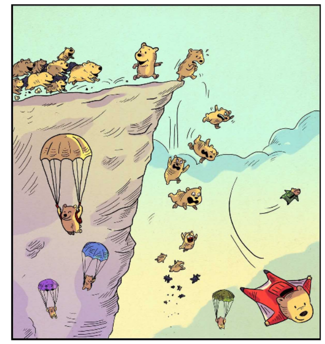
Synergies with Reinforcement Learning

Sebastian Risi and Risto Miikkulainen

February 20, 2026

Introduction to RL and NE

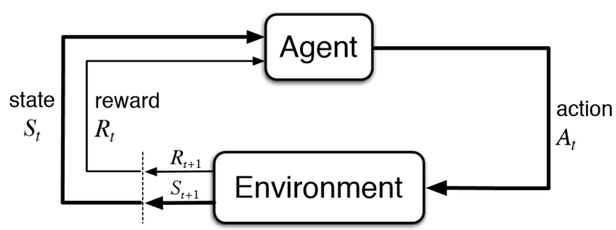
- Reinforcement Learning (RL) and Neuroevolution (NE) are two key methods to optimize neural networks.
- RL uses trial-and-error with rewards/punishments, while NE optimizes networks through evolutionary algorithms.
- Both approaches have distinct strengths and weaknesses, and they can be combined for better performance.



(Tan 2017)

Key Strengths and Weaknesses of RL

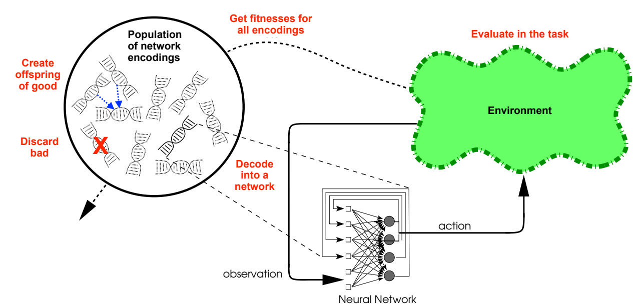
- RL excels at solving sequential decision-making tasks and dynamic environments.
- Useful in domains where the environment model is unknown or complex (e.g., robotics, game playing).
- Challenges:
  - High data and computational requirements.
  - Sensitive to hyperparameters and unstable training.
  - Struggles with high-dimensional state/action spaces.



(Sutton & Barto 2018)

Key Strengths and Weaknesses of NE

- NE optimizes both network topology and parameters simultaneously.
- More robust to local minima compared to gradient-based methods in RL.
- Strengths:
  - Diverse policies through repeated evolution.
  - Suitable when the network structure is unknown.
- Limitations:
  - Less effective for real-time adaptation.
  - Lower sample efficiency in dense reward environments.



(Gomez 2025)

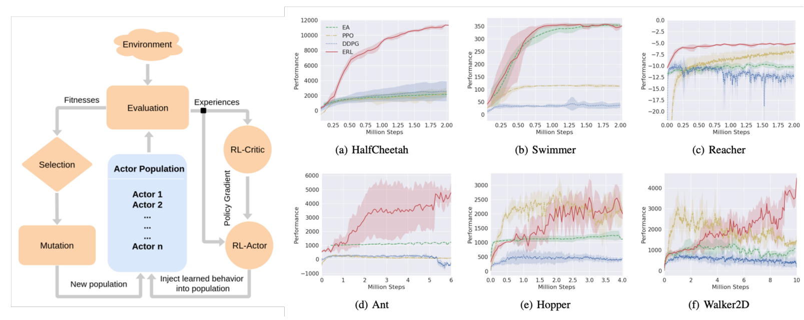
Synergistic Combinations

- RL and NE can be combined to leverage their strengths.
- Hybrid methods such as Evolutionary Reinforcement Learning (ERL) improve exploration and sample efficiency.
- We explore examples of these combinations.

(Freep!k 2025)

Evolutionary Reinforcement Learning (ERL)

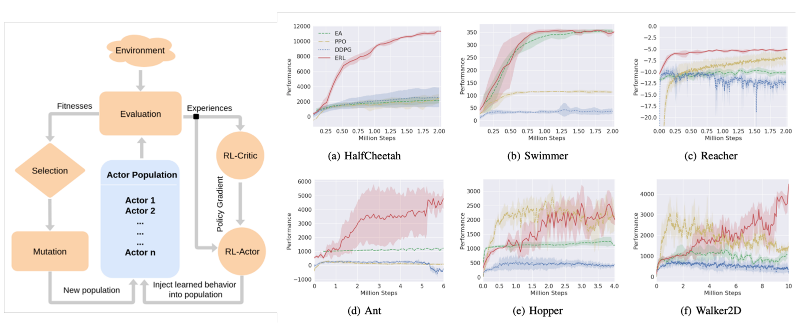
- ERL combines evolutionary algorithms (EA) with deep RL to tackle exploration issues.
- Periodically injects RL agent's gradient information into the evolutionary population.
- Balances exploration and exploitation with evolutionary diversity and gradient-based learning.



(Khadka & Tumer 2018)

Benefits of ERL

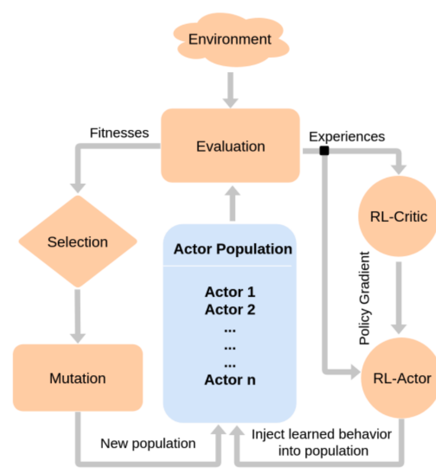
- EA provides effective exploration and handles sparse rewards better than RL.
- RL improves sample efficiency through gradient-based learning.
- ERL outperforms pure EA and RL approaches in various continuous control tasks.



(Khadka & Tumer 2018)

ERL Exploration and Training Process

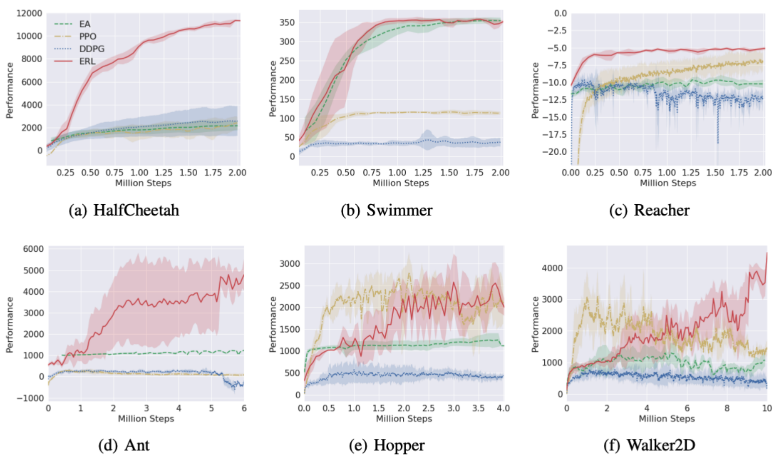
- EA explores in parameter space, RL explores in action space.
- Replay buffer stores state-action-reward transitions for RL training.
- Synchronization phase copies RL actor network weights back to the EA population.



(Khadka & Tumer 2018)

ERL in Continuous Control Tasks

- ERL significantly outperforms state-of-the-art DRL methods such as DDPG and PPO.
- Effective in tasks with sparse rewards and deceptive fitness landscapes.

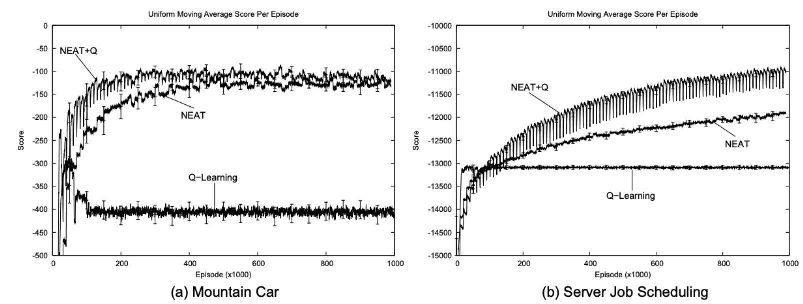


(a) HalfCheetah (b) Swimmer (c) Reacher
(d) Ant (e) Hopper (f) Walker2D

(Khadka & Tumer 2018)

Evolving Value Networks for RL

- Many RL algorithms rely on value functions to estimate cumulative rewards.
- NEAT evolves both network weights and architectures for better value networks.
- NEAT+Q-learning can outperform standard value function approximation methods.

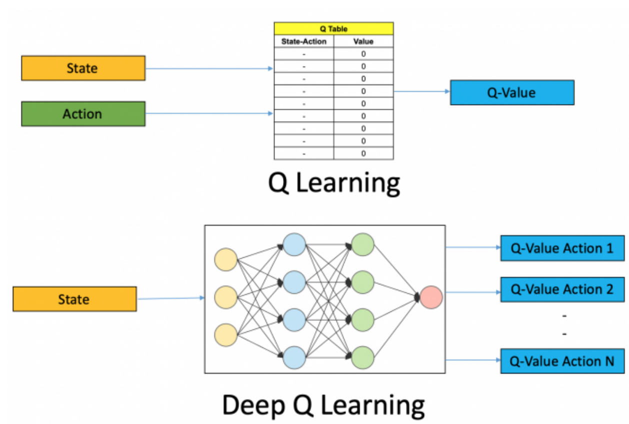


(a) Mountain Car (b) Server Job Scheduling

(Whiteson 2006)

Q-learning Overview

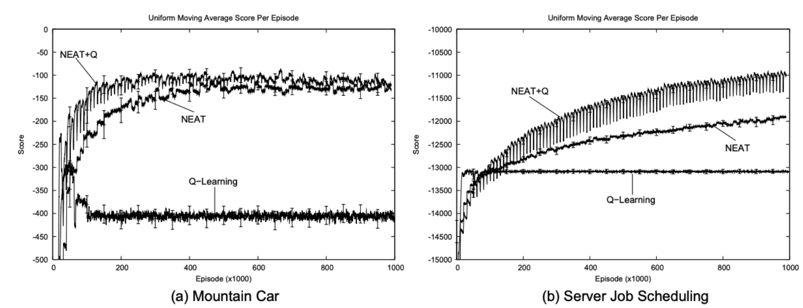
- Q-learning is a model-free RL algorithm aiming to learn the optimal action-value function $Q(s, a)$.
- The agent updates its Q-values based on observed rewards and future states.
- Neural networks can approximate Q-tables for high-dimensional spaces.



(Foy 2020)

NEAT+Q learning Performance

- NEAT evolves network architectures that help Q-learning learn more efficiently.
- Q-learning with NEAT outperforms manually designed networks in tasks like the mountain car and server job scheduling.

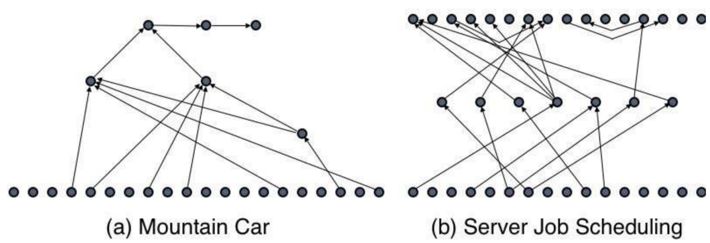


(a) Mountain Car (b) Server Job Scheduling

(Whiteson 2006)

NEAT+Q: Evolved Network Topologies

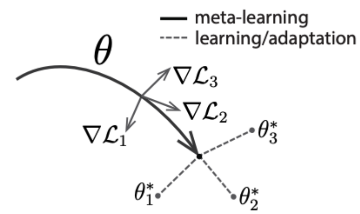
- NEAT evolves sparse, irregular network topologies that are hard to design manually.
- These evolved networks excel in various RL tasks.



(Whiteson 2006)

Evolutionary Meta-Learning

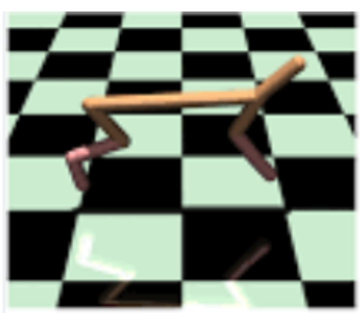
- Meta-learning aims to evolve networks that can rapidly adapt to new tasks.
  - Same term as before but more specific meaning in RL.
- Model-Agnostic Meta-Learning (MAML) finds good starting points for learning.
- Evolutionary methods like MAML-Baldwin and ES-MAML improve on MAML by using evolutionary algorithms.



(Finn et al. 2017)

MAML-Baldwin Approach

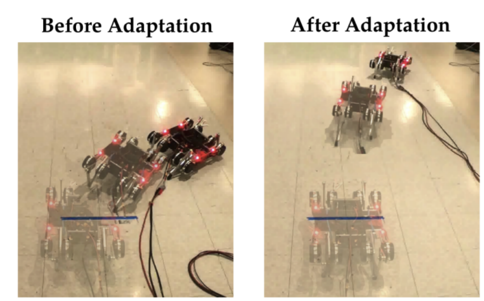
- MAML-Baldwin combines an evolutionary algorithm in the outer loop with RL in the inner loop.
- Evolves initial weights that can adapt to different tasks during the agent's lifetime.
- E.g. in the half-cheetah tasks, adapts to changing directions within seconds of simulation.



(https://github.com/shunzh/pytorch-maml-rl)

ES-MAML Overview

- ES-MAML uses evolutionary strategies (ES) in both the outer and inner loops.
- It is conceptually simple, avoids second-order derivatives of MAML and MAML-Baldwin, and is effective in noisy environments.
- Example: Adapting to reduced motor power or payload changes.
- ES-MAML outperforms standard MAML in noisy and real-world scenarios.



(Song et al. 2020)
Demo link: `https://youtu.be/_QPMCDdFC3E`

Conclusions on Synergistic Combinations

- RL and NE can be combined to tackle the limitations of each approach.
- NE explores more broadly, RL refines more carefully.
- Hybrid methods like ERL, NEAT+Q, (ES-)MAML(-Baldwin) show promising results in various domains.

(Freep!k 2025)

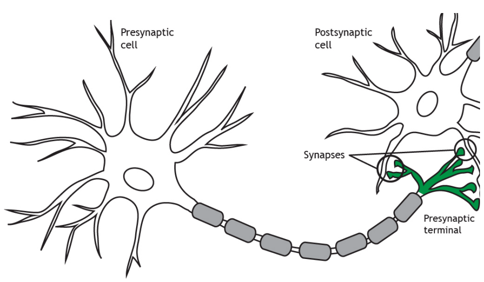
Evolving Neural Networks to Reinforcement Learn

- Hybrid RL and NE approaches can still take many trials to learn.
- Idea: Evolve neural networks that can learn their own learning rules, allowing them to adapt during their lifetime
- Evolution handles slow environmental changes; learning allows adaptation to fast changes.

(Timplaru 2025)

Evolving Hebbian Learning Rules

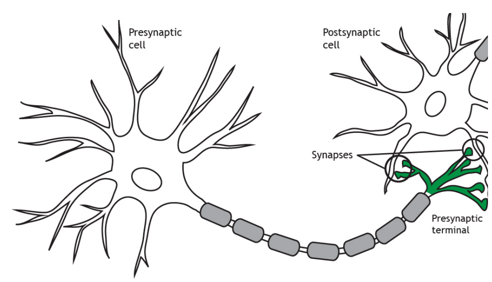
- Hebbian learning adjusts weights based on the activation of neurons.
- Evolution optimizes both initial weights and how those weights change during learning.
- Example rule: $\Delta w_{i \to j} = \eta x_i x_j$, where η is the learning rate, x_i the activity of the presynaptic neuron, and x_j the activity of the postsynaptic neuron.



(Henley 2021)

Scaling Hebbian Networks

- Recent advances in evolution strategies allow scaling Hebbian networks.
- A more general Hebbian rule $\Delta w_{ji} = \eta[Ao_j o_i + Bo_j + Co_i + D]$, includes five parameters for each connection.
- Evolution optimizes these parameters, allowing the network to adapt to more complex environments.

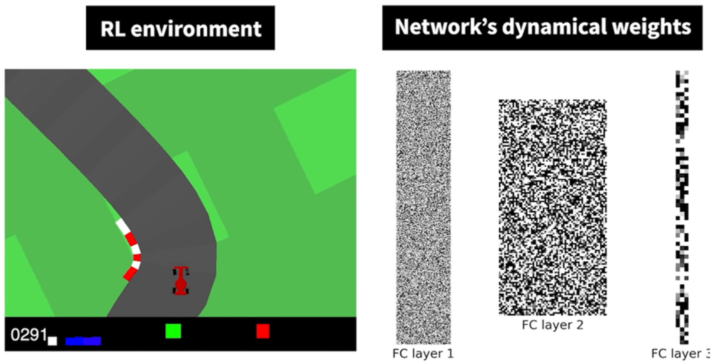


(Henley 2021)

Adaptation Without Rewards

- The evolved Hebbian network adapts without explicit reward feedback.
- The network starts from random weights and adjusts based on activity.
- Adaptation occurs in fewer timesteps compared to traditional RL methods — even in real time.
- Evolution sets up the learning so the task is solved.

Embedded Demo:

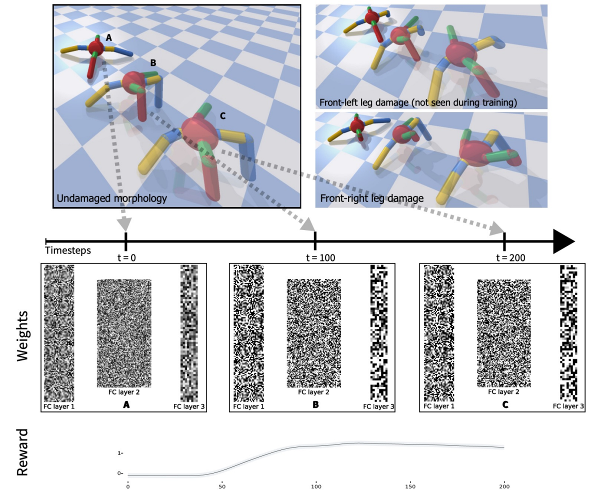


(https://github.com/enajx/HebbianMetaLearning)

Hebbian Networks for Complex Tasks

- Evolved Hebbian networks handle complex, dynamic environments.
- Example: Quadrupedal robot control using Hebbian rules.
- The network adapts to morphological changes, such as limb damage.

Embedded Demo:



(Najarro & Risi 2020)

Comparing Hebbian and Feedforward Networks

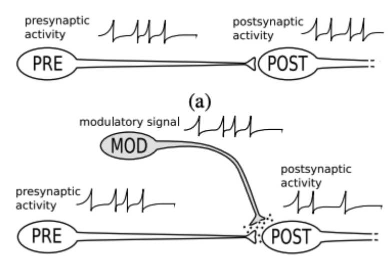
- Standard feedforward networks struggle to adapt to new robot morphologies.
- Hebbian networks quickly adapt, achieving high performance across different morphologies.
- The adaptive capability of Hebbian networks comes from the evolved learning rules.

Quadruped Damage	Seen / Unseen during training	Learning Rule	Distance travelled	Solved
No Damage	Seen	Hebbian	1051 ± 113	True
No Damage	Seen	static weights	1604 ± 171	True
Right front leg	Seen	Hebbian	1019 ± 116	True
Right front leg	Seen	static weights	1431 ± 54	True
Left front leg	Unseen	Hebbian	452 ± 95	True
Left front leg	Unseen	static weights	68 ± 56	False

(Najarro & Risi 2020)

Neuromodulation in Biological Systems

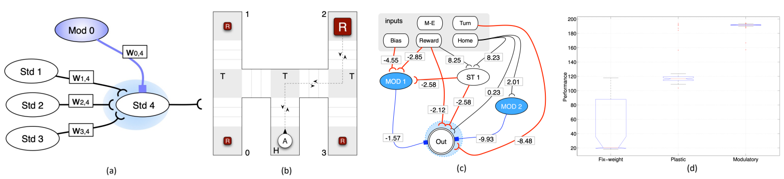
- Neuromodulation: Regulates neural activity via neurotransmitters and chemicals.
- Influences synaptic strength, neuron excitability, and network dynamics.
- Critical for learning, memory, and adaptation to new experiences.



(Soltoggio et al. 2008)

Neuromodulation in Evolved Networks

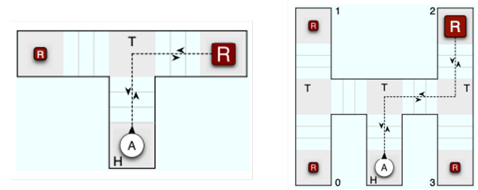
- Neuromodulation modifies Hebbian plasticity in evolving networks.
- Plasticity can be switched on/off based on rewards to allow learning.
- Structural mutations introduce neuromodulatory nodes in the network.



(Soltoggio et al. 2008)

Neuromodulation in the T-Maze Task

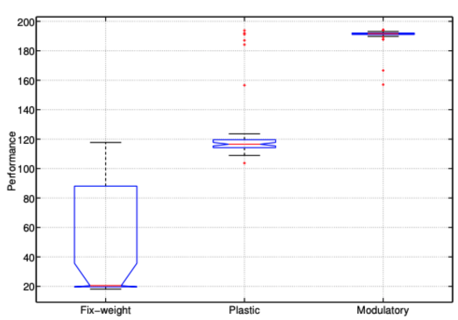
- Task: Single and double T-Maze navigation where the position of a high reward R changes, requiring adaptation.
- Neuromodulation boosts performance in complex tasks (double T-Maze).
- Networks evolve the ability to switch plasticity based on rewards.



(Soltoggio et al. 2008)

Results of Neuromodulation

- Neuromodulated networks outperform Hebbian networks in the more complex double T-Maze task.
- Neuromodulation helps separate circuits for control and adaptation.
- Disabling neuromodulation leads to loss of adaptive behavior in evolved networks.



(Soltoggio et al. 2008)

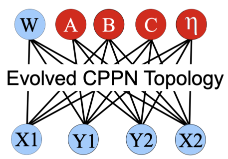
Challenges in Encoding Plasticity

- Direct encoding requires discovering learning rules for every synapse.
- Regularities in connectivity patterns may extend to plasticity rules.
- Indirect encoding can generalize plasticity across a network.
- HyperNEAT encodes weight patterns based on network geometry.
- Adaptive HyperNEAT extends this to plasticity encoding.

Learning Rule Parameters in Adaptive HyperNEAT

- CPPN produces not just weights but also plasticity parameters.
- Parameters include learning rate η, correlation term A, presynaptic factor B, and postsynaptic factor C.
- Weights are updated based on the generalized Hebbian learning rule:

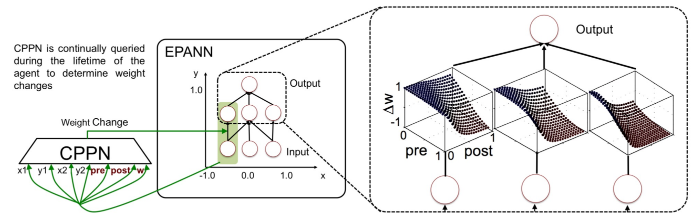
$$\Delta w_{ij} = \eta \cdot [A o_i o_j + B o_i + C o_j]$$



(Risi & Stanley 2010)

Generalized Adaptive HyperNEAT

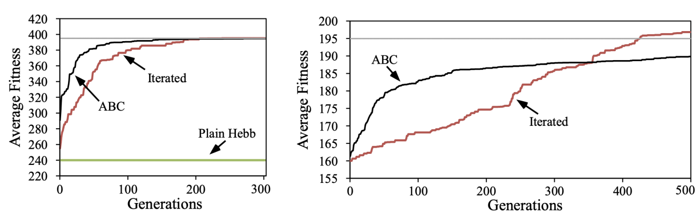
- Generalized model can encode arbitrary learning rules.
- CPPN takes presynaptic activity, postsynaptic activity, and connection weight as inputs and outputs weight changes.



(Soltoggio et al. 2018)

Performance in T-Maze Task

- Task: T-Maze with and without non-linear reward encoding.
- General Adaptive HyperNEAT (Iterated) outperforms Hebbian networks, and HyperNEAT ABC model in the non-linear T-Maze.
- Learns non-linear learning rules based on network geometry and node activity.

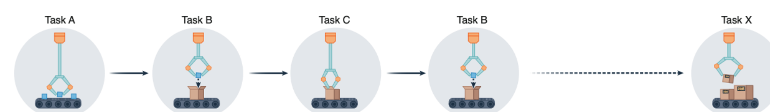


(a) T-Maze (b) T-Maze with Nonlinear Reward Signature

(Risi & Stanley 2010)

Evolving Neural Networks to Continually Learn

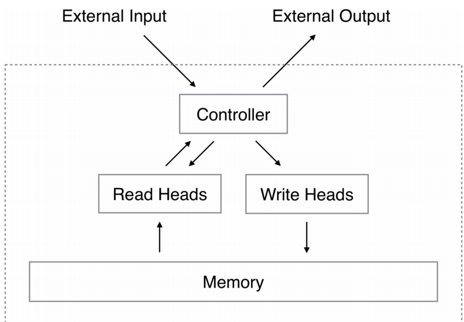
- Key challenge in AI: learning new tasks without forgetting previous ones.
- *Catastrophic forgetting* is a major issue in most current neural networks.
- Need for mechanisms that allow memory retention across tasks.



(Kudithipudi et al. 2022)

Foundation: Memory-Augmented Neural Networks

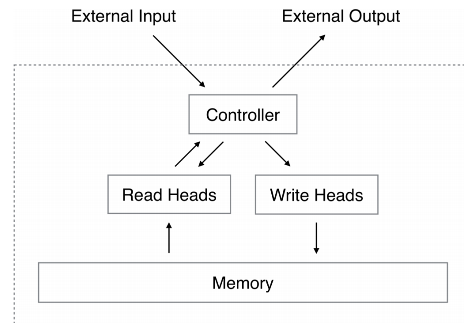
- Neural Turing Machine (NTM) combines traditional neural networks with external memory.
  - External memory allows the network to store and retrieve data over time.
  - Read and write heads interact with memory.
- Fully differentiable; trained with gradient descent.
- Capable of performing tasks like copy, sort, and associative recall.



(Graves et al. 2014)

Differentiable vs Evolved NTM

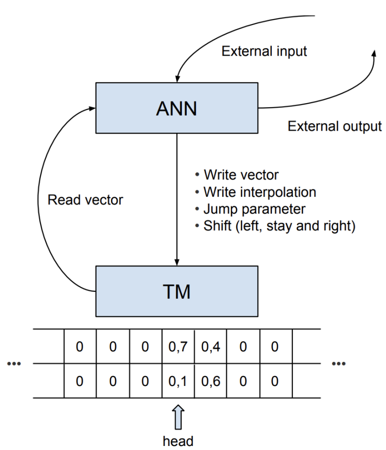
- Differentiable NTM has limitations: fixed memory size, "soft" attention.
- Evolved Neural Turing Machine (ENTM) uses neuroevolution to improve generalization and flexibility.
- ENTM allows hard attention and theoretically unlimited memory capacity.



(Graves et al. 2014)

ENTM Operations

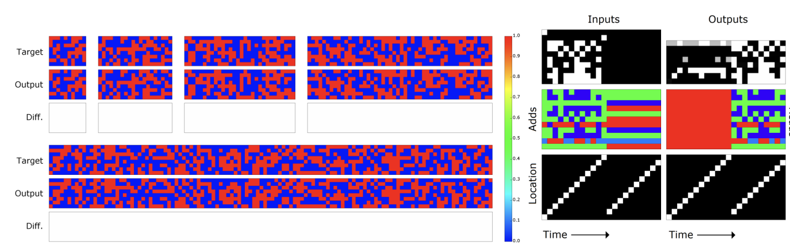
- **Write:** Updates memory vector at the head's location.
- **Content Jump:** Head jumps to memory location most similar to write vector.
- **Shift:** Moves the memory head left, right, or maintains position.
- **Read:** Reads content from the memory vector for use in the next cycle.



(Lüders et al. 2016)

Performance in Copy Task

- Copy task: memorize and retrieve a sequence of binary vectors.
- Evolved NTM generalizes perfectly to long sequences.
- Evolved network is smaller and simpler compared to the original NTM.

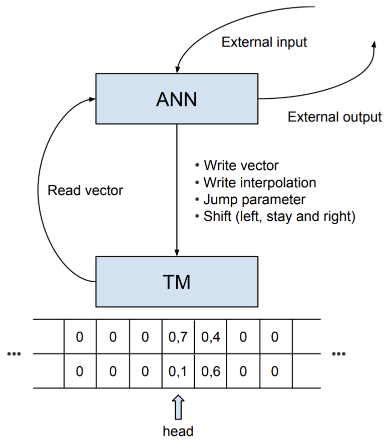


(a) Successful Copy (b) Memory Additions

(Greve et al. 2016)

Continual Learning with ENTM

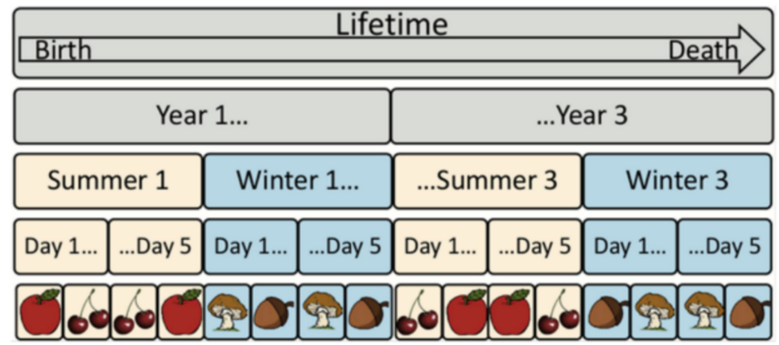
- External memory helps solve the problem of catastrophic forgetting.
- Memory allows storing new information without overwriting old information.



(Lüders et al. 2016)

Season Task Example

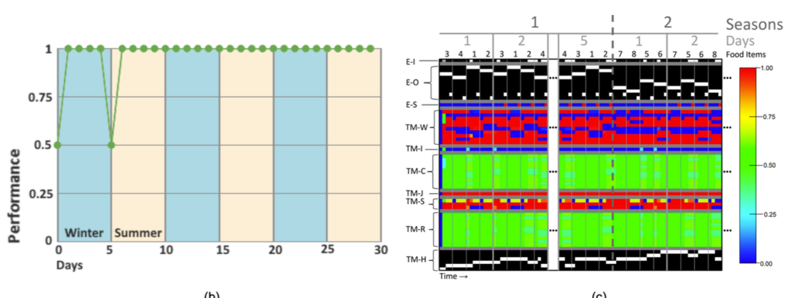
- Continual learning in the Season task: Learn which food items are nutritious or poisonous across different seasons.
- Test agent's ability to retain knowledge across changing environments.



(Lüders et al. 2016)

Season Task Example

- ENTM excels at learning new associations while retaining old ones.
- Learns the seasons quickly, retains over time.

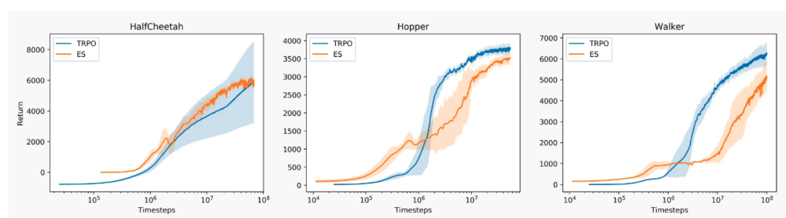


ENTM inputs and outputs over time

(Lüders et al. 2016)

Scaling Up Neuroevolution in RL Tasks

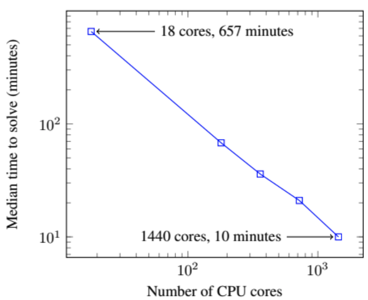
- Advances in hardware accelerators like GPUs have driven scaling in deep learning.
- Neuroevolution (NE) approaches are catching up by leveraging parallel computing resources.
- NE is competitive with RL on larger tasks by scaling across CPUs and GPUs.



(Salimans et al. 2016)

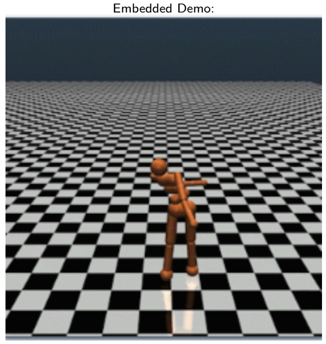
Parallelism in Neuroevolution

- Evolution Strategies (ES), Genetic Algorithms (GA), and even Random Search (RS) can benefit from parallelism.
- ES can scale effectively with thousands of CPUs by reducing communication overhead.
- E.g. found optimal solutions in 10 minutes on humanoid tasks with massive parallelism.



(Salimans et al. 2017)

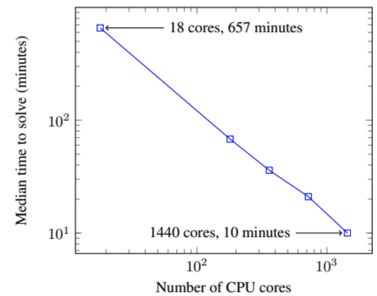
Embedded Demo:



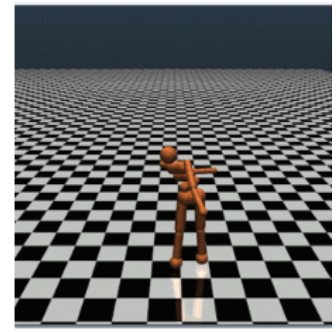
(https://openai.com/index/evolution-strategies/)

ES Advantages

- ES advantages over RL include handling sparse rewards, long time horizons, and no backpropagation.
- Invariant to the frequency of actions.
- Applies to a broader range of tasks.



(Salimans et al. 2017)



(https://openai.com/index/evolution-strategies/)

Simple GA in Atari Games

- Next step from ES isSimple GA: No crossover, no evolving topologies, just simple truncation selection and additive Gaussian noise.
- Demonstrated competitive results on Atari games by optimizing a deep CNN with 4M parameters.

	DQN	ES	A3C	RS	GA	GA
Frames	200M	1B	1B	1B	1B	6B
Time	~7-10d	~ 1h	~ 4d	~ 1h or 4h	~ 1h or 4h	~ 6h or 24h
Forward Passes	450M	250M	250M	250M	250M	1.5B
Backward Passes	400M	0	250M	0	0	0
Operations	1.25B U	250M U	1B U	250M U	250M U	1.5B U
amidar	**978**	112	264	143	263	377
assault	4,280	1,674	**5,475**	649	714	814
asterix	4,359	1,440	**22,140**	1,197	1,850	2,255
asteroids	1,365	1,562	**4,475**	1,307	1,661	2,700
atlantis	279,987	**1,267,410**	911,091	26,371	76,273	129,167
enduro	**729**	95	-82	36	60	80
frostbite	797	370	191	1,164	**4,536**	**6,220**
gravitar	473	**805**	304	431	476	764
kangaroo	7,259	**11,200**	94	1,099	3,790	**11,254**
seaquest	**5,861**	1,390	2,355	503	798	850
skiing	-13,062	-15,443	-10,911	-7,679	†**-6,502**	†**-5,541**
venture	163	760	23	488	**969**	†**1,422**
zaxxon	5,363	6,380	**24,622**	2,538	6,180	7,864

(Such et al. 2017)

Broad Comparison in Atari Games

- GA, ES, DQN, and A3C each performed best on different Atari games.
- No clear winner across the board, but different strengths in different games. Highlights the potential for hybridizing RL and NE methods.

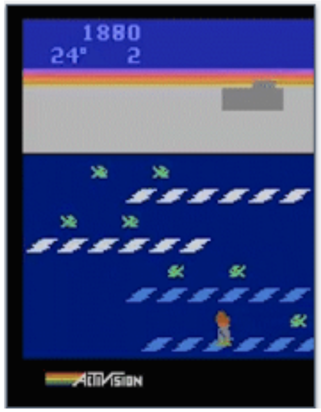
	DQN	ES	A3C	RS	GA	GA
Frames	200M	1B	1B	1B	1B	6B
Time	~7-10d	~ 1h	~ 4d	~ 1h or 4h	~ 1h or 4h	~ 6h or 24h
Forward Passes	450M	250M	250M	250M	250M	1.5B
Backward Passes	400M	0	250M	0	0	0
Operations	1.25B U	250M U	1B U	250M U	250M U	1.5B U
amidar	**978**	112	264	143	263	377
assault	4,280	1,674	**5,475**	649	714	814
asterix	4,359	1,440	**22,140**	1,197	1,850	2,255
asteroids	1,365	1,562	**4,475**	1,307	1,661	2,700
atlantis	279,987	**1,267,410**	911,091	26,371	76,273	129,167
enduro	**729**	95	-82	36	60	80
frostbite	797	370	191	1,164	**4,536**	**6,220**
gravitar	473	**805**	304	431	476	764
kangaroo	7,259	**11,200**	94	1,099	3,790	**11,254**
seaquest	**5,861**	1,390	2,355	503	798	850
skiing	-13,062	-15,443	-10,911	-7,679	†**-6,502**	†**-5,541**
venture	163	760	23	488	**969**	†**1,422**
zaxxon	5,363	6,380	**24,622**	2,538	6,180	7,864

(Such et al. 2017)

Random Search is Surprisingly Effective

- On several Atari Games, even random search outperformed RL!
- Local search sometimes finds sophisticated policies.
  - Example: Frostbite game strategy discovered by random search.
  - Similar results in Backgammon.
- Suggests that sometimes following gradients may hinder optimization.

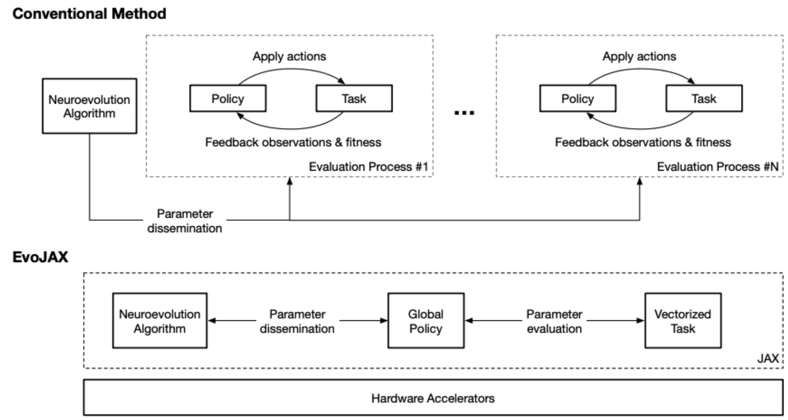
Embedded Demo:



(https://www.uber.com/en-FI/blog/deep-neuroevolution)

Scaling to GPUs and TPUs

- While NE has mostly relied on CPU parallelism, there is potential for GPU/TPU acceleration.
  - Can bring another level of speed and capability.
  - Possible through libraries like JAX, i.e. EvoJAX and EvoSAX.
  - JIT compilation and vectorized operations.



(Tang et al. 2022)

EvoJAX in Action

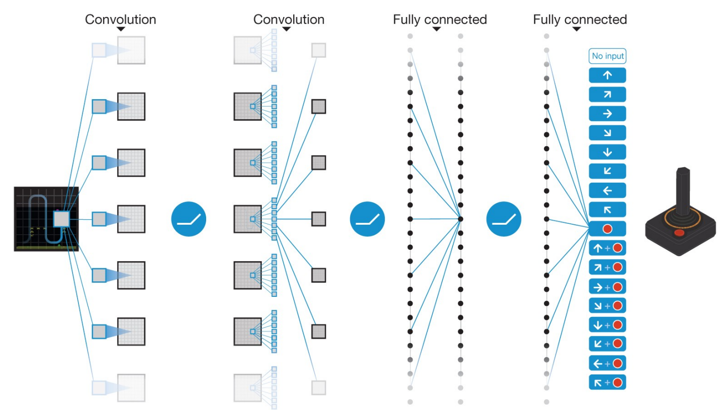
- EvoJAX allows scaling NE across GPUs with parallel fitness evaluations.
- Demonstrated effectiveness in training large neural networks.
- Significant speedup and scalability compared to traditional CPU-based NE approaches.
- Accessible in Colab notebooks!

	Baseline	EvoJAX
MNIST	36 min	3 min
Cart-Pole Swing Up (Hard Version)	37 min	2 min
Locomotion (Ant)[1]	201 min	9 min

(Tang et al. 2022)

An Alternative History of NE

- Imagine if DeepMind had used a GA instead of RL for their Atari breakthrough.
- How would the trajectory of AI research have changed?
- Highlights the untapped potential of neuroevolution in large-scale tasks.



(Mnih et al. 2015)

Conclusion: Reinforcement Learning and Neuroevolution

- **Differences:**
  - RL uses gradient-based optimization and learns through trial-and-error in an environment.
  - NE is a gradient-free, population-based method that explores the policy space using evolutionary processes.
- **Synergistic Combinations:**
  - Hybrid methods such as ERL, NEAT+Q, (ES-)MAML(-Baldwin) combine NE's exploration power with RL's gradient-based finetuning.
  - NE can help overcome RL's issues with sparse rewards and long time horizons.

Conclusion: Reinforcement Learning and Neuroevolution

- **Successes:**
  - ES and GAs scaled to thousands of CPUs, solving complex tasks like 3D humanoid locomotion in minutes.
  - NE demonstrated competitive results with RL in Atari games, with GA achieving high scores in games like Frostbite.
  - Evolutionary Neural Turing Machines (ENTMs) showed promising performance in continual learning tasks.
- **Future Opportunities:**
  - Hybridizing NE and RL to achieve robust exploration and efficient exploitation.
  - Scaling indirect encodings (e.g., HyperNEAT) to tackle more complex tasks.
  - Leveraging hardware acceleration (e.g., JAX, GPUs/TPUs) for more scalable NE solutions.
  - Exploring open-ended evolution for continuous, autonomous learning.