SCIENTIFIC REPORTS

OPEN

Evolving autonomous learning in cognitive networks

Leigh Sheneman[1,2] & Arend Hintze[1,2,3]

Received: 7 July 2017
Accepted: 14 November 2017
Published online: 01 December 2017

There are two common approaches for optimizing the performance of a machine: genetic algorithms and machine learning. A genetic algorithm is applied over many generations whereas machine learning works by applying feedback until the system meets a performance threshold. These methods have been previously combined, particularly in artificial neural networks using an external objective feedback mechanism. We adapt this approach to Markov Brains, which are evolvable networks of probabilistic and deterministic logic gates. Prior to this work MB could only adapt from one generation to the other, so we introduce feedback gates which augment their ability to learn during their lifetime. We show that Markov Brains can incorporate these feedback gates in such a way that they do not rely on an external objective feedback signal, but instead can generate internal feedback that is then used to learn. This results in a more biologically accurate model of the evolution of learning, which will enable us to study the interplay between evolution and learning and could be another step towards autonomously learning machines.

Natural organisms not only have to fit their environment, but also have to adapt to changes in their environment which means that they have to be plastic. While plasticity occurs in many forms, here we focus on *neural plasticity* which we define as an organisms ability to use "experiences" to improve later decisions and behavior. Being able to solve a T-maze repetitively, remembering where food is located, avoiding places where predators have been spotted, and even learning another language are all cognitive abilities that require an organism to have neural plasticity. We will show how this neural plasticity can evolve in a computational model system. Neural plasticity allows natural organisms to learn due to reinforcement of their behavior[1]. However, learning is tied to specific neural mechanisms- working memory (WM), short-term memory (STM) and long-term memory (LTM). While learning was initially perceived as a new "factor" in evolution[2], potentially even independent, it has since been well integrated into the *Modern Synthesis of Evolution*[3]. Evolution and learning can have a positive effect on each other[4,5], however, this is not necessarily always the case[6]. This has several implications: Evolution begin with organisms that could not adapt during their lifetime, which means that they had no neural plasticity. The only feedback that the evolutionary process receives is differential birth and death. As a consequence, learning will only evolve if it can increase the number of viable offspring, and it can only do so if there is a signal that predictably indicates a fitness advantage[7].

Organisms receive many signals from their environment which have to be filtered and interpreted. Irrelevant signals should be ignored while others require adaptive responses. This can be done through instincts or reflexes in cases where fixed responses are necessary. In other cases, information has to be stored and integrated in order to inform later decisions, which requires memory and learning. To distinguish between actions that lead to advantageous results and those that are disadvantageous organisms need positive or negative feedback. However, none of the signals organisms receive are inherently "good" or "bad"; even a signal as simple as food requires interpretation. The consumption of food has to trigger positive feedback within the organism in order to function as a reward. The machinery that triggers the feedback is an evolved mechanism and is often adaptive to the environment. If food were a global positive feedback signal, it would reinforce indiscriminate food consumption. Organisms would not be able to avoid food or store it for later, but instead eat constantly.

Another important detail we have to consider is the difference between learning and memory. While memory is information about the past, learning is the process that takes a sensorial percept and, typically by reinforcement, retains that information for later use. Specifically, sensory information is stored in working memory (WM)[8,9]. Imagine this as the flurry of action potentials that go through the brain defining its current state. Information that

[1]Department of Computer Science and Engineering, Michigan State University, East Lansing, 48824, USA. [2]BEACON-Center for the Study of Evolution in Action, Michigan State University, East Lansing, 48824, USA. [3]Department of Integrative Biology, Michigan State University, East Lansing, 48824, USA. Correspondence and requests for materials should be addressed to A.H. (email: hintze@msu.edu)

a living organism needs to store for a moment is believed to reside in STM[9,10], but how information transforms from WM to STM is not fully understood[8,10,11]. Natural systems use their LTM if they want to keep information for longer. Presumably, information from STM becomes reinforced and thus forms LTM, this is sometimes referred to as consolidation[9,10,12]. The reinforcement process takes time and therefore is less immediate than STM. In addition, memories can be episodic or semantic[9,10,13] and can later be retrieved to influence current decisions. While information in the working memory can be used to influence decisions, it does not change the cognitive substrate. However, long term potentiation (or other neural processes) use this information to change the neural substrate by presumably forming or modifying connections.

In summary, if we want to model the evolution of learning in natural organisms properly we need to take the following statements seriously:

- evolution happens over generations while learning happens during the lifetime of an organism
- evolution is based on differential birth and death (selection) and learning evolved to increase the number of viable offspring and/or to avoid death
- organisms do not receive an objective "positive" or "negative" signal, but instead evolved mechanisms to sense and interpret the world so that they can tell what actions were positive and which ones were not
- memory is information about the past and can be transient
- information in the WM does not change the cognitive machinery, while learning changes the substrate to retain information for longer, which turns transient into permanent information

Machine Learning

Computer science and engineering are typically not concerned with biological accuracy but more with scalability, speed, and required resources. Therefore, the field of machine learning is much more of a conglomerate of different methods which straddle the distinct concepts we laid out above. Machine learning includes methods such as data mining, clustering, classification, and evolutionary computation[14]. Typically, these methods try to find a solution to a specific problem. If we provide an explicit reference or example class we refer to it as supervised learning since the answer is known and the fitness function quantifies the difference between the provided solution and the ones the machine generates. For unsupervised learning we provide a fitness function that measures how well a machine or algorithm performs without the need to know the solution in advance. Genetic algorithms (GAs), which are a form of evolutionary search, work in supervised or unsupervised contexts, whereas learning algorithms are typically supervised. A special class are learning to learn algorithms, which improve their learning ability while adapting to a problem[15] but do not necessarily apply evolutionary principles.

Genetic algorithms clearly optimize from one generation to the other, while learning algorithms on the other hand could be understood as lifetime learning. *Q*-learning[16] optimizes a Markov Decision Processes by changing probabilities when a reward is applied. Typically, delayed rewards are a problem, that deep-*Q* learning and memory replay try to overcome[17]. Artificial neural networks can be trained by using back propagation[18–20], the Baum-Welch algorithm[21,22], or gradient decent[23,24] which happens episodically, but on an individual level and not to a population that experiences generations. Multiplicative weights algorithm strengthens or weakens connections in a neural network-based on the consensus of a pool of experts[25,26], again on an individual level.

At the same time, memory and learning are often treated interchangeably. Recurrent artificial neural networks can store information in their recurrent nodes (or layer) without changing their weights, which would be analogous to WM. Similarly, the system we use, Markov Brains (MB), can form representations about their environment and store this information in hidden states (WM), again transiently without changing its computational structure[27] (For a general explanation of Markov Brains see https://arxiv.org/abs/1709.05601[28]). Changes to the weights of an ANN, or the probabilities of a Markov process, or the probabilities of a POMDP[29] reflect better learning, since those changes are not transient and change all future computations executed by the system.

We also find a wide range of evolvable neural network systems[30–32] (among many others) which change from generation to generation and allow for memory to form by using recurrent connections. Alternatively, other forms of evolving systems interact and use additional forms of memory[33,34]. In order to evolve and learn, other systems allow the topology and/or weights of the neural network to change during evolution while also allowing weight changes during their lifetime[34–44]. Presenting objective feedback to adapt these systems during their lifetime allowed their performance to improve. As a consequence, the machinery that interprets the environment to create feedback was of no concern, but, as stated above, natural organisms also need to evolve that machinery to learn. We think it is quite possible to change these systems to not rely only on external feedback. Instead, they themselves could create the feedback signal as part of their output. However, none of the systems mentioned above is an evolvable MB (for a comparison see Fig. 1A vs. B).

In our approach we use MBs[45], which are networks of deterministic and probabilistic logic gates, encoded in such a way that Darwinian evolution can easily improve them. MBs have been proven to be a useful tool to study animal behavior[46,47], neural correlates[27,48,49], evolutionary dynamics[50], decision making[51,52], and can even be used as a machine learning tool[53,54]. One can think of these MBs as artificial neural networks (ANN)[55] with an arbitrary topology that uses Boolean logic instead of logistic functions. Through sensors these networks receive information about their environment as zeros or ones, perform computations and typically act upon their environment through their outputs. We commonly refer to MBs that are embodied and through that embodiment[56] interact with their environment as agents (others use the term animat which is synonymous). MBs use hidden states to store information about the past similar to recurrent nodes in an ANN. The state of these hidden nodes has to be actively maintained which makes the information volatile. The information in the hidden states can be used to perform computations and functions as memory. This form of memory resembles WM or STM more than LTM due its volatile nature. In the past, the entire structure of a MB would be encoded by the genome and

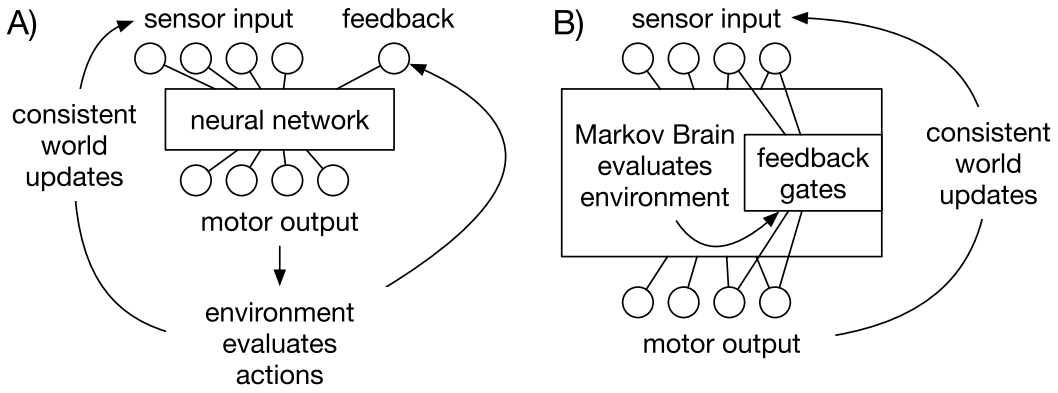

Figure 1. Comparison of two different approaches to feedback generation. Traditional methods have the environment evaluate the actions of a neural network (or similar system) and then generate feedback which is provided as an external signal, similar to adding another input (panel A). Our approach (panel B) integrates both the feedback generation and the feedback gates into the Markov Brain. This way, the world only provides consistent information about itself which can be evaluated so that feedback generation does not need to be offloaded to the environment, but becomes part of the entire machine. The entire connectivity as well as feedback gates are integrated as part of what needs to evolve.

would not change over the lifetime of the agent. Here we introduce what we call feedback gates, which allow MBs to use internal feedback to store information by changing their probabilistic logic gates (see Methods for a detailed description of feedback gates). Like other systems these updates do not change the topological structure of the node network but rather the probabilities within the gates; similar to how learning in ANN is achieved through weight changes. However, feedback is not an objective signal coming from the environment but must be generated as part of the evolved controller. The feedback gates only receive internally generated feedback to change their behavior. This linkage between inputs, evaluation of the environment to generate feedback, how feedback gates receive this information, and how everything controls the actions of the agent evolves over time (see Fig. 1B).

Here we introduce feedback gates that allow MBs to change their internal structure which is akin to learning and forming long-term memories during the lifetime of an organism. The closest similarity can be found in *Q*-learning or deep-*Q* learning, which changes the probabilities within a Markov Decision process even if rewards are delayed[14] (for a detailed explanation of the feedback gate function see Materials and Methods as well as Supplementary Fig. 1). We will show that feedback gates function as expected and allow agents to evolve the ability to decipher sensor inputs so that they can autonomously learn to navigate a complex environment.

Results

Natural organisms have to learn many things over their lifetime including how to control their body. Here the environment used to evolve agents resembles this problem. Agents have to learn to use their body in order to navigate properly. The environment is a 2D lattice (64×64 tile wide) where a single tile is randomly selected as the goal an agent must reach. The lattice is surrounded by a wall so agents can not escape the boundary, and $\frac{1}{7}$ of the lattice is filled with additional walls to make navigation harder. From the goal the Dijkstra's path is computed so that each tile in the lattice can now indicate which of its neighboring tiles is the next closest to the goal. In cases where two neighbor tiles might have the same distance, one of these tiles is randomly selected as the next closest. For ease of illustration we can now say that a tile has an arrow pointing towards the tile that should be visited next to reach the goal in the shortest number of tiles.

The agent, controlled by a MB, is randomly placed on a tile that is 32 tiles away from the goal and facing in a random direction (north, west, south, or east). Agents can see the arrow of the tile they are standing on. The direction indicated by the tile is relative to the that of the agent, so that a tile indicating north will only be perceived as a forward facing arrow if the agent also faces north. The agent has four binary sensors that are used to indicate which relative direction the agent should go to reach the goal.

The agent can move over the lattice by either turning 90 degrees to the left or right, or by moving forward. So far, in order to navigate perfectly, the agent would simply need to move forward when seeing a forward facing arrow, or turn accordingly. Instead of allowing the agent to directly pick a movement, it can choose one of four intermediate options (A, B, C, or D) at any given update. At the birth of an agent, these four possible options are mapped to the four possible actions: move forward, turn left, turn right, do nothing. As a result, the complexity of the task increases when the agent has to learn which of the 24 possible option-to-action maps currently applies to navigate the environment properly. The agent is not given any direct feedback about its actions; a mechanism must evolve to discern the current mapping and which is rather difficult.

In prior experiments[27,45–51,53], MBs were made from deterministic or probabilistic logic gates that use a logic table to determine the output given a particular input. Deterministic gates have one possible output for each input, while probabilistic gates use a linear vector of probabilities to determine the likelihood for any of the possible outputs to occur. To enable agents to form LTM and learn during their lifetime we introduce a new type of

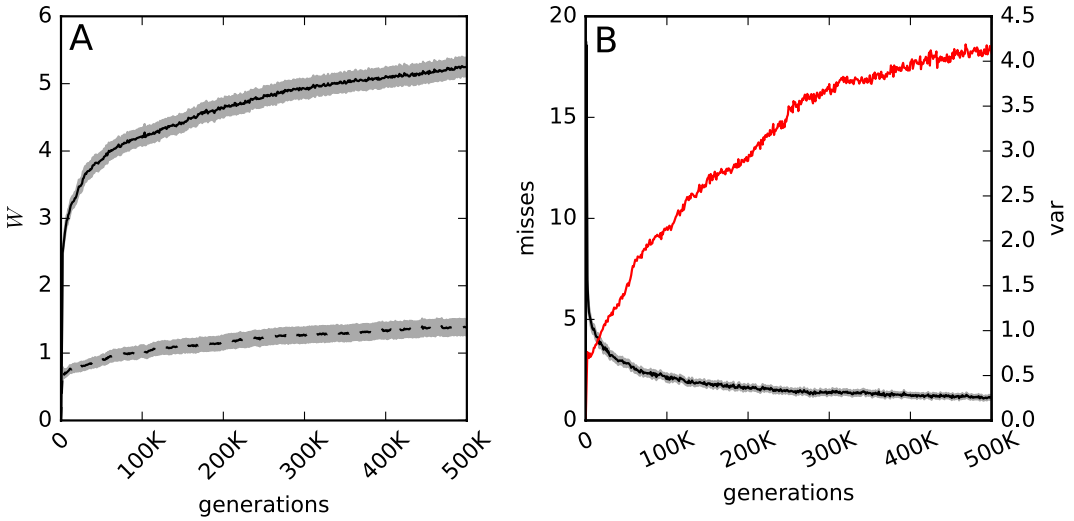

Figure 2. Performance over evolutionary time. Panel A solid line shows how often the goal was reached (*W*) by all 300 replicate experiments across all 24 possible environments for agents on the line of descent. The dotted line is the same average performance for the same agents when their feedback mechanism was disabled. The underlying gray-shade indicates the standard error. Panel B shows how often on average the 300 replicate agents on the line of descent could not reach the goal a single time in any of the 24 possible environments as a black line. In red is the variance in performance on the line of descent as an average over all 300 replicate experiments.

gate: feedback gate. These gates are different from other probabilistic gates, in that they can change their probability distribution during their lifetime based on feedback (for a detailed description, see below). This allows for permanent changes which are akin to LTM. While MBs could already retain information by using hidden states, now they can also change "physically". MBs must evolve to integrate these new gates into their network of other gates and find a way to supply feedback appropriately.

Feedback Gate Usage. To test if the newly introduced feedback gates help evolution and increase performance, we compare three different evolutionary experimental conditions. Agents were evolved over 500,000 generations that could use only deterministic logic gates, only probabilistic logic gates, or all three types of gates–deterministic, probabilistic, and feedback gates to solve the task.

When analyzing the line of descent (LOD; see materials and methods), we find a strong difference in performance across the three evolutionary conditions (see Supplementary Information Fig. 2). None of the 300 agents that were evolved using only probabilistic logic gates were capable of reaching the goal in any of the 24 mappings. The agents that were allowed to use only deterministic logic gates failed to reach the goal in 225 of the 300 experiments, but the remaining 75 agents never reached the goal more than five times. Agents allowed to use all gates including feedback gates only failed to reach the goal in 75 experiments and in the remaining 225 experiments they reached the goal on average 5 times; with the best performer reached the goal 9 times on average. A Wilcoxon rank sum test[57] comparing the final distributions of performances for each experimental condition showed that we can reject the hypothesis that they were drawn from the same distribution (with the lowest p-value smaller than 10^{-20}). This shows that agents that were allowed to use feedback gates outperform all other conditions by far.

It is not entirely surprising to us that agents using only probabilistic gates struggle in this task because agents using probabilistic gates generally evolve slower, which might explain the effect. However, to our surprise, we found a couple of agents who were only allowed to use deterministic gates evolved to solve the task at least a couple of times. In the group that could use all three types of gates, we found 3 agents that could reach the goal on average 5 times using only probabilistic gates, as opposed to the 9 times the top agent with feedback gates could reach the goal on average. This shows two things: the task can be solved using only the gate inputs (i.e. WM) and providing agents with feedback gates during evolution allows them to reach the goal more often. This is an important control because from a computational point of view there is no qualitative difference between WM and LTM as both methods allow for recall of the past.

Populations allowed to use feedback gates quickly evolve the ability to reach the goal in any of the 24 possible environments. The variance of their performance supports the same idea, that agents do not become better by just performing well in one environment, but instead evolve the general ability to learn the mapping each environment presents (See Fig. 2 panel B).

Now that we have shown that agents with feedback gates are capable of evolving a solution to navigate in this environment, we have to ask if they actually utilize the feedback gates. For that, all agents on the LOD were tested again but their feedback gates were kept from changing their probability tables. Comparing these results with the agents performance when using the feedback gates regularly reveals that the agents rely heavily on their feedback gates (see Fig. 2 panel A). This implies that the evolved agents indeed store information about the environment for longer periods using their feedback mechanism. If they would have used hidden states as the only means to save information for later, blocking the feedback signal would not have caused a loss of function. Therefore feedback gates indeed allow for the formation of LTM.

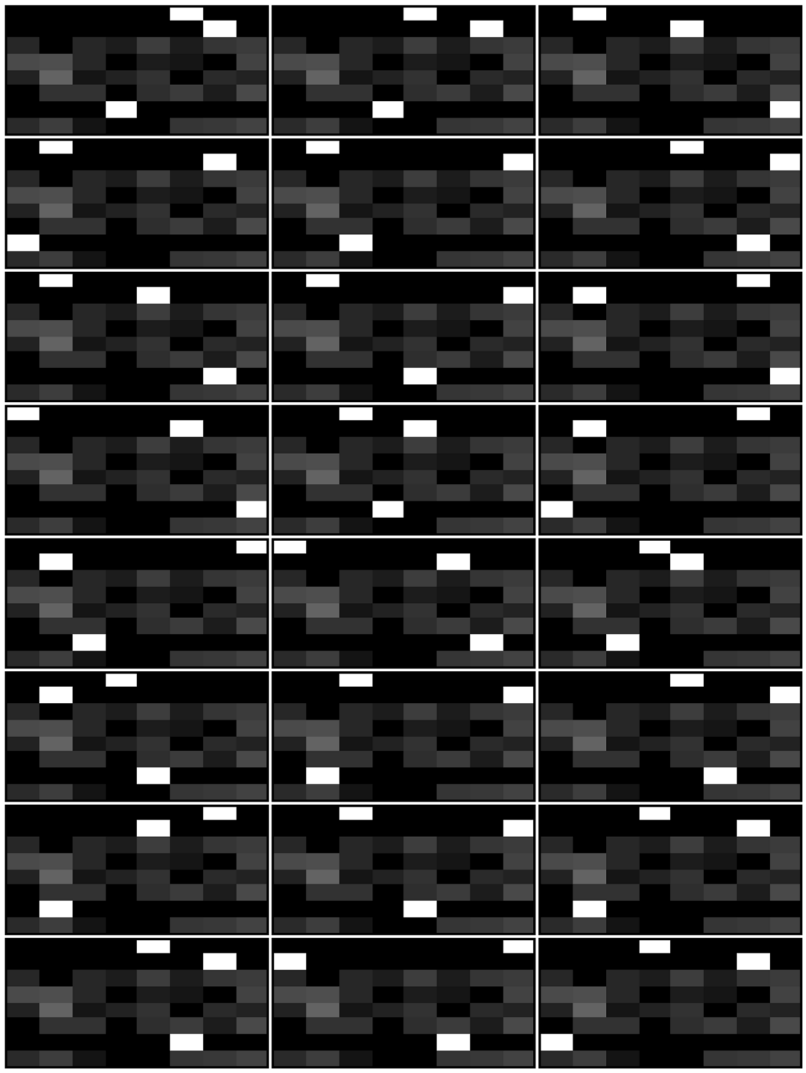

Figure 3. Probability tables of a feedback gate after it learned each of the 24 possible mappings. Each of the 24 gray scale images corresponds to a feedback table adapted during the lifetime of an agent to a different mapping. The darker the color, the lower the probability, observe that rows have to sum to 1.0. Some rows, apparently those of input combinations that were never experienced, remain unadapted. Rows one, two, and seven of each table converge to a single high value surrounded by low probabilities.

Feedback gates change over time. We find that the probability tables modified by feedback become specifically adapted to each of the 24 possible mappings the agents get tested in. See Fig. 3 as an example of the best performing agent using only one feedback gate. Some rows in the probability tables converge to having a single high value that is specific to the environment the agent experienced (for more details see Supplementary Information Figs 3–4). This shows, that indeed feedback gates become specifically adapted to the environment the agent experiences. It also indicates, that agents change their computational machinery according to their environment and do not rely solely on WM to perform their task.

The change to the feedback gates' probability tables can be quantified by measuring the mutual information each table conveys at birth and after the agent completes the task. We find that the mutual information is generally lower at birth (~0.25) and higher at the end of the task (~0.8). This signifies that the agents have more information about the environment at death than they did at birth, as expected. We then compute the difference between both measurements, ($\overline{\Delta}$), which quantifies the change of mutual information over the lifetime of the agent. When discretizing these values for different levels of performance, we find a strong correlation ($r = 0.922$) between performance and increase in mutual information (see Fig. 4). This shows that agents who perform better increase information stored in their feedback gates over their lifetime.

Differences between agents using feedback gates and those who do not. We wanted to test if feedback gates improve the agents ability to learn. Those agents that evolved to use feedback gates generally perform very well in the environment, however, we also find other agents that only use deterministic gates and still have an acceptable performance. This either suggests that feedback gates are either not necessary or that they

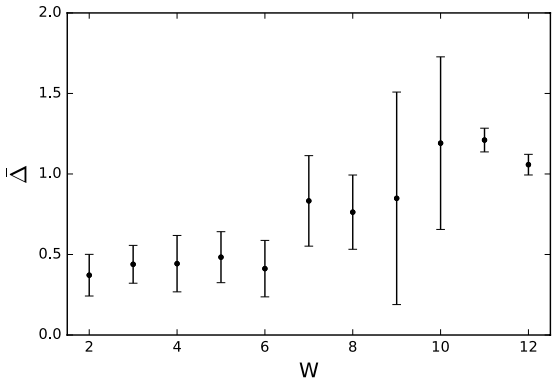

Figure 4. Change in mutual information of feedback gates for different levels of performance. The change in performance ($\overline{\Delta}$) is shown on the y-axis for different performances (W). The performance for all 300 final organisms using all types of gates was measured and put into ten bins, ranging from the lowest performance 2 to the highest performance of 12, with a bin size of 1. Error-bars indicate the variance.

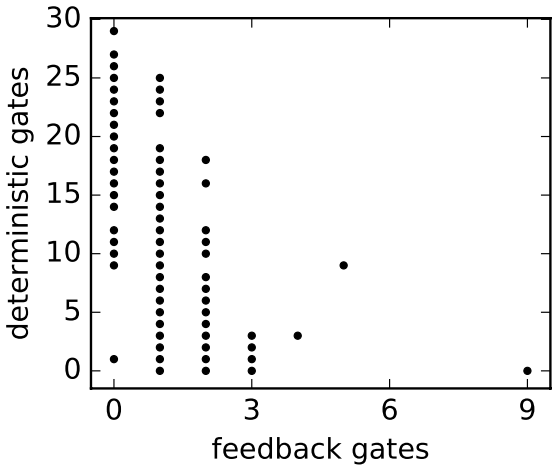

Figure 5. Correlation of number of feedback gates to deterministic gates. Each dot represents the number of feedback gates the last agent on the line of descent has versus the number of deterministic gates it has in 300 replicates. The more feedback gates an agent has the less deterministic gates it evolves; the feedback gates allow agents to decrease brain size while still solving the task. The Pearson correlation coefficient between the number of feedback gates and deterministic gates is -0.52 with p-value of $8.64e^{-06}$.

do not provide enough of an selective advantage to be used every time. It is also possible that there is more than one algorithmic solution to perform well in this navigation task. All of these points suggest that a more thorough analysis and comparison of the independently evolved agents is necessary.

We find that agents that do not use feedback gates require a much greater number of logic gates than those who do (see Fig. 5). This seems intuitive, since feedback gates can store information in their probability tables whereas agents that do not use them need to store all information in their WM. This suggests that there might be a difference in the evolved strategy between those agents that use feedback gates and those who do not. When observing the behavior of the differently evolved agents we find that there are two types of strategies (see Supplementary Information Figs 5–6 for details). Agents that evolved brains without feedback gates use a simple heuristic that makes them repeat their last action when the arrow they stand on points into the direction they are facing. Otherwise, they start rotating until they move off a tile, which often results in them standing again on a tile that points into the direction they are facing, which makes them repeat the last action (see Supplementary Information Fig. 7).

Agents that evolved to use feedback gates appear to actually behave as if they learn how to turn properly. They make several mistakes in the beginning, but after a learning period perform flawlessly. Of the 300 replicate evolutionary experiments where agents were allowed to use all types of logic gates 56 did not evolve using feedback gates. Comparing the actions those 56 agents take with the remaining 244 which do use feedback gates we first find that as expected the group using feedback gates reached the goal more often on average (5.33 times, versus 4.89 times for those agents not using feedback gates) which suggests a difference in behavior. The usage of actions is also drastically different during evolution (see Fig. 6). Agents using feedback gates reduce the instances where they do nothing, minimize turns and maximize moving forward. Agents not using feedback gates are less efficient

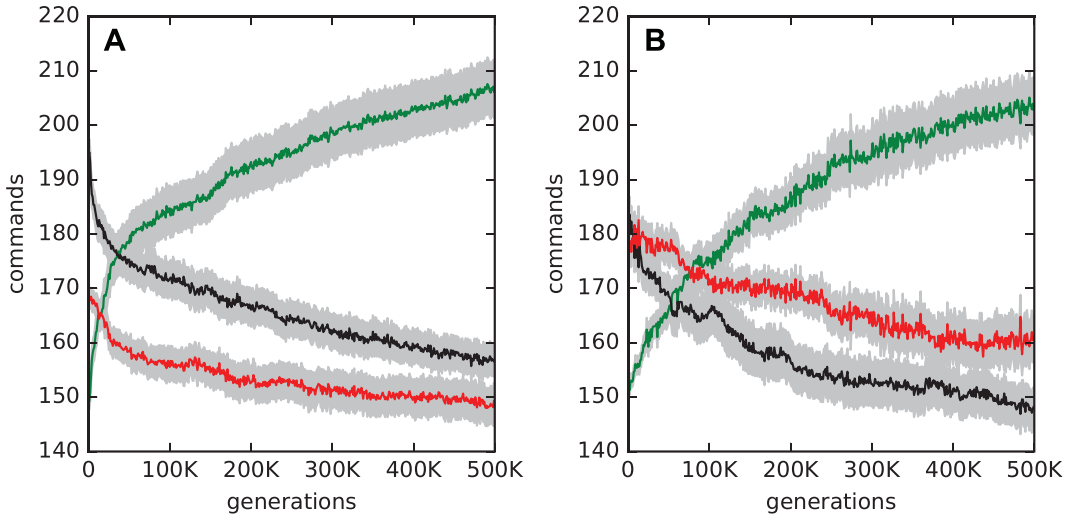

Figure 6. Different actions performed while navigating over evolutionary time. Both panels show the average use of forward (green), do nothing (red), and turn (black) commands over generations. Gray background indicates the standard error. Panel A shows those 244 agents that do use feedback gates, Panel B shows the remaining 56 independently evolved agents that do not use feedback gates. Agents on the LOD were analyzed.

because they rely on forward movements while minimizing times where they do nothing and turns. In conjunction with the observations made before we conclude that indeed agents not using feedback gates use some form of heuristic with a minimal amount of memory while agents using feedback gates learn to navigate the environment properly and quickly (see Supplementary Information Fig. 8 and Section 8 for a more details regarding the evolved solutions).

Discussion

In prior experiments, Markov Brains could evolve to solve various complex tasks and even form memories that represent their environment[27]. However, these MBs use binary hidden states to store information about the environment similar to WM or STM, but lacked a feedback mechanism that allowed them to change their internal structure, very much like long term potentiation would lead to LTM. Other systems like artificial neural networks already allowed for a similar process, where weights could be adapted due to feedback. We augmented MBs with feedback gates that use the multiplicative weight update algorithm to change their probabilities given positive or negative feedback.

It is important to note that this feedback is not directly provided by the environment to the agents but must be internally generated by the MBs. In addition, MBs need to integrate these feedback gates into their cognitive machinery during evolution. We showed that we can successfully evolve MBs to use feedback gates. These gates generated internal feedback is and they change their internal probabilities as expected. However, we also find competing strategies, which instead of evolving to learn deploy some form of heuristic. In the future it might be interesting to study under which evolutionary conditions either heuristics or learning strategies evolve (similar to Kvam 2017). We used a biological inspired task and we see the future application of this technology in the domain of computational modeling to study how intelligence evolved in natural systems and to eventually use neuroevolution as the means to bring about general artificial intelligence. ANNs that are specifically optimized by machine learning will solve specific classification tasks much faster than the model introduced here. The idea is not to present a new paradigm for machine learning, but to implement a tool to study the evolution of learning. While using MBs augmented with feedback gates will probably not be competitive with other supervised learning techniques, it remains an interesting question: How would typical machine learning tools perform if challenged with the task presented here? (see Supplementary Information Fig. 9 for a comparison with Q learning on a single environment).

One problem encountered by systems that receive external feedback is "catastrophic forgetting"[58,59]. When the task changes, feedback will cause the system to adapt to their new objective but they forget or unlearn former capabilities. Requiring adaptive systems to interpret the environment in order to generate internal feedback might be a way to overcome this problem; assuming that the fitness function or evolutionary environment not only changes but also actively rewards organisms who do not suffer from "catastrophic forgetting".

We now have the ability to evolve machines inspired by natural organisms that can learn without an objective external signal. This gives us a tool to study the evolution of learning using a computational model system instead of having to use natural organisms. It will also allow us to study the interplay between evolution and learning (aka Baldwin effect) and explore under which circumstances evolution benefits from learning and when it does not; we propose to use this model to study these questions in the future. Another dimension we will investigate in the future is the ability of MBs to change their connections due to feedback, not just the probabilities within their gates.

Conclusion

As stated before, combining evolution with learning is not a new idea. We think that it is in principle very easy for other systems to internalize the feedback. For example, it should be easy to evolve an artificial neural network to first interpret the environment, and then use this information to apply feedback on itself. However, we need to ask under which circumstances this is necessary. By providing training classes for supervised learning situations we can already create (or deep learn) machines that can learn to classify these classes. In addition, we often find these classifiers exceed human performance[60,61]. If we are incapable of providing examples of correctly classified data we use unsupervised learning methods and only need to provide a fitness function that quantifies performance. But we know that fitness functions can be deceptive and designing them is sometimes more of an art than a science. When interacting with future AI systems we should find a different way to specify what we need them to do. Ideally they should autonomously explore the environment and learn everything there is to know without human intervention - nobody tells us humans what to research and explore, evolution primed us to pursue this autonomously. The work presented here is one step into this direction and will allow us to study evolution of learning in a biological context as well as explore how we can evolve machines to autonomously learn.

Methods

Markov Brains are networks of probabilistic and deterministic logic gates encoded by a genome. The genome contains genes and each gene specifies one logic gate, the logic it performs and how it is connected to sensors, motors, and to other gates[27,45,50]. A new type of gate, the feedback gate, has been added to the Markov Brain framework (https://github.com/LSheneman/autonomous-learning),and this framework has been used to run all the evolutionary experiments. The Markov Brain framework has since been updated to MABE[62]. See below for a detailed description of each component:

Environment. The environment the agents had to navigate was a 2D spatial grid of 64×64 tiles. Tiles were either empty or contained a solid block that could not be traversed. The environment was surrounded by those solid blocks to prevent the navigating agent from leaving that space. At the beginning of each agent evaluation a new environment was generated and $\frac{1}{7}$ of the tiles were randomly filled with a solid block. The randomness of the environments maintains a complex maze-like structure across environments, but no two agents saw the exact same environment.

A target was randomly placed in the environment, and Dijkstra's algorithm was used to compute the distance from all empty tiles to the target tile. These distances were used to label each empty block so that it had an arrow facing to the next closest tile to the target. When there was ambiguity (two adjacent tiles had the same distance) a random tile of the set of closest tiles was chosen. At birth agents were randomly placed in a tile that had a Dijkstra's number of 32 and face a random direction (up, right, down, or left). Due to the random placement of blocks it was possible that the goal was blocked so that there was no tile that is 32 tiles away, in which case a new environment was created, which happened only very rarely.

Agents were then allowed to move around the environment for 512 updates. If they were able to reach the target, a new random start orientation and location with a Dijkstra's number of 32 was selected. Agents used two binary outputs from the MB to indicate their actions– 00, 01, 10, or 11. Each output was translated using a mapping function to one of four possible actions- move forward, do nothing, turn left, or turn right. This resulted in 24 different ways to map the four possible outputs of the MB to the four possible actions that moved the agent. The input sensors gave information about the label of the tile the agent was standing on. Observe that the agent itself had an orientation and the label was interpreted relative to the direction the agent faced. There were four possible arrows the agent could see–forward, right, backward, or left–and were encoded as four binary inputs, one for each possible direction. Beyond the four input and two outputs nodes, agents could use 10 hidden nodes to connect their logic gates. Performance (or fitness) was calculated by exposing the agent to all 24 mappings and testing how often it reached the goal within the 512 updates it was allowed to explore the world. At every update agents were rewarded proportional to their distance to the goal (d), and received a bonus (b) every time they reached the goal, thus the fitness function becomes:

$$W = \prod_{n=0}^{n<24} \left(\left(\sum_{t=0}^{t<512} \frac{1}{1+d} \right) + b \right) \tag{1}$$

Selection. After all agents in a population were tested on all 24 option-to-action mappings at each generation, the next generation was selected using tournament selection where individuals are randomly selected and the one with the best fitness transmits offspring into the next generation[63]. The tournament size was set to five.

Mutation. Genomes for organisms in the first generation were generated randomly with a length of 5000 and 12 start codons were inserted that coded for deterministic, probabilistic, and feedback gates. Each organism propagated into the next generation inherited the genome of its ancestor. The genome had at least 1,000 and at most 20,000 sites. Each site had a 0.003 chance to be mutated. If the genome had not reached its maximum size stretches of a randomly selected length between 128 and 512 nucleotides got copied and inserted at random locations with a 0.02 likelihood. This allowed for gene duplications. If the genome was above 1000 nucleotides, there was a 0.02 chance for a stretch of a randomly selected length between 128 and 255 nucleotides to be deleted at a random location.

Feedback Gates. At every update of a probabilistic gate, an input i resulted in a specific output o. To encode the mapping between all possible inputs and outputs of a gate we used a probability matrix P. Each element of this

matrix P_{io} defined the probability that given the input i the output o occurred. Observe that for each i the sum over all o must be 1.0 to define a probability:

$$1.0 = \sum_{o=0}^{O} P_{io} \tag{2}$$

where O defines the maximum number of possible outputs of each gate.

A feedback gate uses this mechanism to determine its output at every given update. However, at each update we consider the probability P_{io} that resulted in the gate's output to be causally responsible. If that input-output mapping for that update was appropriate then in future updates that probability should be higher. If the response of the gate at that update had negative consequences, then the probability should be lower. As explained above, the sum over all probabilities for a given input must sum to one. Therefore, a single probability can not change independently. If a probability is changed, the other probabilities are normalized so that Eq. 2 remains true.

But where is the feedback coming from that defines whether or not the action of that gate was negative or positive? Feedback gates posses two more inputs, one for a positive signal, and one for a negative signal. These inputs can come from any of the nodes the Markov Brain has at its disposal, and are genetically encoded. Therefore, the feedback can be a sensor or an output of another gate. Receiving a 0at either of the two positive or negative feedback inputs has no effect, whereas reading a 1 triggers the feedback.

In the simplest case of feedback a random number in the range $[0, \delta]$ is applied to the probability P_{io} that was used in the last update of the gate. In case of positive feedback the value is increased; in the case of negative feedback the value is decreased. The probabilities cannot exceed 0.99 or drop below 0.01. The rest of the probabilities are then normalized.

The effects of the feedback gate are immediately accessible to the MB. However, because MBs are networks the signal that a feedback gate generates might need time to be relayed to the outputs via other gates. It is also possible that there is a delay between an agent's actions and the time it takes to receive new sensorial inputs that give a clue about the situation being improved or not. Thus, allowing feedback to occur only on the last action is not sufficient. Therefore, feedback gates can evolve the depth of a buffer that stores prior P_{io} values up to a depth of 4. When feedback is applied all the probabilities identified by the elements in the queue are altered. The δ is determined by evolution and can be different for each element in the queue.

Line of descent. An agent was selected at random from the final generation to determine the line of descent (LOD) by tracing the ancestors to the first generation[64]. During this process, the most recent common ancestor (MRCA) is quickly found. Observe that all mutations that swept the population can be found on the LOD, and the LOD contains all evolutionary changes that mattered.

References

1. Hebb, D. O. *The organization of behavior: A neuropsychological theory* (Psychology Press, 2005).
2. Baldwin, J. M. A new factor in evolution. *The american naturalist* **30**, 441–451 (1896).
3. Sznajder, B., Sabelis, M. & Egas, M. How adaptive learning affects evolution: reviewing theory on the baldwin effect. *Evolutionary biology* **39**, 301–310 (2012).
4. Hinton, G. E. & Nowlan, S. J. How learning can guide evolution. *Complex systems* **1**, 495–502 (1987).
5. Fontanari, J. & Meir, R. The effect of learning on the evolution of asexual populations. *Complex Systems* **4**, 401–414 (1990).
6. Santos, M., Szathmáry, E. & Fontanari, J. F. Phenotypic plasticity, the baldwin effect, and the speeding up of evolution: The computational roots of an illusion. *Journal of theoretical biology* **371**, 127–136 (2015).
7. Dunlap, A. S. & Stephens, D. W. Reliability, uncertainty, and costs in the evolution of animal learning. *Current Opinion in Behavioral Sciences* **12**, 73–79 (2016).
8. Ma, W. J., Husain, M. & Bays, P. M. Changing concepts of working memory. *Nature Neuroscience* **17**, 347–356 (2014).
9. Nadel, L. & Hardt, O. Update on Memory Systems and Processes. *Neuropsychopharmacology* **36**, 251–273 (2010).
10. Kandel, E. R., Dudai, Y. & Mayford, M. R. The Molecular and Systems Biology of Memory. *Cell* **157**, 163–186 (2014).
11. Squire, L. R. & Wixted, J. T. The Cognitive Neuroscience of Human Memory Since H.M. *Annual Review of Neuroscience* **34**, 259–288 (2011).
12. Abraham, W. C. & Robins, A. Memory retention – the synaptic stability versus plasticity dilemma. *Trends in Neurosciences* **28**, 73–78 (2005).
13. McKenzie, S. & Eichenbaum, H. Consolidation and Reconsolidation: Two Lives of Memories? *Neuron* **71**, 224–233 (2011).
14. Russell, S. J. & Norvig, P. *Artificial intelligence: a modern approach (3rd edition)* (Prentice Hall, 2009).
15. Schmidhuber, J. Evolutionary principles in self-referential learning. *On learning how to learn: The meta-meta- hook.) Diploma thesis, Institut f. Informatik, Tech. Univ. Munich* (1987).
16. Watkins, C. J. C. H. *Learning from delayed rewards*. Ph.D. thesis, University of Cambridge England (1989).
17. Mnih, V. *et al*. Playing atari with deep reinforcement learning. *arXiv preprint arXiv* **1312**, 5602 (2013).
18. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Networks* **61**, 85–117 (2015).
19. Zhang, H., Wu, W. & Yao, M. Boundedness and convergence of batch back-propagation algorithm with penalty for feedforward neural networks. *Neurocomputing* **89**, 141–146 (2012).
20. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
21. Kaleh, G. K. & Vallet, R. Joint parameter estimation and symbol detection for linear or nonlinear unknown channels. *IEEE Trans. Communications ()* **42**, 2406–2413 (1994).
22. Baggenstoss, P. M. A modified Baum-Welch algorithm for hidden Markov models with multiple observation spaces. *IEEE Transactions on Speech and Audio Processing* **9**, 411–416 (2001).
23. Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* **8**, 229–256 (1992).
24. Kaelbling, L. P., Littman, M. L. & Moore, A. W. Reinforcement learning: A survey. *Journal of artificial intelligence research* **4**, 237–285 (1996).
25. Arora, S., Hazan, E. & Kale, S. The Multiplicative Weights Update Method: a Meta-Algorithm and Applications. *Theory of Computing* (2012).
26. Freund, Y. & Schapire, R. E. Adaptive game playing using multiplicative weights. *Games and Economic Behavior* **29**, 79–103 (1999).

27. Marstaller, L., Hintze, A. & Adami, C. The Evolution of Representation in Simple Cognitive Networks. *Neural Computation* **25**, 2079–2107 (2013).
28. Hintze, A. *et al*. Markov brains: A technical introduction. *arXiv preprint arXiv* **1709**, 05601 (2017).
29. Kaelbling, L. P., Littman, M. L. & Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial intelligence* **101**, 99–134 (1998).
30. Sims, K. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, 15–22 (ACM, 1994).
31. Stanley, K. O. & Miikkulainen, R. Evolving neural networks through augmenting topologies. *Evolutionary computation* **10**, 99–127 (2002).
32. Gauci, J. & Stanley, K. O. Autonomous evolution of topographic regularities in artificial neural networks. *Neural computation* **22**, 1860–1898 (2010).
33. Spector, L. & Robinson, A. Genetic programming and autoconstructive evolution with the push programming language. *Genetic Programming and Evolvable Machines* **3**, 7–40 (2002).
34. Greve, R. B., Jacobsen, E. J. & Risi, S. Evolving neural turing machines. In *Neural Information Processing Systems: Reasoning, Attention, Memory Workshop* (2015).
35. Yao, X. Evolving artificial neural networks. *Proceedings of the IEEE* **87**, 1423–1447 (1999).
36. Stanley, K. O., Bryant, B. D. & Miikkulainen, R. Evolving adaptive neural networks with and without adaptive synapses. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, vol. 4, 2557–2564 (IEEE, 2003).
37. Gomez, F. & Schmidhuber, J. Evolving modular fast-weight networks for control. *Artificial Neural Networks: Formal Models and Their Applications–ICANN* **2005**, 750–750 (2005).
38. Urzelai, J. & Floreano, D. Evolution of adaptive synapses: Robots with fast adaptive behavior in new environments. *Evolution* **9** (2006).
39. Soltoggio, A., Bullinaria, J. A., Mattiussi, C., Dürr, P. & Floreano, D. Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In *Proceedings of the 11th International Conference on Artificial Life (Alife XI)*, LIS-CONF-2008-012, 569–576 (MIT Press, 2008).
40. Lüders, B., Schläger, M. & Risi, S. Continual learning through evolvable neural turing machines. In *NIPS 2016 Workshop on Continual Learning and Deep Networks (CLDL 2016)* (2016).
41. Tonelli, P. & Mouret, J.-B. On the relationships between synaptic plasticity and generative systems. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 1531–1538 (ACM, 2011).
42. Risi, S. & Stanley, K. O. A unified approach to evolving plasticity and neural geometry. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, 1–8 (IEEE, 2012).
43. Coleman, O. J. & Blair, A. D. Evolving plastic neural networks for online learning: review and future directions. In *Australasian Joint Conference on Artificial Intelligence*, 326–337 (Springer, 2012).
44. Greve, R. B., Jacobsen, E. J. & Risi, S. Evolving neural turing machines for reward-based learning. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference*, 117–124 (ACM, 2016).
45. Edlund, J. A. *et al*. Integrated Information Increases with Fitness in the Evolution of Animats. *PLoS Comput Biol* **7**, e1002236 (2011).
46. Olson, R. S., Hintze, A., Dyer, F. C., Knoester, D. B. & Adami, C. Predator confusion is sufficient to evolve swarming behaviour. *Journal of The Royal Society Interface* **10**, 20130305–20130305 (2013).
47. Hintze, A. *et al*. Evolution of Autonomous Hierarchy Formation and Maintenance. In *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, 366–367 (The MIT Press, 2014).
48. Joshi, N. J., Tononi, G. & Koch, C. The minimal complexity of adapting agents increases with fitness. *PLoS Comput Biol* (2013).
49. Albantakis, L., Hintze, A., Koch, C., Adami, C. & Tononi, G. Evolution of Integrated Causal Structures in Animats Exposed to Environments of Increasing Complexity. *PLoS Comput Biol* **10**, e1003966–19 (2014).
50. Schossau, J., Adami, C. & Hintze, A. Information-Theoretic Neuro-Correlates Boost Evolution of Cognitive Systems. *Entropy* **18**, 6–22 (2016).
51. Kvam, P., Cesario, J., Schossau, J., Eisthen, H. & Hintze, A. Computational evolution of decision-making strategies. *arXiv preprint arXiv* **1509**, 05646 (2015).
52. Kvam, P. & Arend, H. Rewards, risks, and reaching the right strategy: Evolutionary paths from heuristics to optimal decisions. *Evolutionary Behavioral Sciences, invited submission for the Special Issue on Studying Evolved Cognitive Mechanisms* ((under review)).
53. Chapman, S., Knoester, D. B., Hintze, A. & Adami, C. Evolution of an artificial visual cortex for image recognition. *ECAL* 1067–1074 (2013).
54. Chapman, S. D., Adami, C. & Wilke, C. O. & KC, D. B. The evolution of logic circuits for the purpose of protein contact map prediction. *PeerJ* **5**, e3139 (2017).
55. Russell, S. & Norvig, P. Ai a modern approach. *Learning* **2**, 4 (2005).
56. Clark, A. *Being there: Putting brain, body, and world together again* (MIT press, 1998).
57. Wilcoxon, F., Katti, S. & Wilcox, R. A. Critical values and probability levels for the wilcoxon rank sum test and the wilcoxon signed rank test. *Selected tables in mathematical statistics* **1**, 171–259 (1970).
58. French, R. M. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences* **3**, 128–135 (1999).
59. Ellefsen, K. O., Mouret, J.-B. & Clune, J. Neural modularity helps organisms evolve to learn new skills without forgetting old skills. *PLoS Comput Biol* **11**, e1004128 (2015).
60. Ciregan, D., Meier, U. & Schmidhuber, J. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 3642–3649 (IEEE, 2012).
61. Silver, D. *et al*. Mastering the game of go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).
62. Hintze, A. & Bohm, C. Mabe. https://github.com/ahnt/MABE (2016).
63. Blickle, T. & Thiele, L. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation* **4**, 361–394 (1996).
64. Lenski, R. E., Ofria, C., Pennock, R. T. & Adami, C. The evolutionary origin of complex features. *Nature* **423**, 139–144 (2003).

Acknowledgements

This work is in part supported by the NSF BEACON-center for the study of evolution in action under Cooperative Agreement DBI-0939454. We would like to thank Christoph Adami, Charles C. Ofria, and Fred C. Dyer for useful discussions.

Author Contributions

L.S. and A.H. conceived of the experiments and wrote the manuscript. L.S. performed all computational experiments and data analysis.

Additional Information

Supplementary information accompanies this paper at https://doi.org/10.1038/s41598-017-16548-2.

Competing Interests: The authors declare that they have no competing interests.

Publisher's note: Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.