The Attention Seekers

Bit-Pedal Walker

Biswajit Banerjee
sumon@gatech.edu

Dennis Frank
dfrank6@gatech.edu

Leyao Huang
lhuang395@gatech.edu

Abstract

In this study, we present a novel approach to evolving neural network architectures for control tasks by combining neuroevolution techniques with aggressive weight quantization. Specifically, we employ the NeuroEvolution of Augmenting Topologies (NEAT) algorithm to search for effective network topologies, while constraining the weights to a minimal 1.58-bit {-1, 0, 1} representation. Our key insight is that the network architecture itself can encode inductive biases well-suited for a given task, while ultra-low bit quantization allows highly compact and efficient models.

We evaluate our approach on the challenging BipedalWalker-v3 environment from OpenAI Gym, where an agent must learn to control a bipedal robot to navigate varied terrains. Through empirical analysis, we demonstrate that our 1.58-bit quantized NEAT models achieve competitive performance compared to traditional weight-agnostic neural networks, while requiring a drastically smaller memory footprint. Notably, the learned architectures exhibit emergent behaviors and strategies for robust locomotion. Our work highlights the potential of combining neuroevolution with severe quantization as a promising direction for developing resource-efficient solutions to complex control problems.

1. Introduction

In recent years, the intersection of Evolutionary Strategies (ES) with Deep Learning (DL) has significantly interested and prompted exploration within the field of Artificial Intelligence (AI). This has resulted in the emergence of Evolutionary Deep Learning (EDL) [6], a novel paradigm that diverges from traditional gradient-based approaches and draws inspiration from nature.

The current field of AI is dominated by gradient based optimization but ES is also an important branch that is efficient in optimizing complex problems. In general, ES imitates the evolution rule of "survival of the fittest" from nature to evolve candidate solutions to obtain the most optimal result. Generally speaking, ES algorithms mainly include swarm intelligence (SI), genetic algorithm (GA), differential evolution (DE), estimation of distribution algorithms (EDA) and many more. In the AI community, as DL and ES are respectively the efficient learning and optimization technologies, their developments are tightly integrated and greatly benefit from each other.

ES and DL primarily diverge at how they are optimized. In case of DL we have one model which is optimized through back-propagation over epochs whereas in ES many models co-evolve across generations to create newer, more advanced models. The issue with ES is scalability as since there is a huge population of models, the memory footprint explodes with model complexity.

Drawing on ES, Adam Gaier and David Ha have demonstrated the efficacy of evolving shared-weight scheme Weight Agnostic Neural Networks (WANN) [2] across various applications, including the Bipedal Walker challenge. The concept of WANNs suggests that the network architecture can directly encode task-specific solutions, as facilitated by evolutionary algorithms. In this ambitious undertaking, we re-imagine WANNs in a more memory efficient way.

Our project will thus focus on a minimal architecture space characterized by 1.58-bit non-shared weights, inspired by the recent advancements in 1.58-bit LLMs by Shuming Ma *et al*. [3]. The use of

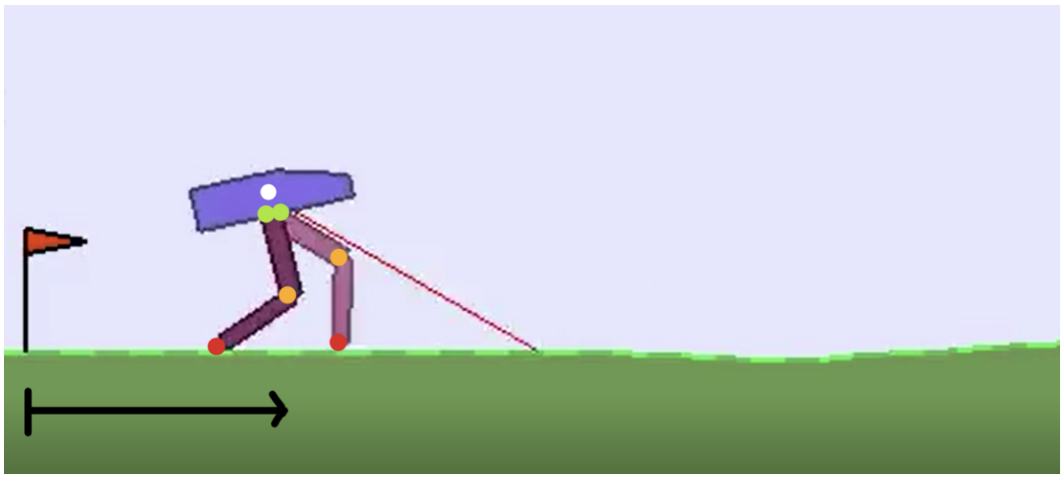

Figure 1. Simulation of a Bipedal Walker in the BipedalWalker-v3 environment.

1- or 1.58-bit LLMs has been shown to deliver competitive performance compared to traditional floating-point weights, while significantly reducing memory and energy requirements.

1.1. Related Work

- **Proximal Policy Optimization (PPO)**: PPO aims to improve the stability and efficiency of the policy gradient by limiting the size of policy updates, which is helpful for maintaining stable training progress in complex environments [5].

- **Augmented Random Search (ARS)**: ARS is known for its simplicity and effectiveness, particularly suitable for continuous action domains. It adjusts policy parameters directly, assessing impacts to enhance performance without computing gradients, thus reducing computational overhead and simplifying training [4]. This method is especially effective for locomotion tasks such as `BipedalWalker-v3`.

- **Deep Q-Networks (DQN)**: Although typically used for discrete action spaces, modifications of DQN have been applied to continuous environments through action discretization. This method uses a deep neural network to approximate the Q-value function, learning optimal policies based on the maximization of expected rewards.

2. Approach

2.1. Environment

In order to evolve our networks, we needed an environment to train and test them on. We chose the OpenAI Gym's `BipedalWalker-v3` environment as featured in [2] so that we can provide direct comparison of our models. The OpenAI Gym simulates a Bipedal Walker comprising five linkages, as shown in the Figure 3. The Walker's state is represented by a 24-element space vector encompassing various parameters: angular position θ, horizontal and vertical velocities $(\hat{x}, \hat{y})$, hull angular velocities ω, angular positions η and velocities ψ of each joint (colored in green and orange in Figure 3), leg contact states with the ground γ (colored in red), and 10 LIDAR rangefinder measurements λ (shown as a red line) that detect the surface ahead [1]. The control actions available to the Walker are encapsulated in a four-element vector that specifies the torque commands for the motors at each of the four joints.

In this project, we primarily employ the standard configuration of the `BipedalWalker-v3` environment, characterized by terrains with both uphill and downhill slopes. We reserve the more challenging hardcore variant, which features obstacles and gaps, exclusively for testing purposes. Also, we restrained our simulation to 1600 frames.

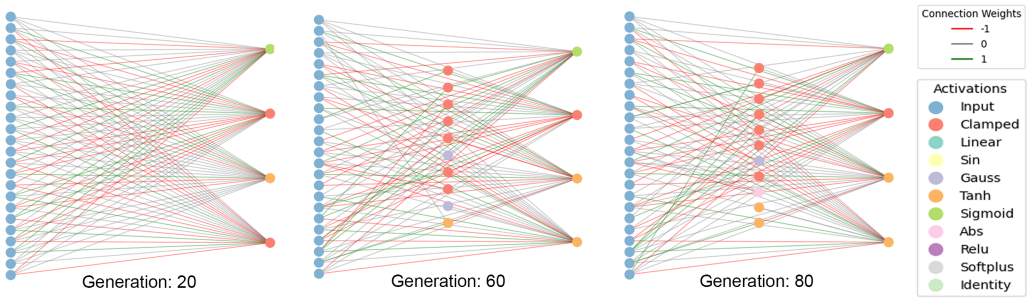

Figure 2. Evolution of Neural Network architecture over generation

2.2. NEAT

Inspired by the Weight Agnostic Neural Networks (WANN) approach proposed by Gaier and Ha [2], we developed a framework to instantiate and evolve feed-forward neural network architectures using the NeuroEvolution of Augmenting Topologies (NEAT) algorithm. We initialized our networks with a minimal topology consisting of the 24 observation nodes connected to the 4 action nodes and incrementally increased the architectural complexity of populations as governed by a configuration file with custom-specified parameters. Notably, the default activation function was set to the clamped variant, which could mutate to other non-linear functions such as linear, sine, Gaussian, tanh, sigmoid, abs, and ReLU, with a mutation probability of 0.4. This diversity in activation functions facilitated the exploration of various non-linear transformations, thereby enhancing the network's capability to adapt to intricate control tasks. Furthermore, the mutation settings for network connections were strategically configured to favor increasingly complex architectures, with probabilities of 0.6 and 0.4 assigned to adding and deleting nodes/connections, respectively. The species differentiation mechanism, a crucial aspect of NEAT, was driven by a compatibility threshold of 3.0, which promoted genetic diversity by clustering similar topologies into distinct species during the reproduction phase. Our implementation draws substantial motivation from the NEAT-Python repository (neat-python), serving as a foundation upon which we built our extensions.

2.3. Reinforcement Learning

OpenAI Gym Bipedal Walker and NEAT were then combined in a reinforcement learning framework such that a network genome would predict the next action and the Bipedal Walker environment would perform a step with it. This could be generated iteratively until success of the simulation defined as reaching frame 1600 or failure as defined as the head of the Walker hitting the ground.

2.4. Fitness

The fitness function is the criterion that evaluates the performance of the model in the environment and thus how fit it is to contribute to the next generation. We used NEAT sampling to create the next generation by selecting two well performing models (network genomes) and crossing them with the majority of nodes (genes) being taken from most fit genome. As we can not back propagate through this crossing example, we created our own class for creating the next generation.

Softmax Reproduction: An important variant we explored is where we compute the softmax of the fitness of all genomes from one generation to perform a weighted sampling of a genome contributing its genes to the next generation (including 5% probability of random mutation). In this way we ensure the global population converges to a minima without the overall fitness curve being stochastic.

2.5. Quantization

Neural Network weight quantization is a useful operation as it can provide similar performance to non-quantized networks while minimizing memory footprint and vastly reducing the number of floating point operations. We applied the following methods to quantize our application and compare the results.

2.5.1 Integer Quantization

Integer quantization is just rounding off the values to their nearest integer equivalent value. Specifically, we tried two approaches:

Post-training quantization: We first train the generation using floating point weight and bias values and converge to a minima where the network performs really well (*i.e.* total reward 250+). Post completion, we then round off the network weights and biases to integers and reevaluate their performance from which we observed around 17% performance drop.

Intra-training quantization: We created a Genome class (IntGenome) that during training samples weights from a Gaussian distribution and rounds them to the nearest integer value. Using this process we were able to come close to the floating point level of performance (with an average fitness of $\pm 10\%$) with relatively less memory footprint.

2.5.2 Bit Quantization

The BitNet has demonstrated its capabilities not only on small architectures but even on Large Language models (LLM) as demonstrated by Shuming Ma *et al.* [3]. We aim to follow in their footsteps and implement WANNs in a non-weight sharing manner, where weights will be constrained to a distribution of {-1, 0, 1}.

Post-training quantization: As previous discussed in the Integer Quantization section, we employ post-training quantization strategy, but this time to {-1, 0, 1}, to observe network performance. We see a major drop (as expected) where the Walker was able to continue to walk but not effectively. Hence we see the point made by the WANNs that the architecture encodes some innate behaviours.

Intra-training quantization: We also created a Genome class (BitGenome) that during training samples connection weights from a discrete distribution of {-1, 0, 1}. We also chose to set the bias of nodes to 0 as to we didn't want bias to negate the effect applied by connection weights to the inputs. This is our flagship model.

2.6. Hyper-parameters

In addition to these strategies to meet our evolutionary objective, we also performed a hyper-parameter search on two relevant parameters to further refine the performance of our model. Namely, we test survival threshold of 1, 0.5, or 0.1, and beta threshold for credit assignment of 0.005, 0.03, and 0.09.

3. Experimental Setup and Results

3.1. Setup

We first trained 200 networks over 300 generations with architecture evolving and performance improving with each successive generation. We initially tuned hyper-parameters for the Default Genome but kept them consistent for and only modified the required parameters for the other models. This assures us consistent results and a fair comparison.

To evaluate the models, we set up a visual Bipedal Walker environment and tested our different approaches. Figure 3 shows the fitness of an example network across generations. The reporting panel on top right of each image shows the generation, followed by the total reward (fitness) it accumulated over the number of steps mentioned below. We render the whole walk and display the last state. Our supplementary material contains the videos of their different walking styles.

3.2. Performance

We monitored the fitness which is average accumulated reward for each of those network over 5 scenarios, where each scenario is a randomly generated environment. Image 4 shows the fitness increasing over generation for three different approaches using combinations of the following configurations:

Genome: The genome defines the network architecture. The blue line represents default genomes which takes in any value int or float as weight and bias. It acts as our reference to how default algorithm learns. The green and orange line represents the Bit Quantize genomes which have weight restriction to be any of {-1, 0, 1} values and bias as zero.

Reproduction: The reproduction defines how the networks produce the next generation. The blue and green line represents default reproduction that takes the two most fit genomes and crosses them to create a new genome. The orange line represents the Softmax sampling method as mentioned above.

As is clearly visible the Bit Quantize genome dom-

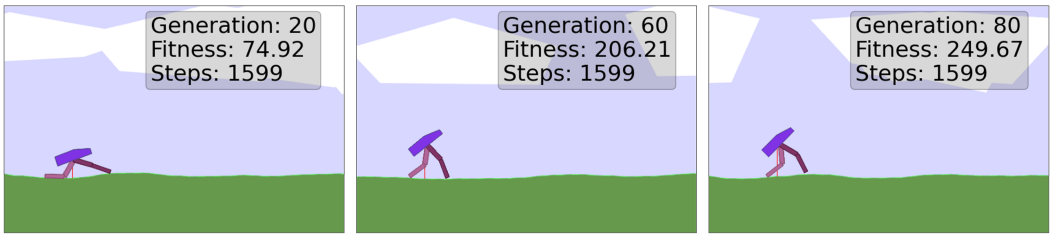

Figure 3. Demonstration of bipedal walker over different generation

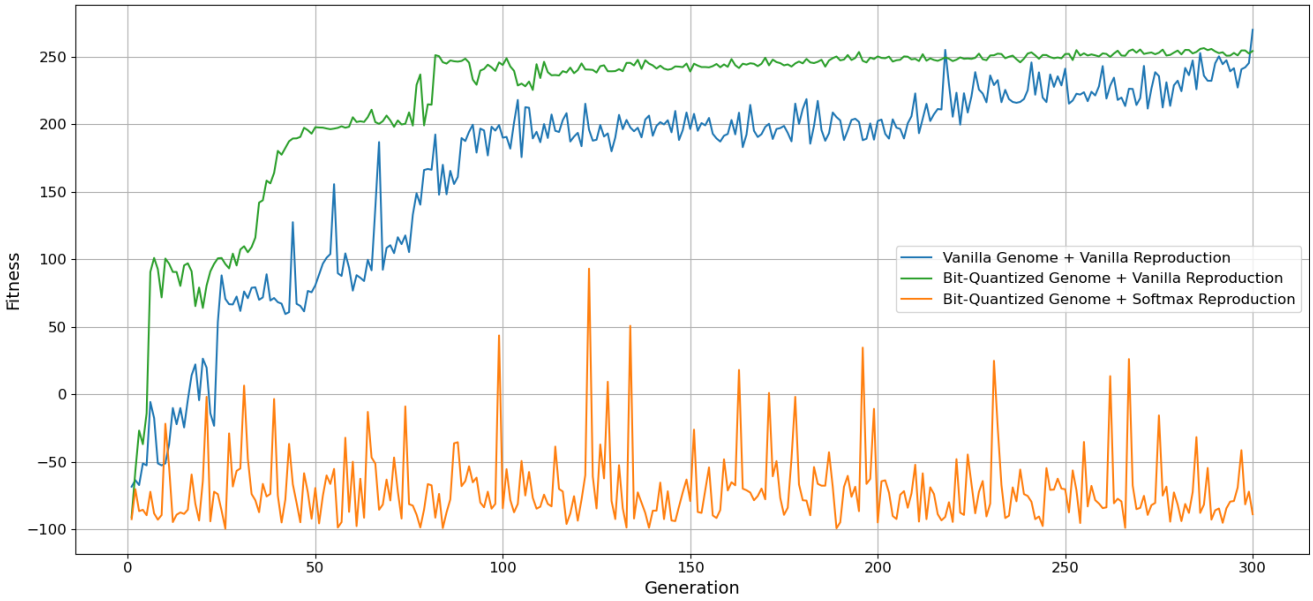

Figure 4. Fitness of Bipedal Walker in normal mode

inates each generation. It not only provides better performance but also our best performing model has the very small approximate size of 22 bytes (which is number of connections * 1.5 bit + activation functions). The orange line looks more or less random which shows that this sampling technique didn't provide any fruitful results. We believe further refining this approach might provide a way to back propagate through the generations and optimize evolution.

3.3. Evaluations

Table 1. Results at 300th Generation

	Nodes	**Connections**	**Fitness**
Default	41	23	270
Bit-Quantized	55	101	254.29
Softmaxed	32	96	-88.75
WANN Paper	102	210	261

We have seen that at every generation the Bit Quantize is ahead in both fitness and network complexity. At the end of the generations, we see a spike for the default genome while the Bit Quantize genome stays saturated. We also compared against the values shared by the WANN paper although we are not sure of how many generations they used for their best evolution. The above metric can be improved more using multiple experiments.

4. Discussion

Although table 1 is one example run, we have seen through multiple tests that this trend is consistent. We have packaged our implementation of the bit-neat approach with multiple notebooks that will help reproduce the above results.

Although Softmax sampling performed poorly, we think it was important to test this variation as it provided differentiable parameters to the evolutionary training. However, the method itself turned out to pro-

vide too chaotic of architectures to provide meaningful usage. We believe further optimization of this approach could yield fruitful results.

We also evaluated the models in hardcore mode of Bipedal Walker where the environment spawns random hurdles in which we noticed similar performance despite the harsh terrain. Most interesting of all are the number of ways Walkers learned to navigate through the environment as can be seen in the videos of our supplementary material. We also observed some emergent behaviours where the network that evolved in flat terrain tried to jump over hurdles in hardcore mode.

4.1. Challenges

Exploring the search space of quantized architectures is non-trivial due to the discretized weight space of -1, 0, 1. This could make the optimization landscape more rugged and difficult to navigate effectively. Also, quantized networks may get trapped in poor local optima more easily compared to higher precision models. Maintaining population diversity is critical. We faced these challenges several times. Increasing number of scenarios proved to be helpful to get the evolution out of local optimas.

As generations evolve, we see helpful genes propagated properly and trivial ones getting killed. But an unfortunate downside is that sometimes useful genes might go extinct due to the nature of the algorithm. Thus, understanding the inductive biases encoded by the evolved quantized architectures and how they facilitate effective behavior is also non-trivial. Providing theoretical guarantees on the performance and convergence properties of quantized neuroevolution remains an open challenge.

5. Conclusion

This study presents the implementation of a quantified NEAT (NeuroEvolution of Augmenting Topologies) approach for the control of a Bipedal Walker within a simulated environment. Our methodology demonstrated swift and effective learning outcomes for this specific task. Considering the intricacies inherent in genetic algorithms, this approach has the potential to significantly expedite operations such as selection, crossover, and mutation, especially when handling extensive populations or intricate genomes. Such capabilities offer promising prospects for deployment in scalable and memory-efficient real-world applications.

References

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. 2

[2] Adam Gaier and David Ha. Weight agnostic neural networks. *Advances in neural information processing systems*, 32, 2019. 1, 2, 3

[3] Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024. 1, 4

[4] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. *CoRR*, abs/1803.07055, 2018. 2

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. 2

[6] Zhi-Hui Zhan, Jian-Yu Li, and Jun Zhang. Evolutionary deep learning: A survey. *Neurocomputing*, 483:42–58, 2022. 1