

The Evolution of Training Parameters for Spiking Neural Networks with Hebbian Learning

Katarzyna Kozdon¹, Peter Bentley^{1,2}

¹University College London, Gower Street, London WC1E 6BT United Kingdom

²Braintree Ltd, 7 Gower Street, London WC1E 6DP, United Kingdom

k.kozdon@cs.ucl.ac.uk

Abstract

Spiking neural networks, thanks to their sensitivity to the timing of the inputs, are a promising tool for unsupervised processing of spatio-temporal data. However, they do not perform as well as the traditional machine learning approaches and their real-world applications are still limited. Various supervised and reinforcement learning methods for optimising spiking neural networks have been proposed, but more recently the evolutionary approach regained attention as a tool for training neural networks.

Here, we describe a simple evolutionary approach for optimising spiking neural networks. This is the first published use of evolutionary algorithm to develop hyperparameters for fully unsupervised spike-timing-dependent learning for pattern clustering using spiking neural networks. Our results show that combining evolution and unsupervised learning leads to faster convergence on the optimal solutions, better stability of fit solutions and higher fitness of the whole population than using each approach separately.

Introduction

Spiking neural networks (SNNs) and evolutionary algorithms (EAs) are two classical methods which are undergoing a revival. Due to their intrinsic properties discussed below, SNNs are a promising tool for processing time-series data. However, the same properties also make them harder to train to successfully perform complex tasks which non-spiking neural networks (NNs) can be trained to perform (Mnih et al., 2015). Most attempts to improve the performance of SNNs (Bohte, Kook and PoutrHan, 2000; Ponulak and Kasiński, 2010) at least partly abandon the unsupervised learning paradigm which was inspired by the brain (Hebb, 1950).

In this study, we use an EA to optimise learning hyperparameters of unsupervised SNNs. While using EAs to develop NNs is not a new idea (Belew, McInerney and Schraudolph, 1992), they are classically used either as a training method for developing networks' weights or to develop architecture of the network. Our aim is to provide a proof of concept that EAs can be used to develop learning hyperparameters of SNNs, and to develop a tool which will aid further exploration of the full potential of SNNs with brain-inspired learning. Furthermore, we aim to create a system in which both evolution and learning contribute to

achieving the full potential of the model, thus mimicking development of visual pattern recognition reported in, amongst others, cats (Hubel and Wiesel, 1970), primates (Wilson and Riesen, 1966) and humans (Kalia *et al.*, 2014), where the evolved nervous system is optimised in response to post-natal visual experience.

After a brief overview of SNNs and EAs for NNs, we provide an overview of our model and details of the experiments, followed by results and conclusions.

Background

Biological Neurons and Spiking Neural Networks

SNNs are the third and most recent generation of NNs and were inspired by the firing paradigm of the biological neurons. In SNNs, each neuron accumulates inputs and fires only upon reaching its firing threshold; the change in the internal state of the firing neuron is rapid and approximately identical every time, and has the appearance of a spike (Hodgkin and Huxley, 1952). Thus, the state of each neuron is analogue but passing information between neurons is binary. This accumulation of the signals within the neurons offers a novel way of incorporating the temporal relations within the dataset into the network's activity. It has been proposed that this transient collective synchronisation makes SNNs particularly suitable for unsupervised processing of spatio-temporal patterns (Hopfield and Brody, 2001).

In the brain, there are two main types of neurons: excitatory and inhibitory. Excitatory neurons increase the internal state of their targets, whereas inhibitory neurons decrease it; the former makes their target neurons more likely to reach their firing threshold, whereas the latter decreases the chance of spiking.

Unsupervised Learning

SNNs frequently use biologically-inspired unsupervised learning algorithms. One of the standard approaches is Hebbian learning, especially the spike-timing-dependent plasticity (STDP) rule. A change of weight between pairs of neurons is a function of the difference of their firing times. If the presynaptic neuron fires before its target neuron thus contributing towards its activation, the weight increases –

this is known as long term potentiation (LTP)(Bi and Poo, 1988; Markram et al., 1997). Conversely, if the firing order is reversed and the presynaptic neuron is silent prior to its target activation, the weight decreases leading to long term depression (LTD). In addition to those two standard rules, in this study we use a third, more recently observed in the brain rule: if the presynaptic neuron is inhibitory, the weight always increases if the pair of neurons fire within a certain time-window, irrespectively of the order of firing (inhibitory LTP, inhLTP) (Bi and Poo, 1988; D’amour and Froemke, 2015).

Evolutionary Algorithms for Neural Networks

Biological evolution relies on the assumptions that there exists variability in some hereditary features, and some variants of these features promote organism’s reproduction or survival (“survival of the fittest”) thus becoming more dominant in the population. Inspired by nature, standard EAs work by creating a population of possible solutions, and then alternately assessing their fitness and further exploring the most promising solution spaces by replacing a proportion of the population with near-identical clones of the fittest solutions. Unlike supervised learning methods, EAs do not require sets of correct input-output pairs. EAs more closely resemble reinforcement learning (RL) methods, and are rapidly gaining popularity as a scalable alternative to RL in deep neural network (DNN) training (Salimans et al., 2017). The solutions developed with EAs were more robust to parameter perturbations due to the fact that EAs find a parameter space containing populations of close to optimal solutions rather than a single set of optimal parameters (Lehman et al., 2017). EAs with exploratory behaviour were shown to reduce the number of solutions stuck in the local minima (Conti et al., 2017). When it comes to SNNs, an EA directly developing synaptic weights was demonstrated to reduce complexity of the solutions in memory-dependent tasks in a simulated 2D world (Saggie, Keinan and Ruppin, 2004).

In addition to using EAs as learning algorithms which train the synaptic weights, it is also possible to use EAs to develop the whole architecture of a network. NeuroEvolution of Augmenting Topologies (NEAT) (Stanley, Bryant and Miikkulainen, 2003) is one of the most popular genetic algorithms (GA) for architecture development and relies on directly encoding every parameter of the network. It starts with a basic network and gradually adds complexity to it, thus avoiding creating overly complex networks when increase in the complexity does not improve fitness. It has been successfully used in SNNs, adaptive NNs (Stanley, Bryant and Miikkulainen, 2003), DNN and neural Turing Machines (Greve, Jacobsen and Risi, 2016).

In addition to using EAs to train or develop networks’ architecture, EAs have also been used to develop networks’ learning methods. In this approach, EA samples the global parameter space and the neural network’s learning algorithm optimises the proposed solutions through the local search (Belew, McInerney and Schraudolph, 1992). EAs have been used to develop the hyperparameters of backpropagation (Rumelhart, Hinton and Williams, 1986) and conjugate gradient learning algorithms in traditional

NNs (Belew, McInerney and Schraudolph, 1992). They also have been used to develop learning rules in supervised NNs (Chalmers, 1990). In SNNs, EAs were used to adjust a STDP parameter for each synapse (Floreano and Mondada, 1996) as well as develop parameters of a pre-determined Hebbian learning rule (Baxter, 1992). However, in the latter case, training had a fully supervised element in the form of setting the input and output values to the desired values in order to direct Hebbian-like learning in the hidden layer. In both cases, weights were initialised with new, random values for each population and were not inherited.

Overview of the Model

Kozdon and Bentley (Kozdon and Bentley, 2017) described an ensemble of unsupervised SNNs for parallel processing of spatio-temporal data. During that study, the networks’ learning parameters had to be manually adjusted whenever the type of data changed. The need for adjusting the parameters was the consequence of using STDP: during training, data which strongly activates the network leads to strengthening of the weights until all neurons fire at all times, whereas data which weakly activates the network decreases the weights until the network becomes silent. To overcome this bottleneck and to explore whether evolution might provide a useful alternative, in this work we test automatically developing the STDP hyperparameters using an evolutionary approach.

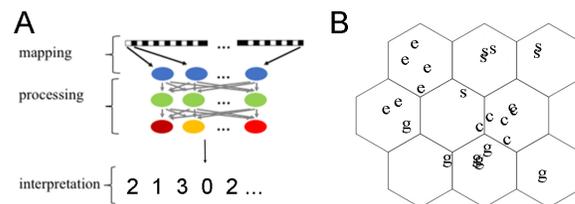


Figure 1 A) We used a three-layer feed-forward network. Vectorised binary bitmap was mapped onto the input layer, one pixel per one neuron. Spiking activity of the output layer was processed to have a form of a vector of the sum of spikes at a given time point. B) Activity vectors were clustered using Self Organizing Maps. Here vectors of spiking activity in response to bitmaps depicting moving cross, ellipse, grid and square were clustered. Precision was calculated as the proportion of cases which belonged to the dominant population in the node, weighted by the number of cases in the whole map. The total score was the sum of scores of all nodes.

Table 1: Predetermined range of the STDP parameters

Parameter	Min value	Max value
discharge	0.006	1.0
LTP	1.0	2.0
inhLTP	1.0	2.0
LTD	0.0	1.0

Spiking Neural Network

We use a previously described setup (Kozdon and Bentley, 2017) utilising fully-connected three-layer feed-forward SNNs consisting of integrate-and-fire neurons (Abbott, 1999) (Fig. 1a), 15% of which were inhibitory. Weights are initialised with random values between 0 and 2, and clamped between 0 and 4. An STDP unsupervised algorithm is used to adjust the weights, and the rules are defined separately for excitatory and inhibitory neurons, in accordance with the published biological observations (Bi and Poo, 1988; D'amour and Froemke, 2015). If a neuron and its target fire during the same iteration, the weight is multiplied by the parameter LTP. Conversely, if the target neuron fires in the absence of the presynaptic neuron firing, the synaptic weight between them is multiplied by the parameter LTD, which is lower than 1.

However, for pairs of neurons consisting of an inhibitory presynaptic neuron and excitatory postsynaptic neuron, if the target neuron fires during the same iteration, an iteration before or after the inhibitory neuron, the weight between them is multiplied by the parameter inhLTP.

The spiking activity of the output layer is represented as a vector of a total number of spikes at a given time point. Vectors encoding spiking in response to different pattern classes are clustered using Kohonen maps (Self Organising Maps, SOM) (Wehrens and Buydens, 2007) and clustering precision is calculated (Fig. 1b).

Evolution of Learning Parameters

Throughout this study, we focus on the four main parameters which affect STDP, namely LTP, LTD, inhLTP and discharge size. As described previously (Kozdon and Bentley, 2017) LTP, LTD and inhLTP parameters of our model define the rate of change of the weights in the STDP paradigm. Discharge is the size of a single output from a single neuron, and its size multiplied by the weight correlates with the likelihood of the target neurons to fire and, consequently, with the likelihood of the connections being strengthened. As these “genes” are epistatic – the phenotype they give depends on the activity of the other genes – in order to avoid separating sets of values which perform well together, we decided not to perform cross-over and instead we alter the genotype using only mutation. However, we are not excluding the possibility that this model could be adjusted to further benefit from the addition of cross-over.

At the beginning of the evolutionary process, we generate a population of SNNs with randomly initialised weights. Each network has one chromosome with four genes representing the LTD, LTP, inhLTP and discharge parameters; their initial values are random floats contained in the pre-defined for each parameter range. Table 1 lists the ranges of values the parameters are restricted to. For the discharge parameter, the values were restricted to the previously tested range of values which permitted spiking but did not lead to instant oversaturation of the signal. For LTP and inhLTP, we selected range of values which allows doubling of the synaptic strength, and for LTD values which allow the synaptic strength to decrease up to the point of the synapse becoming silent.

During training phase, weights are plastic. Sets of spatio-temporal patterns are mapped onto the network’s input layer and weights are adjusted according to the STDP rules determined by the hyperparameters. The spatio-temporal patterns used are inspired by the in vivo experiments on pattern recognition in rats (Thomas *et al.*, 2004), cats (Hubel and Wiesel, 1970) and primates (Wilson and Riesen, 1966), during which the animals are placed in front of a screen and made to watch simple geometric patterns such as stripes, circles and squares move on the screen while the animals’ brain activity is being recorded.

During testing, fitness is established using a set of sixteen patterns containing all possible combinations of shape and direction, then by creating a Kohonen map with spiking activities of the SNN in response to each pattern, and calculating clustering precision of each network.

From the second generation onwards, next generation is created based on the fitness of the SNNs from the previous generation. Top third of the networks is used to populate next generation in one of three ways:

1. The parent is retained. Both the learning hyperparameters and weights remain the same (the latter requires the child not to undergo training between its creation and fitness assessment).
2. The hyperparameters remain the same but the child undergoes a round of training thus potentially changing its weights.
3. One of the four hyperparameters is mutated and the child undergoes training in order for the change of the learning parameters to influence the weights.

The size of the mutation is determined as

$$m = x/\text{generation} \quad (1)$$

where in $x = 0.05$ in order to balance the size of the change in all generations with the range of the values each parameter can take (range for all is ≈ 1 , based on earlier experiments, the smallest meaningful change is ≈ 0.001). The direction of the change is selected randomly. New values are not bound by the initial parameter value constraints listed in Table 1.

If any of the organisms in the best third of the population has fitness lower than 50 (lower than an average randomly-generated organism), it is removed from the genetic pool and the slots which would belong to its three children are initialised with different random values instead.

The division into the three above categories was chosen in order to 1) preserve good solutions and test them on a new data set to see if the organisms could generalise 2) retain fit hyperparameters while taking advantage of the possibility of training improving the fitness even further 3) explore the hyperparameter space in the neighbourhood of fit solutions. As the hyperparameters affect the SNNs only during training, training has to take place here, and it is not possible to test different hyperparameters but the same weights.

Experimental Details

Experiment 1: Evolving Learning Parameters for Classification of Spatio-Temporal Patterns

The objective of the first experiment was to determine whether the collaboration of an EA and STDP leads to the development of an algorithm capable of clustering shapes by the direction they move. The cycle consisted of creating a generation, training organisms and establishing their fitness, and was repeated twenty times.

A population of thirty fully connected forward SNNs was initialised with random weights and random values of the learning hyperparameters. Each network had 500 input neurons, 500 hidden neurons and 10 output neurons. In the first generation, all networks were trained using 20 x 25 pixel binary bitmaps (Fig. 2) which were mapped onto the 500 input neurons (pixel one onto neuron one etc.) in such a way that “black” pixels always made the neurons fire, and “white” pixels did not cause any excitation. The bitmaps contained one of four geometric patterns – cross, ellipse, grid and rectangle. Each shape consisted of 40 black pixels and was placed at a random location within the bitmap. Then the shape would continuously move left, right, top or bottom with the speed of 1 pixel per iteration. One training cycle consisted of 20 patterns, each being shown for five frames. Fitness was defined as the ability to cluster the inputs by the movement direction.

Experiment 2: Evolving Small Populations

Based on the results of experiment 1, we decided to decrease the population size in order to reduce the probability of optimal parameters appearing by chance in the first generation. This was done in order to test if optimal parameters can be developed if they were not present in the population, and not just propagate in the population after being randomly generated in the first generation.

The experiment was carried as previously, with the exception of population size being decreased from 30 organisms to 12 per generation.

Experiment 3: Evolving Fitness in the Absence of Training

To test if consistent selection of fit child solutions alone was sufficient to develop increasingly fit solutions, we used the setup of experiment 1 and turned training off. In this set up, weights of the networks did not change. As the hyperparameters affect the networks’ performance through affecting STDP during training, the hyperparameters were not affecting fitness either, and any increase in fitness was solely due to the EA filling the population with organisms which were good at generalising and consistently performed well in tests, and random initialisation of very fit organisms when organisms with fitness below 50% were removed from the parent pool.

Experiment 4: Evolution in the Absence of Architecture Inheritance

The objective of this experiment was to determine the role of the inherited weights developed by STDP vs the learning

Table 2 Fitness during the evolutionary and learning process (experiment 1)

Generation	Best [%]	Top 3 [%]	Worst [%]
1	97.5, SD = 5.6	92.9, SD = 3.5	27.5, SD = 25.6
10	100, SD = 0	97.9, SD = 3.6	48.8, SD = 9.3
20	100, SD = 0	99.2, SD = 1.8	47.5, SD = 7.1

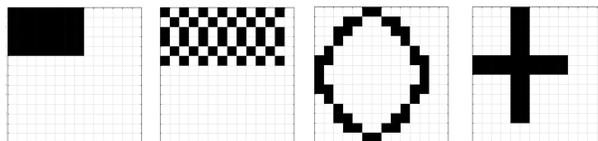


Figure 2 Examples of bitmaps used as input data. A) First data set consisted of square, grid, ellipse and cross. Each shape was composed of 40 black pixels, it was placed at a random location within the visual field (20 x 25 pixels, one for each input neuron) and moved up, down, left or right with the speed of 1 pixel per frame. Experiments 1-5 defined fitness as the ability to cluster the inputs by movement direction, experiment 6 used clustering by shape.

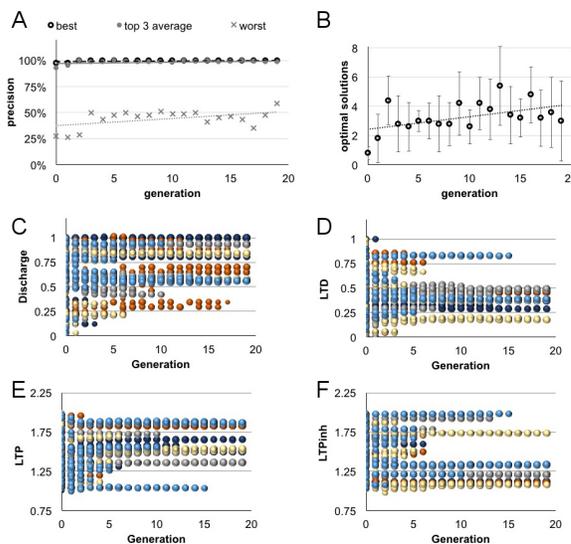


Figure 3 Experiment 1 Performance of SNNs during optimisation including evolution and unsupervised learning A) Precision of best, three best and worst network. B) number of SNNs in the population which were 100% precise C-F) Convergence of parameter values during the evolutionary process. Each colour indicates one of the five repeats; size indicates fitness.

hyperparameters evolved by the EA in a population shaped by both evolution and training. In all previous experiments, children inherited weights of their parents (even though sometimes those weights were then modified by training). In this experiment, weights were not inherited by children. Thus, children would only inherit the capacity to learn but not the experience of their parents.

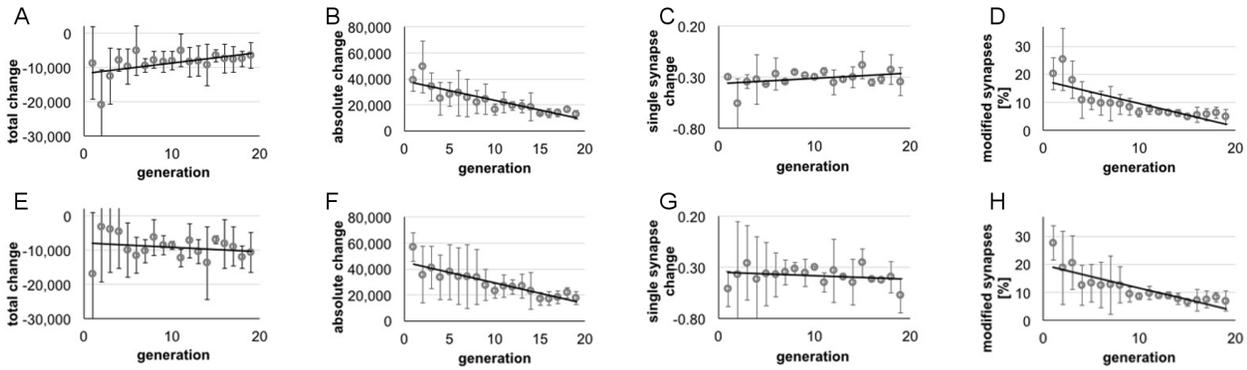


Figure 4 Experiment 1 Weight differences between the parents vs children with a mutated parameter and an additional round of training (A-D) and the parents vs children with unmodified parameters but with an additional round of training (E-H). N = 5, ten parent-child pairs per category per time point, 255000 synapses per network.

Experiment 5: Random Parent Selection

In this experiment organisms were not sorted according to fitness and parents were selected at random to analyse the role of survival of the fittest in achieving optimal solutions.

Experiment 6: Clustering Input by Shape

In order to test if the EA can successfully optimise the learning hyperparameters for other tasks, we changed our definition of fitness from the ability to cluster the inputs by the movement direction to the ability to cluster by the shape used. In this experiment, the mutation parameter x was increased from 0.05 to 0.1 and the number of mutated genes increased from 1 to 2 in order to compensate for the fact that the initial solutions were further from the optimal values.

Results and Discussion

Fitness During Evolution and Learning

Results of experiment 1 showed that with a population of 30 organisms, networks with 100% clustering precision appeared in the first generation, amongst the organisms with randomly generated initial hyperparameters, and after one round of training were present in 4 out of 5 experiments (Fig. 3a, Table 2; due to overlap between the best and top 3 average categories, SD is not shown in Fig. 3). Best organism achieved stable 100% precision (SD = 0 %) after 3 generations. The overall fitness of the whole population steadily increased during the evolution and training process, and the total number of 100% fit solutions in the population increased from 0.8 (SD = 0.45) to 3.0 (SD = 2.7) (Fig. 3b). Evolution initially explored the whole permitted range of parameter space, and the convergence towards a narrower range of parameters was seen during evolution (Fig. 3c –f). Weight differences between the parents and children with a mutated gene (Fig. 4 A-D) and average change per synapse increased with time, but the absolute value of the changes and number of changed synapses decreased. This indicates that later mutations caused larger, bi-directional changes focused on a smaller number of synapses – the latter partly due to an increasing number of synapses being silenced.

Weight differences between the parents and children with an additional round of training and the number of synapses changed decreased with time indicating that the effect of additional training was decreasing with time.

Emergent Fitness in Smaller Populations

As in the experiment 1 the population was big enough to lead to a random creation of optimal and near optimal solutions in the first generation. We then wanted to more

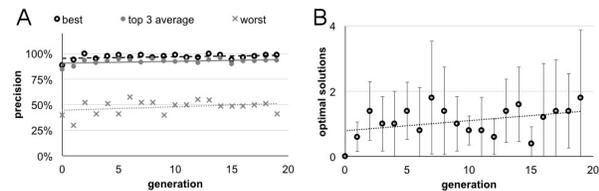


Figure 5 Experiment 2 Fitness in a small population with suboptimal initial parameters. A) Fitness of the population. B) Number of solutions with 100% precision.

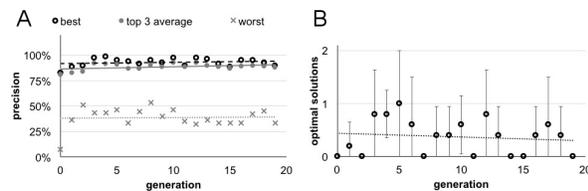


Figure 6 Experiment 3 Fitness in the absence of training. A) Fitness of the population. B) Number of solutions with 100% precision.

Table 3 Fitness in the absence of training (experiment 3)

Generation	Best [%]	Top 3 [%]	Worst [%]
1	82.5, SD = 2.8	80.8, SD = 4.3	7.5, SD = 16.7
10	97.5, SD = 3.4	92.9, SD = 1.9	46.2, SD = 25.9
20	90.0, SD = 5.6	87.9, SD = 4.6	33.7, SD = 31.1

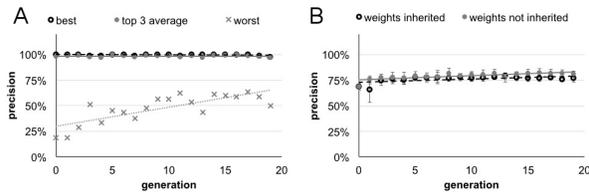


Figure 7 Experiment 4 Fitness in networks which inherit hyperparameters but not weights. A) Fitness of the population. B) Average population fitness in evolution with and without training.

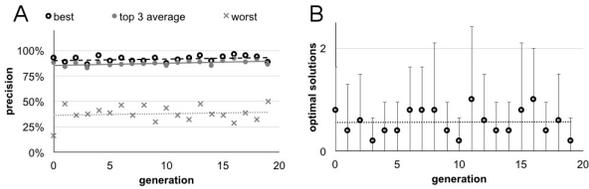


Figure 8 Experiment 5 Fitness without the survival of the fittest. A) Fitness of the population. B) Number of solutions with 100% precision.

closely look at the emergence of optimal solutions when no near optimal solutions were present in the initial population. In order to test this, we decreased the size of the population and included only the instances where the precision of the fittest organism in the first generation was below 95%. In generation 1, the fittest networks had precision of 88.8 % (SD = 5.3) (Fig. 5A). In all experiments, a solution with precision of 100% was found in generation 2 (SD = 0). The number of optimal solutions in the population had a positive trend (Fig. 5B), but it was lower than in experiment 1.

Fitness Without Training

To further examine the source of the increase in fitness of the population, we kept selecting the fittest organisms in each generation but did not train them. Fitness in the first generation was lower in comparison to results achieved with training (82.46%, SD = 2.8% vs 92.5%, SD = 5.6%, which indicates that training alone improved fitness of networks with randomly generated learning hyperparameters. Selecting the fittest organisms led to an increase in fitness in all categories but this increase was not stable and SD did not decrease (Table 3). Overall, the best results were introduced to the population in the first generation and having children alone did increase the overall fitness of the population. Training and mutation are needed to create optimal solutions and systematically reduce SD.

Inheritance of Learning Skills

In all previous experiments, both weights (experience) and the STDP parameters (learning skills) were inherited by the children. In this experiment, children inherited only the hyperparameters. Performance of best and top three networks was not significantly different from the baseline (experiment 1) (Fig. 7B). These results indicate that the

evolved hyperparameters led to successful learning irrespectively of the network's weights.

Fitness Without the Survival of the Fittest.

In this experiment parents were chosen at random, irrespectively of their fitness. Precision of the best, three best and worst networks was significantly lower than in the baseline experiment 1 (Fig. 8A). The slight observed increase in fitness with time may be due to the higher overall number of training cycles in the later generations. The number of optimal solutions in the population remained constant and was lower than the baseline (Fig. 8B).

Shape Detection

In contrast to experiments 1-5, here we defined precision as the ability to cluster the spatio-temporal patterns based on the shape and not movement direction. Precision in generation 1 was lower than when clustering by movement direction and did not reach 100% average during 20 generations (Fig. 9A). However, a positive trend was observed for the precision of the whole population (Table 4) and precision reached in generation 20 was 20% higher than reached by the same SNN model with manually adjusted parameters (Kozdon and Bentley, 2017).

When comparing values of the hyperparameters between networks optimised for direction detection (experiment 1) vs shape detection, we can see that their initial average values were about 0.5 for discharge and LTD and 1.5 for LTP and inhLTP (i.e. the average of random values was in the middle of the permitted range). During evolution, the average hyperparameter values moved away from these values and towards the preferred parameter space. Networks

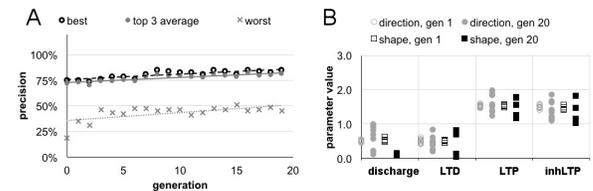


Figure 9 Clustering spatio-temporal inputs by shape. A) Fitness of the population B) Values of the hyperparameters, optimizing for recognizing direction (grey) vs recognizing shape (black). Empty markers indicate random average values in generation 1, filled markers evolved values in generation 20.

Table 4 Fitness of networks optimised for shape detection

Generation	Best [%]	Top 3 [%]	Worst [%]
1	75.0, SD = 6.2	72.1, SD = 4.8	18.7, SD = 17.0
10	82.5, SD = 2.8	80.0, SD = 2.8	46.3, SD = 3.4
20	85.0, SD = 5.6	81.6, SD = 3.7	45.0, SD = 5.2

optimised for shape detection preferred lower discharge values (0.1, SD = 0.3 vs 0.61, SD = 0.3) and LTD values closer to the limits of the permitted range. Preferred parameter spaces for LTP and inhLTP were not significantly different.

Conclusions

In this study, we tested a setup combining an EA for developing learning hyperparameters and SNNs with STDP. Previous attempts to adjust the hyperparameters using manual selection and random search were not successful. However, evolution enabled the SNN to achieve high precision with minimal computational effort for the direction and shape detection tasks.

Importantly, the evolutionary process led to filling the population with networks which were performing well in multiple generations and thus were better at generalising. Through the mutation process, it further explored the parameter space in the neighbourhood of the promising solutions. Both evolution and unsupervised training played a role in developing optimal solutions thus working at both the level of identifying the fittest networks in the population and learning in individual networks.

Despite the fact that we were not developing architecture directly, changing the hyperparameters changed the firing patterns during training and in consequence the weights. Some weights decreased to 0 and could not recover, which is equivalent to pruning connections in the original fully-connected SNN and changing its topology.

When detecting movement direction, the EA arrives at near-optimal solutions very early during the evolution and training process, which is consistent with the proposed suitability of the SNNs for performing this type of task, and with the previous observations that SNNs perform better than chance even without training (Kozdon and Bentley, 2017).

The link between the genotype – the learning hyperparameters - and phenotype is emergent and depends on the activity of the network, which in turn is determined by both the genotype and experience. This makes our evolutionary approach unusual in comparison to standard EAs, and more closely resemble the process through which the brain developed.

It has been proposed that non-inherited learning and heritable capacity to learn can guide evolution by improving fitness and altering the search space in which evolution operates (Baldwin, 1896; Hinton and Nowlan, 1987). Moreover, human babies are believed to undergo two main stages of brain optimisation (Wexler, 2010): the first one is developing what we recognise as the human brain with the stereotyped patterns of neuronal pathways (developed by evolution); the second is adapting neuronal weights in response to experience, including learning from parents. Thus, our model of inheritance can be said to contain also this second stage of development.

Over all, we believe that by automating development of learning hyperparameters for SNNs, our model can replace informed and random search and overcome short-term research bottlenecks, but more importantly it is a tool which allows to further test the unexplored potential of the brain-

inspired unsupervised learning in SNNs in what is the most obvious way: through evolution.

References

- Abbott, L. F. (1999) ‘Lapicque’s introduction of the integrate-and-fire model neuron (1907)’, *Brain Research Bulletin*. Elsevier Science Inc., 50(5–6), pp. 303–304. doi: 10.1016/S0361-9230(99)00161-6.
- Baldwin, J. M. (1896) ‘A New Factor in Evolution’, *The American Naturalist*. The University of Chicago Press/The American Society of Naturalists, 30(354), pp. 441–451. doi: 10.1086/276408.
- Baxter, J. (1992) ‘The Evolution of Learning Algorithms for Artificial Neural Networks’, *Complex systems: From biology to computation*, pp. 313–326.
- Belew, R. K., McInerney, J. and Schraudolph, N. N. (1992) ‘Evolving Networks: Using the Genetic Algorithm with Connectionist Learning’, *Artificial Life II*, 10, pp. 511–547.
- Bi, G.-Q. and Poo, M.-M. (1988) ‘Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type’, *The Journal of Neuroscience: the official journal of the Society for Neuroscience*, 18(24), pp. 10464–10472.
- Bohte, S., Kook, J. and PoutrHan, L. (2000) ‘SpikeProp: Backpropagation for Networks of Spiking Neurons’, in: Bruges: European Symposium on Artificial Neural Networks, pp. 419–424.
- Chalmers, D. J. (1990) ‘The evolution of learning: An experiment in genetic connectionism’, *Proceedings of the 1990 Connectionist Models Summer School*. Morgan Kaufmann, pp. 1–20.
- Conti, E., Madhavan, V., Such, F. P., Lehman, J., Stanley, K. O. and Clune, J. (2017) *Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty-Seeking Agents*. Available at: <http://arxiv.org/abs/1712.06560> (Accessed: 16 January 2018).
- D’amour, J. and Froemke, R. (2015) ‘Inhibitory and Excitatory Spike-Timing-Dependent Plasticity in the Auditory Cortex’, *Neuron*, 86(2), pp. 514–528. doi: 10.1016/j.neuron.2015.03.014.
- Floreano, D. and Mondada, F. (1996) ‘Evolution of plastic neurocontrollers for situated agents’, *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, 4, pp. 402–410. doi: 10.1021/cm101132g.
- Greve, R. B., Jacobsen, E. J. and Risi, S. (2016) ‘Evolving Neural Turing Machines for Reward-based Learning’, in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference - GECCO ’16*, pp. 117–124. doi: 10.1145/2908812.2908930.
- Hebb, D. O. (1950) ‘Organization of behavior. New York: Wiley’, *Journal of Clinical Psychology*. Wiley Subscription Services, Inc., A Wiley Company, 6(3), pp. 307–307. doi: 10.1002/1097-4679(195007)6:3<307::AID-JCLP2270060338>3.0.CO;2-K.
- Hinton, G. E. and Nowlan, S. J. (1987) ‘How Learning Can Guide Evolution’, *Complex Systems*, 1, pp. 495–502.
- Hodgkin, A. L. and Huxley, A. F. (1952) ‘A quantitative description of membrane current and its application to conduction and excitation in nerve.’, *The Journal of physiology*. Wiley-Blackwell, 117(4), pp. 500–44.
- Hopfield, J. J. and Brody, C. D. (2001) ‘What is a

- moment? Transient synchrony as a collective mechanism for spatiotemporal integration.’, *Proceedings of the National Academy of Sciences of the United States of America*, 98(3), pp. 1282–1287. doi: 10.1073/pnas.98.3.1282.
- Hubel, D. H. and Wiesel, T. N. (1970) ‘The period of susceptibility to the physiological effects of unilateral eye closure in kittens.’, *The Journal of physiology*. Wiley-Blackwell, 206(2), pp. 419–36.
- Kalia, A., Lesmes, L. A., Dorr, M., Gandhi, T., Chatterjee, G., Ganesh, S., Bex, P. J. and Sinha, P. (2014) ‘Development of pattern vision following early and extended blindness.’, *Proceedings of the National Academy of Sciences of the United States of America*. National Academy of Sciences, 111(5), pp. 2035–9. doi: 10.1073/pnas.1311041111.
- Kozdon, K. and Bentley, P. (2017) ‘Wide Learning’, in *IEEE Symposium Series on Computational Intelligence*. Honolulu: IEEE, pp. 3183–3190.
- Lehman, J., Chen, J., Clune, J. and Stanley, K. O. (2017) *ES Is More Than Just a Traditional Finite-Difference Approximator*. Available at: <https://arxiv.org/pdf/1712.06568.pdf> (Accessed: 16 January 2018).
- Markram, Lubke, H., Frotscher, J. ;, Sakmann, M. ; and Bert (1997) ‘Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs’, *Science*, 275(5297), p. 213.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. and Hassabis, D. (2015) ‘Human-level control through deep reinforcement learning’, *Nature*, 518(7540), pp. 529–533. doi: 10.1038/nature14236.
- Ponulak, F. and Kasiński, A. (2010) ‘Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting’, *Neural computation*, 22(2), pp. 467–510. doi: 10.1162/neco.2009.11-08-901.
- Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986) ‘Learning representations by back-propagating errors’, *Nature*. Nature Publishing Group, 323(6088), pp. 533–536. doi: 10.1038/323533a0.
- Saggie, K., Keinan, A. and Ruppin, E. (2004) ‘Spikes that count: rethinking spikiness in neurally embedded systems’, *Neurocomputing*, 5860, pp. 303–311. doi: 10.1016/j.neucom.2004.01.060.
- Salimans, T., Ho, J., Chen, X., Sidor, S. and Sutskever, I. (2017) ‘Evolution Strategies as a Scalable Alternative to Reinforcement Learning’. doi: 10.1.1.51.6328.
- Stanley, K. O., Bryant, B. D. and Miikkulainen, R. (2003) ‘Evolving adaptive neural networks with and without adaptive synapses’, *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, 4(2), pp. 2557–2564. doi: 10.1109/CEC.2003.1299410.
- Thomas, B. B., Seiler, M. J., Saddy, S. R., Coffey, P. J. and Aramant, R. B. (2004) ‘Optokinetic test to evaluate visual acuity of each eye independently’, *Journal of Neuroscience Methods*, 138, pp. 7–13. doi: 10.1016/j.jneumeth.2004.03.007.
- Wehrens, R. and Buydens, L. M. C. (2007) ‘Self- and super-organizing maps in R: The kohonen package’, *Journal of Statistical Software*. Foundation for Open Access Statistics, 21(5), pp. 1–19. doi: 10.18637/jss.v021.i05.
- Wexler, B. E. (2010) ‘Review Article Neuroplasticity, cultural evolution and cultural difference’, *World Cultural Psychiatry Research Review*, (Summer), pp. 11–22.
- Wilson, P. D. and Riesen, A. H. (1966) ‘Visual development in rhesus monkeys neonatally deprived of patterned light’, *Journal of Comparative and Physiological Psychology*, 61(1), pp. 87–95. doi: 10.1037/h0022873.