nature machine intelligence

ARTICLES

https://doi.org/10.1038/s42256-020-00267-x

Check for updates

Network of evolvable neural units can learn synaptic learning rules and spiking dynamics

Paul Bertens[1] and Seong-Whan Lee[1,2] ✉

Although deep neural networks have seen great success in recent years through various changes in overall architectures and optimization strategies, their fundamental underlying design remains largely unchanged. Computational neuroscience may provide more biologically realistic models of neural processing mechanisms, but they are still high-level abstractions of empirical behaviour. Here we propose an evolvable neural unit (ENU) that can evolve individual somatic and synaptic compartment models of neurons in a scalable manner. We demonstrate that ENUs can evolve to mimic integrate-and-fire neurons and synaptic spike-timing-dependent plasticity. Furthermore, by constructing a network where an ENU takes the place of each synapse and neuron, we evolve an agent capable of learning to solve a T-maze environment task. This network independently discovers spiking dynamics and reinforcement-type learning rules, opening up a new path towards biologically inspired artificial intelligence.

Neural processes such as synaptic learning[1] and overall network dynamics[2,3] are extremely complex and interact in a way that is still not well understood despite much research towards this goal[4–6]. Mathematical models are commonly used to approximate neurons, synapses and learning mechanisms[1]; however, neurons can behave in many different ways and no single model can accurately capture all experimentally observed behaviour. This neural complexity is mostly at a small scale, such as the cellular local interactions that can lead to an intelligent overall system. Although information processing at the network level is generally well understood (and the basis for artificial neural networks[7,8]), it is how individual neurons are actually capable of giving rise to a network's overall behaviour and learning capability that is largely unknown.

Current neural networks

The following subsections describe the neural network models and related algorithms currently in use, from both a neuroscience and machine learning perspective.

Biological neural networks. Complexity quickly arises when looking in more detail at the smaller scale of neurons and synapses. Different ion channels in the neuron are responsible for generating action potentials (spikes) or graded potentials (real valued), mainly sodium (Na^+), potassium (K^+) and calcium (Ca^{2+})[9]. These channels are also responsible for triggering the release of chemical neurotransmitters, the main method of communication between neurons in biological neural networks. Many types of neurotransmitters exist that each have distinct roles, for example, dopamine[10] (related to learning), $GABA_B$[11] (inhibitory), acetylcholine[12] (motor neurons) and glutamate[13] (excitatory). Axons and dendrites can also actively transmit other information, for example, vesicles[14] (which hold neurotransmitters) and mitochondria[15] that can change a neuron's behaviour. Deriving models that are capable of capturing all of this behaviour is thus an active research area[14,16–19]. These mechanisms might, however, simply be the result of biological constraints and evolutionary processes, and it is still uncertain which exact structures need to exist for intelligent behaviour to emerge.

Models in the field of computational neuroscience generally focus on modelling individual neurons and synaptic learning behaviour. Simple integrate-and-fire (IAF) neurons[20] assume that input potential is summated over time and a spike occurs once this reaches some threshold. More complex models also try to capture the Na^+ and K^+ channels, as was achieved in the Hodgkin–Huxley model[21], which generates more realistic action potentials.

Hebbian plasticity was one of the first proposed models[22] to describe synaptic learning, it states that the synaptic weight increases if the presynaptic neuron fires before the postsynaptic neuron. This change in synaptic weight can then increase or decrease the firing rate of the postsynaptic neuron in the future when receiving a similar stimuli. A more extensive model that also takes the spike timing into account is spike-timing-dependent plasticity[1] (STDP), which updates the weights depending on the spike timing between the pre- and postsynaptic neuron. If a presynaptic spike fires before the post synaptic spike it causes long-term potentiation, which increases the synaptic weight proportional to the timing difference. If the presynaptic spike fires after the postsynaptic spike, it causes long-term depression, which decreases the synaptic weight proportional to the timing difference. Please refer to the Methods for a more mathematical description of the STDP model.

Artificial and recurrent neural networks. Deep-learning models have been very successful in many practical applications[8] and although spiking neural networks have been developed[23,24], so far they have seen limited success due to their high computational demand and difficulty in training. Artificial neural networks—the basis for deep learning models—are especially effective in performing function approximation and are known to be universal function approximators[25]. Recurrent neural network (RNN) architectures have also been investigated to process sequential data and allow memory to be stored between successive time-steps[8], most commonly long short-term memory units (LSTMs)[26] and gated recurrent units (GRUs)[27]. These add gating mechanisms to standard RNNs to allow for easier learning of long-range dependencies and to avoid vanishing gradients in backpropagation.

[1]Department of Brain and Cognitive Engineering, Korea University, Seoul, South Korea. [2]Department of Artificial Intelligence, Korea University, Seoul, South Korea. ✉e-mail: sw.lee@korea.ac.kr

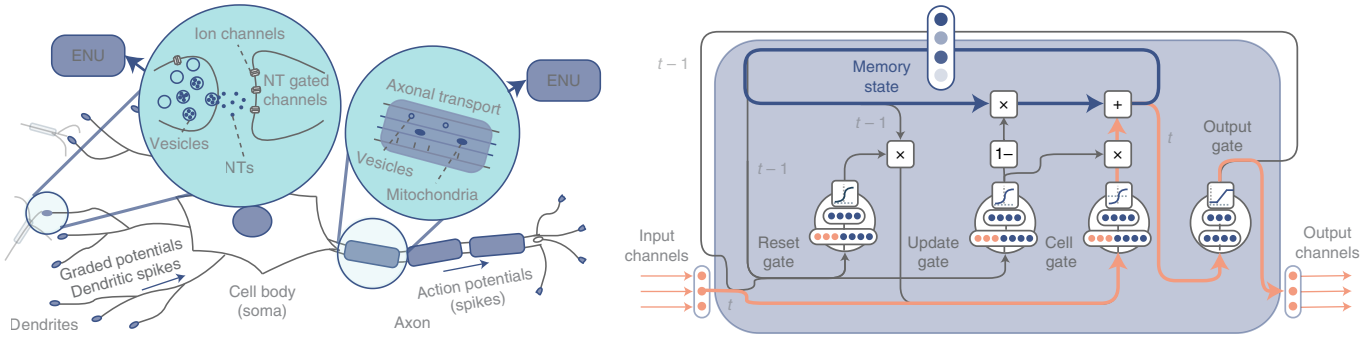


Fig. 1 | A biological neuron and an ENU that can be used to approximate its internal mechanisms. Different evolvable gating mechanisms protect the internal memory of the ENU (storing dynamic parameters). This memory can potentially encode and store the neuronal behaviour, and determines how information is processed. This could be considered analogous to biophysical events or states (for example, the membrane potential, synaptic weights, protein states and so on). The reset gate can forget past information, the update gate determines how much we change the memory state (our dynamic parameters) and the cell gate determines the new value of the dynamic parameters. An additional output gate is also used to reduce the number of output channels and to allow for spike generation to potentially evolve. In this diagram, we have four dynamic parameters, three input channels and three output channels. NT, neurotransmitter. *t*, time step.

However, deep-learning-based models suffer from several limitations[4]. They require backpropagation of errors through the whole network, have difficulty performing one-shot learning (learning from a single example) and suffer from catastrophic forgetting of a previous task once trained on a new task. They are also extreme abstractions of biological neural networks, only considering a single weight value on the connections and having a single real valued number as their output.

Evolutionary algorithms. Evolutionary algorithms have been widely applied to a variety of domains[28,29], and many related optimization algorithms exist that are based on the evolutionary principles of mutation, reproduction and survival of the fittest. Evolution strategies have recently been successfully applied to train deep neural networks, despite the large amount of parameters of most neural network architectures[30]. The advantage of evolution strategies is that they allow for non-differentiable objective functions, and, when training RNNs, do not require backpropagation through time. This means we can optimize and evolve the parameters of a model to solve any desired objective we want and potentially learn over arbitrarily long sequences.

A new type of biologically inspired neural network

Past attempts to model biological neural networks have mostly been focused on manually deriving mathematical rules and abstractions based on experimental data. In this paper we take a different approach. We approximate the functionality of different components in neural networks using artificial RNNs, such that each synapse and neuron in the network is an RNN. We would ultimately like to evolve a type of mini-agent (the RNN) that—when duplicated and connected together in an overall neural network—exhibits intelligent learning behaviour. To achieve this, we propose an RNN-based evolvable neural unit (ENU), which is able to learn to store relevant information in its internal state memory and perform complex processing on the received input using that memory. As all ENU weights are shared across all synaptic or neural compartments, learning different behaviour can only occur by learning (evolving) to store and update dynamic parameters in the ENU memory state. We demonstrate that such an ENU could be evolved to approximate STDP and IAF neurons. We then also show that by combining multiple ENUs into a larger network, they can evolve to learn to solve a maze environment task. These networks of ENUs evolve spiking dynamics and synaptic learning rules that can update the way information flows throughout the network (which is dependent on the external reward received). They thus successfully evolve complex information processing and reinforcement-type learning behaviour, opening up a new path towards a more flexible and powerful neural network.

ENUs. As the basis for an evolvable RNN to approximate neural and synaptic behaviour, we build on previous work on modelling long-term dependencies through GRUs[27]. We extend on this model and add an extra output gate that applies a non-linear activation function and feeds this back to the input, which simultaneously also reduces the number of possible output channels (improving computational complexity). They are implemented in such a way that allows them to be combined and evolved efficiently into a larger overall network. We term these units ENUs.

Using a gating structure allows us to have fine control over how different input influences the internal memory state (storing dynamic parameters), which in turn controls how that input is processed and how the dynamic parameters are updated. The input and output of the ENU is a vector, where each value in the vector can be considered a type of channel used to transmit information between different ENUs. The internal memory state is also a vector that can store multiple values. It enables us to condition the received input on the dynamic parameters stored in the memory state, which control both the output and gating behaviour of the ENU (see Fig. 1). Each gate in the model is a standard single layer artificial neural network with k output units and an evolvable weight matrix w, where k is equal to the internal memory state size. These gates can process information from the current input channels and previous memory state. This allows us to, for example, evolve a function that adds some value to a dynamic parameter in the memory state, but only if there is a spike at a certain input channel. Evolving such units to perform complex functions then becomes considerably easier than in standard RNNs, which can suffer from vanishing values and undesired updates to their internal memory state[26]. Protecting updates to the memory state is especially important in our case, as they define the dynamic parameters that determine the behaviour of our ENU, and should be able to persist over their entire lifetime. Please refer to the Methods for a more detailed description of (and the equations associated with) the gating mechanisms

Intuitively, this ENU-type architecture allows us to, for example, evolve spike-based behaviour through storing and summing received input into the memory state then, once some threshold is

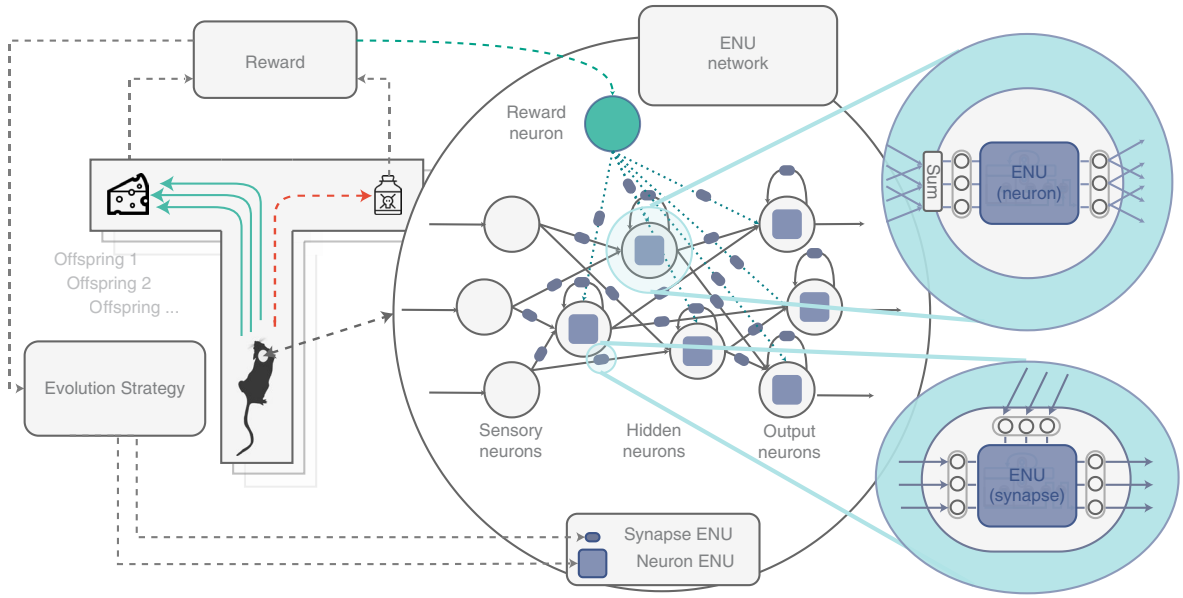


Fig. 2 | Network of ENUs. An example of evolving reinforcement-based learning to solve a T-maze environment task. Multiple offspring are simulated in parallel to evaluate the fitness (offspring 1, offspring 2 and so on). Reward from the environment determines the fitness of the agent (that is, the mouse). This reward is also passed to the reward neuron in the ENU network. The reward neuron connects to the other ENU neurons, allowing them to evolve to update their incoming synapse ENUs according to the reward obtained. Several sensory input neurons are used to detect colours in front of the agent, whereas the output neurons determine the action the agent has to take (forward, left or right).

reached, the output gate can evolve to activate and output a value, a so-called spike. This spike is then fed back into the input, allowing the reset and update gate to evolve to reset the internal memory state.

For synaptic compartments, the internal state could evolve to memorize when a pre- and postsynaptic spike occurs, and also store and update some dynamic weight parameter that can change how the input is processed and passed to the postsynaptic neuron. Furthermore, multiple input and output channels allow flexibility in evolving different type of information processing mechanisms, using, for example, a type of neurotransmitter, or even functions analogous to dendritic and axonal transport.

Network of ENUs. By combining multiple ENUs into a single network, we can construct a ENU-based neural network (ENU-NN). This network connects multiple neural and synaptic compartment ENU models together, as seen in Fig. 2. All soma and synaptic neuronal compartments share the same ENU gate parameters, and we only evolve the weights of these shared gates. This means we have two figurative chromosomes to evolve, one for the synapses and one for the neurons, shared across all synapses and neurons; however, they each have unique internal memory state variables, which allows for complex signal processing and learning behaviour to occur, as the compartments can evolve to update their internal states depending on the local input they receive. Synaptic plasticity (the synaptic weights) in such a model would thus no longer be encoded in the weights of the network directly, but could instead evolve to be stored and dynamically updated in the internal states of the synapse ENUs as a function of the current and past input.

To evolve reinforcement-type learning behaviour in such a network, we can construct a sparse recurrent network with several input sensory neurons that detect, for example, different colours in front of the agent (which has an ENU-NN) in a given environment. We can also designate some ENU neurons as output motor neurons that determine the action the agent should take. Furthermore, to allow reward feedback we can have a reward neuron that uses different ENU input channels to indicate environmental rewards obtained (see Fig. 2).

The main components of the ENU-NN are as follows:

- Neuron ENU: analogous to the biological cell body and axon. It is responsible for transforming the summated input from its connected synapses. It can evolve to use multiple channels to transmit information analogous to biophysical events (for example, spikes or neurotransmitters).
- Synapse ENU: analogous to the biological dendrite and synapse. It transforms information from the pre- to postsynaptic ENU neuron, similar to weights in artificial neural networks; however, it is allowed to use multiple channels to learn to transmit information analogous to spikes, graded potentials or other types of dendritic transport. It also has a feedback connection from the postsynaptic neuron, allowing it to potentially evolve STDP-type learning rules. It can thus integrate information from both the pre- and postsynaptic ENU neuron to update its behaviour.
- Integration step: sums the output per channel of each incoming synapse connection, which is then processed by the postsynaptic neuron ENU.

Learning synaptic learning rules and spiking dynamics

We first use individual models of the proposed ENU to approximate existing mathematical functions of IAF neurons and STDP. This demonstrates the flexibility of the proposed ENUs to potentially evolve similar neuronal computations as observed in the brain. We then combine several ENUs into a network as described in Fig. 2, which are evolved from scratch to solve a maze environment task. This network of ENUs is evolved without any predefined rules on the synaptic or neuronal computations, it is only evolved to maximize fitness in the given environment. Results for each of the experiments are given below. Please refer to the Methods for

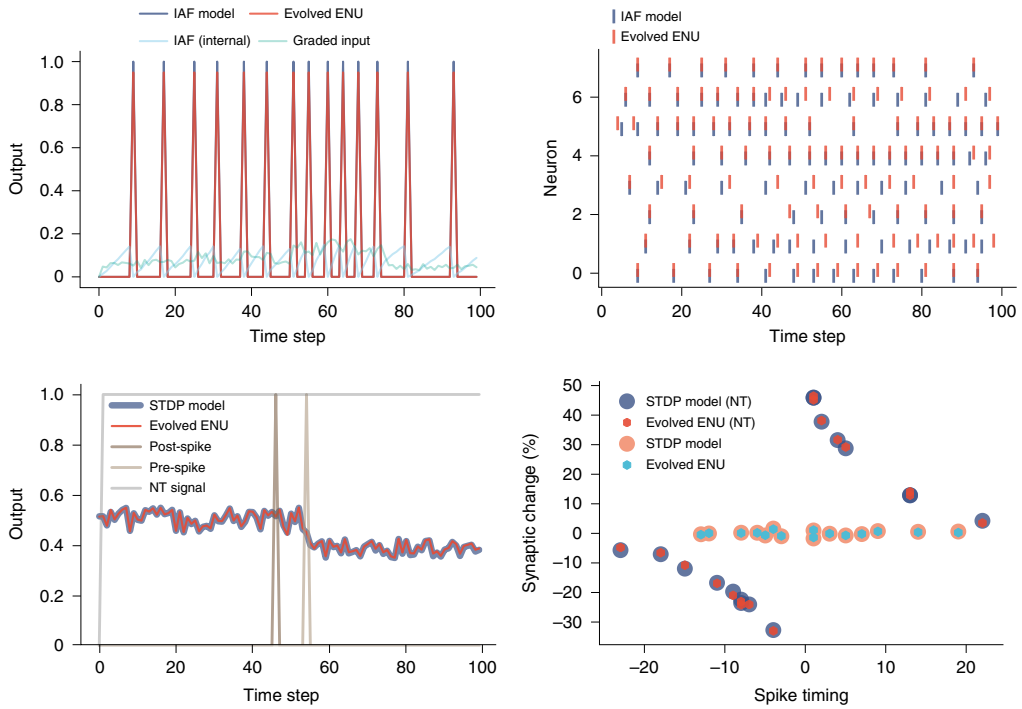


Fig. 3 | Result of evolving IAF neurons and neuromodulated STDP. For a single input example (left) and multiple inputs (right). The evolved ENU model closely matches the actual IAF model (top), properly integrating information and spiking at the right time once a threshold is reached. In the case of the STDP model (bottom), the ENU evolved to update the synaptic weight only when the NT signal was present in the input, and evolved to update the weights relative to the timing of the pre- and postsynaptic spike, matching the original STDP function.

a more detailed description of the exact experimental set-up and parameters.

Evolving IAF neurons using a single ENU. Results of evolving IAF neurons for 3,000 generations are shown in Fig. 3. It can be seen that the ENU is able to accurately mimic the underlying IAF model, properly integrating the received graded input and outputting a spike at the right time. This shows an ENU is capable of evolving to approximate spiking behaviour. A more detailed comparison of the firing rate with the input current of both the underlying IAF model and approximated ENU can be found in Extended Data Fig. 2.

Evolving neuromodulated STDP using a single ENU. Figure 3 illustrates the results after evolving an STDP-type learning rule for 10,000 generations. If the presynaptic spike occurs after the postsynaptic spike, the output intensity of the graded input potential is reduced, as in the standard STDP rule (and vice versa). The evolved ENU was also able to only update the output when a given artificial input neurotransmitter is present, giving us a type of neuromodulated STDP[31]. To approximate such behaviour, it had to evolve the multiplication operation of its internal memory state with the input, as initially the ENU had no such operation. This demonstrates that an ENU is capable of evolving complex synaptic type learning through memorizing pre- and postsynaptic spikes in its internal memory, and by storing and updating some dynamic parameter that changes how the incoming input is processed (analogous to a synaptic weight).

Evolving reinforcement learning in a network of ENUs. Results for evolving reinforcement learning behaviour in a network of ENUs after 30,000 generations in a T-maze environment are shown in Fig. 4. The agent starts at the bottom of the T-maze and it has two possible locations for food and poison that get randomly swapped (see also Fig. 2). A single generation is one episode in the environment (one simulation run) and lasts for 400 time steps. All of the dynamic parameters of each synapse and neuron reset each generation, meaning it always has to relearn which sensory neuron leads to negative or positive rewards and which output neuron performs what action as if it is reborn (a blank slate).

Interestingly, we obtain spike-like patterns on the output neurons even though we never strictly enforce such behaviour. We only

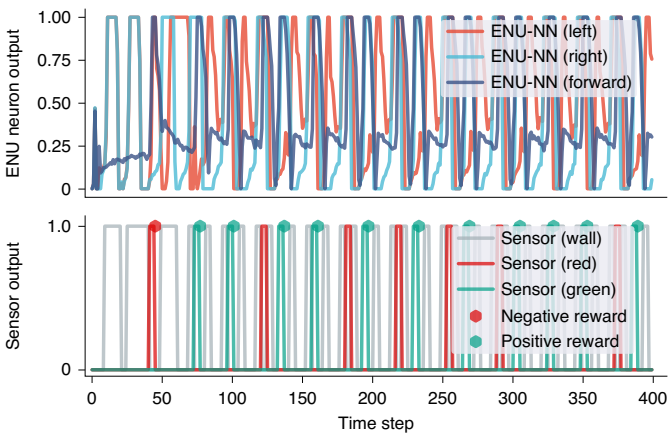


Fig. 4 | Results of evolving reinforcement-type learning behaviour in a network of ENUs. Both the output of the agent (top) and input (bottom) are shown. Spike-like patterns on the output neuron are evolved completely independently, even though we did not directly optimize for such behaviour. Furthermore, it can be seen that the agent performs one-shot learning when eating the poison, only eating it once and subsequently always avoiding it.

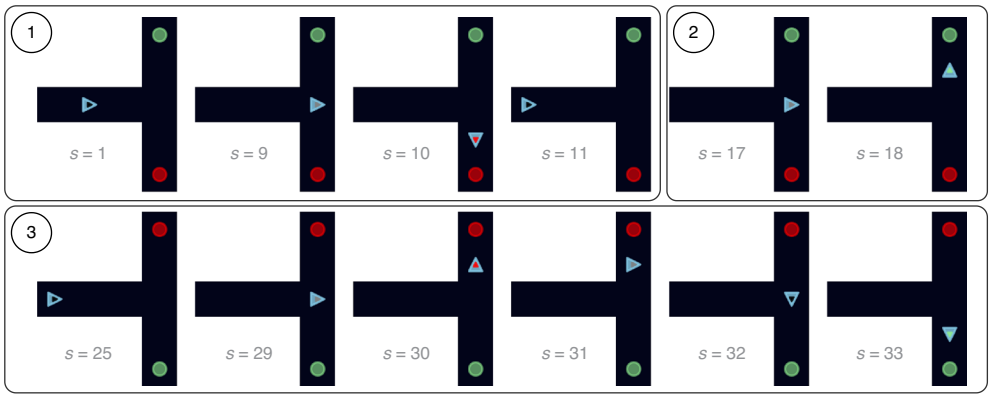


Fig. 5 | Example of the ENU-NN agent's evolved behaviour. The number of steps taken in the environment at each panel is indicated by *s*. The agent evolved to learn to: remember which food gives a negative reward (1), avoid such reward in the future by taking a different route (2), and adapt to turn around once the food and poison are switched (3).

optimize to maximize the reward in the environment. This could partially be explained by the sensory input neurons outputting spikes as well; however, the resulting output is not identical and it still had to evolve to process those input spikes and integrate them properly across the network and at the output. An example of the steps taken by the agent is also given in Fig. 5. We can see that the agent at first detects red, eats the poison and subsequently receives a negative reward. After that the agent learns to go the other way to get the food instead (detecting green). Once the food is switched it detects the poison but does not eat it (and so does not receive a negative reward), as it has now learned to turn around and obtain food on the other side instead.

The performance (and fitness) is measured by how good the agent is at getting food and avoiding poison. Agents (which have ENU neural networks) will thus have a higher fitness if they are able to learn quicker from negative and positive rewards in the environment. This means that agents that are better able to learn to store the right information in the internal memory of each individual ENU, and update their behaviour appropriately, will have a higher overall fitness. Evolution strategies then update the shared parameters of the ENU gates in the direction of the mutated parameters of these highest-performing agents. This leads it to improve its ability to process information in the ENU network and learn from reward feedback. It has to evolve such in a way that all ENU compartments with the shared gate parameters can cooperate and update their unique internal state to achieve a higher fitness. It thus evolves a type of universal function that is the same across all neurons and synapses and updates their internal dynamic parameters depending on the input. Please refer to the Methods for a more detailed explanation of evolution strategies and how the ENU gate parameters are updated.

The agent is able to learn through the evolved ENU-NN that the sensory neuron that detects the colour red (and activation of the output neuron that results in eating it) leads to a negative reward. This means that it evolved a mechanism for updating the ENU synapse between those neurons to have an inhibitory effect, such that the next time it detects red, the previous action that led to eating the poison is not chosen (it is inhibited) and instead another action is taken.

Ablation and comparison study. To evaluate the importance of different components of the proposed ENU-NN, we compared its performance with standard GRU and LSTM units (see Table 1). The results show that the ENU was better able to mimic the IAF neuron behaviour and the STDP learning rule. As was mentioned in Fig. 1, the motivation for adding the feedback output gate was to ensure we get a type of reset behaviour in the inner memory state of the ENU, to learn spiking-based behaviour once some threshold is reached. Having such an output gate with a feedback connection makes learning such behaviour considerably easier, which explains the improvement in performance when evolving IAF neurons. Although it was not expected to necessarily help with evolving an STDP-type learning rule, results do show an improved convergence rate and performance using an ENU over a standard GRU or LSTM. We also performed additional ablation experiments that remove the feedback connection of the ENU (NFENU) and found that the convergence rate and performance is also worse. This shows that having this extra feedback connection has a positive effect on evolving the type of behaviour we desire. We also attempted to share the parameters between the ENU neuron and ENU synapse (SHAREDENU); it was found that this reduced performance considerably. This is in line with our initial motivation for separating the parameters, in that they serve very distinct functions within the network. The ENU neuron is supposed to process the summed synaptic inputs, whereas the synapse ENU transforms and transmits information from the pre- to postsynaptic neuron. A more detailed convergence analysis for each model can be found in Extended Data Fig. 1.

Further experiments were also performed that evolved ENUs to approximate a more complex arbitrary synaptic update rule and to solve a more complex double T-Maze environment. Please refer to Extended Data Figs. 3–5 and Extended Data Table 1 for a more detailed analysis of these results.

Table 1 | Comparison and ablation study final performance

	IAF	STDP	T-Maze NN
ENU	**−18.22 ± −2.44**	**−8.42 ± 0.50**	**36.65 ± 0.027**
GRU	−47.73 ± 7.59	−96.34 ± 0.50	26.16 ± 0.066
LSTM	−43.54 ± 7.30	−41.52 ± 1.13	26.33 ± 0.019
NFENU	−32.34 ± 8.34	−60.55 ± 4.58	31.18 ± 0.018
SHAREDENU	N/A	N/A	21.37 ± 0.034

Comparing the IAF, STDP and T-Maze experiments. The mean and standard deviation of the fitness are shown for 30 trial runs. For the IAF and T-Maze experiments, the difference between LSTMs and GRUs was found to be not significant ($p = 0.52$). The ENU was found to perform better ($p < 0.001$) in all experiments over the compared models and ablation studies.

Discussion

There are distinct differences between a network of ENUs and a standard DNN, the latter of which uses only a single value for the synaptic weight parameter (which is static) and neuron output.

The same holds for spiking neural network models with STDP-type learning rules. The activation function and update rules of these models are also fixed. In our case, each dynamic parameter stored inside ENUs is a vector capable of storing multiple values that can influence the operation performed by the synaptic and neuronal ENU compartments. Each ENU evolves the ability to update those dynamic parameters based on the reward obtained and local input received, unlike DNNs, which require backpropagation over the entire network given some supervised learning signal or reward signal[32].

There has also been extensive work on evolving neural networks in a broad range of domains[29,33,34]; however, these approaches generally only evolve some small set of (hyper)parameters of the neural network model. For example, they evolve the hyperparamaters of a standard predefined synaptic update rule such as that of STDP[35–39] or predefined spiking models[40–44]. Similarly, reward-based learning has also been evolved but employs the same approach of evolving predefined mathematical models[45,46].

The main limitation of these previous methods is that they make assumptions on the neuronal function and behaviour. Biological neural processing, however, is extremely complex and abstract mathematical models are only capable of capturing some subset of the observed behaviour.

The most important distinction of our work in relation to previous methods is thus that each ENU within the network is not an abstract point-wise neuron with a few hyperparameters determining its behaviour (such as which activation function to use, or the hyperparameters of a spiking neuron model). Instead, each neuron is a fully RNN—with thousands of evolvable parameters—that is capable of evolving and approximating potentially any function. Likewise, our synapses are not abstract single-weight synapses. Each of our ENU synapses are also fully RNNs, with internal memory states and gating mechanisms that determine how to update those internal states. Furthermore, by having multiple channels of communication between neurons and synapses, we allow for more flexibility in learning and information processing behaviour. This could explain why biological networks might have evolved to use so many types of different neurotransmitters[47,48].

The network of ENUs proposed in this paper offers a new direction for potentially more powerful and biologically realistic neural networks, which could ultimately lead not only to a greater understanding of neural processing, but to an artificially intelligent system that can learn and act across a wide variety of domains.

Methods

Evolution strategies. As our desired goal is to solve reinforcement-learning-based environments and the task is not directly differentiable, we cannot use standard backpropagation through time to optimize our network. We would also like to be able to learn over long time-spans, which makes backpropagation through time impractical due to vanishing gradient issues. This makes evolution strategies ideally suited[49–52] and we take a similar approach to that of a previous work[30]. The gradient is approximated through random Gaussian sampling across the parameter space and the weights are updated following this approximate gradient direction (see Algorithm 1). We can also use standard stochastic gradient descent methods such as momentum[53] on the resulting approximate gradients obtained. Due to weight sharing we only have to evolve the gate parameters of two ENUs: one for the synapse the other for the neuron, giving us a relatively small parameter space.

Algorithm 1. Evolution strategies**Input:** base parameters θ_k, learning rate η, and standard deviation σ**for** $k=1$ **to** N **do** Sample offspring mutations $\epsilon_i, \ldots, \epsilon_n$ from $\mathcal{N}(0, I)$ Compute fitness $F_i = F_r(\theta_k + \sigma\epsilon_i)$ for $i=1, \ldots, n$ Calculate approximate gradient $G_k = \sum_{i=0}^{n} F_i\epsilon_i$ Update base parameters $\theta_{k+1} = \theta_k + \eta G_k$**end for**

Fitness ranking. We use fitness ranking to reduce the effect of outliers[30]. We can sort and assign rank values to each offspring according to their fitness, which determines their relative weight in calculating our approximate gradient. We then get a transformed fitness function $F_r(\theta_i) = \frac{\text{rank}(F(\theta_i))^{\alpha}}{\sum_{i=0}^{n}\text{rank}(F(\theta_i))^{\alpha}}$, where rank() assigns a linear ranking from 1.0 to 0 (high to low); α is a hyperparameter that determines the shape of this distribution. In our case $\alpha=5$, such that around the top 20% best-performing agents account for 80% of the gradient estimate.

Batching. Mini batches were also used to better estimate the fitness of the same offspring across multiple environments, so the fitness of each offspring was determined by the mean fitness across m mini-batch environments.

Gating in an ENU. The exact equations for the ENU gates and updating of the memory state are as follows:

$$\begin{aligned}
\mathbf{z}_t &= \phi(W_z \cdot [\mathbf{h}_{t-1}, \mathbf{o}_{t-1}, \mathbf{x}_t]) \\
\mathbf{r}_t &= \phi(W_r \cdot [\mathbf{h}_{t-1}, \mathbf{o}_{t-1}, \mathbf{x}_t]) \\
\widetilde{\mathbf{h}}_t &= \tanh(W_c \cdot [\mathbf{r}_t \odot \mathbf{h}_{t-1}, \mathbf{o}_{t-1}, \mathbf{x}_t]) \\
\mathbf{h}_t &= (1-\mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \widetilde{\mathbf{h}}_t \\
\mathbf{o}_t &= \text{clip}((W_o \cdot \mathbf{h}_t), 0, 1)
\end{aligned} \tag{1}$$

where $\mathbf{x}_t$ is the input, $\mathbf{z}_t$ the update gate, $\mathbf{r}_t$ the reset gate, $\widetilde{\mathbf{h}}_t$ the cell gate, $\mathbf{h}_t$ the new memory state and $\mathbf{o}_t$ the output gate; ϕ is the sigmoid function, the centre dots are for matrix multiplication and $\odot$ is element-wise multiplication; $\mathbf{W}_*$ are the weight matrices of each gate and the parameters to be evolved. We can also apply clipping (restricting output values to be between 0 and 1), as we use evolution strategies to optimize our parameters and do not require a strict differentiable activation function. This clipping results in a thresholded output and prevents exploding values.

Implementation. To compute the network of ENUs, we can reshape each neuron and synaptic compartment output to batches and perform standard matrix multiplication to compute the full network of ENUs in parallel (as our ENU parameters are shared). The connection matrix determines how neurons are connected and in this case can be either random or have a fixed sparse topology. It determines how we broadcast the output of a neuron to each synapse connected to it (broadcasting copies the output of neurons to different synapses as multiple synapses can be connected to the same neuron); however, the denser the connection matrix, the more synapses we have and thus the higher our computational cost. Please refer to Extended Data Fig. 6 for a more detailed computation diagram.

Evolution strategies can be performed efficiently on a graphical processing unit through multidimensional matrix multiplications, allowing us to evaluate thousands of mutated networks in parallel. The weight matrix in this case is three-dimensional, where the third dimension stores each offspring's mutated weights. Normal matrix multiplication is of the form $(N, K)\times(K, M)\rightarrow(N, M)$, where N is the batch size, K the number of inputs units and M the number of output units; in the three-dimensional case we get $(P, N, K)\times(P, K, M)\rightarrow(P, N, M)$, where P is the number of offspring. PyTorch[54] was used for computing and evolving our ENU-NN, whereas the rest were implemented mainly in Numpy[55], including custom vectorized experimental environments.

From a single ENU to a network of ENUs. As described in Fig. 2, the network consists of two types of ENUs: we have a neuron ENU, which shares parameters across all neurons, and a synapse ENU, which shares parameters across all synapses. These are both identical and follow the ENU description in Fig. 1 and equation (1). The only difference is in how they are connected within the network. These ENU neurons and synapses all get combined through the connection matrix, which determines how the output of each ENU neuron is sent to each ENU synapse (step 1 in the computation diagram in Extended Data Fig. 6) and which synapses belong to each neuron.

The synapse ENU takes a pre- and postsynaptic neuron ENU as its input, transforms it and outputs a new vector (step 2). It is important to note that the output value of the ENU is a vector of multiple output channels and not just a single value, we therefore get a matrix where the rows represent each of the ENUs and the columns represent each of the output channels (as described in Fig. 1). The postsynaptic neuron connection is the extra connection back to the synapse (as seen in Fig. 2). This is important for allowing STDP-type learning rules to emerge, as it requires timing between post- and presynaptic spikes. These outputs of the synapse ENUs then get summed up together (step 4) and processed further by each neuron ENU (step 5). This process gets repeated at every time step, updating the internal memory state and output of each ENU. The synapse ENU is thus responsible for processing information from the presynaptic neuron to the postsynaptic neuron, and the neuron ENU is responsible for transforming the aggregated input of its synapses and transmitting it. These distinct ENUs thus serve a similar specialized role as the synapses and neurons in biological neural networks; however, in our case their underlying function is evolved and approximated by the proposed ENU.

Experiments. First we evolve an ENU to mimic the behaviour of a simple IAF and STDP model. These serve as a demonstration of the flexibility of the ENU model and their potential to model neuronal-like information processing behaviour. We then combine multiple neural and synaptic ENUs into a single network and evolve reinforcement-type learning behaviour purely through local dynamics. This network of ENUs, however, does not assume any predefined spiking models on the neurons or synaptic learning rules, and instead has to evolve these types of rules completely independently within the network.

Evolving IAF neurons. The ENU receives random uniform noise as graded input potential at every time step (sampled uniformly between 0.01 and 0.11). It then has to mimic the IAF model receiving the same input. Once the sum of the input potentials reaches a certain threshold (0.5 in our experimental setting), we reset the internal state of the IAF and output a spike (see also Extended Data Fig. 7). The simplest IAF variant with linear summation is given below:

$$\mathrm{IAF}(u_t) = \begin{cases} 0, & \text{if } u_t < \text{th} \\ 1, & \text{if } u_t \geq \text{th} \end{cases} \quad (2)$$

where th is the threshold and u_t is the membrane potential; u_t is the sum of the input potential x_t over time (that is, $u_t = u_{t-1} + x_t$) and resets to 0 if $u_t \geq$ th.

Learning spike-like behaviour can generally be difficult as the task is not directly differentiable. Optimizing the mean squared error between the IAF model and ENU output directly is infeasible as even a small shift in spikes would cause a decremental effect on the error; however, as we are using evolution strategies, we can have more flexible loss functions and optimize the timing and intensity of each spike. We therefore optimize the interspike interval instead, which allows for incremental fitness improvements and gradually gets each spike to better match the desired timing of the IAF model. We also add an extra term that requires each spike to be as close as 1 as possible (matching the IAF spike). The fitness function can therefore be described as follows:

$$F_{\mathrm{IAF}} = -\left(\sum_i^n (y_i - \hat{\mathbf{y}}_i)^2 + \sum_i^n (t_i - \hat{\mathbf{t}}_i)^2\right) \quad (3)$$

where i is the index of each spike, y_i is the IAF model output spike as described in equation (2), and $\hat{\mathbf{y}}_i$ is the ENU estimated spike; t_i and $\hat{\mathbf{t}}_i$ are the timing of the spikes in the IAF model and ENU output, respectively (that is, the time since the last spike).

Evolving STDP. For evolving a STDP-type learning rule we require multiple input channels for the model. The basic equation for the STDP rule is as follows:

$$\omega(t) = \begin{cases} \mathrm{A}e^{\left(\frac{-t}{\tau}\right)}, & \text{if } t > 0 \\ -\mathrm{A}e^{\left(\frac{t}{\tau}\right)}, & \text{if } t < 0 \end{cases} \quad (4)$$

where A and τ are constants that determine the shape of the STDP function (in our case A was set to 0.5 and τ to 10), t is the relative timing difference between the pre- and postsynaptic spike, and ω is the resulting synaptic weight value.

First we input random uniform noise as graded potential (randomly sampled between 0.45 and 0.55). We then also generate a random input spike from the presynaptic neuron at some time t_i and a random backpropagating spike from the postsynaptic neuron at t_b. We also use a neuromodulated variant of the standard STDP rule, which is known to play a role in synaptic learning[31,56] and only allows STDP-type updates to occur if a given input neurotransmitter is present (see also Extended Data Fig. 7).

In total we thus have four input channels and a single output channel that is the transformed graded input potential multiplied by some weight w. The weight value is updated according to the standard STDP rule, which is dependent on the timing between the pre- and postsynaptic spike. The fitness is then the mean squared error between the desired STDP model output and ENU output. So the fitness function becomes:

$$F_{\mathrm{STDP}} = -\sum_t^n (y_t - \hat{\mathbf{y}}_t)^2 \quad (5)$$

where t is each time step, y_t is the STDP model output updated according to equation (4) (which depends on the input spike timings) and $\hat{\mathbf{y}}_t$ is the output of the ENU model in response to the same input signals.

Further experiments were also performed that evolve an ENU to approximate a more complex synaptic update rule (complex STDP). This was to investigate whether the ENU can approximate more arbitrary rules. Specifically to create a more complex function, we modify the standard synaptic update $\omega(t)$ by applying a cosine transformation depending on whether the dopamine signal is present or not. This provides us with a new synapic update function $\omega^*(t)$: if the dopamine signal is present then $\omega^*(t) = |\omega(t)| - 0.1(\cos(20\omega(t)) - 1)$, else $\omega^*(t) = 0.5\cos(10\omega(t))$. These parameters were chosen to purposely generate a more complex wave-like pattern, however, it is important to note that the underlying mathematical model does not necessarily have to be known to fit the ENU, we could fit on samples from any unknown underlying model. It demonstrates the flexibility of ENUs in approximating different types of functions, and that evolving a synaptic ENU within a larger overall network could thus potentially learn any type of synaptic update rule. The experimental setting was the same as in the standard STDP experiment previously described, however, the underlying function to approximate is now a more complex arbitrary function that behaves different based on the neurotransmitter signal present at the input. Please refer to Extended Data Figs. 3 and 7 for more details.

Evolving reinforcement learning in a network of ENUs. To evolve reinforcement-type learning behaviour, we design an experimentally commonly used T-maze task[57] (see Fig. 2). The T-maze is implemented as a grid-like system, where each action by the agent can move it at most one grid cell further. In our case the T-maze had a width of five steps and a height of four steps. The goal of the agent (that is, the mouse) is to explore the maze and find food; once the agent eats the food they will receive a positive reward and be reset to the initial starting location. On the other hand, if the food was actually poisonous then the agent will receive a negative reward (and also be reset). It is then up to the agent to remember where the food was and revisit the previous location. After the agent has eaten the non-poisonous food several times there is a random chance that the food and poison will be switched (50% in our experimental setting). The agent thus has to evolve to learn to detect from its sensory input whether the food is poisonous or not, and can only know this by eating the food at least once to receive the associated negative or positive reward.

To this end we provide several inputs to the agent: one that detects the wall of the maze, one that detects green and one that detects red. These inputs are passed to the first channel of the connected synapse ENU. These sensory neurons simply measure the presence of a wall (grey object), red object or green object in the grid location that the agent is facing. For example if a red object is present, the red sensory neuron will output a 1, else it will output a 0. We also have a neuron that provides the resulting reward of eating the food or poison. Positive rewards go to the second channel, whereas negative rewards go to the third channel. For the output, the agent can either go forward, left or right one step, depending on the highest activated output neuron (or do nothing if no output neuron activates).

The fitness of the agent is determined by the reward obtained in the environment; thus the better it is at remembering where the food is and at avoiding eating the poison, the higher the fitness. The fitness function then becomes:

$$F_{\mathrm{T-maze}} = \sum_t^n R_t \quad (6)$$

where R_t is the reward obtained in the environment at each t. Eating food gives a reward of +1, and eating poison gives a reward of −1. We also add an extra term that decays an agent's energy every time step to encourage exploration. Eating food refreshes the energy again and gives the agent 40 time steps to get new food. Once the agent runs out of energy it dies and will no longer be able to move or gather more rewards.

The ENU-NN consists of six ENU neurons, of which three are output neurons. Each neuron has eight ENU synapses that sparsely connect to the other neurons (two to the sensory neurons, two to the hidden neurons, two to the output neurons, one to the reward neuron and one to itself). Please also refer to Fig. 2 for a more detailed diagram. The neuron that has the highest output activity over four time steps determines the action of the agent (this allows for sufficient time of sensory input to propagate through the network). The output is taken from the first channel of the output neuron. We also add noise to the output gate of the ENU, as all ENUs share the same parameters and we want each neuron and synapse to behave slightly different on initialization.

To further investigate the flexibility of the network of ENUs in different environments, further experiments were performed in a more complex double T-Maze environment. The same experimental settings and network configuration was used in this experiment; however, the food and poison could now be randomly located in four locations instead of two. These locations were also randomly changed with 50% probability each time the agent ate food. Please refer to Extended Data Fig. 4 for more details.

It is important to note that the agents are reset each generation and they have no recollection of the past; they have to relearn which sensory neuron has what meaning and which output neuron performs what motor command. The network input and output neurons also get randomly shuffled each generation. This avoids agents potentially exploiting the topology to learn fixed behaviour. We are thus evolving an ENU-NN to perform reinforcement-type learning behaviour (evolving to learn), instead of directly learning fixed behaviour in the synaptic weights.

Experimental details. The experimental parameters were chosen through initial experimentation such that we achieved a good balance between computation time and performance. For optimization of the ENU parameters, evolution strategies were used as described in algorithm 1. We used 1,024 offspring, Gaussian mutations with standard deviation of 0.01, a learning rate of 1.0 and momentum of 0.9. For both the synaptic and neuronal compartments an ENU with a memory size of 32 units was used, with 16 output units (that is, 16 output channels).

As we have three gates of 32 units controlling the memory state, each receiving 16 input units, 16 feedback units and 32 memory state units, we get $3\times(16+16+32)\times32 = 6{,}144$ parameters for the memory state gates, and a further $32\times16 = 512$ parameters for the output gate, resulting in 6,656 parameters for the neuron ENU and synapse ENU. We thus have in total approximately 12,000 parameters to evolve in the ENU-NN. As the parameters are shared across the entire network, we do not need to optimize the parameters of each individual ENU. This means the number of evolvable parameters stay fixed regardless of network size.

The experiments were run on a single Titan V graphical processing unit with an i7 12-core central processing unit. In case of the T-maze experiment the mean fitness was taken across eight random environments for each offspring. For evolving the IAF and STDP model, the mean over 32 environments was taken (allowing

us to plot a STDP-type curve from multiple observations). One episode in the environment is one generation and each episode the ENU internal memory states are reset. In case of the IAF and STDP environment each episode lasts for 100 time steps, whereas for the T-maze experiment the episode lasted for 400 time steps.

Reporting summary. Further information on research design is available in the Nature Research Reporting Summary linked to this article.

Data availability

Data was directly generated from the code through simulations, no external datasets were used. Figures 3–5 were generated directly from the simulation.

Code availability

A Code Ocean compute capsule, which contains a pre-built compute environment and the source code, is available at https://doi.org/10.24433/CO.1361267.v1.

Received: 15 December 2019; Accepted: 28 October 2020;
Published online: 10 December 2020

References

1. Abbott, L. F. & Nelson, S. B. Synaptic plasticity: taming the beast. *Nat. Neurosci.* **3**, 1178–1183 (2000).
2. Börgers, C. & Kopell, N. Synchronization in networks of excitatory and inhibitory neurons with sparse, random connectivity. *Neural Comput.* **15**, 509–538 (2003).
3. Fell, J. & Axmacher, N. The role of phase synchronization in memory processes. *Nat. Rev. Neurosci.* **12**, 105–118 (2011).
4. Hassabis, D., Kumaran, D., Summerfield, C. & Botvinick, M. Neuroscience-inspired artificial intelligence. *Neuron* **95**, 245–258 (2017).
5. Spruston, N. Pyramidal neurons: dendritic structure and synaptic integration. *Nat. Rev. Neurosci.* **9**, 206–221 (2008).
6. Sacramento, J., Costa, R. P., Bengio, Y. & Senn, W. Dendritic cortical microcircuits approximate the backpropagation algorithm. In *Advances in Neural Information Processing Systems* 8721–8732 (Curran Associates, 2018).
7. Jain, A. K., Mao, J. & Mohiuddin, K. M. Artificial neural networks: a tutorial. *Computer* **29**, 31–44 (1996).
8. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).
9. Catterall, W. A. Structure and function of voltage-gated ion channels. *Ann. Rev. Biochem.* **64**, 493–531 (1995).
10. Flagel, S. B. et al. A selective role for dopamine in stimulus–reward learning. *Nature* **469**, 53–57 (2011).
11. McCormick, D. A. Gaba as an inhibitory neurotransmitter in human cerebral cortex. *J. Neurophysiol.* **62**, 1018–1027 (1989).
12. Levitan, I. B. & Kaczmarek, L. K. *The Neuron: Cell and Molecular Biology* (Oxford Univ. Press, 2015).
13. Nedergaard, M., Takano, T. & Hansen, A. J. Beyond the role of glutamate as a neurotransmitter. *Nat. Rev. Neurosci.* **3**, 748–755 (2002).
14. Hilfiker, S. et al. Synapsins as regulators of neurotransmitter release. *Philos. Trans. R. Soc. B* **354**, 269–279 (1999).
15. Hollenbeck, P. J. & Saxton, W. M. The axonal transport of mitochondria. *J. Cell Sci.* **118**, 5411–5419 (2005).
16. Mountcastle, V. B. The columnar organization of the neocortex. *Brain* **120**, 701–722 (1997).
17. Vale, R. D. The molecular motor toolbox for intracellular transport. *Cell* **112**, 467–480 (2003).
18. Collingridge, G. L., Isaac, J. T. & Wang, Y. T. Receptor trafficking and synaptic plasticity. *Nat. Rev. Neurosci.* **5**, 952–962 (2004).
19. Kepecs, A., Wang, X.-J. & Lisman, J. Bursting neurons signal input slope. *J. Neurosci.* **22**, 9053–9062 (2002).
20. Abbott, L. F. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain Res. Bull.* **50**, 303–304 (1999).
21. Hodgkin, A. L. & Huxley, A. F. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiol.* **117**, 500–544 (1952).
22. Hebb, D. O. *The Organization of Behavior* Vol. 65 (Wiley, 1949).
23. Maass, W. Networks of spiking neurons: the third generation of neural network models. *Neural Networks* **10**, 1659–1671 (1997).
24. Bellec, G., Salaj, D., Subramoney, A., Legenstein, R. & Maass, W. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Advances in Neural Information Processing Systems* 787–797 (Curran Associates, 2018).
25. Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Networks* **4**, 251–257 (1991).
26. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997).
27. Cho, K. et al. Learning phrase representations using RNN encoder–decoder for statistical machine translation. Preprint at https://arxiv.org/abs/1406.1078 (2014).
28. Back, T. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms* (Oxford Univ. Press, 1996).
29. Stanley, K. O., Clune, J., Lehman, J. & Miikkulainen, R. Designing neural networks through neuroevolution. *Nature Mach. Intell.* **1**, 24–35 (2019).
30. Salimans, T., Ho, J., Chen, X., Sidor, S. & Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. Preprint at https://arxiv.org/abs/1703.03864 (2017).
31. Frémaux, N. & Gerstner, W. Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules. *Front. Neural Circuits* **9**, 85 (2016).
32. Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015).
33. Hausknecht, M., Lehman, J., Miikkulainen, R. & Stone, P. A neuroevolution approach to general Atari game playing. *IEEE Trans. Comput. Intell. AI Games* **6**, 355–366 (2014).
34. Igel, C. Neuroevolution for reinforcement learning using evolution strategies. In *The 2003 Congress on Evolutionary Computation, 2003. CEC '03* Vol. 4, 2588–2595 (IEEE, 2003).
35. Bengio, S., Bengio, Y., Cloutier, J. & Gecsei, J. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks* Vol. 2 (Univ. Texas, 1992).
36. Mouret, J.-B. & Tonelli, P. Artificial evolution of plastic neural networks: a few key concepts. In *Growing Adaptive Machines* 251–261 (Springer, 2014).
37. Risi, S. & Stanley, K. O. Indirectly encoding neural plasticity as a pattern of local rules. In *International Conference on Simulation of Adaptive Behavior* 533–543 (Springer, 2010).
38. Di Paolo, E. A. Evolving spike-timing-dependent plasticity for single-trial learning in robots. *Philos. Trans. R. Soc. A* **361**, 2299–2319 (2003).
39. Carlson, K. D., Richert, M., Dutt, N. & Krichmar, J. L. Biologically plausible models of homeostasis and STDP: stability and learning in spiking neural networks. In *The 2013 International Joint Conference on Neural Networks (IJCNN)* 1–8 (IEEE, 2013).
40. Floreano, D., Epars, Y., Zufferey, J.-C. & Mattiussi, C. Evolution of spiking neural circuits in autonomous mobile robots. *Int. J. Intell. Syst.* **21**, 1005–1024 (2006).
41. Rounds, E. L. et al. An evolutionary framework for replicating neurophysiological data with spiking neural networks. In *International Conference on Parallel Problem Solving from Nature* 537–547 (Springer, 2016).
42. Carlson, K. D., Nageswaran, J. M., Dutt, N. & Krichmar, J. L. An efficient automated parameter tuning framework for spiking neural networks. *Front. Neurosci.* **8**, 10 (2014).
43. Buhry, L. et al. Automated parameter estimation of the Hodgkin–Huxley model using the differential evolution algorithm: application to neuromimetic analog integrated circuits. *Neural Comput.* **23**, 2599–2625 (2011).
44. Venkadesh, S. et al. Evolving simple models of diverse intrinsic dynamics in hippocampal neuron types. *Front. Neuroinformatics* **12**, 8 (2018).
45. Soltoggio, A., Durr, P., Mattiussi, C. & Floreano, D. Evolving neuromodulatory topologies for reinforcement learning-like problems. In *2007 IEEE Congress on Evolutionary Computation* 2471–2478 (IEEE, 2007).
46. Blynel, J. & Floreano, D. Exploring the T-maze: evolving learning-like robot behaviors using CTRNNs. In *Workshops on Applications of Evolutionary Computation* 593–604 (Springer, 2003).
47. Doya, K. Metalearning and neuromodulation. *Neural Networks* **15**, 495–506 (2002).
48. Soltoggio, A., Bullinaria, J. A., Mattiussi, C., Dürr, P. & Floreano, D. Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In *Proc. 11th International Conference on Artificial Life (Alife XI)* 569–576 (MIT Press, 2008).
49. Back, T., Hoffmeister, F. & Schwefel, H.-P. A survey of evolution strategies. In *Proc. 4th International Conference on Genetic Algorithms* Vol. 2 (Morgan Kaufmann,1991).
50. Beyer, H.-G. & Schwefel, H.-P. Evolution strategies—a comprehensive introduction. *Natural Comput.* **1**, 3–52 (2002).
51. Wierstra, D. et al. Natural evolution strategies. *J. Mach. Learning Res.* **15**, 949–980 (2014).
52. Lehman, J., Chen, J., Clune, J. & Stanley, K. O. ES is more than just a traditional finite-difference approximator. In *Proc. Genetic and Evolutionary Computation Conference* 450–457 (ACM, 2018).
53. Sutskever, I., Martens, J., Dahl, G. & Hinton, G. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning* 1139–1147 (JMLR, 2013).
54. Paszke, A. et al. *Automatic Differentiation in PyTorch* (Open Review, 2017).
55. Oliphant, T. E. *A Guide to NumPy* Vol. 1 (Trelgol, 2006).
56. Pawlak, V., Wickens, J. R., Kirkwood, A. & Kerr, J. N. Timing is not everything: neuromodulation opens the STDP gate. *Front. Synaptic Neurosci.* **2**, 146 (2010).

57. Deacon, R. M. & Rawlins, J. N. P. T-maze alternation in the rodent. *Nat. Protocols* **1**, 7–12 (2006).

Acknowledgements
We thank J. Kalafotovich, H. Bin Ko and R. Hormazabal for their review of the manuscript and related comments and discussions. This work was supported by the Institute for Information and Communications Technology Planning and Evaluation grant funded by the Korean government (MSIT) (no. 2017-0-01779, a machine learning and statistical inference framework for explainable artificial intelligence; no. 2019-0-01371, development of brain-inspired AI with human-like intelligence; and no. 2019-0-00079, Department of Artificial Intelligence, Korea University).

Author contributions
P.B. conceived the network of ENUs and implemented related experiments. S.-W.L. discussed the results and supervised the research. P.B. and S.-W.L. wrote the paper.

Competing interests
The authors declare no competing interests.

Additional information
Extended data is available for this paper at https://doi.org/10.1038/s42256-020-00267-x.

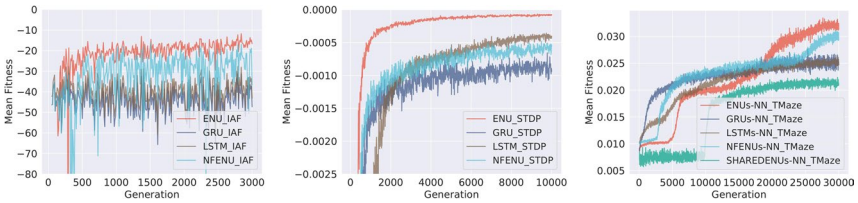
Supplementary information is available for this paper at https://doi.org/10.1038/s42256-020-00267-x.

Correspondence and requests for materials should be addressed to S.-W.L.

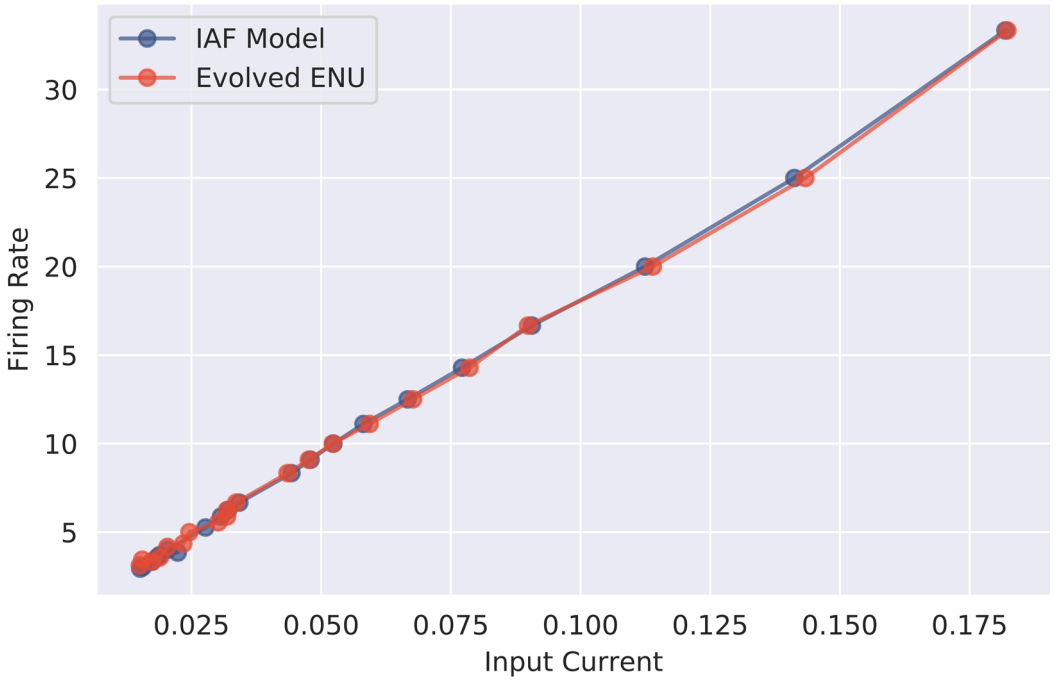
Peer review information Nature Machine Intelligence thanks the anonymous reviewers for their contribution to the peer review of this work.

Reprints and permissions information is available at www.nature.com/reprints.

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

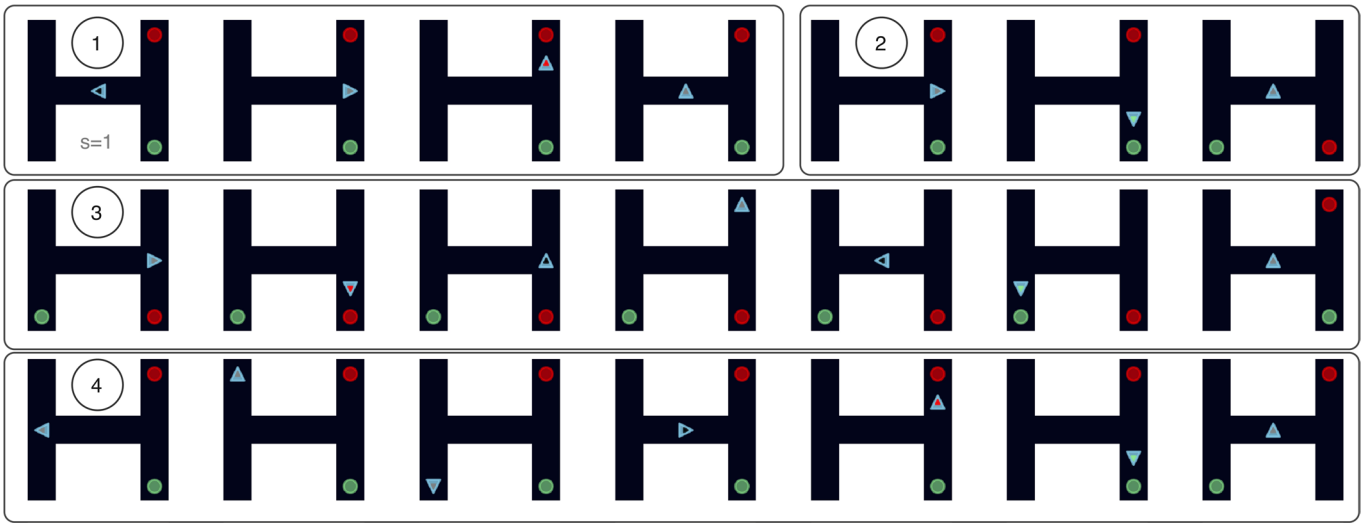


Extended Data Fig. 1 | Comparison and ablation study training progress. Comparing standard GRUs, LSTMs and our proposed Evolvable Neural Unit (ENU). Generally the ENU consistently outperforms the other models in terms of final performance. Additionally to investigate the effect of the feedback connection from the output gate, we removed such connection in the No Feedback ENU (NFENU), showing the importance of this connection. In case of the Network of ENUs, we also ran additional experiments that shared the parameters between the neuron and synapse model (the SHAREDENUs-NN). It shows that having separate ENUs for both the synapses and neurons significantly improve performance, and that without such specialization the network fails to converge.

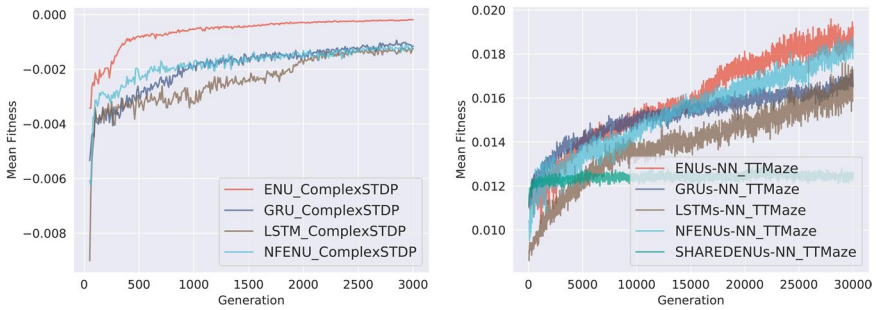


Extended Data Fig. 2 | Mean input current vs Firing Frequency after evolving Integrate and Fire neurons. shows that the evolved IAF model firing frequency pattern in response to the input current closely matches the underlying model it evolved to approximate.

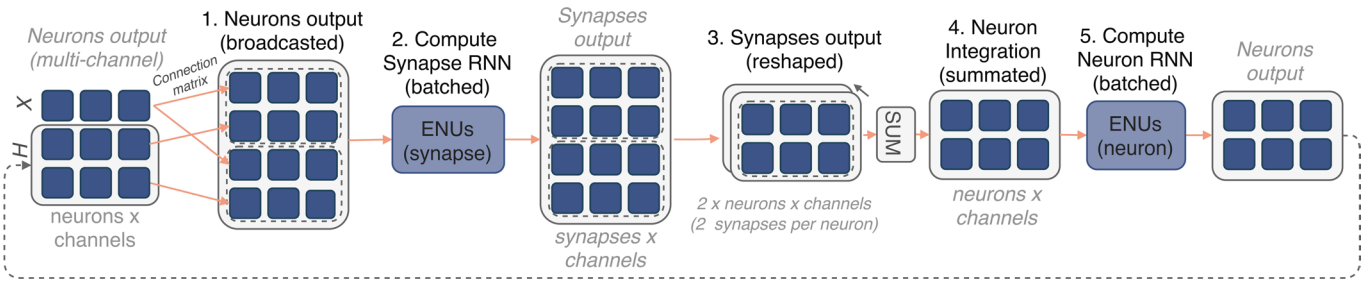
Extended Data Fig. 3 | Complex synaptic update rule example. Comparing evolving an ENU for 1000 generations (left) vs 3000 generations (right). The ENU learned to approximate a complex neuromodulated STDP type learning rule. When the neurotransmitter is present at the input (NT) the rule follows a symmetric type STDP rule. However, when the NT signal is absent it follows completely different dynamics. It is maximum at a spike timing difference of around 0 and 10ms, while around 5ms the synaptic change is essentially disabled. This shows we do not require the manual derivation of a possibly complex exact mathematical function that explains the synaptic behaviour. Instead, ENUs can potentially evolve any type of complex arbitrary neuromodulated synaptic update learning rules when evolved within a larger complex network.



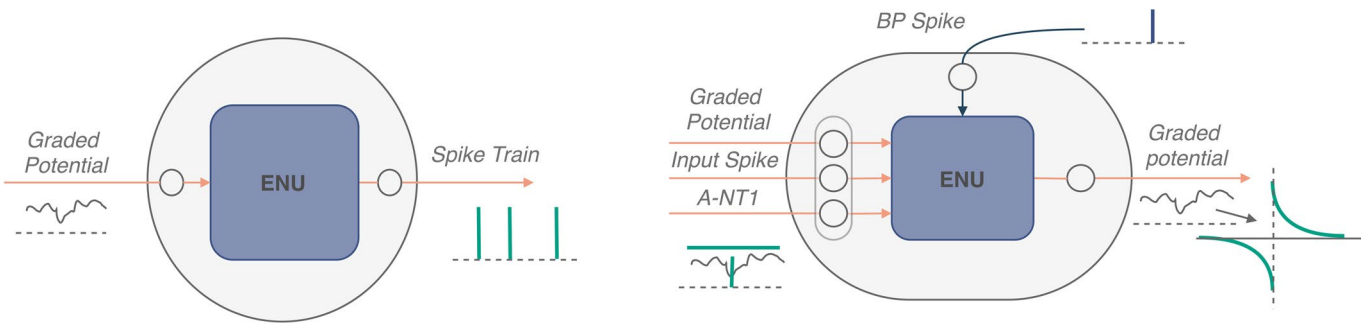
Extended Data Fig. 4 | Double T-Maze evolved learning behaviour. Double T-Maze evolved learning behaviour. Example of steps taking in the double T-maze environment by an evolved Network of ENUs. The agent can be seen to have successfully evolved to explore the environment to find and eat the initial poison (1). It then explores an alternative path to find non-poisonous food instead (2), indicating it has properly learned from a single example to associate the previous actions taken with a negative reward. Since food and poison can randomly change location, the agent goes back to the previous food location, but detects poison instead. As it previous obtained a negative reward with the action of eating the poison, it internally modified the synapse ENUs internal memory states to alter its behaviour, and successfully learned to turn around and find food in another part of the maze (3). It also evolved proper exploration behaviour if no food or poison is found in a section of the maze, successfully navigating to the other side (4).



Extended Data Fig. 5 | Convergence analysis Ecomplex STDP and Double T-Maze experiment. The ENU can be seen to generally converge faster in both experimental settings. In case of evolving a complex synaptic update rule (Complex STDP), the ENU significantly outperforms the other models. When the feedback connection is removed (NFENU), the performance also drops. This indicates the importance of the feedback connection, which was also observed in the previous standard STDP experiment in Fig. 4. In case of the Double T-Maze experiment, the ENU also converges faster with this feedback connection. The LSTM generally takes longer to converge compared to the GRU model, which could be explained by the fact that LSTMs are slightly more complex than GRUs. When the parameters are shared between the synapse and neuron ENUs, the network fails to converge (SHAREDENU). This was also observed in the standard T-Maze experiment, and further indicates the need for the specialization of the synaptic and neuronal behaviour.



Extended Data Fig. 6 | Computation flow diagram of a Network of ENUs. Shows a computation example with 4 ENU synapses and 2 ENU neurons, each having 3 channels. The sensory input neurons X are concatenated with all the ENU neurons H to get our input batch. A connection matrix is then applied that broadcasts (copies) the neurons' output to each connected synapse ENU (1). On this resulting matrix we can then apply standard matrix multiplication and compute our synapse ENUs output in parallel (2). We can reshape this and sum along the first axis, as we have the same number of synapses for each neuron (3). This gives us the integrated synaptic input to each neuron ENU (4). Finally, we apply the neuron ENUs on this summated batch and obtain the output for each neuron in the ENU network (5). Each ENU has multiple outputs, so we have multiple channels that are processed by the ENU (the columns of each matrix), and we also have multiple neuron and synapse ENUs computed in parallel (the rows of each matrix).



Extended Data Fig. 7 | IAF and STDP experimental setup. For evolving the IAF ENU a single random graded potential is given as input (left). The goal of the ENU is then to approximate the underlying IAF rule. In case of evolving the STDP rule (right) multiple input channels are used: the graded input potential, the input spike, the neuromodulation signal (A-NT1) and the backpropagating spike. The target is then to output the modified graded input potential matching the STDP rule.

	Complex STDP	Double T-Maze
ENU	**-19.55 +- 1.11**	**18.79 +- 0.24**
GRU	-108.31 +- 7.26	16.91 +- 0.15
LSTM	-125.63 +- 4.42	16.42 +- 0.39
NFENU	-123.33 +- 6.74	18.39 +- 0.16
SHAREDENU	N/A	12.35 +- 0.06

Extended Data Table 1 | Final performance of Complex STDP approximation and Double T-Maze experiment. Shows the mean and standard deviation of the performance in each environment over 30 trial runs. The ENU consistently outperforms the compared models ($p<0.005$). On the Double T-Maze Experiment, the LSTM and GRU model perform similarly ($p=0.012$), which was also observed in previous experiments. See also Fig. 5 for a more detailed convergence analysis.

nature research

Corresponding author(s): Paul Bertens
Seong-Whan Lee

Last updated by author(s): Jan 9, 2020

Reporting Summary

Nature Research wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Research policies, see Authors & Referees and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed
☒	☐ The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☒	☐ A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☒	☐ The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐ A description of all covariates tested
☒	☐ A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☒	☐ A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☒	☐ For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐ For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐ For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐ Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	Custom code was written in Python
Data analysis	Custom code was written in Python

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors/reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Research guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A list of figures that have associated raw data
- A description of any restrictions on data availability

Data was directly generated from the code through simulations, no external data sets were used. Figures 4-7 were generated directly from the simulation.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	*Describe how sample size was determined, detailing any statistical methods used to predetermine sample size OR if no sample-size calculation was performed, describe how sample sizes were chosen and provide a rationale for why these sample sizes are sufficient.*
Data exclusions	*Describe any data exclusions. If no data were excluded from the analyses, state so OR if data were excluded, describe the exclusions and the rationale behind them, indicating whether exclusion criteria were pre-established.*
Replication	*Describe the measures taken to verify the reproducibility of the experimental findings. If all attempts at replication were successful, confirm this OR if there are any findings that were not replicated or cannot be reproduced, note this and describe why.*
Randomization	*Describe how samples/organisms/participants were allocated into experimental groups. If allocation was not random, describe how covariates were controlled OR if this is not relevant to your study, explain why.*
Blinding	*Describe whether the investigators were blinded to group allocation during data collection and/or analysis. If blinding was not possible, describe why OR explain why blinding was not relevant to your study.*

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology
☒	☐ Animals and other organisms
☒	☐ Human research participants
☒	☐ Clinical data

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging