise connection probabilities for a fairly large set of genetically different types of neurons. We build a mathematical model for this style of indirect encoding, a probabilistic skeleton, and show that it suffices for programming a repertoire of quite demanding computing capabilities into neural networks. These computing capabilities emerge without learning, but are likely to provide a powerful platform for subsequent rapid learning. They are engraved into neural networks through architectural features on the statistical level, rather than through synaptic weights. Hence they are specified in a much lower dimensional parameter space, thereby providing enhanced robustness and generalization capabilities as predicted by preceding work.

1 Introduction

Artificial neural networks typically receive their computational capabilities through adaptation of a very large set of parameters: through training of their synaptic weights with a very large number of examples, starting from a tabula rasa initial state. In contrast, neural networks of the brain are already at birth highly structured, and they have innate computing capabilities [1], see [2], [3], [4], [5], [6], [7], [8] for experimental data. 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,

*These authors contributed equally.

and that they can stand up and walk right after birth. But it has remained an open question how the genetic code achieves this.

We re-examine experimental data on the architecture of generic microcircuits in the mammalian neocortex [9] [10]. Their architecture was found to be genetically encoded through connection probabilities between a fairly large number of genetically different types of neurons, i.e., classes of neurons with similar gene expression profiles [11]. We show that these genetically encoded architectural features of neural networks of the neocortex are in principle able to engrave substantial computing capabilities into these networks. We create for that purpose a mathematical model for genetically controlled features of neural network architectures, a probabilistic skeleton. We show that probabilistic skeletons are able to install powerful computing capabilities in brain-like neural network models: recurrent networks of spiking neurons (RSNNs). In fact, they provide a surprisingly expressive "programming language" for that. For example, the architectural features that it controls can endow RSNNs with the capability to compute on spike times, to recognize particular spike patterns that might for example represent poisonous food odors, to carry out generic computations of stereotypical cortical microcircuits, and to enable quadruped locomotion.

Probabilistic skeletons specify computing capabilities of neural networks in the low dimensional parameter space of general architectural network features. The strength of an individual synaptic connection in the network is only scaled through the connection probabilities of the neurons involved. This biologically derived indirect code removes some of the brittleness that plagues trained deep neural networks, as predicted by [1]. In particular, it makes them robust to weight perturbations and online noise in synaptic connections. Furthermore, it drastically reduces the required number of synapses and wire length. These implications provide new perspectives for understanding biological neural networks, but also suggests new design paradigms for highly energy-efficient neuromorphic hardware with unreliable analog units.

We will first present the precise definition of a probabilistic skeleton, and then show that it controls architectural features that suffice for endowing RSNNs with a variety of computing capabilities that are conjectured to be innate. Finally, we analyze general properties of this new method and suggest directions for further research.

2 Results

2.1 Probabilistic skeletons provide a mathematical model for aspects of network architectures that are under genetic control

Current models of cortical microcircuits [9, 10] are based on two types of data: A set of neuron types -estimated to be well over 100 within a single neocortical area [11] - and a table of connection probabilities for any pair of neuron types as in panel A of Fig. 4 in [10], that is reproduced here as Fig. 1a. The entries of this table provide base connection probabilities that are valid if the somata have a horizontal distance of at most $75\mu m$. If the horizontal distance is larger, these base connection probabilities are multiplied with

an exponentially decaying function of their distance. Examples for such functions are shown in panel C of Fig. 4 in [10], reproduced here as Fig. 1b.

A probabilistic skeleton is a mathematical model for this indirect style of encoding network architectures. It requires the specification of some number K of neuron types, the prevalence of each neuron type (see lower part of Fig. 1c), and base connection probabilities between a pair of neuron types (see upper part of Fig. 1c). In addition, it specifies a parameter σ that scales the exponential decay of connection probabilities with the horizontal distance between the somata according to equation 4 in Methods; see Fig. 1d for samples of exponential decays that were used. A probabilistic skeleton does not specify the values of individual synaptic weights, just three scaling parameters w_{in}, w_E, w_I that scale the strength of connections from input neurons and from excitatory and inhibitory connections in the recurrent network, see Fig. 1e. Importantly, a probabilistic skeleton does not define any specific neural network. Rather, it is a probabilistic generative model from which an unbounded number of different neural networks with common architectural features can be sampled.

To sample a neural network from a probabilistic skeleton, one specifies its number N of neurons and arranges them on a 2D sheet (because their spatial distances are relevant for their connection probabilities). A suitable method for arranging neurons of different types in a uniform manner is suggested by a well-known ontogenetic feature of the neocortical 2D sheet: It is structured as an array of minicolumns that extend vertically across all neocortical layers [12], [13], [14]. Each minicolumn in a cortical area, see Fig. 1f for an illustration, contains about the same number of neurons, from all major neuron types that are prevalent in that area. Fig. 1g shows the resulting 2D arrangement of neurons in our model, where each minicolumn -appearing as a disk in the top-down view of Fig. 1g- contains in general the same number M of neurons from all neuron types, according to the prevalence bar of the probabilistic skeleton (see bottom part of Fig. 1c). For tasks with small numbers of input or output neurons, they were placed into selected subsets of the minicolumns. The horizontal distance between neurons that are located in the same minicolumn (disk) is negligible, and hence provided by base connection probabilities. Disks are arranged orthogonally for simplicity, with a distance of 60 µm between neurons in neighboring disks.

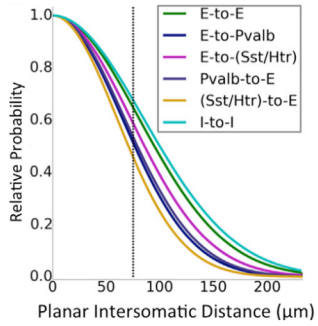
Multiple synaptic connections are common in the neocortex, see Fig. 7A of [9]. Hence we draw for each pair i, j of neurons not just once, but S times from the corresponding connection probability, see equation 4. We used $S = 8$ in our experiments, but the exact value had little impact. The resulting multiplicity m_{ij} of synaptic connections from neuron i to neuron j enables a rough differentiation of connection strengths that can be argued to arise from the genetic code. Examples for the binomial distributions of these multiplicities are shown in Fig. 1h.

Since a probabilistic skeleton only captures aspects of the architecture of neocortical neural networks that are under genetic control, one can use this concept to examine the impact of genetically encoded architectural features on computational properties of a neural network. If a probabilistic skeleton endows its neural network samples with a specific computing capability, this computing capability can be argued to be within the reach of genetic control (”innate”).

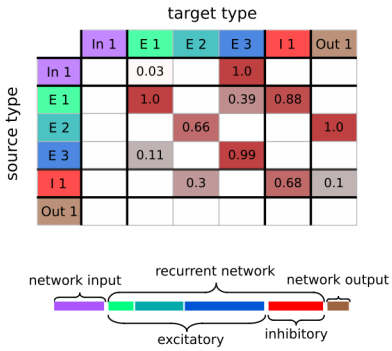
a

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

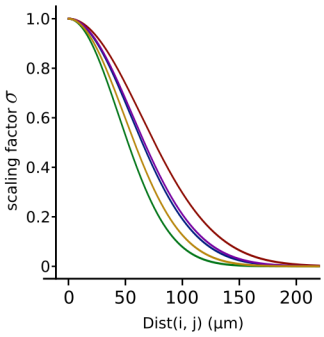
b



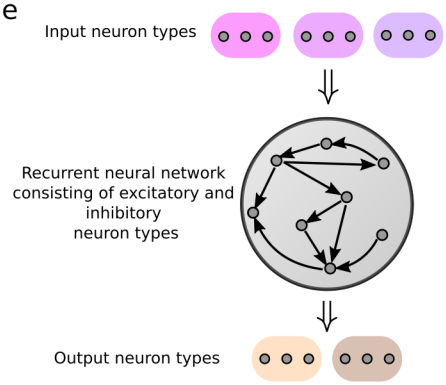
c



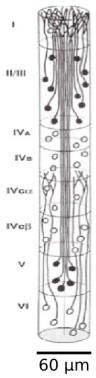
d



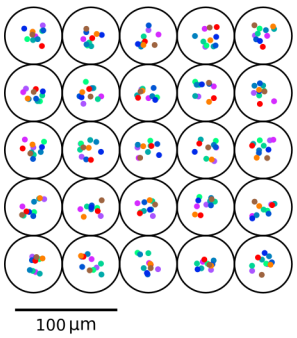
e



f



g



h

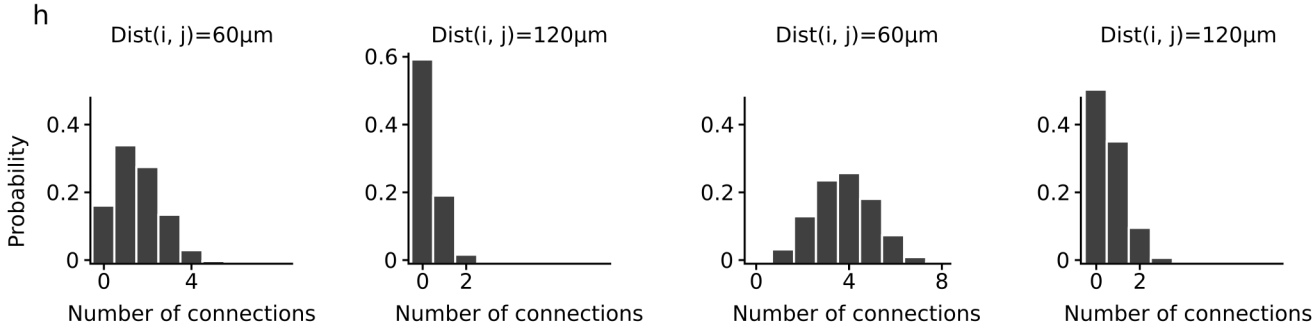


Figure 1: | Illustration of the concept of a probabilistic skeleton. **a** Base connection probabilities between 17 types of neurons in mouse V1 (reproduced from [10]). White grid cells indicate unknown values. **b** Scaling of connection probabilities with the horizontal distance of their somata for mouse V1 (reproduced from [10]). **c** Top: Sample base connection probability table of a probabilistic skeleton for the case of $K = 6$ neuron types. White grid cells indicate here that the corresponding base connection probability has the value 0. Rows and columns labeled "in" refer to input neuron types, the label "out" refers to output neuron types. Bottom: Prevalence-bar of a probabilistic skeleton. Its length defines the number M of neurons in a minicolumn. **d** Sample options for distance-dependent scaling functions, with different values of σ in equation (4). These functions turned out to work well for computing tasks that we considered. **e** Large-scale architecture of a neural network sample from a probabilistic skeleton, with the recurrent network in the center, as well as unidirectional synaptic connections from all input neuron types and to all output neuron types. **f** Sketch of a cortical minicolumn according [12], showing somata of neurons on different layers and vertically running dendrites. **g** Topdown view of our 2D arrangement of neurons, each containing the same combination of neurons of different types (color code as in panel c). Horizontal displacements of neurons within a minicolumn (disk) are just for illustration, they are actually placed at the center. **h** Examples for binomial distributions from which the number m_{ij} of synaptic connections from a neuron i of type I to a neuron j of type J are drawn 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 the spatial distance $\mathrm{Dist}(i, j)$ between their somata. |

2.2 Algorithmic approach for optimizing probabilistic skeletons

The iterative method illustrated in Fig. 2 is used to examine whether a specific computing capability can be induced by a probabilistic skeleton. Since the fitness function is in general not differentiable, we have used evolution strategies [15] as optimization algorithm for probabilistic skeletons. This algorithm combines elements of stochastic search with empirical estimates of gradients of the fitness function. It is used to optimize a parameter vector $\boldsymbol{\theta}$ that consists of the base connection probabilities $p_{I\to J}$ and prevalences p_I for all neuron types I and J, and the 3 scaling parameters w_{in}, w_E, w_I for synaptic weights. The remaining parameters K and σ of a probabilistic skeleton were separately optimized.

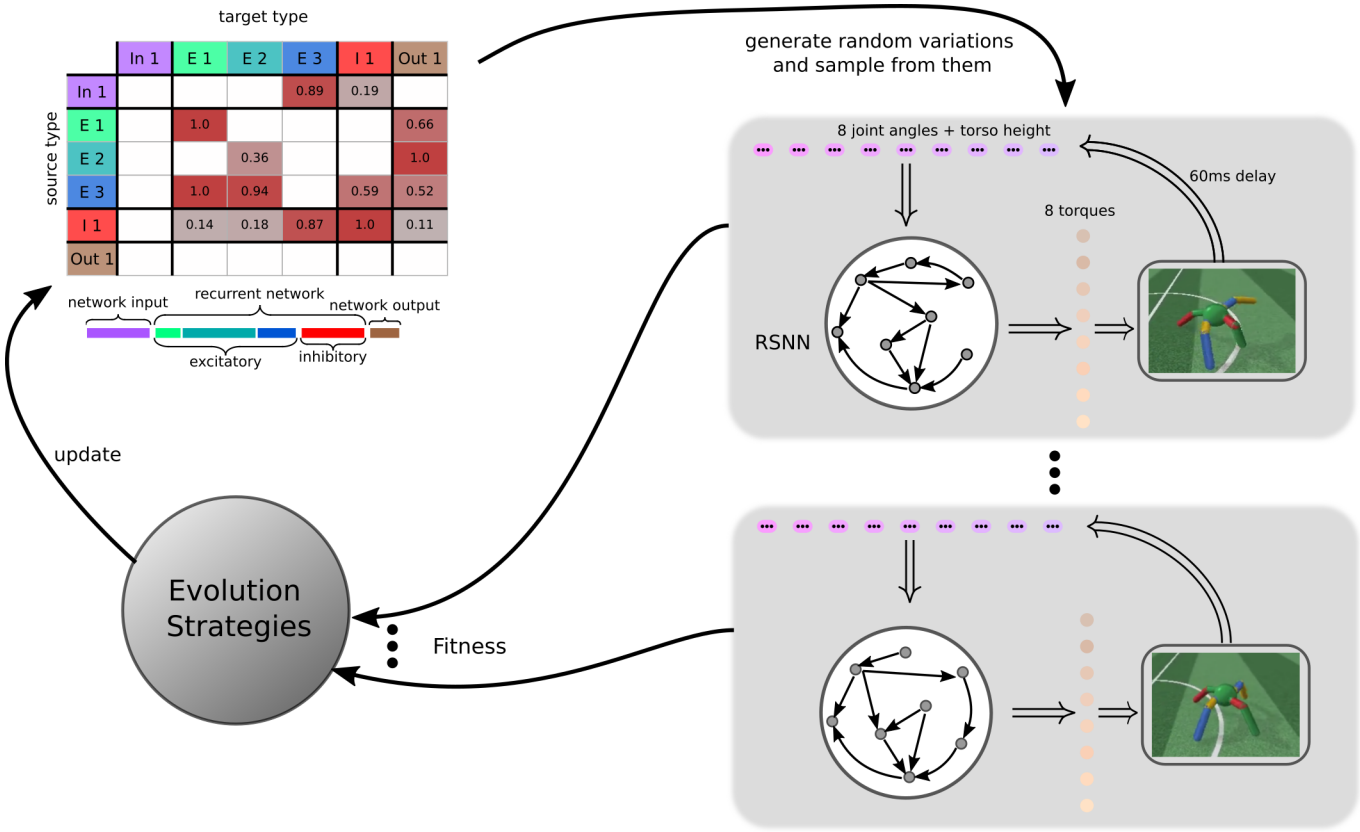


Figure 2: |Illustration of our algorithmic approach for optimizing a probabilistic skeleton for a computing task. The motor control task of Fig. 6 is used as illustration. Several RSNNs are sampled from the current probabilistic skeleton, and their capability to solve the given task, i.e., their fitness, is measured. Evolution strategies modify the probabilistic skeleton based on these fitness values. Then the loop is iterated.|

One can use probabilistic skeletons to generate any type of neural network. We examine here applications to recurrent networks of spiking neurons (RSNNs). Among simple neural network models, these can be argued to be the most brain-like. In addition, it is more difficult to construct or train RSNNs for a specific computing task, so the task becomes more challenging. One reason is that -in contrast to artificial neural networks- their computations are event-based rather than clocked: A spike represents an event in space and time, and time can be used as additional resource for representing and transforming salient information. It is also substantially more difficult to induce computational function in RSNNs through gradient descent of their synaptic weights because of the non-differentiability of spiking neuron models.

2.3 Computations on spike times.

We want to install in RSNNs through their architecture the capability to classify and remember the time interval between two waves of input spikes, see the top row of Fig. 3a.

The first wave of input spikes arrives right at the beginning of a trial, and the second wave can arrive anytime during the subsequent 200ms. These 200ms are partitioned into 4 bins of 50ms length, indicated by yellow vertical boundaries in the top row of Fig. 3a. There are 4 output neurons types, each assigned to one of the 4 bins, and that one which produces the most spikes during the last 30ms of the 200ms long trial encodes the decision of the network. The time scale of this task is in the range of behavioral responses, but a clever network organization is needed in order to enable a network of standard neuron and synapse models to discern and classify such fairly large time differences up to 200ms, and to produce the decision at a specific time after the onset of a trial, without an external clock or prompt.

We found that RSNN samples from a probabilistic skeleton with just 10 types of neurons in the recurrent network and altogether 164 parameters, shown in Fig. 3b, can solve this task with a classification accuracy of 97% (standard deviation 2.1%). The fitness of the probabilistic skeleton was tested during its optimization on RSNN samples with 304 neurons. Thus the number of parameters of the probabilistic skeleton (164) represented just a tiny fraction of the 59,904 parameters (= number of potential synaptic connections of the RSNN samples) that would have to be adjusted during training of the RSNN. A typical spike raster of an RSNN sample is shown in Fig. 3a. One sees that temporal distances between the two waves of input spikes and the decision time are bridged by persistent activity in specific recurrent neuron types.

Fig. 3b, c suggest that the logic of interactions between the neuron types is quite complex. The 4 rightmost columns of the table of base connection probabilities in Fig. 3b show that each of the 4 types of output neurons receives synaptic inputs from 4 different -although overlapping- sets of recurrent neuron types. The complex interactions within the recurrent network manage to activate during the decision period just the right set of recurrent neuron types. In the trial shown in Fig. 3a the 3rd output type is correctly activated during the decision period, because the neuron types E2, E6, E7 from which it receives the most synaptic connections all fire persistently during this period. This is facilitated by the fact that E2 and E7 have strong internal recurrent connections, and E6 is activated by them. During the last 10ms also E5 start to fire, which activates the 2nd output type, without being able to cause an incorrect decision. Spike rasters from further sample trials are shown in the Supplement (Fig. S1, S2, S3). Altogether we see that architectural features that are under genetic control are able to induce in RSNNs sophisticated computational capabilities on spike times.

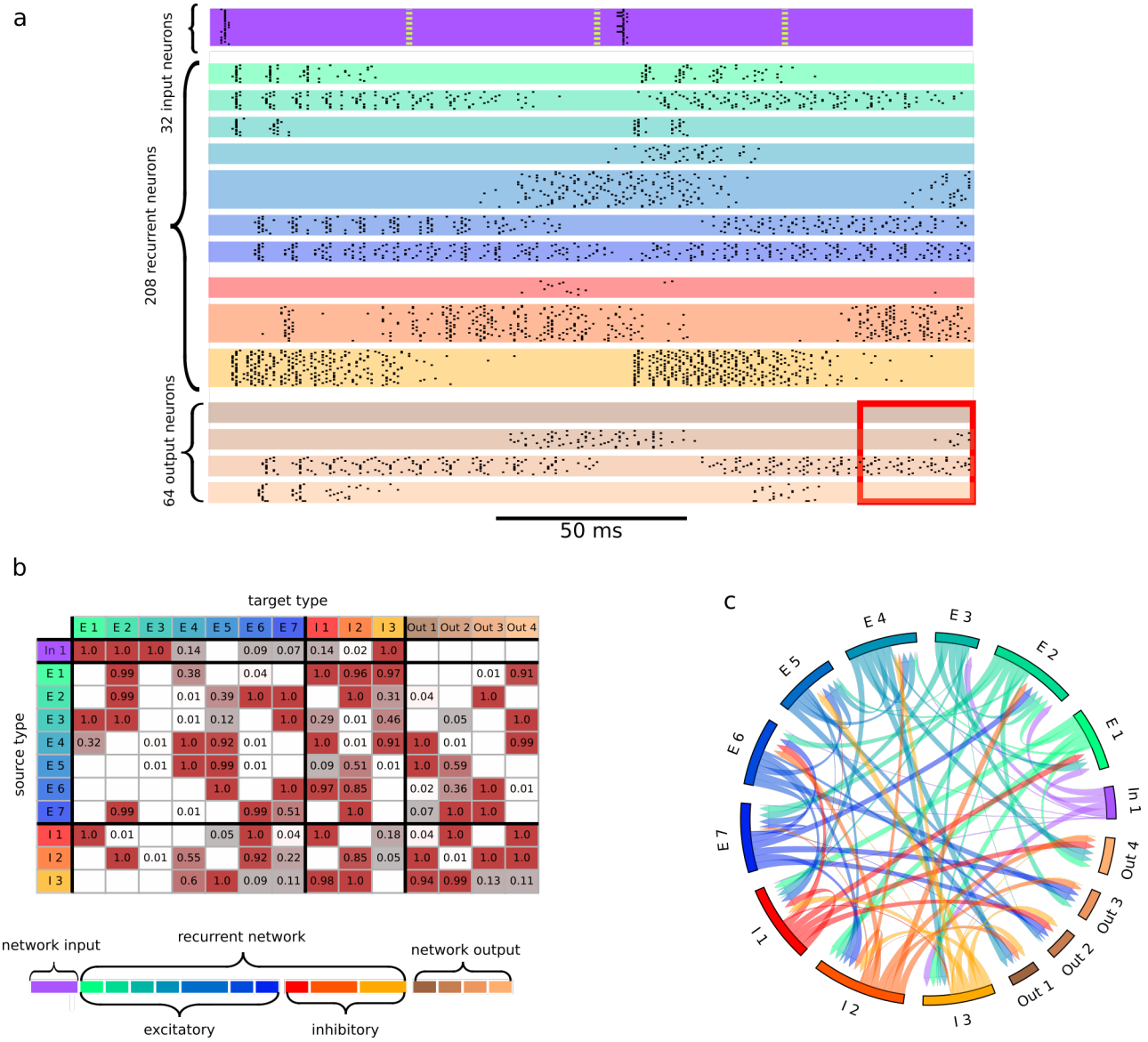


Figure 3: |Induction of innate computing capability on spike times through a probabilistic skeleton. **a** The task is to classify the time of the 2nd wave of input spikes into 4 bins of length 50 ms, indicated by dotted lines in the top row. The network is supposed to indicate its decision through stronger firing of the corresponding type of output neurons during the last 30ms of a trial, indicated as a red frame at the bottom. Firing activity is shown for all 304 neurons of an RSNN sample of the probabilistic skeleton. Emergent sequential activation of different assemblies of neurons is clearly visible. **b** Probabilistic skeleton for this task that was optimized using RSNN samples with 304 neurons. **c** The connectivity graph induced by this probabilistic skeleton. It is plotted in the same style (chord diagram) as connectivity data for cortical microcircuits in Fig. 7C of [9]. The thickness of ribbons is proportional to the number of synaptic connections; the thickness of their arrow-heads encodes for the target neuron type the distribution of incoming connections over source neuron types. The lengths of arcs for different neuron types reflect their expected fraction of incoming and outgoing synaptic connections.|

2.4 Can the genetic program install the capability to recognize specific spike patterns in RSNNs?

We are addressing here the question of whether the genetic code can install in RSNNs the capability to recognize specific sensory input patterns, even without prior experience. These especially salient sensory input patterns could represent odors of poisonous food, views of predators, or faces of conspecifics [8]. Most sensory inputs, including olfactory inputs (see e.g., Fig. 5 of [16]), arrive in RSNNs of the brain in the form of spatio-temporal spike patterns. Hence we wondered whether a probabilistic skeleton can endow RSNNs with the capability to recognize variations of specific but arbitrarily chosen spatio-temporal spike patterns, such as the ones shown for classes 1 and 2 in Fig. 4a. At the same time, the RSNN should not respond in the same way to other spatio-temporal spike patterns, even if they have the same firing rates, such as the samples of class 3 in Fig. 4a. The spike pattern templates for classes 1 and 2 were frozen Poisson spike trains. Samples from the corresponding classes were generated by adding, deleting, and shifting spikes of these templates in time. Spike patterns of class 3 were freshly generated with the same Poisson firing rates as the spike templates for classes 1 and 2. The network decision was expected to be encoded like in the preceding task: There are 3 output neuron types, and the one that produces the most spikes during the last 30ms of a trial determines the network decision.

We found that a probabilistic skeleton with 9 recurrent neuron types (shown in Fig. 4b, c) can install this capability in RSNNs samples with a classification accuracy of 91% (standard deviation 3.1%). The spike rasters of RSNN samples with 144 neurons are shown in Fig. 4d, e, f. One sees again that each type of output neurons receives synaptic inputs from somewhat different recurrent neuron types. For example, output neurons of type 3 are excited through strong connections from recurrent type E5, yielding in the trial shown in Fig. 4f the correct network decision. Type E5 is also highly active in the trial shown in Fig. 4d. But output neurons of type 3 also receive inhibitory input from type I2, whose activation prevents them from firing incorrectly in this trial. Altogether we see again that emergent assembly activations bridge the delay until the network has to make a decision.

The probabilistic skeleton for this task had 157 parameters, which is very efficient given that the specification of the spike templates for classes 1 and 2 requires already a substantial number of parameters. In contrast, the RSNN samples that were considered during the optimization of the probabilistic skeleton had 144 neurons, hence 13,392 potential synaptic connections. Altogether we see that low-dimensional architectural features of neural networks are able to imprint into them special handling of particular spatio-temporal spike patterns, a feat that one would have expected to require fine-tuning of synaptic weights through training in a high-dimensional parameter space.

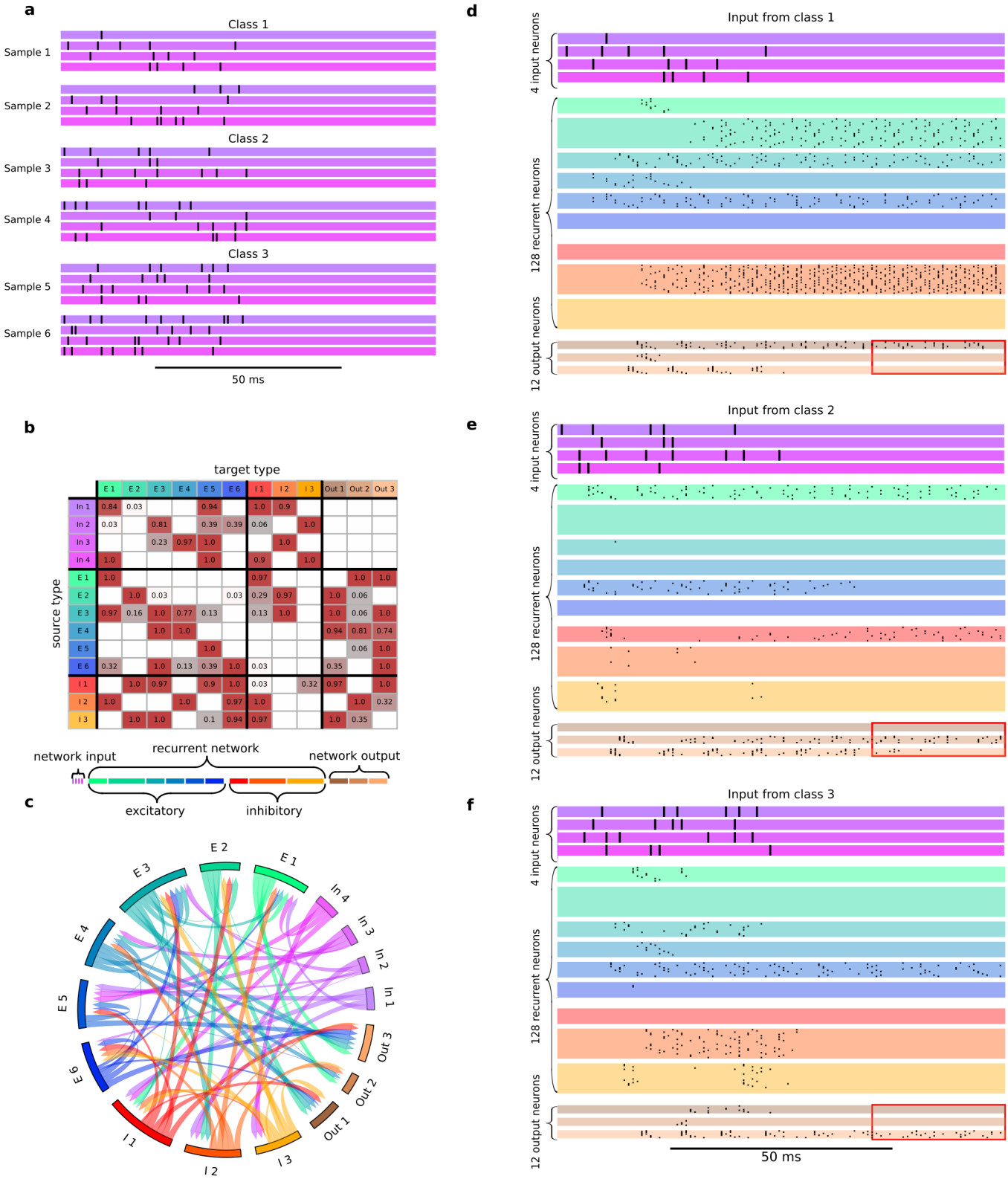


Figure 4: |Innate spike pattern classification capability. **a** Two samples from each of the three classes of spike input patterns. The first two classes consist of variations of specific but arbitrarily chosen spike patterns, the third class consists of distractor spike patterns with the same firing rates. **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 neurons of RSNN samples with 144 neurons, in sample trials for spike inputs from classes 1-3. The 30 ms time window during which the network decision is expected is indicated by the red frame at the bottom of the spike rasters.|

2.5 Generic 2D computing capabilities.

The neocortex forms a 2D sheet, with topographic maps between cortical areas and from peripheral sensory neurons. Hence it is commonly conjectured that the neocortex is genetically endowed with basic computing capabilities that make use of this 2D organization [12]. We wondered whether a probabilistic skeleton can capture such basic 2D computing capabilities. Contrast enhancement and amplification of local maxima, i.e., applying a Mexican hat filter, is a standard example for such a 2D computing capability. We found that a probabilistic skeleton with just 4 recurrent neuron types (shown in Fig. 8a) endows RSNN samples with the capability to approximate the computation of a Mexican hat filter, achieving for example for 5x5 input patterns a correlation of 0.89 with this target (standard deviation 0.01). Fig. 5 c and d depict two examples for the resulting transformation of the input patterns shown at the top to the outputs shown at the bottom by an RSNN sample from the probabilistic skeleton with 1152 neurons. The grey values of pixels in the input- and output patterns were encoded through firing rates. While this RSNN sample had 1152 neurons and 995, 328 potential synaptic connections, the probabilistic skeleton encoded this computation with just 32 parameters. Although this probabilistic skeleton had been optimized for computations on 5x5 inputs (red square in Fig. 5b), RSNN samples could also process the shown 12x12 input patterns very well (cross correlation 0.71 with target). Hence one can argue that the underlying probabilistic skeleton captures a generic 2D computing capability. In particular, the probabilistic skeleton managed to organize the somewhat delicate interaction of excitatory and inhibitory neurons that is needed to execute this task. The phasic firing pattern of the output neurons in Fig. 5a suggests that the resulting RSNN implements the local contrast enhancement through an iterated temporal competition between excitatory neurons in the recurrent network, where the strongest activated neurons fire first, thereby inhibiting weaker competitors -but transiently also themselves.

Cortical areas typically receive synaptic inputs in the form of multiple topographic maps from other cortical areas, and also from the visual or somatasensory periphery. Coincidence detection between bottom-up and top-down projections is conjectured to be a fundamental operation in perception [17], see Fig. 5e for an illustration. This computation requires the RSNN to approximate the computation of products of input values at the same location in the two input patterns (see Methods and target outputs at the bottom of Fig. 5g and h), rather than sums: Hence is somewhat nontrivial to implement with spiking neurons. Fig. 5f-h shows that also this fundamental 2D computing capability can be induced through a probabilistic skeleton (shown in Fig. 8b). Fig. 5g depicts the computation of an RSNN sample on the two 2D patterns input1 and input2 shown at the top and middle of panel h. Input- and output values were again encoded by firing rates. The RSNN produced the output shown below them. The target output is depicted at the bottom. Panel i shows another example of this computation by the RSNN. The RSNN achieved on 5x5 patterns a correlation with the target output of 0.777 (standard deviation 0.014), which is not optimal, but substantially higher than what is achievable with a linear computation (sum), see Fig. 8a. This probabilistic skeleton had 121 parameters. It had been optimized, like the one for local contrast

enhancement, for 5x5 input patterns, but also achieved for 12x12 input patterns a cross-correlation of 0.71 with the target. The RSNNs for the 12x12 inputs had 2160 neurons and 3,732,480 potential synaptic weights. Altogether the results of this section provide evidence that fundamental computational operations that have been conjectured to be employed by cortical microcircuits throughout the neocortical sheet can very well be engraved into these circuits by genetically controlled features of their architecture, in a quite low-dimensional parameter space.

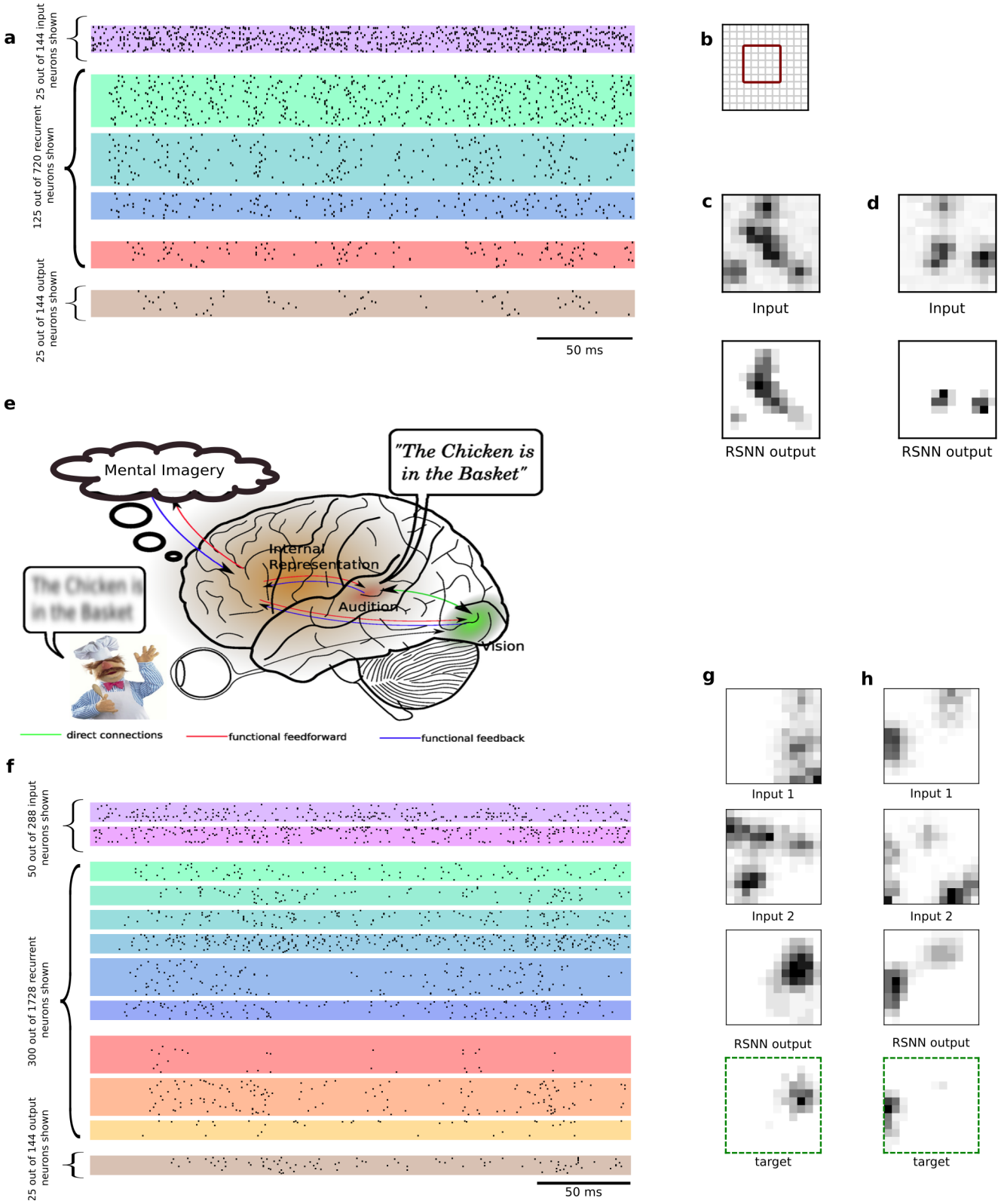


Figure 5: |Generic 2D computations: Local contrast enhancement and detection of coincidences in two input streams. **a, c** Spike raster plot of an RSNN sample solving the local contrast enhancement task, for the 2D input pattern shown at the top of panel c, producing the 2D output shown at the bottom of panel c. Grey values were encoded through Poisson firing rates. **b** The red box in the 12x12 grid indicates the 5x5 pattern size for which the probabilistic skeleton had been optimized, the 12x12 grid the pattern size on which its RSNN sample had been tested. **d** A different 2D input and output pattern for the same RSNN as in c. **e** Motivation of the coincidence detection task as fundamental computational primitive of generic brain computations on expectations and multi-modal sensory inputs; figure adapted from [18]. **f** Spike raster of an RSNN sample of the probabilistic skeleton for solving the 2D coincidence detection task. **g** Pairs of 2D input patterns, for which this RSNN produced the 2D output patterns shown below them. Target output is shown at the bottom. **h** Another sample for the same RSNN.|

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

Innate rudimentary motor control capability, for example, to stand up and walk right after birth, is essential for survival in many species. In contrast to the previously discussed computing tasks, biological motor control requires a transformation of multiple spike-input streams -that represent sensory inputs and feedback- into multiple time-varying spike output streams that control mussels in a closed loop, hence in real-time. We chose a standard benchmark task for motor control: Enabling a quadruped ("ant") to walk by controlling the 8 joints of its 4 legs through a suitable stream of torques. The RSNN controller received 9 spike input streams that encoded -with a delay of 60 ms to make the task more challenging and biologically realistic- through population coding 9 dynamically varying variables: The angles of the 8 joints as well as the vertical position of the torso, see Fig. 6a. Further information about population coding can be found in the Suppl. in section 2.7 We found that a probabilistic skeleton with just 15 types of neurons in the recurrent network, specified by 635 parameters, see Fig. 6b, is able to encode this motor control capability. We refer to movie of the ant locomotion for the resulting locomotion of the quadruped when its joints were controlled by an RSNN sample from this probabilistic skeleton. One can see in the input/output sample shown in Fig. 6d that the computational transformation which this task requires is quite complex. A sample spike raster of this RSNN in Fig. 6c shows that the population coding of the continuous-valued input variables induced a rather complex spatial dynamics of firing activity in most of the neuron types.

Note that the 635 parameters of the probabilistic skeleton correspond to the number of synaptic weights of a recurrent neural network with 25 neurons. However, this would hardly be able to transform the 9 time series of feedback signals into the required 8 time series of output signals. We employed RSNN samples from the probabilistic skeleton whose recurrent network consisted of 250 neurons. Direct tuning of their synaptic weights for this control task would result in a $114,500$ dimensional encoding of the control

algorithm. The compressed encoding of the control strategy enhanced the robustness of the RSNN controller: After randomly deleting 30% of the recurrent and output neurons of the RSNN, it was still able to control the ant locomotion, although the ant was walking somewhat slower, see (Movie of ant after 30% deletion). Altogether we have seen in this section that also demanding real-time computations in a closed loop with the environment, as required for locomotion, can very well be genetically encoded, through means that are already known.

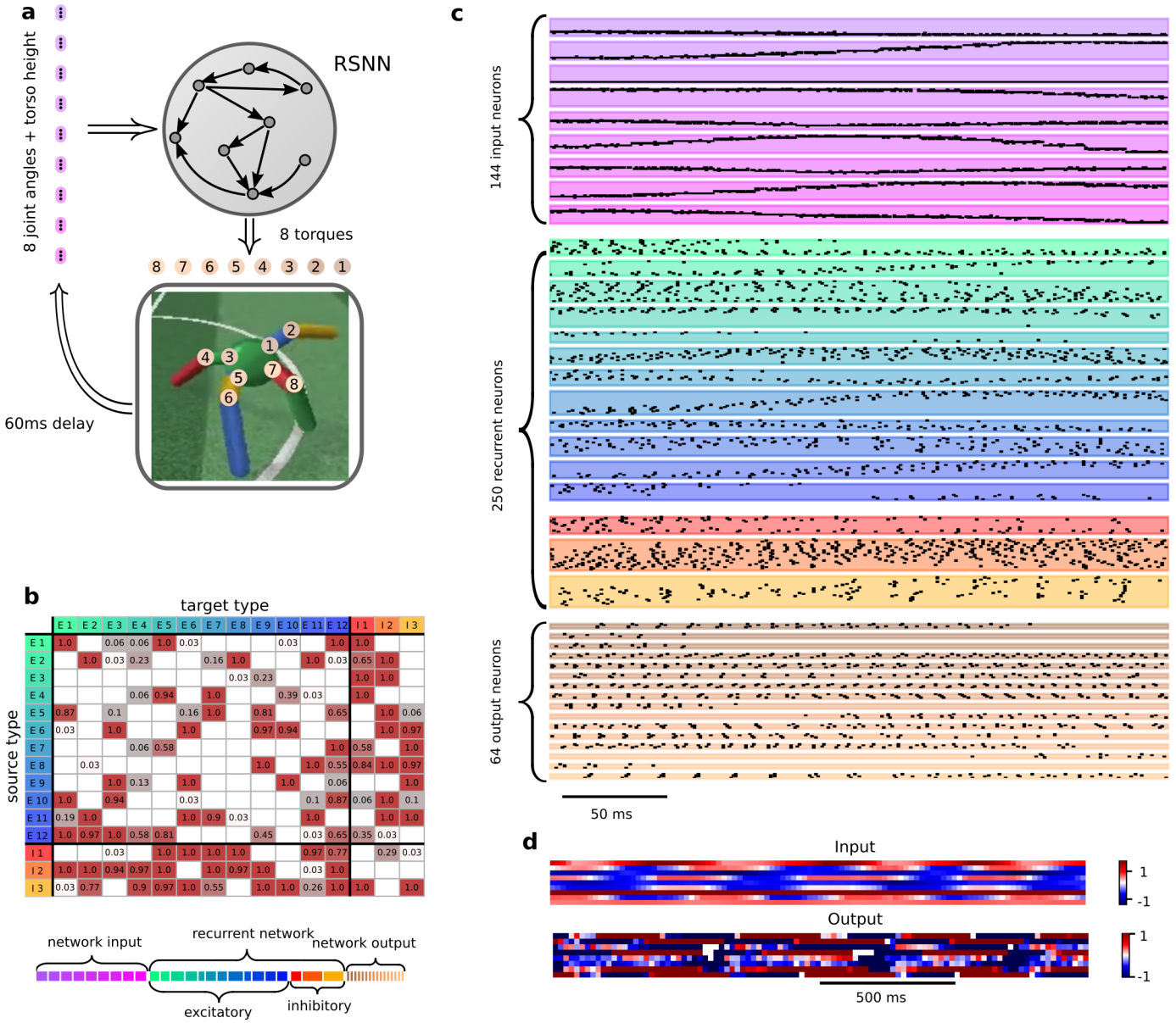


Figure 6: |Example for innate motor control capability through a probabilistic skeleton. **a** System architecture, indicating network inputs and outputs, as well as the 8 joints that are controlled by the RSNN outputs. **b** Probabilistic skeleton for solving this motor control task (base connection probabilities for its numerous input- and output neuron types are shown in the Suppl., Fig S5) **c** Spike raster of an RSNN sample with 458 neurons drawn from this probabilistic skeleton. Population coding of the 9 continuous-valued input variables induced spatially structured firing activity in most of the neuron types. **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 of recurrent neuron types is a new measure for gauging the complexity of the connection structure of neural networks that results from the probabilistic skeleton approach. Fig. 7a shows that the performance of RSNN samples tends to depend above some rather small threshold very little on the exact number of neuron types for which the probabilistic skeleton has been optimized. Note that we excluded the motor control task from the control experiments in this section because of its substantially higher computing demands.

It is reasonable to assume that the connectivity structure of neural networks in the brain was not optimized for a specific number N of neurons, both because of differences in brain sizes between and within species, and because of changing network sizes during ontogeny. Fig. 7b shows that even if the probabilistic skeleton is optimized just for a single number N of neurons in RSNN samples (indicated by a star), computing capabilities that are induced through a probabilistic skeleton tend to generalize to substantially larger neural networks. The decays in performance for large values of N arise partially because neural representations and spatial distributions of network inputs did not grow accordingly. If these grow with the network size, and even more if the probabilistic skeleton is optimized for RSNN samples of different sizes, even better invariance of computational performance can be expected.

Biological neural networks are subject to continuously ongoing spine motility, i.e., pruning of synaptic connections and generation of new connections, both spontaneously and during learning [19, 20] Hence we wondered whether probabilistic skeletons provide robustness to random changes in the strengths of synaptic connections. Fig. 7c shows that the computational performance of RSNN samples is little affected by pruning up to 10% of their synapses, and decays gracefully for more radical pruning. If pruning is accompanied by simultaneous random generation of new synaptic connections, as suggested by the experimental data for biological neural networks, robustness of performance becomes even better. If one changes the number of synaptic connections for each connected pair of neurons randomly up or down, the RSNN samples tolerated perturbations of up to 75% in the number of synaptic connections for each pair of neurons without significant changes in performance, see Fig. 7d.

Another characteristic feature of neural networks of the brain is that they have to carry out their computations in the presence of a substantial amount of noise in synaptic transmission, caused by the fact that the probability of presynaptic release of a vesicle of neurotransmitters in response to a spike of the presynaptic neuron is in the neocortex typically around 0.5 [21]; in other words, synaptic transmission is highly unreliable. Since neural networks that have been trained in the traditional way typically have problems with that, we wondered whether probabilistic skeletons endow RSNN samples with more robustness to noise in synaptic transmission. Fig. 7e shows that their computing capability is little affected by random changes up to 75% in the strengths of individual synaptic connections for each presynaptic spike, and decays gracefully for even larger noise amplitudes. This result suggests that encoding computational function through

the architecture, rather than through individual synaptic weights, automatically endows neural networks in the brain with high tolerance to continuously ongoing noise due to spine motility. Altogether the results shown in Fig. 7 illustrate a new range of properties that emerge in RSNNs when one installs their basic computational capabilities by methods that appear to be employed by nature.

A further inherent property of RSNN samples from probabilistic skeletons is that their number of synapses and total wire length grow linearly with the number of neurons (see Suppl. section 2.5 and 2.6), rather than quadratically as in most neural network designs. Furthermore, since these RSNN samples have a well-defined spatial structure, one can relate the number of neurons per square mm, the number of synapses per neuron, and their total wire length to experimental data. It turns out that they stay in all three counts below the measured numbers from the neocortex, by a 1-3 orders of magnitude (see Methods). One reason is that the number M of neurons in a minicolumn was in our examples well below the 80 -120 neurons that are typically found in a neocortical minicolumn. Also, the number K of neuron types stayed in our examples well below the 111 neuron types identified in mouse V1 [11]. Hence brain data are consistent with a hypothesis that a generic cortical microcircuit can be understood as a superposition of several RSNN samples, each from a different probabilistic skeleton that endows the microcircuit with an additional innate computing capability.

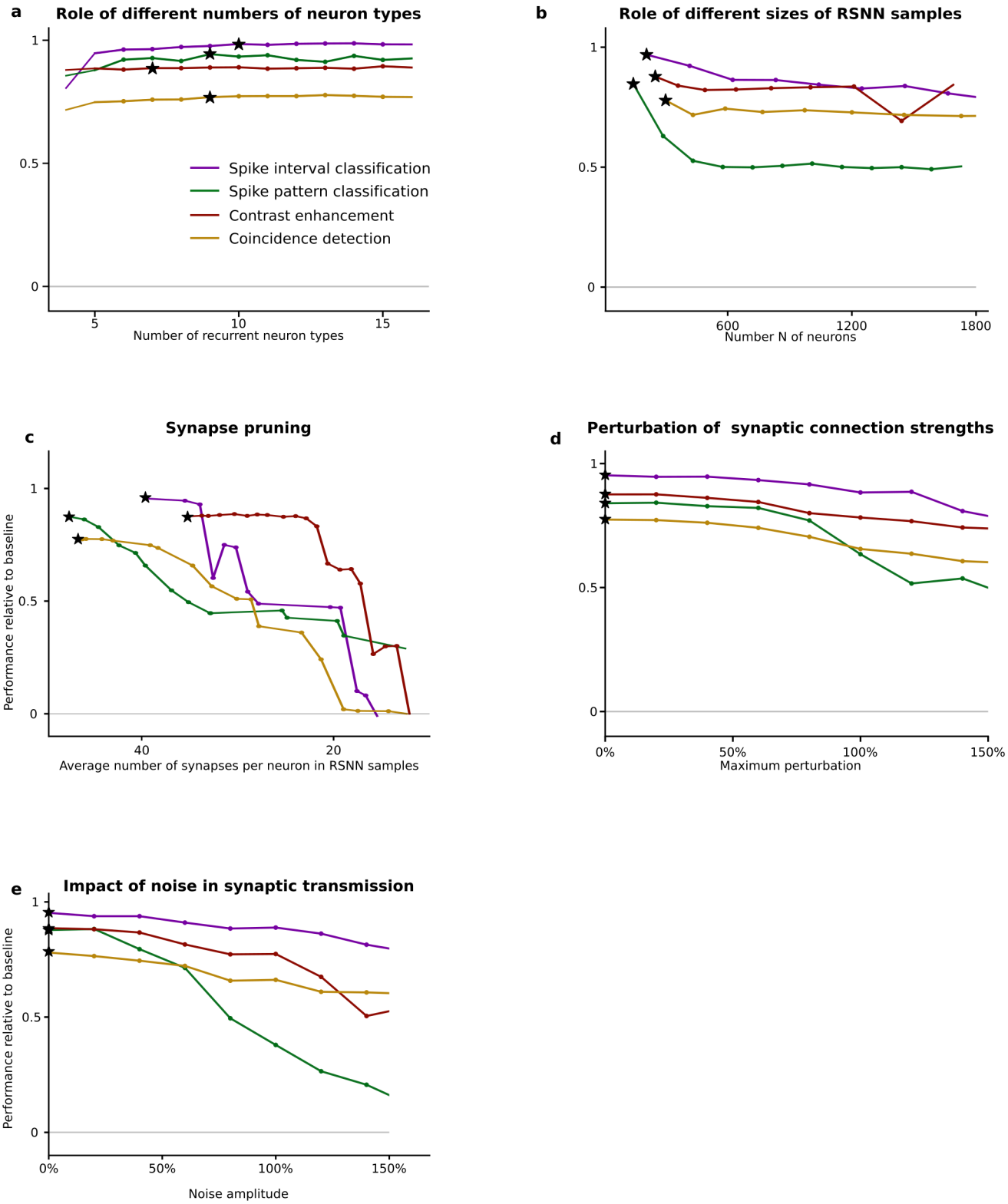


Figure 7: |General functional properties of RSNN samples from probabilistic skeletons. Performance of RSNN is measured relative to random guessing as a common baseline for all tasks, see Methods for details. **a** Mean performance values achieved by optimizing probabilistic skeletons with different numbers of recurrent neuron types. **b** 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. **c** Degradation of 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 marks the number of synapses per neuron before pruning. **d** Performance after random perturbation of the strengths of synaptic connections, plotted as a function of the maximal amount of change in the number of synaptic connections for each neuron pair (i,j), expressed as a fraction of the original number m_{ij} of synaptic connections. **e** Performance in the presence of independent noise in synaptic transmission for each presynaptic spike and each synapse. Noise amplitude was measured similarly as for panel d; see Methods for details. |

3 Discussion

We have shown that a simple mathematical model for genetically controlled neural network features suffices for endowing RSNNs with substantial computing capabilities, in particular with standard examples of innate computing capabilities. This method provides a highly compressed indirect code for specifying network function that is diametrically opposed to the standard procedure in machine learning, where computing capabilities are directly induced in the very large parameter space of synaptic weights through extensive training on huge datasets, starting from a tabula rasa initial state. Our results imply that a largely overlooked structural difference between biological and artificial neural networks plays an essential role for that: Biological neural networks consist of a fairly large number of genetically different neuron types, whereas artificial neural networks usually consist of just one or very few neuron types. Surprisingly, this structural difference is relevant even if one ignores the diverse morphological and neurophysiological features of different biological neuron types: Just the possibility to define network architectures in terms of connection probabilities between different neuron types provides a powerful method for programming computational function into neural networks.

The concept of a probabilistic skeleton provides a rigorous basis for exploring this biological link from network architecture to network function. It defines a probabilistic generative model that can generate an unbounded number of neural networks -with different numbers of neurons and different synaptic weights- whose architectures share some functionally salient features on the statistical level. Arguably, such a probabilistic model better captures the impact of the genetic code on network function than deterministic models. They also raise an interesting theoretical question: Are there functionally desirable network architectures that can be generated with significant probability by a

suitable generative model, but are hard to construct deterministically? It is already known that certain graphs with attractive information routing capabilities, such as expanders and concentrators, are hard to generate through deterministic wiring rules, but have a non-zero probability of being generated through probabilistic wiring rules [22].

Neural network samples from probabilistic skeletons differ in another aspect from commonly considered network architectures: Their number of synapses and total wire length grow just linearly with the number of neurons. This appears to be essential for physical implementations of large neural network instances, both in brains and in neuromorphic hardware, but is rarely met by existing design methodologies. It arises in samples from probabilistic skeletons through extensive uses of topographic maps between different neuron types and external inputs, which enable an interesting style of 2D in-place-computing.

Our concept of a probabilistic skeleton only aims at capturing a minimal fragment of the functionally relevant genetically encoded program for neural network architectures. It will be interesting to see which further computing capabilities of neural networks emerge if one allows neurons of different types to have different electrophysiological and/or morphological properties, as they have in the neocortex [9, 10]. We found that allowing different neuron types to use different $GLIF_3$ point neuron models of [10] did not enhance performance for the tasks that we considered. Additional extensions of the concept of a probabilistic skeleton can phase in different spatial decay constants for connection probabilities for different neuron types, as indicated in Fig. 4 of [10], as well as selected long-range connections between spatially separated network modules. Furthermore, it is well-known that cortical microcircuits employ not only genetically different types of neurons, but also many genetically different synapse types [23], [24]. Different synapse types were found to exhibit important differences in their short term [25], [26] and long term dynamics [27]. Hence they are likely to affect both computing capabilities and learning aspects of neural network samples.

Probabilistic skeletons provide an approach for meeting the challenge of [1]: Understand the functional impact of the "genomic bottleneck", i.e., of the fact that the number of bits which the genome uses for encoding the neural networks is really small in comparison with their number of synapses. This challenge has also been addressed by other recent work. [28] models its impact on a more abstract level, assuming that the existence of synaptic connections can be inferred deterministically from binary codes for neurons through linear operations. The model of [29] is less abstract than ours and that of [28]. It is based on individual neuron-to-neuron compatibility rules based on transcription factors that are especially salient in smaller brains. A functional advantage of the genomic bottleneck was demonstrated in both of these approaches in terms of enhanced generalization capabilities of trained feedforward artificial neural networks. We have demonstrated in Fig. 7 implications of the genomic bottleneck for more brain like recurrent networks of spiking neurons, such as robustness to changes in the number of neurons or synapses, changes in the strengths of synaptic connections, and noise in synaptic transmission. This inherent robustness should be contrasted with the inherent brittleness of deep neural networks [30]. It also suggests a new approach for the design of neuromorphic chips that are based on extremely energy-efficient but imprecise mem-

ristors or other novel physical devices [31]. Another possible technological application arises in the domain of organoids [32].

Probabilistic skeletons provide a biologically derived method for indirect encoding of functional neural networks. It will be interesting to compare this biological style of indirect encoding with approaches that have recently been proposed in machine learning, especially in the context of meta-learning [33] [34]. Their style of indirect encoding is also likely to produce nice expansions of indirect coding approaches in the area of neuroevolution [35, 36].

4 Methods

Neuron types

There are 3 kinds of neuron types: 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. Each recurrent neuron type consists only of excitatory neurons or only of inhibitory neurons. Note that input or output types only consist of excitatory neurons.

Neuron and synapse models

Recurrent and output neurons are modelled as discrete-time versions of standard Leaky-Integrate-and-Fire (LIF) neuron models, More precisely of the GLIF_1 model from [37]. The definition of the continuous neuron model can be found in the Suppl. in section 2.1. Control experiments with the GLIF_3 model from [10] produced qualitatively similar results.

For the discrete time version of neuron $j \in \{1, \ldots, N\}$ 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

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

$$\tag{2}$$

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

$$z_j(t) = H\left(V_j(t) - v_{th}(t)\right) \tag{3}$$

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.

The previously defined neuron model use the following set of 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 [10], and the raw data is available in [38]. 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.

The same synapse model as in [10] has been used. Additional information about the synapse model can be found in the Suppl. in section 2.2.

Details to the definition of a probabilistic skeleton

A probabilistic skeleton consists of

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

(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 p_I of each neuron type I, i.e., a number representing the fraction of neurons belonging to type I in a generic minicolumn, see the bottom plot of Fig. 1c. Further details can be found in the Suppl., section 2.3.

(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 that controls the decay of connection probabilities with the horizontal distance between somata.

A probabilistic skeleton is a generative model, which defines a distribution over neural networks of different sizes and with different synaptic connections that share common architectural features.

One samples a neural network from a probabilistic skeleton according to the following rules:

1. Pick a number n_{mcol} of minicolumns and a number $M \geq K$ of neurons per minicolumn. This determines the number of neurons $N = n_{mcol} \cdot M$ in the sample network.

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:

$$\mathbb{P}[\text{Synapse from i to j}] = p_{I \to J}\, e^{-\frac{\text{Dist}(i,j)^2}{\sigma^2}} \quad . \tag{4}$$

   This yields the number m_{ij} of synaptic connections from i to j.

The functional form of the dependence of connection probabilities on $\text{Dist}(i, j)$ approximates the corresponding data from [10], see panels b and d in Fig. 1. 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 [9] most synaptically connected neurons do in fact have multiple synaptic connections. The effective strength (weight) 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 m_{ij} of synaptic connections from i to j that results from drawing $S = 8$ times from the distribution given in equ. (4).

Optimization method

Probabilistic skeletons were optimized for specific computing tasks with the Separable Natural Evolution Strategy (Separable NES), which had been introduced in [15]. The algorithm is given below in pseudo code. For the optimization of the d-dimensional vector $\boldsymbol{\theta}$ of parameters of the probabilistic skeleton 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 adapts the Gaussian distributions to capture more of those parts of the parameter space where the fitness of the offsprings is higher. The fitness function F depends on the computational task for which the probabilistic skeleton is optimized. 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 [39] and [40]. The learning rate for $\boldsymbol{\sigma}$ was chosen as $\eta_\sigma = 0.01$. As suggested in [40] mirrored sampling has been employed, see, e.g., [41]. 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 k=1,...,$\lambda/2$ **do**
      Init $\mathbf{s} \in \mathbb{R}^{(\lambda,d)}$ as $\mathbf{s}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{s}_{(k+\lambda/2)} = -\mathbf{s}_k$
      $\boldsymbol{\theta}_k \leftarrow \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \mathbf{s}_k$
      Compute Fitness $F(\boldsymbol{\theta}_k)$
    end for
    Compute gradients
    $\nabla_{\boldsymbol{\mu}} \leftarrow \sum_{k=1}^{\lambda} F(\boldsymbol{\theta}_k)\mathbf{s}_k$
    $\nabla_{\boldsymbol{\sigma}} \leftarrow \sum_{k=1}^{\lambda} F(\boldsymbol{\theta}_k)(\mathbf{s}_k^{\mathrm{T}}\mathbf{s}_k - 1)$
    Update parameters
    $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + \eta_{\boldsymbol{\mu}}\boldsymbol{\sigma}\nabla_{\boldsymbol{\mu}}$
    $\boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma} \exp\{\frac{\eta_{\boldsymbol{\sigma}}}{2}\nabla_{\boldsymbol{\sigma}}\}$
  end for

For the optimization of the base connection probabilities $p_{I\to J}$ one needs to make sure that they are always assigned values in $[0, 1]$. For that purpose real valued auxiliary parameters $\kappa_{IJ} \in \mathbb{R}$ are optimized, from which the base connection probabilities are obtained by using the sigmoid function:

$$p_{I\to J} = \frac{1}{1 + \mathrm{e}^{-\kappa_{IJ}}}. \tag{5}$$

The value of the number K of neuron types and of the scaling parameter σ from equation (4) were optimized through a separate hyperparameter search.

Details to the construction of the connectivity graph (chord diagram) of a probabilistic skeleton

The chord diagrams in Fig. 3c and Fig. 4c are drawn in the style of Fig. 7C of [9], using the base connection probabilities $p_{I\to J}$. The thickness of each chord reflects the value of the corresponding base connection probability and the length of each segment (arc) on the perimeter is proportional to the expected fraction of synapses from and to the corresponding neuron type. Such a chord graph of a probabilistic skeleton should not be confused with the connectivity graphs of individual RSNN samples from this probabilistic skeleton. The latter also depends on the selected number N of neurons, the value of σ, and numerous outcomes of random drawings. More precisely, the arc length of the type I is proportional to:

$$\frac{\sum_J p_{I\to J} + \sum_J p_{J\to I}}{2\sum_{IJ} p_{I\to J}}$$

.

Experiments

Details to computations on spike times.

Task description: The goal is here to classify the temporal distance between two waves of input 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 wave of spikes at the beginning of a 200ms long trial, and the second wave at any other time during the trial. For each input neuron some Gaussian noise with mean zero and variance 1 ms has been added to the spike times to avoid that all input neurons spike at the exact same time.

Performance measure: The percentage of correctly classified distances between the two waves of input spikes is used as a performance measure. The standard deviation of the performance on this and the subsequent tasks was obtained by averaging over the performance of 50 different RSNNs sampled from the same probabilistic skeleton evaluated on 100 inputs each.

Fitness function: Best optimization results are achieved when a different fitness measure than accuracy is used. To compute the fitness the softmax function was applied to the vector $(r_1, r_2, r_3, r_4)^T$ of spike counts of the 4 output neuron types during the last 30ms of a trial to obtain the class probabilities $(p_1, p_2, p_3, p_4)^T$. To compute the fitness the target class y was first one-hot encoded to the target class vector $\mathbf{y}$, i.e. to a vector where all entries are 0 except the element at position $y - 1$, which has the value 1. An example of one-hot encoding can be found in the Suppl., section 2.4. The fitness function is given by the negative cross entropy loss. For a single example with one-hot-encoded target class $\mathbf{y}$ the fitness is defined as:

$$F(\boldsymbol{\theta}) = \sum_{k=1}^{4} y_k \log(p_k). \tag{6}$$

Details of the probabilistic skeleton and its optimization process: A decay constant of $\sigma = 77.7$ was used for this task. The scaling parameters for synaptic strengths were $w_{in} = 14.6$, $w_E = 15.49$, $w_I = 6.92$. The 304 neurons of RSNN samples during optimization were arranged in $n_{mcol} = 16$ minicolumns on a 4x4 grid, where $M = 19$. In every minicolumn there are two input neurons and there is one output neuron per type.

During the optimization of the probabilistic skeleton the activity of output neurons was not only considered during the last 30ms. Instead, initially all spikes of output neurons were counted during the full 200ms of a trial. In the course of the optimization this period was gradually reduced to the last 30ms.

Details to installing in RSNNs the capability to recognize specific spike patterns?

Generation of spike inputs: Two clearly distinct ensembles of Poisson spike trains from 4 neurons with a rate of 50 Hz were frozen as templates. Spike input patterns of classes 1 and 2 were generated by creating variations of these spike templates: For every input neuron two time steps from the first 50ms were chosen, and a new spike was inserted at them or the spike was deleted if there was a spike at this time step. Subsequently the spike times of all spikes in the template were shifted by a random amount drawn from a Gaussian with mean zero and variance 0.5 ms and rounded to the nearest integer value. The third class consisted of random Poisson spike trains over 50ms with a rate of 50 Hz.

Input: The network received as input a spike pattern of 4 input neurons over 50ms from one of the three classes, drawn with uniform probability from the three classes.

Performance Measure: The same performance measure as for the preceding task was used.

Fitness function: A corresponding fitness function as for the preceding task was used.

Details of the probabilistic skeleton and its optimization process Parameters $w_{in} = 14.38$, $w_E = 7.85$, $w_I = 7.90$ and $\sigma = 129.73$ were used. RSNN samples that were tested during the optimization of the probabilistic skeleton consisted of 148 neurons, which were arranged in a 3x4 grid of $n_{mcol} = 12$ minicolumns, each minicolumn consisting of $M = 12$ neurons. There was one input neuron in every corner of the grid, hence the corresponding columns had one neuron more than M.

Details to local contrast enhancement

Task description: In this task 2D patterns $\mathbf{x} \in [0,1]^{\sqrt{n_{mcol}} \times \sqrt{n_{mcol}}}$ are presented to the network, where n_{mcol} is the number of minicolumns; these are arranged in a square grid. For optimizing the probabilistic skeletons, $n_{mcol} = 25$ was selected, resulting in a 5x5 grid.

Input generation To generate a pattern $\mathbf{x}$ first a fixed number N_{points} of coordinate pairs $(m,n) \in \{2,\ldots,\sqrt{n_{mcol}}-1\}^2$ were drawn randomly. In a second step, the value 1.0 was assigned to these points, all other points were 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. $N_{points} = 2$ was used for 5x5 input patterns. The resulting 2D patterns were presented to the RSNN through corresponding 2D arrays of Poisson spike trains. Their rates between 0 and 200 Hz were linearly scaled by the grey-values of the corresponding grid cells of the 2D pattern. The length of a trial was 300 ms.

Targets for the computation: Target values were computed by applying the Mexican hat filter defined below, with symmetric padding to the 2D input pattern, followed by clipping at zero and normalization to $[0, 1]$. The Mexican was defined by:

$$\text{filter} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{7}$$

Output: There were n_{mcol} output neurons, one in each minicolumn, which were arranged in a square grid. The output value of each output neuron i was computed by counting its spikes and normalizing that value:

$$\hat{s}_i = \sum_t z_i(t), \tag{8}$$

$$s_i = \frac{\hat{s}_i}{\max(\hat{s})}. \tag{9}$$

Performance Measure and fitness function: Fitness was the normalized cross correlation between target signal $\mathbf{y}$ and output signal $\mathbf{s}$:

$$F(\boldsymbol{\theta}) = NCC = \frac{\sum_{k=1}^{n_{mcol}} s_k y_k}{\sqrt{\sum_{k=1}^{n_{mcol}} s_k^2 \sum_{k=1}^{n_{mcol}} y_k^2}}$$

This value 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 identical. Hence we used it also for measuring performance.

Details of the probabilistic skeleton and its optimization process: The probabilistic skeleton, shown in Fig. 8a, had $K = 4$ types and was optimized for RSNN samples with $n_{mcol} = 25$ minicolumns (arranged in a square grid) and $M = 8$ neurons per column, resulting in $N = 200$ neurons of RSNN samples. Other parameters of the probabilistic skeleton were $w_{in} = 1.70$, $w_E = 1.71$, $w_I = 2.15$, and $\sigma = 73.6$.

Details to the coincidence detection

Task description: Two 12x12 input patterns were simultaneously presented to the network. The goal was to mark positions in the 12x12 grid where both input patterns were reasonably strong, using a product operation.

Input: The input patterns were generated using the same algorithm as for the contrast enhancement task, with the only difference that a minimum distance of 3 was

imposed on the points used to generate the input pattern.

Computing the targets: The target outputs of the coincidence detection task were computed by the following formula:

$$\mathbf{y} = \frac{\boldsymbol{\phi}(\mathbf{x}_1 \odot \mathbf{x}_2)}{\max(\boldsymbol{\phi}(\mathbf{x}_1 \odot \mathbf{x}_2))} \tag{10}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are the two input patterns. $\odot$ refers to the element wise product, and ϕ is a function that returns element wise the identity of the input if the element is higher than the average of all vector coordinates of the input vector, else it returns zero, as described in equation 11:

$$\boldsymbol{\phi}(\mathbf{v}) = \begin{pmatrix} \phi(v_1) \\ \vdots \\ \phi(v_n) \end{pmatrix}, \quad \phi(v_k) = \begin{cases} v_k & v_k \geq \text{mean}(\mathbf{v}) \\ 0 & \text{else.} \end{cases} \tag{11}$$

Output, performance measure and fitness function: The same output convention and fitness function, and performance measure as for the contrast enhancement task were used.

Details of the probabilistic skeleton and its optimization process: The probabilistic skeleton, shown in Fig. 8b, used $K = 12$ neuron types and $M = 15$ neurons per mini-column. Its other parameters were $w_{in} = 2.28$, $w_E = 3.08$, $w_I = 1.92$, and $\sigma = 70$.

Note, that in the coincidence detection task, the target output is not the sum of two input values, but their product. This is salient, since simply computing a sum and thresholding its value is a trivial task for an RSNN. However, approximating a product is substantially more challenging. In Fig. 8c it becomes apparent that simply computing the sum would not be a good strategy, as it would yield a low fitness value.

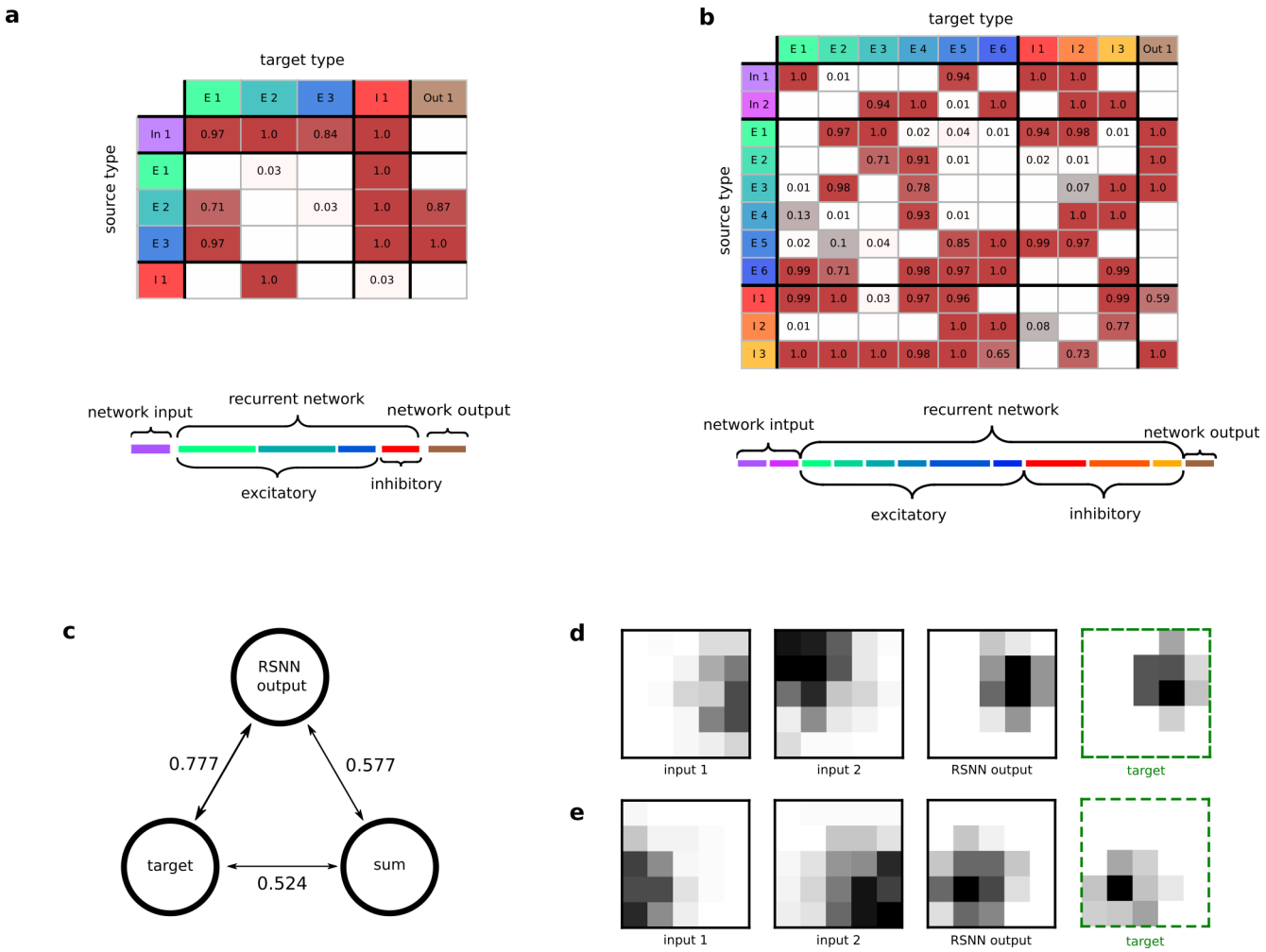


Figure 8: |Details to the induction of 2D computing capabilities through probabilistic skeletons. a Probabilistic skeleton for local contrast enhancement task. **b** Probabilistic skeleton for the coincidence detection task. **c** Normalized cross entropy between the output of the RSNN for the coincidence detection task, the target and the average of the two input patterns. This shows that approximating the coincidence detection target with a linear operation yields a substantially smaller correlation. **d** and **e** show 2D input patterns for the coincidence detection task along with the RSNN output and the target output. A 5x5 grid was used in this figure, the format for which the probabilistic skeleton had been optimized.|

Details to innate motor control capabilities through probabilistic skeletons

Task description: For the simulation of the environment (AntMuJoCoEnv-v0) the PyBullet physics engine [42] was used. The agent is a quadruped walker and is usually referred to as 'ant' in the literature. It consists of four legs with four joints, which are

attached by another four joints to a torso, modelled as a sphere. The center of the sphere defines the location of the plant on a 2D plane. The goal of this task is to achieve a high movement speed over the whole trial period, while also avoiding to touch the ground. An episode is terminated if the center of its torso moves below a height of 0.2m, or if the maximum number of time steps has been reached.

Spatial structure of RSNN samples The population coding of continuous-values input variables induced a prominent 1D dynamics in populations of input neurons, and there seems to be no natural way to map these 2D input arrays properly into a 2D structured RSNN for computational processing. For this reason, and because such basic motor control capabilities are likely to be encoded in the spinal cord and other subcortical structures, we used for this task a 1D arrangement of neurons in order to define their spatial distances, rather than neocortical minicolumns. More precisely, the neurons of input and recurrent types were evenly-spaced distributed over a 1D line segment $[0, 660]$ µm. The locations of output neurons, organized for each output variable into two output types consisting of 4 neurons at the same location (see below), were optimized alongside the other parameters of the probabilistic skeleton. The distance measure $\mathrm{Dist}(i, j)$ for neurons i and j was computed as the absolute value of the difference between their 1D coordinates.

Input: Time in the simulated environment was discretized to time steps of 17 ms length. For this reason the network received each continuous-valued input value for 17 ms through population coding in one of the 9 input neuron types, each having 16 neurons. It should be noted that population coding is commonly employed in the brain to encode continuous-valued variables [43]. The input was provided by the current state of the simulated environment. Its state space was 111 dimensional. We excluded most of them, for example angular velocities, to have a more compact and arguably biologically more realistic network input.

Output: The action space of the controller is given by $\mathcal{A} = [-1, 1]^8$, which corresponds to 8 torques applied to the 8 joints of the ant. An output torque $y \in [-1, 1]$ of the model is computed by using two output neuron types, each consisting of 4 output neurons, representing negative and positive torques to a joint, denoted by J_- and J_+. This corresponds to motor commands in the form of firing rates to 2 antagonistic muscles for a joint. Firing activity of output neurons of the RSNN were decoded as signal to the simulated environment by computing the normalized linear combination of the spike rates over a 17 ms time step of the environment:

$$y = \frac{\sum_{t=1}^{17} e^{-\frac{17-t}{\tau_{out}}} \left(s^{J_-}(t) - s^{J_+}(t)\right)}{\sum_{t=1}^{17} e^{-\frac{17-t}{\tau_{out}}} \max\left(s^{J_-}(t), s^{J_+}(t)\right)}, \tag{12}$$

where $\tau_{out} = 10$.

Performance measure and fitness function: The performance measure was the same as the fitness value. The fitness was given by the total reward received from the environment, summed up over time. At every time step of 17ms length the agent received a reward

$$F(\boldsymbol{\theta}) = v_{\text{fwd}} - 0.1 j_l + 1, \tag{13}$$

where v_{fwd} is the velocity of the center of the ant in the x direction, $j_l :=$ number of joints which are at the limit. A constant reward of 1 was added for each time step in order to induce long lasting locomotion without premature abortion of an episode because the torso touched the ground.

RSNN samples with 458 neurons from the optimized probabilistic skeleton produced an average fitness of 517 (standard deviation of 51.85) using 250 steps in the environment, where the average was computed over 100 trials. The version of the model where 30% of the recurrent and output neurons are randomly deleted achieved a fitness of 421.

Details of the probabilistic skeleton: The probabilistic skeleton consisted of $K = 40$ types, and was optimized for RSNN samples with $N = 458$ neurons. Every input type was constrained to only form connections to one recurrent type, which did not receive synaptic inputs from another input type. The other parameters were $w_{in} = 4.75$, $w_E = 4.5$, $w_I = 2.3$ and $\sigma = 80.0$.

Note: The version of the ant locomotion task that we considered differed somewhat from the version that is commonly considered in the literature [44]. There one does not assume a delay in feedback from the environment. Also, the more limited observation space that we used made it harder for the model to know in which direction it was facing, especially at a later point in the trial. This made it harder to move especially along the x-axis, which was the only direction in which locomotion was rewarded.

Details to Figure 7 (General results and principles)

To compare the different tasks it is necessary to use for different computing tasks a common performance scale. This can be achieved by defining the baseline for every task as the performance level of a random output. For example, the computations on spike times task required a decision between 4 classes, hence picking a random class would give for a uniform distribution of classes an expected accuracy of 25%. Analogously the baseline accuracy for the spike pattern classification, which involves three classes, is 33.33%. For the contrast enhancement task and the coincidence detection task it is possible that a probabilistic skeleton has a fitness of 0.0, hence this was chosen to be the baseline for these tasks.

The performances on these different tasks were scaled by calculating for each task the difference between the theoretically best possible performance (either accuracy or normalized cross correlation) to the baseline performance and normalizing this difference to $[0, 1]$.

Panel a: For each number of recurrent neuron types 80 probabilistic skeletons were optimized for every task, and the best performing ones were used for the plot.

For panel b the number of neurons N was varied by increasing the number of minicolumns n_{mcol}. The number of neurons per column M was not changed.

The pruning experiments for panel c were done by deleting synapses that had 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 a value on the x-axis.

Panel d: Each value m_{ij} was perturbed by a positive or negative value drawn uniformly from the range around m_{ij} bounded by the fraction marked on the x-axis. The resulting value was rounded to the next non-negative integer.

For panel e, the effective weight of each individual synapse was independently perturbed for each presynaptic spike. The amplitude of this perturbation was measured as fraction x of its current value, and the maximal fraction is indicated on the x-axis of the panel. For each value of x the noise value was drawn uniformly from the interval $[-x, x]$. The resulting perturbed weight was set to zero if the perturbation caused its sign to change.

Details to the comparison of neuron density, synapses numbers, and wire length with experimental data from the neocortex. According to Fig. 2B of [45] the number of neurons under a square mm of the neocortical sheet is in the mammalian brain around 100,000. The number of synapses per neuron was estimated in [46] to be 7777, and the total length of axons per neuron was estimated to be 4.4cm. We have compared these experimental data with corresponding estimates that arise for RSNN samples from probabilistic skeletons for the computing tasks that we considered (see Table 1 in the Suppl.). For example, the RSNN for coincidence detection, whose firing activity and performance was shown in Fig. 5 f-h, has 2160 neurons, occupies a square patch of $0.5184 mm^2$, has $360,100$ synapses, and a total wire length of 17.5m. Thus its number of neurons per square mm is by a factor 22 smaller than in the mammalian brain, the number of synapses is by a factor 1008 smaller, and its total wire length is by a factor 118 smaller than in the data. Thus, these numbers are in a reasonable range, but significantly smaller than in the experimental data. The main reason for that is that the number of neuron types that are needed for each of the computing tasks that we considered is substantially smaller than the estimated 111 neuron types in mouse V1 [11]. Consistent with that, the number M of neurons in a minicolumn was in our examples well below the 80 -120 neuron in a typical neocortical minicolumn. Note that the number of synapses and total wire length grow superlinearly with the number of neuron types, (see Suppl. section 2.5 and 2.6). In addition, we only counted wire length in the horizontal direction, and ignored long-range connections.

Acknowledgements

We would like to thank Anton Arkhipov, Dániel Barabasi, Guozhang Chen, Peter Jonas, Eben Kadile, Robert Legenstein, Jason MacLean, Risto Miikkulainen, Franz Scherr, Kenneth Stanley, and Yuqing Zhu for helpful comments on a prior version of this manuscript. This research was partially supported by the Human Brain Project (Grant Agreement number 785907) of the European Union. Computations were carried out on the Human Brain Project PCP Pilot Systems at the Juelich Supercomputing Centre, which received co-funding from the European Union (Grant Agreement number 604102) and on the Vienna Scientific Cluster (VSC).

Author contributions

WM and CS designed the approach, CS and DL carried out the experiments and analyzed the results, WM, CS and DL wrote the paper.

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] 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.

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

[11] Bosiljka Tasic et al. "Shared and distinct transcriptomic cell types across neocortical areas". In: *Nature* 563.7729 (2018), pp. 72–78.

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

[13] 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.

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

[15] Tom Schaul, Tobias Glasmachers, and Jürgen Schmidhuber. "High dimensions and heavy tails for natural evolution strategies". In: *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. 2011, pp. 845–852.

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

[17] Matthew Larkum. "A cellular mechanism for cortical associations: an organizing principle for the cerebral cortex". In: *Trends in neurosciences* 36.3 (2013), pp. 141–151.

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

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

[20] Anthony Holtmaat and Karel Svoboda. "Experience-dependent structural synaptic plasticity in the mammalian brain". In: *Nature Reviews Neuroscience* 10.9 (2009), pp. 647–658.

[21] Tiago Branco and Kevin Staras. "The probability of neurotransmitter release: variability and feedback control at single synapses". In: *Nature Reviews Neuroscience* 10.5 (2009), pp. 373–383.

[22] Shlomo Hoory, Nathan Linial, and Avi Wigderson. "Expander graphs and their applications". In: *Bulletin of the American Mathematical Society* 43.4 (2006), pp. 439–561.

[23] 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.

[24] 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.

[25] Henry Markram et al. "Interneurons of the neocortical inhibitory system". In: *Nature reviews neuroscience* 5.10 (2004), pp. 793–807.

[26] Maksym V Kopanitsa et al. "A combinatorial postsynaptic molecular mechanism converts patterns of nerve impulses into the behavioral repertoire". In: *BioRxiv* (2018), p. 500447.

[27] Holly J Carlisle et al. "Opposing effects of PSD-93 and PSD-95 on long-term potentiation and spike timing-dependent plasticity". In: *The Journal of physiology* 586.24 (2008), pp. 5885–5900.

[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] 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.

[30] 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.

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

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

[33] Andrei A Rusu et al. "Meta-learning with latent embedding optimization". In: *arXiv preprint arXiv:1807.05960* (2018).

[34] Adam Katona et al. "Utilizing the Untapped Potential of Indirect Encoding for Neural Networks with MetaLearning". In: *Evostar 2021* (2021).

[35] David Ha, Andrew Dai, and Quoc V Le. "Hypernetworks". In: *arXiv preprint arXiv:1609.09106* (2016).

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

[37] 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.

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

[39] Daan Wierstra et al. "Natural evolution strategies". In: *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. IEEE. 2008, pp. 3381–3387.

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

[41] 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.

[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: *arXiv preprint arXiv:1506.02438* (2015).

[45] C Nikoosh Carlo and Charles F Stevens. "Structural uniformity of neocortex, revisited". In: *Proceedings of the National Academy of Sciences* 110.4 (2013), pp. 1488–1493.

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