Bachelorthesis

Investigating Neuroevolution of Ternary Neural Networks as an Alternative to Reinforcement Learning

Frederic Thoma

Würzburg, October 30, 2025

Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik IX - Games Engineering
Prüfer: Prof. Dr. Sebastian von Mammen
Betreuer: Johannes Büttner

Contents

Chapter 1

Introduction / Motivation

AI systems are increasingly constrained by energy, latency, and deployment cost rather than marginal gains on benchmarks. Ternary neural networks (TNNs) with weights in {-1, 0, +1} address these constraints by shrinking model size, reducing memory traffic, and replacing many floating-point multiplies with cheaper operations. However, training TNNs is difficult: ternarization is non-differentiable, yielding zero gradients almost everywhere and undefined values at thresholds. Neuroevolution, e.g., evolutionary strategies (ES) and genetic algorithms (GA), offers a gradient-free alternative that searches directly in discrete weight spaces. This makes training TNNs with ES/GA a seemingly natural fit. Moreover, the discretized parameterization induces a dramatically smaller search space than full precision, which may accelerate exploration and improve the chances of finding competent solutions under tight evaluation budgets.

This thesis asks whether TNNs evolved with ES/GA can match full-precision neural networks (FPNNs) evolved with the same ES/GA procedures under equal budgets. We further ask whether TNNs under neuroevolutionary training can solve the same suite of tasks that FPNNs can under identical conditions or whether there exist tasks for which ternarization imposes a fundamental performance ceiling. We evaluate policy/value networks on a curated set of Gymnasium environments spanning discrete and continuous actions of varying difficulty. We compare performance (average return, success rate), sample efficiency (area under the learning curve, steps-to-threshold), stability across seeds, and model size (number of parameters). In addition to neuroevolutionary comparisons, we include a strong reinforcement learning baseline Proximal Policy Optimization (PPO) as a reference for task solvability and attainable performance under dense-gradient training. Our goal is to quantify the accuracy–efficiency trade-off of TNNs and identify regimes by task, sparsity, and evolutionary hyperparameters where they outperform, match, or lag full-precision counterparts.

Chapter 2

Related Work

Evolution Strategies as a Scalable Alternative to Reinforcement Learning: Salimans, Ho, Chen, *et al.* [1] show that simple, black-box evolution strategies using population perturbations and communication-light updates can match policy-gradient RL while scaling efficiently across massive parallel compute. This directly motivates our use of ES/GA for non-differentiable ternary networks and our head-to-head comparisons with full-precision models and PPO under matched evaluation budgets.

Ternary Neural Networks for Resource-Efficient AI: Alemdar, Leroy, Prost-Boucle, *et al.* [2] propose TNNs with weights/activations in {-1, 0, +1}, trained via a teacher–student, layer-wise method, and show that they remove multiplications and improve energy efficiency with competitive accuracy. For our work, it grounds the case for TNNs' memory/compute savings that we aim to exploit, while we swap in ES/GA as a gradient-free training route to handle ternarization and then compare against full-precision models under equal budgets.

Gymnasium: A Standard Interface for Reinforcement Learning Environments: Gymnasium [3] provides a lightweight, widely adopted API for defining observation/ action spaces and reproducible seeding across diverse tasks. Using Gymnasium ensures that TNNs and FPNNs are evaluated under identical conditions and its monitoring utilities make runs easily comparable.

Genetic Algorithms for NN Training: Genetic algorithms directly optimize feedforward network weights and can outperform backpropagation on difficult tasks in certain settings [4]. Here, GA serves as a discrete, gradient-free baseline that naturally aligns with ternary weight search.

Population-Based Incremental Learning (PBIL): PBIL replaces explicit crossover with

an evolving probability vector and often performs strongly on diverse optimization problems [5]. It yields a simple, stable GA-style optimizer for large parameter spaces such as neural network policies.

Cross-Entropy Method (CEM): CEM updates a sampling distribution by moving it toward elite samples, effectively minimizing divergence to concentrate probability mass on high-performing regions [6]. We use CEM as a strong black-box baseline that scales well and is widely applied in policy search.

CoSyNE: Cooperative Synapse Neuroevolution coevolves synapses rather than whole networks, forming teams of weight-vectors that are repeatedly recombined via permutation and selection to efficiently search high-dimensional parameter spaces [7]. For TNNS, CoSyNE's synapse-level credit assignment and permutation-based mixing make ternary weight search modular and robust—differing from other black-box algs by explicitly decomposing the network and exploiting cooperative coevolution to preserve useful building blocks while exploring aggressively.

Chapter 3

Methodology

3.1 PPO-Trained Baseline Network

To contextualize the neuroevolutionary results, we train a strong reinforcement learning baseline using Proximal Policy Optimization (PPO). Unlike the TNNs, this baseline uses full-precision weights and dense-gradient updates, providing a reference for task solvability and attainable performance under gradient-based learning.

3.2 Neuroevolution Algorithms

We train both the TNNs and the FPNNs using gradient-free Evolutionary Strategies and Genetic Algorithms. For context on task solvability and attainable performance, we compare both models against the previously described PPO-trained baseline. These are the algorithms that we use:

- **Simple Genetic Algorithm (SGA)** (population-based, direct search): evolve a pool of networks via selection, crossover, and mutation; uses only fitness signals (no gradients), making it well-suited to discrete/quantized weights.
- **Population-Based Incremental Learning (PBIL)** (estimation-of-distribution): maintain a probability vector over parameters, sample candidate networks, then shift the vector toward the best performers; simple, low-variance updates and few hyperparameters.
- **Cross-Entropy Method (CEM)** (elite-driven distribution update): iteratively sample from a parametric distribution (e.g., Gaussian/Bernoulli), keep the elite set, and refit the mean (and optionally covariance); fast, stable convergence on rugged fitness landscapes.

- **CoSyNE** (cooperative coevolution, synapse-level recombination): decompose the network into per-synapse weight vectors that are evolved in parallel and repeatedly permuted into full networks; preserves useful building blocks and improves credit assignment; well-suited to TNNS since ternary synapses can be mixed and selected modularly without gradients, enabling efficient search in high-dimensional discrete spaces.

3.3 Benchmarks / Environments

We evaluate on four standard Gymnasium control benchmarks of increasing difficulty:

1. **CartPole:** discrete, low-dimensional state; short horizon.
2. **LunarLander:** discrete, more complex dynamics, longer horizon.
3. **CarRacing:** pixel observations with continuous control; long horizon.
4. **BipedalWalker:** continuous multi-joint control with contacts; long horizon.

Each experiment will be repeated five times, and we will report the average performance and the standard deviation. If time and scope permit, we will also test for significant differences.

Chapter 4

Work Packages

- **WP1: Create TNN and FPNN in PyTorch**
  Description: Implement both the FPNN and the TNN in PyTorch; add minimal tests to confirm basic functionality.
  Dependency: Project setup (repo, environment, etc.)
  Result artifact: PyTorch implementations of FPNN and TNN with unit tests.
  Time requirement: 2 weeks

- **WP2: Implementations for all ES/GA algorithms**
  Description: Find implementations for all chosen ES/GA algorithms. If there are no implementations, implement them. Adjust algorithms for compatibility with TNNs if necessary.
  Dependency: WP1
  Result artifact: Implementation for all algorithms, ready to be used for training both FPNN and TNN
  Time requirement: 2 weeks

- **WP3: Train and evaluate FPNN with RL (PPO)**
  Description: Train a strong reinforcement learning baseline using Proximal Policy Optimization. Benchmark its performance on chosen Gymnasium environments.
  Dependency: WP1
  Result artifact: Bechmark results of FPNN trained with PPO on chosen Gymnasium Tasks
  Time requirement: 1 week

- **WP4: Train and evaluate FPNN with selected evolutionary algorithms**
  Description: Train the FPNN using each of the chosen evolutionary algorithms. Evaluate how the trained FPNN performs on each selected Gymnasium environment for each algorithm.

Dependency: WP1, WP2
Result artifact: Benchmark results of FPNN under neuroevolutionary training
Time requirement: 2 weeks

- **WP5: Train and evaluate TNN with selected evolutionary algorithms**
  Description: Train the TNN using each of the chosen evolutionary algorithms. Evaluate how the trained TNN performs on each selected Gymnasium environment for each algorithm.
  Dependency: WP1, WP2
  Result artifact: Benchmark results of TNN under neuroevolutionary training
  Time requirement: 2 weeks

- **WP6: Collect and analyze benchmark results**
  Description: Consolidate all benchmark data (e.g., into a spreadsheet or database), compute summary statistics, and create comparison plots/diagrams.
  Dependency: WP1 - WP5
  Result artifact: Cleaned dataset, figures (plots/tables), and a short technical summary of findings.
  Time requirement: 1 weeks

Declaration of originality

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Würzburg, October 30, 2025 Frederic Thoma

Bibliography

[1] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017. DOI: 10.48550/arXiv.1703.03864. arXiv: 1703.03864 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1703.03864 (visited on 10/24/2025).

[2] H. Alemdar, V. Leroy, A. Prost-Boucle, and F. Pétrot, "Ternary neural networks for resource-efficient ai applications," *arXiv preprint arXiv:1609.00222*, 2016. DOI: 10.48550/arXiv.1609.00222. arXiv: 1609.00222 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1609.00222 (visited on 10/24/2025).

[3] M. Towers, A. Kwiatkowski, J. Terry, *et al.*, "Gymnasium: A standard interface for reinforcement learning environments," *arXiv preprint arXiv:2407.17032*, 2024. DOI: 10.48550/arXiv.2407.17032. arXiv: 2407.17032 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2407.17032 (visited on 10/24/2025).

[4] D. J. Montana and L. Davis, "Training feedforward neural networks using genetic algorithms," in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, Detroit, MI, USA: Morgan Kaufmann, 1989, pp. 762–767. [Online]. Available: https://www.ijcai.org/Proceedings/89-1/Papers/122.pdf (visited on 10/24/2025).

[5] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-CS-94-163, Jun. 1994. [Online]. Available: https://www.ri.cmu.edu/publications/population-based-incremental-learning-a-method-for-integrating-genetic-search-based-function-optimization-and-competitive-learning/ (visited on 10/24/2025).

[6] R. Y. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Methodology and Computing in Applied Probability*, vol. 1, no. 2, pp. 127–190, 1999. DOI: 10.1023/A:1010091220143. [Online]. Available: https://link.springer.com/article/10.1023/A:1010091220143 (visited on 10/24/2025).

[7] F. Gomez, J. Schmidhuber, and R. Miikkulainen, "Accelerated neural evolution through cooperatively coevolved synapses," *Journal of Machine Learning Research*, vol. 9, pp. 937–965, 2008. [Online]. Available: `https://jmlr.org/papers/v9/gomez08a.html`.