Testing the Genomic Bottleneck Hypothesis in Hebbian Meta-Learning

Rasmus Berg Palm RASMB@ITU.DK
Elias Najarro ENAJ@ITU.DK
Sebastian Risi SEBR@ITU.DK
IT University of Copenhagen

Abstract

Hebbian meta-learning has recently shown promise to solve hard reinforcement learning problems, allowing agents to adapt to some degree to changes in the environment. However, because each synapse in these approaches can learn a very specific learning rule, the ability to generalize to very different situations is likely reduced. We hypothesize that limiting the number of Hebbian learning rules through a "genomic bottleneck" can act as a regularizer leading to better generalization across changes to the environment. We test this hypothesis by decoupling the number of Hebbian learning rules from the number of synapses and systematically varying the number of Hebbian learning rules. The results in this paper suggest that simultaneously learning the Hebbian learning rules and their assignment to synapses is a difficult optimization problem, leading to poor performance in the environments tested. However, parallel research to ours finds that it is indeed possible to reduce the number of learning rules by clustering similar rules together. How to best implement a "genomic bottleneck" algorithm is thus an important research direction that warrants further investigation.

Keywords: Hebbian learning, Meta learning, Genomic Bottleneck

1. Introduction

Deep reinforcement learning has made great progress recently on very hard problems like Go, Starcraft and Dota using deep neural networks (Silver et al., 2016; Vinyals et al., 2019; Berner et al., 2019). However, once learned, the networks are static and highly specific. As such, there is very little capacity to adapt to changes in the environment or to generalize across environments (Justesen et al., 2018). For instance, the state-of-the-art AlphaStar agent trained to play one race in Starcraft cannot play another, even though it is a very similar task, and much less play Go or load a dishwasher. In contrast animals and humans show remarkable flexibility in their ability to generalize across tasks and adapt to changes.

Meta-learning proposes to overcome these limitations by learning-to-learn. That is to learn general learning rules that are broadly applicable and enable an agent to quickly adapt to changes in the environment or new tasks (Finn et al., 2017; Wang et al., 2016; Weng, 2018).

One particularly interesting approach to meta-learning is Hebbian meta-learning. The goal in Hebbian meta-learning is to learn Hebbian learning rules that enable an agent to quickly learn to perform well in its environment and adapt to changes. Hebbian learning is a learning paradigm in which the synaptic strength between neurons is determined by the correlation of their activity (Hebb, 2005); informally, "neurons that fire together, wire together". The Hebbian learning paradigm is promising since it proposes a simple and *general* learning paradigm supported by extensive empirical evidence in biological brains (Prezioso et al., 2018; Caporale and Dan, 2008).

Najarro and Risi (2020) showed promising results using Hebbian meta-learning to solve a difficult car racing reinforcement problem and quadruped locomotion. Impressively the agent's policy network was initialized randomly each episode, so it had to learn the task during its lifetime using only the meta-learned Hebbian learning rules. Further, the learned Hebbian learning rules generalized to an unseen quadruped leg damage scenario, which baseline non-plastic feedforward neural networks could not. However, each synapse had its own learning rule, which allowed the network to learn very specific learning rules for each neuron. This raises the question to which degree the network learned to learn, or whether each learning rule rather encoded very specific dynamics for each synapse.

We hypothesize that the architecture in Najarro and Risi (2020) is limited in its ability to discover general learning rules, but rather encodes very specific dynamics for each synapse. We further hypothesize that in order to learn generally useful learning rules the network must be limited to how many learning rules it can learn. This is inspired by the genomic bottleneck observed in humans and complex animals, where the information that can be stored in the genome is several orders of magnitude smaller than what is needed to determine the final wiring of the brain (Zador, 2019). We hypothesize that limiting the number of learning rules acts as a regularizer, which improves generalization and adaptability.

2. Related Work

Meta-Learning. In meta-learning, the goal is to learn to quickly adapt to a target task given a set of training tasks (Weng, 2018). Common approaches can loosely be categorized into black-box, optimization, and metric-based. Black-box methods learn a function that, conditioned on samples from a new task, outputs a function to solve the new task. For instance by jointly learning a network that can produce a latent summary of a new task and a network that can solve the task given the latent summary (Santoro et al., 2016; Mishra et al., 2017). Optimization-based attempts to learn to quickly optimize on a new task, e.g. by finding an initialization from which optimization on new tasks is fast (Finn et al., 2017; Nichol et al., 2018) or by learning the optimization algorithm (Andrychowicz et al., 2016; Li et al., 2017). Metric based approaches learns an embedding space that facilitates effective distance-based classification, e.g. Siamese networks (Koch et al., 2015) or prototypical networks (Snell et al., 2017). Meta Reinforcement Learning (meta-RL) extends the meta-learning idea to the reinforcement learning setting. Formally the goal is to quickly learn to perform well in a new Markov Decision Process (MDP) given a set of training MDPs. A common approach is to learn a recurrent neural network policy where the hidden activations are not reset between episodes (Wang et al., 2016; Duan et al., 2016). This allows the policy network to discover how the environment behaves across episodes and adapt its policy. Another common approach is meta-learning an initialization of a policy network from which policy gradient descent can quickly adapt to the new MDP (Finn et al., 2017; Song et al., 2019).

Plastic Artificial Neural Networks A less explored meta-learning approach is based on plastic neural networks that are optimized to have both innate properties and the ability to learn during their lifetime. For example, such networks can learn by changing the connectivity among neurons through local learning rules like Hebbian plasticity (Soltoggio et al., 2018, 2007). Often these networks are optimized through evolutionary algorithms (Soltoggio et al., 2018; Najarro and Risi, 2020) but more recently optimizing the plasticity of connections in a network through gradient descent has also been shown possible (Miconi et al., 2018). However, in contrast to the evolving Hebbian learning rules approach in Najarro and Risi (2020), the gradient descent approach (Miconi et al., 2018) was so far restricted to only evolving a single plasticity parameter for each connection instead of a different Hebbian rule.

Fast Weights Artificial neural networks (ANN) have either a slow form of storing information—through updating the weights—or, if they are recurrent, a very fast form of information storage in the form of internal activations. Fast weights seek to introduce an intermediate time-scale to the information storage in ANNs (Hinton and Plaut, 1987; Schmidhuber, 1992) and is motivated by the observation that biological neural networks have learning processes occurring concurrently that span across very different time scales. Recently, this approach has been successfully applied to image recognition tasks as well as a model for content-addressable memory (Ba et al., 2016). In the context of meta-learning, fast weights has been shown to perform meta-RL by having an ANN update the weights of a policy network on a per-task basis (Munkhdalai and Yu, 2017). Additionally, Munkhdalai and Trischler (2018) showed that a fast-weights Hebbian mechanism is capable of performing one-shot supervised learning tasks.

3. Hebbian meta-learning

In Hebbian meta-learning, the goal is to meta-learn Hebbian learning rules that enable an agent to perform well, and adapt to changes, in its environment.

3.1. Hebbian Learning

The agent acts in an episodic reinforcement learning environment and is controlled by a neural policy network. At the start of each episode, the policy network is initialized with random weights, which then undergoes changes according to Hebbian learning rules during the episode. Specifically, we use the ABCD Hebbian weight plasticity formalization (Soltoggio et al., 2007) and the weights are updated at each step of the episode. The change to the weight connecting neuron i and j is

$$\Delta w_{ij} = \eta_{ij} \left(A_{ij} o_i o_j + B_{ij} o_i + C_{ij} o_j + D_{ij} \right) , \tag{1}$$

where o_i and o_j are the pre- and post-synaptic activation of the neurons and $h = \{\eta, A, B, C, D\}$ are the Hebbian parameters that are fixed during the episode.

3.2. Evolutionary Strategies

The Hebbian parameters are meta-learned on an evolutionary time scale t, using evolutionary strategies (ES) (Salimans et al., 2017). In evolutionary strategies the goal is to maximize the expected fitness of a distribution of individuals, $\max_\theta \mathbb{E}_{z \sim p(z|\theta)} F(z)$, where $F(z)$ is the fitness of an individual z and θ parameterize this distribution. We compute the gradient of this objective using the score function estimator and use gradient ascent to maximize it,

$$\nabla_\theta \mathbb{E}_{z\sim p(z|\theta)} F(z) = \mathbb{E}_{z\sim p(z|\theta)} F(z) \nabla_\theta \log(p(z|\theta))$$

$$\theta^{t+1} = \theta^t + \alpha \left[\mathbb{E}_{z\sim p(z|\theta^t)} F(z) \nabla_{\theta^t} \log(p(z|\theta^t)) \right] , \tag{2}$$

where α is the learning rate. The expectation is evaluated using n samples, the population size. In order to use ES then, we must define $F(z)$ and $p(z|\theta)$. In this paper $F(z)$ is always the accumulated reward of an episode.

3.3. Individual Learning Rules

For the case of individual learning rules, N synapses each have their own learning rules $h_i = [\eta_i, A_i, B_i, C_i, D_i]$, which are drawn from independent normal distributions with fixed variance σ^2 and meta-learned means $\mu \in \mathbb{R}^{N\times 5}$. In this case $p(z|\theta) = p(h|\mu) = \prod_{i=1}^{N} \mathcal{N}(h_i|\mu_i, \sigma)$, where μ_i denotes the i'th row of the μ parameters. Inserting into eq. (2) and deriving the gradient this reduces to the expression in (Najarro and Risi, 2020; Salimans et al., 2017),

$$\mu^{t+1} = \mu^t + \alpha \left[\frac{1}{\sigma} \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)} F(\mu^t + \sigma\epsilon)\epsilon \right] ,$$

where ϵ is a sample from a standard normal $\mathcal{N}(0, I)$.

3.4. Shared Learning Rules

To explore the effect of sharing a limited amount of Hebbian learning rules we sample each synapses' learning rule from a Gaussian Mixture Model (GMM) with M clusters, such that

$$\begin{aligned} k &\sim p(k|\lambda) \in [1, ..., M]^N \, , \lambda \in \mathbb{R}^{N\times M} \, , \\ h &\sim \mathcal{N}(h|\mu_k, \sigma) \in \mathbb{R}^{N\times 5} \, , \mu \in \mathbb{R}^{M\times 5} \, , \\ p(z|\theta) = p(h|\mu, \lambda) &= \prod_{i=1}^{N} \sum_{k=1}^{M} \mathcal{N}(h_i|\mu_k, \sigma) p(k|\lambda_i) \, . \end{aligned} \tag{3}$$

Here $p(k|\lambda_i) = e^{\lambda_{ik}} / \sum_{j=1}^{M} e^{\lambda_{ij}}$, i.e. the softmax categorical distribution parameterized by λ_i logits for synapse i and the meta learned parameters are μ and λ. We use automatic differentiation to compute the gradient of the log likelihood of this distribution with respect to μ and λ, and then use eq. (2) to update μ and λ.

This approach to assigning learning rules to synapses is the most flexible and direct but requires $N \times M$ parameters. However, we are only interested in testing the effect of limiting the number of learning rules, not the number of parameters or bits needed to encode an individual, although those are interesting directions for future work.

4. Experiments

The first experiment is to replicate the original results of Najarro and Risi (2020) to ensure a fair comparison. In all experiments, we use the same experimental setup, architectures, and hyper-parameters as in Najarro and Risi (2020) except where noted.

Similar to Najarro and Risi (2020) we experiment on the car racing and quadruped locomotion tasks. We perform leave-one-out cross-validation, with five variations for each task. The five variations for the car racing tasks are (1) default settings, (2,3) twice and half the road friction coefficient, and (4,5) constant force pushing the car west and east. For the quadruped locomotion task, the variations are (1) default settings, (2,3) left and right front leg damage as in Najarro and Risi (2020) and (4,5) 50% longer rear and front legs. We perform leave-one-out cross-validation by leaving one variation out for testing and training on the remaining variations. Specifically, we use eq. (2) to maximize $\mathbb{E}_{z\sim p(z|\theta)}F'(z)$ where F' is the fitness function of a meta-task formed by taking the average fitness across the four training variations of the task.

To test our core hypothesis we evaluate for a varying number of learning rules expressed as a fraction of the number of synapses such that $M = \frac{1}{\rho}N$. We vary $\rho = [1, 16, 32, 64, 128, 256, N]$, where $\rho = 1$ corresponds to a learning rule per synapse and $\rho = N$ corresponds to a single learning rule.

We compare to three baselines in all experiments: (1) A Hebbian meta-learning network with a learning rule per synapse as in Najarro and Risi (2020), (2) an identical static network with learned weights, and (3) a LSTM baseline with hidden states initialized to zero at the start of each episode (Hochreiter and Schmidhuber, 1997). We construct the LSTM baseline by replacing the first densely connected layer in the static baseline with a LSTM layer and keeping everything else the same. All architectures are optimized using ES.

5. Results

We use the EvoStrat computational library (Palm, 2020) for all experiments. Code to reproduce all results are available at github.com/rasmusbergpalm/hebbian-evolution.

5.1. Replication

We successfully replicated the results of Najarro and Risi (2020) for the car racing and quadruped walker environments consistently achieving high rewards in 300 and 500 generations respectively with a Hebbian network with a learning rule per synapse.

5.2. Computational issues

Evaluating eq. 2 for the shared Hebbian learning rules for small values of ρ, becomes very expensive in terms of memory due to the number of parameters of the model. Unfortunately, that means we're not able to perform the experiments for values of $\rho < 32$.

5.3. Ant Environment

The results on the quadruped walker environment "AntBulletEnv-v0" are summarized in table 1. The networks are trained on the 4 left-most morphologies and tested on the "Long

front" morphology corresponding to 50% longer front legs.
Baselines The baselines all do reasonably well. The static network is best on average on the training tasks, whereas the LSTM and Hebbian learning rules are better on the unseen test task, although the standard deviations are large. This is in line with the original results from Najarro and Risi (2020).
Shared learning rules We first test with $\rho = 128$, which fails on both the training and testing morphology. Given that it fails on the training tasks we only evaluate subsequent values of ρ on the default training morphology. We assume that if it fails here it will also fail under the more difficult leave-one-out meta-learning testing scheme.

Table 1: Ant environment results. Average distance traveled. Bold indicate unseen testing morphology. Mean and standard deviation from 100 samples. Rounded to nearest integer.

Model	Default	Damage left	Damage right	Long back	**Long front**
Static	1545 ± 30	839 ± 60	1891 ± 105	1197 ± 31	202 ± 174
LSTM	1097 ± 57	1374 ± 74	1240 ± 140	806 ± 113	314 ± 115
Hebbian	928 ± 88	1102 ± 164	981 ± 140	741 ± 169	364 ± 131
Shared($\rho = 32$)	2 ± 12				
Shared($\rho = 64$)	3 ± 13				
Shared($\rho = 128$)	0 ± 10	-9 ± 18	15 ± 31	1 ± 6	0 ± 4
Shared($\rho = 256$)	2 ± 12				
Shared($\rho = N$)	-1 ± 0				

5.4. Car Racing Environment

Baselines All the baselines learn to perform well, surprisingly also on the unseen test environment variation.
Shared learning rules Similarly to the ant results, we test with $\rho = 128$ first, and it fails to learn anything on both the training and testing tasks. Given similar negative results from the ant environment, we conclude that there is something fundamentally problematic about the shared learning rule experiments, and focus on diagnosing that. As such we do not test for other values of ρ on the car environment.

Table 2: Car racing environment results. Average distance traveled.

Model	Default	Half friction	Double friction	Pushed left	**Pushed right**
Static	784 ± 119	491 ± 202	789 ± 202	738 ± 219	628 ± 228
LSTM	680 ± 247	375 ± 209	711 ± 243	653 ± 273	610 ± 316
Hebbian	674 ± 135	552 ± 190	692 ± 120	725 ± 139	606 ± 186
Shared($\rho = 128$)	6 ± 3	5 ± 1	8 ± 7	6 ± 1	6 ± 2

5.5. Post-hoc analysis

We deviate from our pre-planned experimental setup to see if we can get the shared Hebbian learning rules to work. We experiment on the default Ant morphology using $\rho = 128$. We independently try lowering the learning rate to 0.01 (from 0.2), reducing the number of shared learning rules to 16 and initializing the shared learning rule means from $\mathcal{N}(0, 10)$ (from $\mathcal{N}(0, 1)$). However, none of these modifications helped. We also try randomly assigning shared learning rules instead of learning the assignments. This modification improves performance, however, the networks are still much worse than the baselines (not shown). We don't pursue the random assignment any further since we are interested in learning the assignments.

We note that the Hebbian networks with shared learning rules fail to learn the mixing weights λ, which stay close to a uniform distribution, and that the gradient estimates on the mixing weights are two orders of magnitude smaller than on the learning rule means μ. We manually verify that the $\log p(h|\mu, \lambda, \sigma)$ computations and the gradients $\nabla_\mu \log p(h|\mu, \lambda, \sigma)$ and $\nabla_\lambda \log p(h|\mu, \lambda, \sigma)$ are correct for a simple test case.

Suspecting a difficult optimization landscape we use the Adam optimizer which scales the learning rate individually for each parameter (Kingma and Ba, 2014). With the Adam optimizer, the networks attain poor but non-zero performance. This rules out the most severe bugs in the implementation and experimental setup and lends support to our hypothesis that it is "just" a difficult optimization problem. We experiment with several learning rates and learning rate schedules, achieving the best results using a learning rate of 1.0 and an exponential learning rate decay of 0.9931 which corresponds to halving the learning rate every 100 updates. However, these results are still much worse than the baselines, achieving 555 ± 89 reward on the default Ant morphology.

6. Discussion

The experimental results in this paper do not support our initial hypothesis. However, parallel research to ours found that it is indeed possible to reduce the number of Hebbian learning rules significantly and improve generalization on unseen environments by iteratively clustering similar learning rules and re-training (Pedersen and Risi, 2021). As such, the results suggest that our approach (i.e. modeling the learning rules as drawn from a Gaussian Mixture Model and jointly optimizing the assignment and the learning rules) is a hard optimization problem.

There is a bit of a chicken and egg problem in jointly optimizing the assignment and the learning rules: it is challenging to optimize the learning rules before they are assigned to synapses, and it is challenging to optimize the assignment to synapses before the learning rules are learned. The approach introduced by Pedersen and Risi (2021) avoids this issue by learning a Hebbian learning rule per synapse, while periodically clustering similar learning rules and re-training with the reduced set of learning rules. It could also be possible to avoid this chicken and egg problem by alternately learning the assignment and the learning rules in an Expectation-Maximization (EM) inspired approach (Dempster et al., 1977).

Another possible explanation for the difficulty in directly learning a small number of shared Hebbian learning rules is the lottery ticket hypothesis (Frankle and Carbin, 2018). The lottery ticket hypothesis is based on the observation that neural networks can often be pruned aggressively without losing performance, and that it is in fact a small sub-network of the bigger network which solves the task; however, it is difficult to learn the small sub-networks directly. The hypothesis is that overparameterized neural networks are more likely to contain sub-network, which are initialized in such a way that they can be effectively optimized to solve the task. If finding a well-initialized sub-network corresponds to winning the lottery then bigger networks simply have more tickets and are thus more likely to win.

The main positive finding of this paper is that we reproduce the results from (Najarro and Risi, 2020) with different tasks and a different implementation. We thus add to the growing body of evidence that shows it is indeed possible to solve complex tasks with randomly initialized neural networks that learn using local Hebbian learning rules.

One interesting idea for future work is to re-define an individual $z = \{h, k\}$, i.e. the sampled learning rules h *and* the learning rule clusters they are sampled from k, such that $p(z|\theta) = p(h, k|\theta) = \prod_{i=1}^{N} \mathcal{N}(h_i|\mu_k, \sigma)p(k_i|\lambda_i)$. Without the inner sum, this setup would be much cheaper to optimize for many learning rule clusters. It might also result in an easier optimization landscape. However, it would technically not be correct since an individual's fitness only depends on the sampled learning rules h, not which learning rule cluster it is sampled from k.

A limitation of our approach to Hebbian meta-learning is that the agent cannot adapt to changes in the reward function during its lifetime, since it does not observe the reward. The reward is only observed at the evolutionary time scale. As such all the training and test tasks must share the same reward function. This is in contrast to common benchmarks in meta RL where the reward function differs across training and test MDPs. Extending our method with approaches that can modulate Hebbian learning based on the reward (Abbott, 1990; Krichmar, 2008; Soltoggio et al., 2007, 2018) or indirectly encode plasticity (Risi and Stanley, 2010), could be a promising future research direction.

Changes to the pre-registration proposal. The pre-registration proposal sections (1, 2, 3, and 4) have been changed insignificantly to correct grammar and spelling. The main findings have been added to the abstract.

References

LF Abbott. Modulation of function and gated learning in a network memory. *Proceedings of the National Academy of Sciences*, 87(23):9241–9245, 1990.

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in neural information processing systems*, pages 3981–3989, 2016.

Jimmy Ba, Geoffrey Hinton, Volodymyr Mnih, Joel Z. Leibo, and Catalin Ionescu. Using Fast Weights to Attend to the Recent Past. *ArXiv e-prints*, Oct 2016. URL https://arxiv.org/abs/1610.06258v3.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Natalia Caporale and Yang Dan. Spike timing–dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.*, 31:25–46, 2008.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Donald Olding Hebb. *The organization of behavior: A neuropsychological theory.* Psychology Press, 2005.

Geoffrey E Hinton and David C Plaut. Using fast weights to deblur old memories. In *Proceedings of the ninth annual conference of the Cognitive Science Society*, pages 177–186, 1987.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

Niels Justesen, Ruben Rodriguez Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, and Sebastian Risi. Illuminating generalization in deep reinforcement learning through procedural level generation. *arXiv preprint arXiv:1806.10729*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

Jeffrey L Krichmar. The neuromodulatory system: a framework for survival and adaptive behavior in a challenging world. *Adaptive Behavior*, 16(6):385–399, 2008.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

Thomas Miconi, Jeff Clune, and Kenneth O Stanley. Differentiable plasticity: training plastic neural networks with backpropagation. *arXiv preprint arXiv:1804.02464*, 2018.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

Tsendsuren Munkhdalai and Adam Trischler. Metalearning with Hebbian Fast Weights. *ArXiv e-prints*, Jul 2018. URL https://arxiv.org/abs/1807.05076.

Tsendsuren Munkhdalai and Hong Yu. Meta Networks. *ArXiv e-prints*, Mar 2017. URL https://arxiv.org/abs/1703.00837v2.

Elias Najarro and Sebastian Risi. Meta-learning through hebbian plasticity in random networks. *Neural information processing systems (NeurIPS)*, 2020.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Rasmus Berg Palm. Evostrat. https://github.com/rasmusbergpalm/evostrat, 2020.

Joachim Winther Pedersen and Sebastian Risi. Evolving and merging hebbian learning rules: Increasing generalization by decreasing the number of rules. *arXiv preprint arXiv:2104.07959*, 2021.

M Prezioso, MR Mahmoodi, F Merrikh Bayat, H Nili, H Kim, A Vincent, and DB Strukov. Spike-timing-dependent plasticity learning of coincidence detection with passively integrated memristive circuits. *Nature communications*, 9(1):1–8, 2018.

Sebastian Risi and Kenneth O Stanley. Indirectly encoding neural plasticity as a pattern of local rules. In *International Conference on Simulation of Adaptive Behavior*, pages 533–543. Springer, 2010.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

J. Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087, 2017.

Andrea Soltoggio, Peter Durr, Claudio Mattiussi, and Dario Floreano. Evolving neuromodulatory topologies for reinforcement learning-like problems. In *2007 IEEE Congress on Evolutionary Computation*, pages 2471–2478. IEEE, 2007.

Andrea Soltoggio, Kenneth O Stanley, and Sebastian Risi. Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. *Neural Networks*, 108: 48–67, 2018.

Xingyou Song, Wenbo Gao, Yuxiang Yang, Krzysztof Choromanski, Aldo Pacchiano, and Yunhao Tang. Es-maml: Simple hessian-free meta learning. *arXiv preprint arXiv:1910.01215*, 2019.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Lilian Weng. Meta-learning: Learning to learn fast. *lilianweng.github.io/lil-log*, 2018. URL `http://lilianweng.github.io/lil-log/2018/11/29/meta-learning.html`.

Anthony M Zador. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature communications*, 10(1):1–7, 2019.