Sparsity through evolutionary pruning prevents neuronal networks from overfitting

Gerum, Richard C.; Erpenbeck, Andre; Krauss, Patrick; Schilling, Achim

Published in:
Neural Networks

DOI:
10.1016/j.neunet.2020.05.007

Publication date:
2020

Neural Networks 128 (2020) 305–312

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Sparsity through evolutionary pruning prevents neuronal networks from overfitting

Richard C. Gerum [a], André Erpenbeck [b], Patrick Krauss [c,d,e], Achim Schilling [c,d,*]

[a] Biophysics Group, Department of Physics, Friedrich Alexander University Erlangen-Nürnberg (FAU), Germany
[b] The Raymond and Beverley Sackler Center for Computational Molecular and Materials Science, School of Chemistry, Tel Aviv University (TAU), Israel
[c] Neuroscience Lab, Experimental Otolaryngology, University Hospital Erlangen, Germany
[d] Cognitive Computational Neuroscience Group at the Chair of English Philology and Linguistics, Friedrich-Alexander University Erlangen-Nürnberg (FAU), Germany
[e] Department of Otorhinolaryngology/Head and Neck Surgery, University of Groningen, University Medical Center Groningen (UMCG), The Netherlands

ARTICLE INFO

Article history:
Received 8 November 2019
Received in revised form 31 January 2020
Accepted 4 May 2020
Available online 11 May 2020

Keywords:
Evolution
Artificial neural networks
Maze task
Evolutionary algorithm
Overfitting
Biological plausibility

ABSTRACT

Modern Machine learning techniques take advantage of the exponentially rising calculation power in new generation processor units. Thus, the number of parameters which are trained to solve complex tasks was highly increased over the last decades. However, still the networks fail – in contrast to our brain – to develop general intelligence in the sense of being able to solve several complex tasks with only one network architecture. This could be the case because the brain is not a randomly initialized neural network, which has to be trained from scratch by simply investing a lot of calculation power, but has from birth some fixed hierarchical structure. To make progress in decoding the structural basis of biological neural networks we here chose a bottom-up approach, where we evolutionarily trained small neural networks in performing a maze task. This simple maze task requires dynamic decision making with delayed rewards. We were able to show that during the evolutionary optimization random severance of connections leads to better generalization performance of the networks compared to fully connected networks. We conclude that sparsity is a central property of neural networks and should be considered for modern Machine learning approaches.

1. Introduction

Sparsity is a characteristic property of the wiring scheme of the human brain, which consists of about 8.6×10^{10} neurons (Herculano-Houzel, 2009), interconnected by approximately 10^{15} synapses (Hagmann et al., 2008; Sporns, Tononi, & Kötter, 2005). Thus, from almost 10^{22} theoretically possible synaptic connections, only one of 10 million possible connections is actually realized. This extremely sparse distribution of both neural connections and activity patterns is not a unique feature of the human brain (Hagmann et al., 2008) but can also be found in other vertebrate species such as for example mice and rats (Kerr, Greenberg, & Helmchen, 2005; Oh et al., 2014; Perin, Berger, & Markram, 2011; Song, Sjöström, Reigl, Nelson, & Chklovskii, 2005). Even evolutionary very old species with a quite simple nervous system such as the nematode *C. elegans* with only 302 neurons (Jarrell et al., 2012) and over 7000 connections (White, Southgate, Thomson, & Brenner, 1986) show this sparsity. Besides the described sparsity of virtually all nervous systems, many biological neural networks also show small world properties (Amaral, Scala, Barthélémy, & Stanley, 2000; Bassett & Bullmore, 2006; Latora & Marchiori, 2001; Watts & Strogatz, 1998), such as scale free connectivity patterns (Bassett, Meyer-Lindenberg, Achard, Duke, & Bullmore, 2006; Perin et al., 2011; van den Heuvel & Yeo, 2017).

When dealing with the term sparsity in biology and machine learning, it is necessary to distinguish between three different forms of sparsity: "sparse representation", "input sparsity", and "model sparsity" (Kafashan, Nandi, & Ching, 2016). Sparse representation means that only a small amount of neurons respond to certain stimuli. Thus, even a fully connected network can have the property of sparse representation. One famous but controversial example for sparse representation, often called sparse-coding (Zhang, Yang, & Feng, 2011), is the so called "grandmother cell", being a vivid term for the idea that only a single neuron encodes for one highly complex concept (Barwich, 2019; Quiroga, Kreiman, Koch, & Fried, 2008; Rose, 1996). In recent years much effort has been undertaken to achieve sparse coding of certain

* Correspondence to: Neuroscience Group, Experimental Otolaryngology, Friedrich-Alexander University of Erlangen-Nürnberg, Waldstrasse 1, 91054 Erlangen, Germany.
E-mail address: achim.schilling@uk-erlangen.de (A. Schilling).

https://doi.org/10.1016/j.neunet.2020.05.007

input stimuli to improve machine learning (Babadi & Sompolinsky, 2014; Dasgupta, Stevens, & Navlakha, 2017; Jiao et al., 2018; Jin, Zhou, Gao, & Zhang, 2018; Olshausen & Field, 2004; Pehlevan & Sompolinsky, 2014) and on the other hand sparse coding was also investigated in biology (Crochet, Poulet, Kremer, & Petersen, 2011; Zaslaver et al., 2015). Input sparsity, in contrast, means that the input patterns fed to the neural network are sparse. However, in this study we investigate the development of model sparsity in artificial neural networks, being the analogue to a sparse connectome in biology. For reasons of simplicity, we use from now on the term sparsity to refer to model sparsity in the biological sense, as well as in the context of machine learning.

Sparsity in biology is the result of both, phylogenetic and ontogenetic adaptations. Even though, almost all species' immature nervous systems are already very sparse, this sparsity is even further increased during development and maturation of the agents' nervous systems (Low & Cheng, 2006). In fact, the infant human brain contains two times more synapses than the adult brain (Kolb & Gibb, 2011). Analogously, the immature nervous system of *C. elegans* contains more synapses than the adult form (Oren-Suissa, Bayer, & Hobert, 2016).

But pruning is not restricted to axons and synaptic connections. It even extends to the total number of neurons, which also decreases during development. For instance, the immature nervous system of *C. elegans* initially consists of 308 neurons (Chalfie, 1984), whereas the adult form contains only 302 neurons (Jarrell et al., 2012). And also in humans, the number of neurons decreases during development (Yeo & Gautier, 2004). These ontogenetic changes are referred to as pruning (Paolicelli et al., 2011), and it seems to be a universal phenomenon for all species from *C. elegans* to humans. Furthermore, pruning is found to be mandatory for healthy development (Hong, Dissing-Olesen, & Stevens, 2016). In cases where normal synaptic pruning fails, this may even lead to disorders like schizophrenia (Boksa, 2012).

Since sparse connectivity architectures are realized on all scales and in a vast number of agents of different complexity, it can be assumed that sparse connectivity is a general principle in neural information processing systems, leading to advantages compared to densely connected networks. One major advantage of sparse artificial deep neural networks used for image classification in comparison to fully connected networks, is the reduction of computational costs while at the same time boosting the ability to generalize (Anwar, Hwang, & Sung, 2017; Han, Mao, & Dally, 2015; Mocanu et al., 2018; Wen, Wu, Wang, Chen, & Li, 2016).

However, these pure feed forward network architectures show low biological plausibility as they neither have the ability of dealing with time series data, nor have any memory-like features. Efficient processing of time series data in artificial neural networks is a complex task with a bunch of limitations. The technique of training the neural networks by unfolding the data in time is computational expensive and time consuming and leads to effects such as vanishing or exploding gradients (Hochreiter, 1998; Pascanu, Mikolov, & Bengio, 2012), which have been partly overcome by the introduction of Long–Short-Term-Memory Networks (Schmidhuber & Hochreiter, 1997). However, these networks are difficult to interpret from a biological point of view.

To overcome these limitation, novel biologically inspired approaches for processing time series data were introduced called reservoir computing (Lukoševičius, Jaeger, & Schrauwen, 2012; Verstraeten, Schrauwen, D'Haene, & Stroobandt, 2007). A so called reservoir of neurons with fixed (i.e. not adjusted by training) random recurrent connections is used to calculate higher-order correlations of the input signal which serve as input for a feed forward output layer that is trained with error back-propagation. The properties of the reservoir networks were found to be ideal for biologically inspired parameters with a high sparsity (Alexandre, Embrechts, & Linton, 2009). Thus, these reservoirs work best at the edge of chaos, meaning that the parameters have to be chosen so that they are balanced between complete chaos and absolute periodicity (Bertschinger & Natschläger, 2004; Krauss et al., 2019b; Schrauwen, Verstraeten, & Van Campenhout, 2007).

Much effort has been undertaken to apply the technique of reservoir computing on tasks with delayed rewards such as robot navigation in mazes (Antonelo & Schrauwen, 2012; Antonelo, Schrauwen, & Stroobandt, 2007).

However, the technique of reservoir computing is still based on the fact that the output layer has to be trained in a supervised way using back-propagation and, thus, complex tasks with a delayed reward are difficult to realize. Thus, in this study we used an evolutionary approach to train networks in solving a maze task. Although, reinforcement learning techniques – especially deep reinforcement learning – would be suitable for this kind of hidden-state Markov process, this approach lacks biological plausibility. Thus, deep reinforcement learning needs further techniques for hyper-parameter tuning such as Bayesian optimization (Springenberg, Klein, Falkner, & Hutter, 2016). However, the aim of this study is to analyze self-evolving networks and to characterize the architecture to derive basic principles, which are of relevance for biological systems. Nevertheless, evolutionary techniques could be used to optimize reinforcement learning as well (Young, Rose, Karnowski, Lim, & Patton, 2015).

In our evolutionary system we were able to show that the random severing of connections (evolutionary pruning), without explicitly rewarding sparsity, did lead to a general sparsification of the networks and a better generalization performance. Furthermore, we could demonstrate that evolutionary training works best when the probability for destroying connections is higher but in the same range of the probability for recreating connections.

The paper is structured as follows: In the Method section we describe the used software resources, how the maze task is implemented, the evolutionary algorithm, and the initial neural network architecture. The Results section starts with the analysis of the effect of the different sparsification methods on the performance of the neural networks. We furthermore analyze the architecture of the evolved networks and demonstrate the effect of severance probability and reconnection probability. We added a Conclusion section summarizing the main results, and provide a discussion on the limitations of the study and possible future research directions (Discussion).

2. Materials and methods

2.1. Software resources

All simulations were run on a desktop computer equipped with an i9 extreme processor (Intel) with 10 calculation cores. The complete software was written in Python 3.6 using the libraries sys, os, glob, subprocess, json, natsort, pickle, shutil, NumPy (Van Der Walt, Colbert, & Varoquaux, 2011). Data visualization was done using Matplotlib (Hunter, 2007) and plots were arranged using the Pylustrator (Gerum, 2019).

2.2. Maze task

The task for the agents to perform is a maze based on a rectangular grid of 400 × 22 cells (Fig. 1c). The maze/obstacle task we use here, is similar to the maze task described by Sanchez and coworkers (Sanchez, Pérez-Uribe, & Mesot, 2001), which they also use to analyze the performance of evolutionary approaches.

In our maze task there exist two types of cells, free cells and wall cells. Free cells can be entered and wall cells not. The border of the maze consists of walls to prevent agents from leaving the maze. Starting from the left, every 2 to 10 cells a wall with a

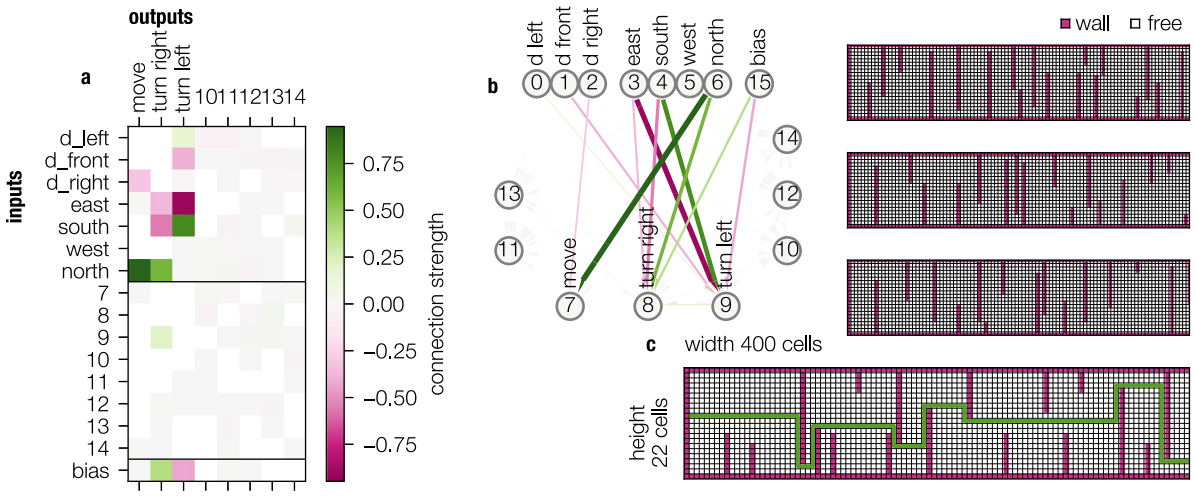

Fig. 1. Example network and mazes. a, Weight matrix of an exemplary network. **b**, The same network displayed as connections. c, Mazes are 400 cells wide and 22 high. Walls (pink) are at all borders and randomly placed in between. The green line depicts one ideal path though the network. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

length between 4 to 20 is inserted. With a probability of 0.25 the wall is inserted from the same side (up or down) as the last wall or with a probability of 0.75 it is inserted from the other side.

Agents start always at the left end of the maze facing to the right. As 'sensory' input each agent receives the distance to the wall in front, to the left and to the right (input neurons 0 to 6, cf. Fig. 1a).

If the distance is larger then 10, it is set to 10 (visual range). It also receives the direction, it is currently looking at, as a one-hot encoded, four-neuron input, i.e. one neuron at a time is in state 1 and the others are in state 0. This input serves as a kind of compass. The seven input neurons do exclusively receive input from the environment, but do not get any input from other neurons, thus, they are reset at each time step.

The agent can output three values for the three possible actions: go straight, turn right, or turn left. The action with the highest value is selected (winner takes all) (output neurons 7 to 9, cf. Fig. 1a). When the agent chooses to go straight and the next field is a wall, it is not moved.

After 400 actions, the covered x-distance of the agent is fed to the fitness function which is proportional to the covered distance (cf. Fig. 1c). Thus, the reward is delayed by 400 time steps.

2.3. Network

The logic of each agent consists of a fully connected network of $N = 16$ neurons (identified as the number of neurons leading to the best performance with fast convergence, cf. Supplements S8, S9, S10), with states s (cf. Fig. 1a,b). The connection weights W are initialized with a random value drawn from a uniformly distribution from the interval $[-\sigma, +\sigma]$ with $\sigma = 4 \cdot \sqrt{\frac{6}{N+N}}/10$. For each time step t in the task, the $3 + 4$ input values (distances (left, front, right) and one-hot encoded direction) are set as the states of the neurons #0 to #6. Then one processing pass through the network is calculated, $s_{t+1} = \mathrm{ReLu}\,(W \cdot s_t)$ (s_t: state of the network at time point t), with $\mathrm{ReLu}(x)$ being the rectified linear function

$$\mathrm{ReLu}(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases} \tag{1}$$

The states s of the neurons #7 to #9 are used as outputs to choose one of the three possible actions to perform in the maze task (connectivity matrix see Fig. 1a). Thus, the action is determined by a winner-takes all-method (in the case of no activation, the "move" action is chosen). Our approach is a policy based approach, as the output of the network directly is the action to take and no quality assessment of different states is undertaken.

2.4. Evolutionary algorithm

For optimizing the networks to fulfill the maze task, we use an evolutionary algorithm (Fekiac, Zelinka, & Burguillo, 2011). Therefore, a pool of 1000 agents (for simulations with different population sizes cf. Supplements S11) is created with a random initialization and 10 mazes are created for the agents to be trained on.

For each iteration, all agents have to perform the maze task and are assigned a fitness, depending on their score in the task. The best half of the agent pool has now the chance to create offspring. The probability to generate an offspring is proportional to their relative fitness compared to the other agents. Agents with a probability of 10% or more are set to a maximal probability of 10% to retain biodiversity. For each of the old agents to be replaced, a parent agent is selected at random according to their reproduction probabilities. Agents can have multiple offspring or no offspring at all.

After offspring generation, each agent is mutated. We used three different mutation types:

- **Weight mutation**: The connection weights W are each mutated by addition of a Gaussian distributed random variable ($\mu = 0$: mean of distribution, $\sigma = |\sigma_{\mathrm{mut}}|$: standard deviation).
- **Mutation rate change**: The mutation rate σ_{mut} is also mutated by multiplication with a Gaussian distributed random variable ($\mu = 1$, $\sigma = \sigma_{\mathrm{mut}}$).
- **Connection mutation**: Existing connections are removed with a probability $p_{\mathrm{disconnect}}$ and non-existing connections are added with a probability of $p_{\mathrm{connect}} = p_{\mathrm{disconnect}}$. Removed connections have a weight of 0 and are not subject to weight mutations. Thus, a removed connection cannot be recovered by a simple mutation step, but can only be recovered by a reconnection mutation.

The fitness (Eq. (2)) is calculated from the squared mean of the square root distances the agent reached in all 10 training mazes (SMR, Eq. (3), this is done to favor generalizing agents which perform okay in all mazes against a specializing agent which

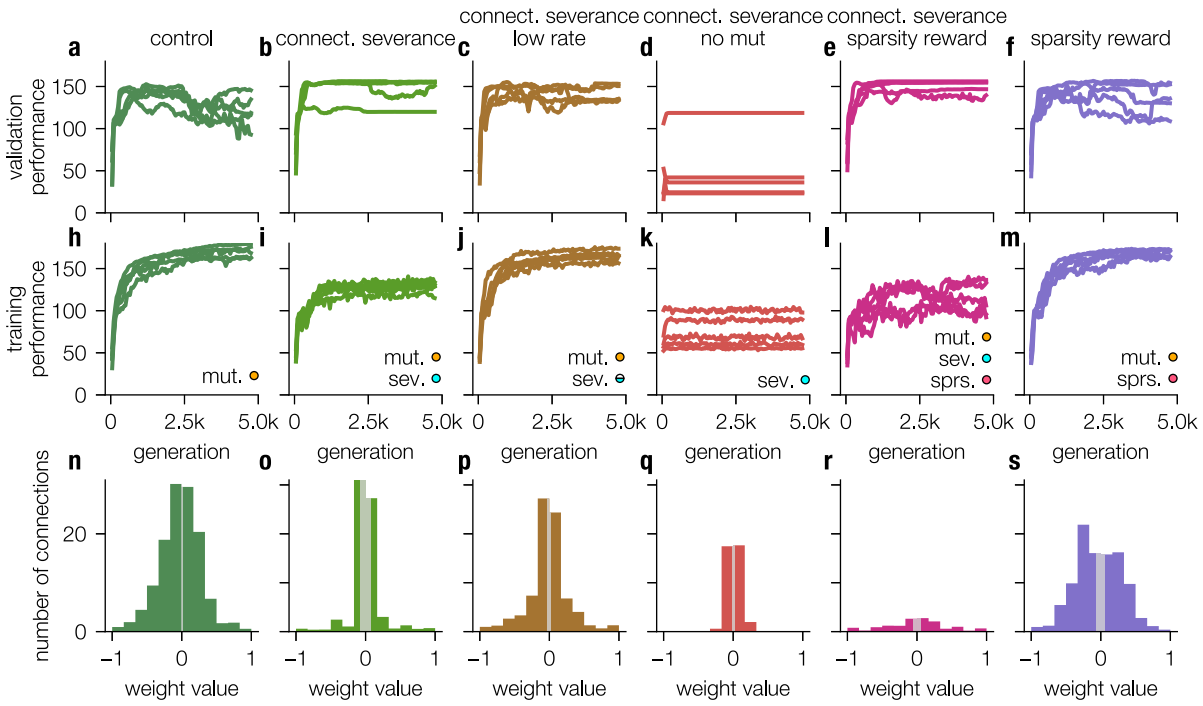

Fig. 2. Performance of networks of the different experiments during training and validation. a–f, Performance over generations for different experiments in 10 validation mazes. Curves show 5 different training seeds, evaluated on the same 10 validation mazes. **h–m**, Performance during training over generations for the 5 different training seeds. Labels stand for the different properties of the condition: "mut." for mutation of the weights, "sev." for severing/restoring connections, "sprs.". for adding a sparsity reward to the fitness function. **n–s**, Histograms of the connection weights for the different experiments. Zero connections are not included in the histograms. Gray shaded areas in the center indicate which weights can be removed without reducing the performance.

Table 1
Settings for the different experiments. The initial mutation rate σ_{mut}, the probability of removing or restoring a connection p_{connect} and the sparsity reward factor f_{sparsity}.

Name	σ_{mut}	p_{connect}	f_{sparsity}
Control	0.01	–	–
Connection severance	0.01	0.01	–
Connection severance	0.01	0.001	–
Connection severance no mut.	–	0.01	–
Connection severance sparsity reward	0.01	0.01	0.1
Sparsity reward	0.01	–	0.1

performs well in only one maze), the maximum mean activation of the neurons and optionally from the mean number of active connections:

$$\text{fitness} = \text{SMR}_{\text{mazes}} - \max_{\text{mazes}}(\text{mean}(\text{activation})) + f_{\text{sparsity}} \cdot \text{sparsity} \quad (2)$$

$$\text{SMR}_{\text{mazes}} = \left(\frac{1}{N_{\text{mazes}}} \cdot \sum_{i=1}^{N_{\text{mazes}}} \sqrt{d_i} \right)^2 \quad (3)$$

($\text{SMR}_{\text{mazes}}$): Root-Mean-Square-Distance, d_i: covered distance in maze i, activation: average activation of neural network, N_{mazes}: number of mazes, f_{sparsity}: sparsity reward factor. All experiments (for the initial parameters see Table 1) are repeated for 5 different seeds of the random number generator that is used to obtain the initial weights and the mutations. The different repetitions were performed on the same 10 training mazes to keep them comparable. The performance of the networks is defined as the average distance covered in the mazes after 400 time-steps. The validation performance is the average distance in unseen validation mazes, whereas the training performance is the average distance in the ten known training mazes.

3. Results

The evolutionary algorithm was able to find solutions enabling the agents to efficiently navigate through the mazes. In all experiments, except the experiment without weight mutations, the agents gradually learned to perform better in the maze tasks over the generations (Fig. 2h–m). The convergence was quite slow, as about 5000 generations were needed to converge to a stable solution. The convergence behavior was mostly independent of the seed of the random number generator, except for the "no mutation" condition, which relied strongly on the initial weights. The performance during training was best for the conditions with no, or low sparsification pressure (Fig. 2h,j,m).

The fitness in validation mazes, that the agents had not seen during training, was more sensitive to the seed than the performance during training. For the experiments with more sparsification pressure (Fig. 2b,e) the validation performance did exceed the fitness during training, showing good generalization, whereas the experiments with lower sparsification pressure (Fig. 2a,c,f) showed more problems with over-fitting, which means that the performance is higher during training than during validation.

Validation performance increased in most runs during training, although some drops in validation performance were observed, which sometimes recovered after a few generations, but in some cases the validation performance continued to fluctuate (see Supplements S2–S6). In general when the mean validation fitness dropped also the variability of the performance/fitness increased, showing that even if some agents found an "overfitting" solution, it was not quickly adopted by all agents, whereas when the solutions were more general, they seemed more stable and were adopted by the whole pool of agents.

The weight distribution of the final generation was different for each experiment. While the experiments with low sparsification pressure, that also showed overfitting, show more connections and more weights with larger absolute values (Fig. 2n,p,s)

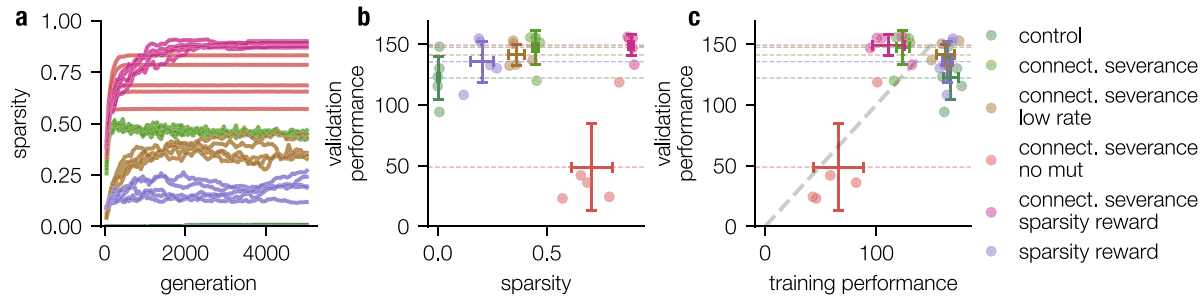

Fig. 3. Correlation of validation performance to sparsity and training performance. a, Sparsity $\left(1-\frac{n_{\text{non-zero}}}{n_{\text{possible}}}\right)$ as a function of the epochs (generations, sparsity is exclusively gained by evolutionary pruning). **b**, Correlation of validation performance to sparsity. Except for the case of "connection severance no mutation" validation performance increases on average with increased sparsity. **c**, Correlation of validation performance to training performance. Higher training performance leads in all experiments also to higher validation performance, indicating that none of the networks runs into severe overfitting.

compared to the other experiments that show more small weight values (Fig. 2o,q). Apart from "connection severance no mut" and "connection severance sparsity reward" (48% and 49% negative weights) all experiments show more negative weights (52%–55%), referring to inhibitory synapses, a fact which indicates more interesting behavior and more efficient information processing (Krauss et al., 2019b).

In most experiments, the sparsity increased over time (Fig. 3a), but in "connection severance" it even slightly decreased after the initial rise and in "sparsity reward" the sparsity fluctuated strongly over time. A higher sparsity was in all cases (except the no mutation case) associated with a higher validation performance (Fig. 3b), showing that sparsity improves the generalization behavior of evolutionary trained networks. A comparison of the training performance to the validation performance also shows that for the sparser cases, the training fitness decreases and in contrast to that, the validation performance increases. Therefore, the sparsification prevents overfitting and enhances the generalization properties of the networks. In addition, it improves computation efficiency in the evolutionary trained networks as the computations, divided between many neurons in the fully connected control group, are, in sparser networks, forced to be carried out on a small subset of neurons.

Furthermore, it could be shown that the networks which perform best in the test mazes develop simple feed-forward structures (cf. Fig. 1a). Additionally, some asymmetry in the connectivity matrix can be observed (cf. Fig. 1b). On the one hand, the bias units prefer the turn towards a certain direction. On the other hand, the connection from the input distance sensor to the turn output (e.g. to turn to the left side d_{left} in example Fig. 4b) is over-represented for one side. Thus, the networks have a preference to go to one side (e.g. to turn left, in the example in Fig. 4b), if there is enough space. The bias unit serves as counterpart, if the agent moves along the upper edge (resp. left side seen when moving along the x-axis) and guarantees that the network can walk away from the wall. The simple network architectures allow for the analysis of the functional tasks of certain neurons. This understanding of the functional tasks of neurons in artificial neural networks could potentially help to understand biological neural networks (Jonas & Kording, 2017; Kriegeskorte & Douglas, 2018). Different experiments develop quite different solutions to solve the maze task (see Supplementary Material S1). Interestingly, in some experiments similar structures emerge, regardless of the seed. This is especially the case in the "connection severance" experiment, where 4 out of 5 solutions are strikingly similar. This hints at the existence of strongly attractive maxima in the space of possible solutions.

Furthermore, we were able to show that training works best, in terms of validation and training performance, when the probability for removing existing connections $p_{\text{disconnect}}$ is in the same range of the probability of recreating removed connections p_{connect} (Fig. 5). Thus, even when $p_{\text{disconnect}} = p_{\text{connect}}$ generalization performance of the networks is increased. Sparsity of the networks can additionally be increased by deleting all unused connections, which do not lead to any changes in performance (combination of pruning and deletion after training, c.f. Fig. 5). This, procedure can be sophisticated as the random severance of connections does not prevent the development of unconnected subnetworks, which should be finally deleted to remove pseudo-complexity of the neural networks. This could be done by simply thresholding the connection weights (c.f. Supplements Fig. S14). However, gradually thresholding the network connections does not help to significantly increase the sparsity of fully connected networks (c.f. control condition in Fig. S14) as these networks were not forced to distribute the information processing over a smaller amount of neurons by the random pruning procedure.

4. Conclusion

In this study we showed that evolutionary pruning of artificial neural networks, evolutionary trained to solve a simple maze task, leads to sparser networks with better generalization properties compared to dense networks trained without pruning. The evolutionary pruning is realized via two mutation mechanisms: First the networks can set a threshold defining the upper limit of the absolute weight value being virtually set to zero. Secondly, some connections are removed (weight set to 0) and also restored at random with given probabilities p_{connect}, $p_{\text{disconnect}}$. The random removal of existing connections leads to more robust networks with better generalization abilities. This effect works best when $p_{\text{connect}} \leq p_{\text{disconnect}}$ and both probabilities are in the same range. In conclusion, evolutionary pruning by random severance of connections can be used as additional mechanism to improve the evolutionary training of neural networks.

5. Discussion

5.1. Limitations

The maze task is still not complex enough to trigger the development of more complex recurrent neural networks which include abilities such as memory. As the agents in the maze are always provided with a "compass" meaning always know in which direction they are pointing, the task can be solved with a Markov like decision process, as the information of one time step is enough to decide the next action. Thus, after 5000 epochs the best performances are achieved by simple feed forward networks (cf. Fig. 4). Nevertheless, it has to be considered that these networks were not forced to develop feed forward architectures but are a result of the Markov properties of the task they were trained

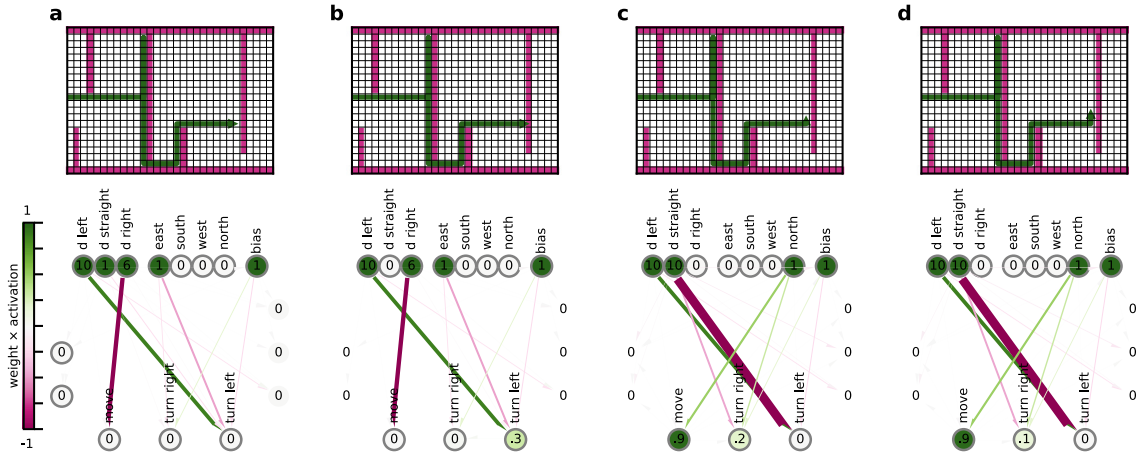

Fig. 4. State of the maze and configuration of the network during a turn. The position and orientation in the maze are denoted by the green triangle. The past trajectory by the green line. Below, the current state of the network is visualized. The values correspond to the current activation of each node and the colored connections to the weight times the activation (note that the weights are static) of the target node (pink negative, green positive). **a**, Network one step before turn. **b**, Network encounters a wall and turns. **c**, Network has just turned and is now heading north. **d**, Network continues walking straight in the new direction. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

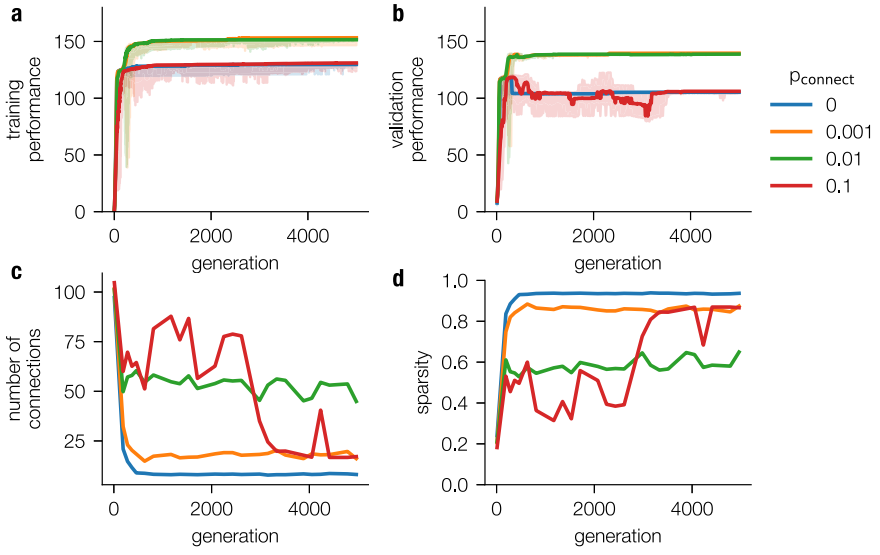

Fig. 5. Training/test performance as a function connection probability p_{connect} and disconnection probability $p_{\text{disconnect}}$ ratio. The training (a) and test (b) performance of the agents as a function of the generation for different values of p_{connect} (0, 0.001, 0.01, 0.1) and a fixed $p_{\text{disconnect}} = 0.01$. Best performance is achieved when $p_{\text{connect}} \leq p_{\text{disconnect}}$ and both are in the same range. (c, d), The sparsity expressed as the number of connections (c) and as a relative count of realized connections (d). A connection is counted active, if the fitness is not decreased when the connection is disabled. Thus, the sparsity of the networks is reduced by deleting sub-networks, which are not used (analogously to Supplements S14).

on. The simulation shows that simple feed forward networks are able to navigate through a maze using only 16 sparsely connected neurons.

5.2. Future research directions

Thus, the here described networks demonstrate that only a small number of neurons is needed to navigate through environment and shows that for example *C. elegans* with its 302 neurons (White et al., 1986) should be indeed able to perform complex tasks. It has already been shown that *C. elegans* shows thermo- and electrotaxis (Gabel et al., 2007) which are not simply a biased random walk in contrast to its' chemotaxis (for certain temperatures thermotaxis also changes to a random walk) (Pierce-Shimomura, Morse, & Lockery, 1999; Ryu & Samuel, 2002). *C. elegans* shows a direct movement along the electric gradient (Gabel et al., 2007). Thus, the amphid sensory neurons of *C. elegans* could be seen as and equivalent to the "compass" neurons in our simulation.

The here described study does not show a biologically inspired neuron configuration, however, demonstrates that 16 neurons are enough to perform relatively simple navigation tasks and gives a hint that the development of sparsity has evolutionary advantages. These findings contrast with current developments in the field of artificial intelligence where the size of the networks due to higher calculation power are scaled up (Xu et al., 2018).

Furthermore, the maze task could be extended so that the task is not a simple Markov process, which means that the next decision cannot be made by simply analyzing the current position in the maze (Littman, 2012; Wierstra & Wiering, 2004). This increase

of complexity can be achieved e.g. by removing the compass neurons. Consequently, a neural network which is able to achieve similar performance than networks with compass neurons need to develop some "memory" features. The networks would have to remember which movements they recently executed and they would have to dynamically recall this information. An even more demanding task would be to force the agents to go directly back to the starting point after they passed the maze. This task requires path integration (Etienne & Jeffery, 2004; Wehner & Wehner, 1986), which in turn requires the ability to flexibly navigate in physical space like insects (Andel & Wehner, 2004; Müller & Wehner, 1988, 1994; Wehner & Wehner, 1986), and at least in mammals episodic memory (Etienne, Maurer, & Séguinot, 1996; McNaughton, Battaglia, Jensen, Moser, & Moser, 2006; Séguinot, Cattet, & Benhamou, 1998). It has been demonstrated that these abilities can only be achieved within highly recurrent networks, as they can be found in the hippocampus (Etienne & Jeffery, 2004). Recently it has been shown that the network architecture of the hippocampus is not limited to spatial navigation, but seems to be domain-general (Aronov, Nevers, & Tank, 2017; Killian & Buffalo, 2018; Nau, Schröder, Bellmund, & Doeller, 2018) and even allows navigation in abstract high-dimensional cognitive feature spaces (Bellmund, Gärdenfors, Moser, & Doeller, 2018; Constantinescu, O'Reilly, & Behrens, 2016; Eichenbaum, 2015; Garvert, Dolan, & Behrens, 2017; Theves, Fernandez, & Doeller, 2019). Future work will have to investigate, whether networks with the above mentioned abilities can also be found by evolutionary algorithms.

Further potential research directions could be to optimize network architectures using evolutionary algorithms to find efficient neural networks, which can be trained supervisely or via reinforcement learning. The counterpart to this approach would be the analysis of evolutionary trained sparse neural networks, and to search for functional units such as motifs (Krauss, Zankl, Schilling, Schulze and Metzner, 2019c), weight statistics (Krauss et al., 2019b), or to analyze complex dynamics like 'recurrence resonance' effects (Krauss, Prebeck, Schilling and Metzner, 2019a), which is up to now often done in untrained networks, that perform no real information processing.

These two strands, would be in line with the philosophy that artificial and biological intelligence are "two sides of the same coin" (Kriegeskorte & Douglas, 2018; Schilling et al., 2018), and that the understanding of brain mechanisms on the one hand and the development of artificial neural network algorithms on the other hand are an iterative process, stimulating each other.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the German Research Foundation (DFG, grant KR5148/2-1 to PK, project number: 436456810), and the Emergent Talents Initiative (ETI) of the University Erlangen–Nuremberg, Germany (grant 2019/2-Phil-01 to PK).

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.neunet.2020.05.007.

References

Alexandre, L. A., Embrechts, M. J., & Linton, J. (2009). Benchmarking reservoir computing on time-independent classification tasks. In *Proceedings of the international joint conference on neural networks* (pp. 89–93). IEEE.

Amaral, L. A., Scala, A., Barthélémy, M., & Stanley, H. E. (2000). Classes of small-world networks. *Proceedings of the National Academy of Sciences of the United States of America*, *97*(21), 11149–11152.

Andel, D., & Wehner, R. (2004). Path integration in desert ants, Cataglyphis: how to make a homing ant run away from home. *Proceedings of the Royal Society of London, Series B*, *271*(1547), 1485–1489.

Antonelo, E., & Schrauwen, B. (2012). Learning slow features with reservoir networks for biologically-inspired robot localization. *Neural Networks*, *25*, 178–190.

Antonelo, E. A., Schrauwen, B., & Stroobandt, D. (2007). Event detection and localization for small mobile robots using reservoir computing. In *Conference on artificial neural networks* (pp. 660–669).

Anwar, S., Hwang, K., & Sung, W. (2017). Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems*, *13*(3), 1–18.

Aronov, D., Nevers, R., & Tank, D. W. (2017). Mapping of a non-spatial dimension by the hippocampal–entorhinal circuit. *Nature*, *543*(7647), 719.

Babadi, B., & Sompolinsky, H. (2014). Sparseness and expansion in sensory representations. *Neuron*, *83*(5), 1213–1226.

Barwich, A.-S. (2019). The value of failure in science: The story of grandmother cells in neuroscience. *Frontiers in Neuroscience*, *13*, 1121.

Bassett, D. S., & Bullmore, E. (2006). Small-world brain networks. *Neuroscientist*, *12*(6), 512–523.

Bassett, D. S., Meyer-Lindenberg, A., Achard, S., Duke, T., & Bullmore, E. (2006). Adaptive reconfiguration of fractal small-world human brain functional networks. *Proceedings of the National Academy of Sciences*, *103*(51), 19518–19523.

Bellmund, J. L., Gärdenfors, P., Moser, E. I., & Doeller, C. F. (2018). Navigating cognition: Spatial codes for human thinking. *Science*, *362*(6415), eaat6766.

Bertschinger, N., & Natschläger, T. (2004). Real-time computation at the edge of chaos in recurrent neural networks. *Neural Computation*, *16*(7), 1413–1436.

Boksa, P. (2012). Abnormal synaptic pruning in schizophrenia: Urban myth or reality? *Journal of Psychiatry & Neuroscience: JPN*, *37*(2), 75.

Chalfie, M. (1984). Neuronal development in Caenorhabditis elegans. *Trends in NeuroSciences*, *7*(6), 197–202.

Constantinescu, A. O., O'Reilly, J. X., & Behrens, T. E. (2016). Organizing conceptual knowledge in humans with a gridlike code. *Science*, *352*(6292), 1464–1468.

Crochet, S., Poulet, J. F., Kremer, Y., & Petersen, C. C. (2011). Synaptic mechanisms underlying sparse coding of active touch. *Neuron*, *69*(6), 1160–1175.

Dasgupta, S., Stevens, C. F., & Navlakha, S. (2017). A neural algorithm for a fundamental computing problem. *Science*, *358*(6364), 793–796.

Eichenbaum, H. (2015). The hippocampus as a cognitive map…of social space. *Neuron*, *87*(1), 9–11.

Etienne, A. S., & Jeffery, K. J. (2004). Path integration in mammals. *Hippocampus*, *14*(2), 180–192.

Etienne, A. S., Maurer, R., & Séguinot, V. (1996). Path integration in mammals and its interaction with visual landmarks. *Journal of Experimental Biology*, *199*(1), 201–209.

Fekiac, J., Zelinka, I., & Burguillo, J. C. (2011). A review of methods for encoding neural network topologies in evolutionary computation. In *Proceedings - 25th European conference on modelling and simulation, ECMS 2011* (pp. 410–416).

Gabel, C. V., Gabel, H., Pavlichin, D., Kao, A., Clark, D. A., & Samuel, A. D. (2007). Neural circuits mediate electrosensory behavior in Caenorhabditis elegans. *Journal of Neuroscience*, *27*(28), 7586–7596.

Garvert, M. M., Dolan, R. J., & Behrens, T. E. (2017). A map of abstract relational knowledge in the human hippocampal–entorhinal cortex. *Elife*, *6*, e17086.

Gerum, R. (2019). Pylustrator: Code generation for reproducible figures for publication. arXiv:1910.00279.

Hagmann, P., Cammoun, L., Gigandet, X., Meuli, R., Honey, C. J., Van Wedeen, J., et al. (2008). Mapping the structural core of human cerebral cortex. *PLoS Biology*, *6*(7), 1479–1493.

Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. (pp. 1–14). arXiv:1510.00149.

Herculano-Houzel, S. (2009). The human brain in numbers: a linearly scaled-up primate brain. *Frontiers in Human Neuroscience*, *3*, 31.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *6*(2), 107–116.

Hong, S., Dissing-Olesen, L., & Stevens, B. (2016). New insights on the role of microglia in synaptic pruning in health and disease. *Current Opinion in Neurobiology*, *36*, 128–134.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science and Engineering*, *9*, 90–95.

Jarrell, T. A., Wang, Y., Bloniarz, A. E., Brittin, C. A., Xu, M., Thomson, J. N., et al. (2012). The connectome of a decision-making neural network. *Science*, *337*(6093), 437–444.

Jiao, Y., Zhang, Y., Chen, X., Yin, E., Jin, J., Wang, X., et al. (2018). Sparse group representation model for motor imagery EEG classification. *IEEE Journal of Biomedical and Health Informatics*, *23*(2), 631–641.

Jin, Z., Zhou, G., Gao, D., & Zhang, Y. (2018). EEG classification using sparse Bayesian extreme learning machine for brain–computer interface. *Neural Computing and Applications*, 1–9.

Jonas, E., & Kording, K. P. (2017). Could a neuroscientist understand a microprocessor? *PLoS Computational Biology*, *13*(1), e1005268.

Kafashan, M., Nandi, A., & Ching, S. (2016). Relating observability and compressed sensing of time-varying signals in recurrent linear networks. *Neural Networks*, *83*, 11–20.

Kerr, J. N., Greenberg, D., & Helmchen, F. (2005). Imaging input and output of neocortical networks in vivo. *Proceedings of the National Academy of Sciences of the United States of America*, *102*(39), 14063–14068.

Killian, N. J., & Buffalo, E. A. (2018). Grid cells map the visual world. *Nature Neuroscience*, *21*(2), 161.

Kolb, B., & Gibb, R. (2011). Brain plasticity and behaviour in the developing brain. *Journal of the Canadian Academy of Child and Adolescent Psychiatry*, *20*(4), 265.

Krauss, P., Prebeck, K., Schilling, A., & Metzner, C. (2019a). Recurrence resonance in three-neuron motifs. *Frontiers in Computational Neuroscience*, *13*.

Krauss, P., Schuster, M., Dietrich, V., Schilling, A., Schulze, H., & Metzner, C. (2019b). Weight statistics controls dynamics in recurrent neural networks. *PLoS ONE*, *14*(4), 1–13.

Krauss, P., Zankl, A., Schilling, A., Schulze, H., & Metzner, C. (2019c). Analysis of structure and dynamics in three-neuron motifs. *Frontiers in Computational Neuroscience*, *13*, 5.

Kriegeskorte, N., & Douglas, P. K. (2018). Cognitive computational neuroscience. *Nature Neuroscience*, *21*(9), 1148–1160.

Latora, V., & Marchiori, M. (2001). Efficient behavior of small-world networks. *Physical Review Letters*, *87*(19), 198701.

Littman, M. (2012). Inducing partially observable Markov decision processes.. In *11th international conference on grammatical inference, Vol. 21* (pp. 145–148).

Low, L. K., & Cheng, H.-J. (2006). Axon pruning: an essential step underlying the developmental plasticity of neuronal connections. *Philosophical Transactions of the Royal Society, Series B (Biological Sciences)*, *361*(1473), 1531–1544.

Lukoševičius, M., Jaeger, H., & Schrauwen, B. (2012). Reservoir computing trends. *KI - Künstliche Intelligenz*, *26*(4), 365–371.

McNaughton, B. L., Battaglia, F. P., Jensen, O., Moser, E. I., & Moser, M.-B. (2006). Path integration and the neural basis of the'cognitive map'. *Nature Reviews Neuroscience*, *7*(8), 663.

Mocanu, D. C., Mocanu, E., Stone, P., Nguyen, P. H., Gibescu, M., & Liotta, A. (2018). Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature Communications*, *9*(1), 2383.

Müller, M., & Wehner, R. (1988). Path integration in desert ants, Cataglyphis fortis. *Proceedings of the National Academy of Sciences*, *85*(14), 5287–5290.

Müller, M., & Wehner, R. (1994). The hidden spiral: systematic search and path integration in desert ants, Cataglyphis fortis. *Journal of Comparative Physiology A*, *175*(5), 525–530.

Nau, M., Schröder, T. N., Bellmund, J. L., & Doeller, C. F. (2018). Hexadirectional coding of visual space in human entorhinal cortex. *Nature Neuroscience*, *21*(2), 188.

Oh, S. W., Harris, J. A., Ng, L., Winslow, B., Cain, N., Mihalas, S., et al. (2014). A mesoscale connectome of the mouse brain. *Nature*, *508*(7495), 207–214.

Olshausen, B. A., & Field, D. J. (2004). Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, *14*(4), 481–487.

Oren-Suissa, M., Bayer, E. A., & Hobert, O. (2016). Sex-specific pruning of neuronal synapses in Caenorhabditis elegans. *Nature*, *533*(7602), 206.

Paolicelli, R. C., Bolasco, G., Pagani, F., Maggi, L., Scianni, M., Panzanelli, P., et al. (2011). Synaptic pruning by microglia is necessary for normal brain development. *science*, *333*(6048), 1456–1458.

Pascanu, R., Mikolov, T., & Bengio, Y. (2012). Understanding the exploding gradient problem. arXiv:1211.5063v1.

Pehlevan, C., & Sompolinsky, H. (2014). Selectivity and sparseness in randomly connected balanced networks. *PLoS One*, *9*(2).

Perin, R., Berger, T. K., & Markram, H. (2011). A synaptic organizing principle for cortical neuronal groups. *Proceedings of the National Academy of Sciences*, *108*(13), 5419–5424.

Pierce-Shimomura, J. T., Morse, T. M., & Lockery, S. R. (1999). The fundamental role of pirouettes in Caenorhabditis elegans chemotaxis. *Journal of Neuroscience*, *19*(21), 9557–9569.

Quiroga, R. Q., Kreiman, G., Koch, C., & Fried, I. (2008). Sparse but not 'grandmother-cell'coding in the medial temporal lobe. *Trends in Cognitive Sciences*, *12*(3), 87–91.

Rose, D. (1996). *Some reflections on (or by?) grandmother cells*. London, England: SAGE Publications Sage UK.

Ryu, W. S., & Samuel, A. D. (2002). Thermotaxis in Caenorhabditis elegans analyzed by measuring responses to defined thermal stimuli. *Journal of Neuroscience*, *22*(13), 1–7.

Sanchez, E., Pérez-Uribe, A., & Mesot, B. (2001). Solving partially observable problems by evolution and learning of finite state machines. In *International conference on evolvable systems* (pp. 267–278). Springer.

Schilling, A., Metzner, C., Rietsch, J., Gerum, R., Schulze, H., & Krauss, P. (2018). How deep is deep enough?–Quantifying class separability in the hidden layers of deep neural networks. arXiv preprint arXiv:1811.01753.

Schmidhuber, J., & Hochreiter, S. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Schrauwen, B., Verstraeten, D., & Van Campenhout, J. (2007). An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th European symposium on artificial neural networks* (pp. 471–482).

Séguinot, V., Cattet, J., & Benhamou, S. (1998). Path integration in dogs. *Animal Behaviour*, *55*(4), 787–797.

Song, S., Sjöström, P. J., Reigl, M., Nelson, S., & Chklovskii, D. B. (2005). Highly nonrandom features of synaptic connectivity in local cortical circuits. *PLoS Biology*, *3*(3), 0507–0519.

Sporns, O., Tononi, G., & Kötter, R. (2005). The human connectome: A structural description of the human brain. *PLoS Computational Biology*, *1*(4), 0245–0251.

Springenberg, J. T., Klein, A., Falkner, S., & Hutter, F. (2016). Bayesian optimization with robust Bayesian neural networks. In *Advances in neural information processing systems* (pp. 4134–4142).

Theves, S., Fernandez, G., & Doeller, C. F. (2019). The hippocampus encodes distances in multidimensional feature space. *Current Biology*, *29*(7), 1226–1231.

van den Heuvel, M. P., & Yeo, B. T. T. (2017). A spotlight on bridging microscale and macroscale human brain architecture. *Neuron*, *93*(6), 1248–1251.

Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, *13*, 22–30.

Verstraeten, D., Schrauwen, B., D'Haene, M., & Stroobandt, D. (2007). An experimental unification of reservoir computing methods. *Neural Networks*, *20*(3), 391–403.

Watts, D. J., & Strogatz, S. H. (1998). Strogatz - small world network nature. *Nature*, *393*, 440–442.

Wehner, R., & Wehner, S. (1986). Path integration in desert ants. approaching a long-standing puzzle in insect navigation. *Monitore Zoologico Italiano-Italian Journal of Zoology*, *20*(3), 309–331.

Wen, W., Wu, C., Wang, Y., Chen, Y., & Li, H. (2016). Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems* (pp. 2074–2082).

White, J. G., Southgate, E., Thomson, J. N., & Brenner, S. (1986). The structure of the nervous system of the nematode caenorhabditis elegans author (s): J. G. White, E. Southgate, J. N. Thomson, S. Brenner source : Philosophical transactions of the royal society of London. series B, biological published by. *Philosophical Transactions of the Royal Society of London*, *314*(1165), 1–340.

Wierstra, D., & Wiering, M. (2004). Utile distinction hidden Markov models. In *Proceedings, twenty-first international conference on machine learning, ICML 2004* (pp. 855–862).

Xu, X., Ding, Y., Hu, S. X., Niemier, M., Cong, J., Hu, Y., et al. (2018). Scaling for edge inference of deep neural networks. *Nature Electronics*, *1*(4), 216–222.

Yeo, W., & Gautier, J. (2004). Early neural cell death: dying to become neurons. *Developmental Biology*, *274*(2), 233–244.

Young, S. R., Rose, D. C., Karnowski, T. P., Lim, S.-H., & Patton, R. M. (2015). Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Proceedings of the workshop on machine learning in high-performance computing environments* (pp. 1–5).

Zaslaver, A., Liani, I., Shtangel, O., Ginzburg, S., Yee, L., & Sternberg, P. W. (2015). Hierarchical sparse coding in the sensory system of Caenorhabditis elegans. *Proceedings of the National Academy of Sciences*, *112*(4), 1185–1189.

Zhang, L., Yang, M., & Feng, X. (2011). Sparse representation or collaborative representation: Which helps face recognition? In *2011 international conference on computer vision* (pp. 471–478). IEEE.