applied sciences

MDPI

Article

Teaching a Real Biped to Walk with Neuro-Evolution After Making Tests and Comparisons on Simulated 2D Walkers

Roland Szabo

Faculty of Electronics, Telecommunications and Information Technologies, Politehnica University Timisoara, Vasile Parvan Av., No. 2, 300223 Timisoara, Romania; roland.szabo@upt.ro; Tel.: +40-256-40-3351

Abstract

The aim of this paper is to test and compare different neuro-evolution methods to train a simulated biped walker to learn to walk. After this step, the best neuro-evolution technique is ported to a real biped, which would train itself to walk. The goal is to reduce the number of falls for the real biped in order to avoid destroying the physical unit. The following four neuro-evolution methods were tested: Deep Q-Learning (DQN), NeuroEvolution of Augmenting Topologies (NEAT), Deep Deterministic Policy Gradients (DDPG), and Augmented Random Search (ARS). The best results from simulations were obtained with the ARS method, but the fastest and easiest to implement on the real biped was the NEAT algorithm.

Keywords: artificial intelligence; artificial neural networks; evolutionary computation; genetic algorithms; machine learning

1. Introduction

In recent years, the domain of bipedal robotics has made remarkable progress, moving from static balance in controlled settings to dynamic, resilient movement across unpredictable surfaces. This paper compiles recent significant studies, categorizing them into advanced control strategies, motion planning, learning-based methods, stability frameworks, innovative mechanism designs, and human-oriented robotic systems.

A prominent framework for controlling the complex dynamics involved in bipedal movement is Model Predictive Control (MPC). The focus of recent research has been on improving MPCs capacity to handle intricate situations and finish body motions. As an example, Yu et al. suggested a hierarchical MPC framework that effectively combined stepping and rolling stability to enable whole-body movement in wheeled biped robots [1]. Likewise, Dallard et al. developed a closed-loop MPC approach that directly optimizes control inputs, improving walking performance and doing away with the need for conventional stabilizers [2]. Li and Nguyen developed an adaptive-frequency MPC system that allows robots to successfully navigate uneven stepping stones [3]. Choe et al. [4] used nonlinear MPC to implement real-time adaptation strategies, which increased reactive capabilities. To strengthen resilience to disruptions, Gu et al. incorporated Signal Temporal Logic (STL) to effectively handle logical conditions and temporal constraints within the MPC framework [5]. Moreover, Daneshmand et al. emphasized the significance of adding swing foot dynamics to variable horizon MPC in order to improve the accuracy of foot placement [6]. Lastly, in multi-agent contexts, Shamsah et al. used Social Zonotope Network MPC to promote socially aware navigation for bipeds in crowded settings [7].

Check for updates

Academic Editor: Grigorios Beligiannis

Received: 20 November 2025
Revised: 17 December 2025
Accepted: 21 December 2025
Published: 30 March 2026

Copyright: © 2026 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Although model-based control retains its effectiveness, data-driven methods, especially Reinforcement Learning (RL) and neural networks, are addressing challenges that are challenging to model analytically. Challa et al. illustrated the usefulness of Long Short-Term Memory (LSTM) networks combined with RGB-D sensors to create human-like walking trajectories [8]. In the area of robust control, Beranek et al. introduced a behavior-based RL strategy to effectively manage unknown external disturbances [9].

Remarkably, complex neural policies are not always essential; Krishna et al. demonstrated that simple linear policies can often suffice for achieving robust walking on difficult terrains when trained well [10]. Addressing the demand for adaptability, Chand et al. concentrated on interactive dynamic walking, formulating techniques to learn gait-switching policies with theoretical generalization assurances [11].

Planning where to step is as significant as controlling limb movements. Efficient planning algorithms have been devised to manage rough and compliant terrain. For underactuated robots negotiating uneven surfaces, Yao et al. introduced a velocity-centered gait planning approach [12]. Hong and Lee used Particle Swarm Optimization (PSO) to enable practical footstep planning in 3D areas for real-time applications [13].

Planning has made use of sophisticated mathematical methods. To control footstep placement on uneven ground, Acosta and Posa employed perceptive mixed-integer control [14]. By creating capturability-based pattern creation that takes into account different center-of-mass heights, Caron et al. addressed the vertical components of planning [15]. Additionally, Crews and Travers investigated the relationship between efficiency and planning, proposing energy management techniques through astute footstep decisions [16]. Underactuated system stability is still a major problem. Stability criteria and reduced-order models have been the focus of several investigations. The Hybrid-Linear Inverted Pendulum (H-LIP) model was used by Xiong and Ames to create 3D underactuated gaits [17]. Specifically for trajectory tracking control, Park et al. presented a novel stability framework [18].

While Mihalec and Yi focused on controllers capable of controlling foot slip dynamics [19], Pi et al. developed adaptive time-delay balance control [20] to address time delays and slip. Safety-critical control has also advanced; Ahmadi et al. used CVaR (Conditional Value-at-Risk) barrier functions to lower risk while moving [21].

From a geometric perspective, Horn et al. improved virtual constraint design to support variable-incline walking [22], while Akbari Hamed and Ames investigated nonholonomic hybrid zero dynamics (HZD) for stabilizing periodic orbits [23]. Furthermore, Khadiv et al. highlighted the efficacy of step timing adaptation as a robust walking control approach [24].

Bipedal robots' reactive logic and hardware are constantly evolving. A multi-locomotion parallel-legged robot was presented by Wang et al., demonstrating the adaptability of creative mechanism designs [25]. Bio-inspiration continues to have a big impact; Zhang et al. investigated the effects of trunk pitch angle and suspension stresses on biomechanics [26], and Zhao et al. developed a wheeled biped inspired by biological systems [27].

In order to improve reactivity for reactive walking without requiring a lot of processing, Lee et al. looked into event-driven Finite State Machines (FSM) [28]. In order to reduce impact forces on uncertain surfaces, Guadarrama-Olvera et al. used anticipatory foot compliance informed by sensing [29].

With an emphasis on exoskeletons and assistive devices, recent research delves deeply into the relationship between humans and bipedal systems, while Soliman and Ugurlu proposed robust feedback control for self-balancing underactuated exoskeletons [30], Li et al. focused on human-in-the-loop control to sustain stable dynamic walking in wearable exoskeletons [31].

Lee and Rosen created models for energy optimization and recycling in lower limb exoskeletons [32]. Energy efficiency is still critical in these systems. Additionally, Li et al.

used HZD and musculoskeletal models to enable natural multicontact walking in assistive devices [33]. In terms of safety, Zhu and Yi presented knee exoskeleton controllers intended to control unplanned foot slides [34]. In order to standardize the evaluation of these diverse systems, Aller et al. developed novel ideas for benchmarking humanoid robot locomotion that went beyond traditional assessment criteria [35].

The literature reviewed shows an active area expanding the limits of robustness, efficiency, and adaptability [36–39]. The integration of stringent control theories (MPC, HZD) with learning-based approaches and designs centered on human interaction holds promise for a future where bipedal robots and exoskeletons function effortlessly in real-world settings [40–42].

Table 1 shows a comparison of the presented method and the state-of-the-art methodology.

Table 1. Comparison of control methodologies for bipedal locomotion.

Reference	Core Methodology	Main Objective	Adaptation/Learning Method
Current Method	Neuro-Evolution	Learn to walk with evolutionary techniques.	Evolutionary optimization involves refining walking parameters through neuro-evolution over successive generations, incorporating the most optimal offspring into the learning process.
Dallard et al. [2]	Model Predictive Control (MPC)	Stable walking without conventional stabilizers.	Real-time optimization involves the MPC controller persistently recalculating the best actions according to the robot's present condition and a dynamic model.
Challa et al. [8]	Optimized-LSTM (RNN)	Human gait generation of trajectory.	Deep Learning (supervised). The LSTM model is trained using human gait data collected from an RGB-D sensor to produce new trajectories.
Beranek et al. [9]	Reinforcement Learning (RL)	Manage walking in the presence of unforeseen external disruptions.	Behavior-Based Reinforcement Learning involves an agent developing a control strategy through trial and error, interacting with its environment to optimize a reward.

Bipedal robots can learn to walk using neuro-evolution techniques. The neuro-evolution algorithms can solve the problem of the ideal movement angles of each joint of the leg of the bipeds in order to walk as naturally as possible and without falling. Practically, they can be applied to teach a biped to walk. The neuro-evolution algorithms can be used to empirically teach a biped to walk instead of figuring out the angles of the joints of the biped to make the robot walk without falling.

2. Problem Formulation

The goal of this work is to present different methods of neuro-evolution. These methods will then be implemented in Python (version 3.15) and used to train a biped to walk. Initially, a biped from a simulated environment will be used. After that, it will

be ported to a real biped robot. The simulated biped environment is the Bipedal Walker from the Box2D environment from OpenAI's Gymnasium toolkit. The goal is to test different neuro-evolution methods in this environment, such as Deep Q-Learning (DQN), NeuroEvolution of Augmenting Topologies (NEAT), Deep Deterministic Policy Gradients (DDPG), and Augmented Random Search (ARS), and then port it to a real biped robot, like BRAT from Lynxmotion.

One good way to teach a biped to walk is to use Reinforcement Learning (RL), because it is a method that uses feedback, rewards, or penalties based on the quality of the action executed. This was the case in which the walking of a biped can be optimized. Also, Reinforcement Learning (RL) is based on neuro-evolution, where neural networks are combined with genetic algorithms, where the new offspring can make a new action on the environment, but an evolved offspring can gradually improve the quality of the action, which gradually can solve many complex problems, in our case the biped to learning walk.

3. Problem Solving

Next, the mathematical background of the biped robot, which learns to walk using neuro-evolution, will be presented. Then, a block diagram of the implementation will be presented. Finally, this chapter will end with practical results, where all four algorithms (DQN, NEAT, DDPG, and ARS) were tested in simulation, and the best resulting algorithm was implemented on the real biped robot.

3.1. Mathematical Background

The Bellman Optimality (Q-Value Function) equation is fundamental in value-based Reinforcement Learning. Calculate the value of performing a particular action (a) in a given state (s), considering not only the immediate reward, but also future prospects. According to the equation, the quality of an action is the sum of the immediate reward $R(s, a)$ obtained now and the highest possible value from the subsequent state (s'), adjusted by a factor known as Gamma (γ). The agent's degree of patience is influenced by the gamma; a gamma close to 0 makes the agent more concerned with short-term profits, while a gamma close to 1 motivates it to pursue long-term goals.

The Q-Value Function (Bellman Optimality) is as follows:

$$Q(s, a) = R(s, a) + \gamma \max_{a'} Q(s', a') \tag{1}$$

The surprise, or the difference between what is anticipated and what actually happens, is measured by this formula. The first part of equation ($R + \gamma \max Q \ldots$) represents the target—the result that occurred (the reward obtained plus the best estimate of the future). The agent expected to occur is shown in the latter section $Q(s, a)$. The difference between these two values is known as the Temporal Difference (TD). The agent has a perfect grasp of the surroundings if the TD error is zero. A high TD error suggests that the agent made inaccurate predictions, which requires additional learning.

The Temporal Difference (TD) error is given by the following formula:

$$\mathrm{TD}(s, a) = R(s, a) + \gamma \max_{a'} Q(s', a') - Q(s, a) \tag{2}$$

The agent fixes its mistakes in this way and updates the previous belief $Q(s, a)$ using some of the TD Error to move it in the direction of the new reality. The learning rate, α (alpha), determines the size of this section. A low Alpha suggests that the agent is more cautious and learns slowly, whereas a high alpha shows that the agent quickly modifies its views in response to new information. The Q-values eventually coincide with the best course of action through ongoing revisions.

https://doi.org/10.3390/app16073336

The Classical Q-Learning Update Rule is presented as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{3}$$

When transitioning from value-based methods to policy-based ones, this equation establishes the main goal of the agent. The notation $J(\theta)$ represents the score related to the existing policy. The expected value ($\mathbb{E}$) of the cumulative rewards obtained along a trajectory (τ) is the definition of this score. The goal of the training algorithm is to maximize this sum by optimizing the policy parameters (θ), which could be weights in a neural network.

The Expected Reward Objective is shown as follows:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} r(s_t, a_t) \right] \tag{4}$$

The agent is guided by this equation to modify its neural connections in order to increase its calculation of the reward gradient (∇), showing how to maximize rewards in the best way. The formula essentially states: If the total return (G_t) is high, modulate *heta* to increase the likelihood of these actions in the future based on your knowledge. Put differently, it ensures that we focus on the value of the actions that are actually carried out ($\log \pi$). However, they are less likely if the return is low. This eliminates the need to define each Q-value and allows the agent to reinforce what has been successful.

The Gradient of Policy (Reinforcement) is expressed as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t \right] \tag{5}$$

The advantage function contributes to the reduction of variance during training by comparing actions with typical actions in comparable circumstances rather than based on their immediate benefit. The value of the state $V(s)$ and the value of a specific action $Q(s,a)$ are compared to achieve this. A positive result implies that the action is better than average, while a negative result suggests that it is worse than average. With this approach, the agent can focus on relative gains instead of fixed metrics. The method directs the agent to modify its neural connections in order to maximize rewards. When developing the gradient, it is possible to determine the best course to maximize the rewards (∇). In order to increase the likelihood of repeating those actions in the future, the equation recommends reviewing the actions taken ($\log \pi$); if the total return (G_t) is high, then adjust the parameters θ. On the other hand, the likelihood of low returns should be decreased.

The advantage function is shown as follows:

$$A(s,a) = Q(s,a) - V(s) \tag{6}$$

A complex algorithm called the Deep Deterministic Policy Gradient (DDPG) was created to handle the continuous control problems that are frequently seen in robotics. To guarantee steady training, it uses a Target Value formula with multiple networks. DDPG uses a Target Network (Q and μ) rather than extracting the subsequent value from the primary network. These target networks are updated much more gradually and function as static copies of the primary networks. DDPG reduces the chase for its own tail oscillation that frequently impedes deep Reinforcement Learning models by computing the target y_t with these networks that are changing steadily and gradually.

The DDPG Target Value has the following form:

$$y_t = r(s_t, a_t) + \gamma Q'(s_{t+1}, \mu'(s_{t+1})) \tag{7}$$

A method that looks at the configuration of the current set of solutions is the first step in the optimization process. Using common statistical methods, the covariance matrix is calculated to accomplish this. The covariance components, which are obtained by multiplying the deviations of x and y from their respective means, show the distribution and orientation of the solution set with the mean serving as its central point. A relationship between the variables is indicated by a non-zero covariance, which implies that the solution cloud is stretched diagonally and gives the algorithm insight into how the parameters are related to one another.

The Mean Estimates are presented below:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{8}$$

$$\mu_y = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{9}$$

The Covariance Matrix Terms can be seen below:

$$\sigma_x^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)^2 \tag{10}$$

$$\sigma_y^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu_y)^2 \tag{11}$$

$$\sigma_{xy} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y) \tag{12}$$

The CMA-ES algorithm determines how to advance the search distribution for the next generation after the current setup is complete. By averaging only the best-performing elite solutions, it first determines the new mean and directs the search toward the most promising area found so far. The formula for updating covariance is the primary innovation. Instead of using the current generation average, it evaluates the distribution of the new elite solutions in order to determine the new dispersion (or shape) of the search. The formula successfully captures the direction of movement based on the calculation of the beginning point of the steps. This encourages the search distribution to grow and expand along the path of improvement, naturally giving the optimization process momentum.

The New Mean Calculation is performed as follows:

$$\mu_x^{(g+1)} = \frac{1}{N_{best}}\sum_{i=1}^{N_{best}} x_i \tag{13}$$

$$\mu_y^{(g+1)} = \frac{1}{N_{best}}\sum_{i=1}^{N_{best}} y_i \tag{14}$$

The revised covariance calculation uses the mean of current generations $\mu^{(g)}$.

$$\sigma_x^{2,(g+1)} = \frac{1}{N_{best}}\sum_{i=1}^{N_{best}}(x_i - \mu_x^{(g)})^2 \tag{15}$$

$$\sigma_y^{2,(g+1)} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} (y_i - \mu_y^{(g)})^2 \tag{16}$$

$$\sigma_{xy}^{(g+1)} = \frac{1}{N_{best}} \sum_{i=1}^{N_{best}} (x_i - \mu_x^{(g)})(y_i - \mu_y^{(g)}) \tag{17}$$

The CMA-ES algorithm decides how to change the search distribution for the next generation after establishing the current configuration. In order to steer the search center toward the most advantageous region found thus far, it starts by computing a new average and focusing only on the elite solutions that perform the best. The equation used to update covariance is the main area for improvement. Assess the placement of these elite solutions in relation to the average of the prior generation rather than the current one when creating the new search layout. By connecting the computation to the beginning of the step, this equation highlights the movement direction and encourages the search distribution to expand and extend along the path of improvement, which naturally gives the optimization process momentum.

The expected objective function has the following form:

$$J(\theta) = \mathbb{E}_\theta[F(z)] = \int F(z)\ \pi(z,\theta)\ dz \tag{18}$$

Working with an environment (the fitness function) that functions as a black box simulator that cannot be differentiated using traditional calculus is one of the main mathematical challenges in this field. The derivatives use the logarithmic derivative method to address this issue. Using this mathematical method, the gradient operator can be separated from the unknown fitness function and applied to the logarithm of the probability distribution (the policy). The probability distribution, which is typically a Gaussian curve, is a well-known mathematical expression that is simple to differentiate. Gradient descent on complicated problems that are usually non-differentiable is made easier with the help of this technique.

The Gradient Derivation (Log-Derivative Trick) is presented below:

$$\nabla_\theta J(\theta) = \nabla_\theta \int F(z)\ \pi(z,\theta)\ dz \tag{19}$$

$$\nabla_\theta J(\theta) = \int F(z)\ \nabla_\theta \pi(z,\theta)\ dz \tag{20}$$

$$\nabla_\theta J(\theta) = \int F(z)\ \frac{\nabla_\theta \pi(z,\theta)}{\pi(z,\theta)}\ \pi(z,\theta)\ dz \tag{21}$$

$$\nabla_\theta J(\theta) = \int F(z)\ \nabla_\theta \log \pi(z,\theta)\ \pi(z,\theta)\ dz \tag{22}$$

$$\nabla_\theta J(\theta) = \mathbb{E}_\theta[F(z)\ \nabla_\theta \log \pi(z,\theta)] \tag{23}$$

Ultimately, the approximate gradient calculation formula, given by Equation (24), is used to translate the theoretical gradient obtained from the log-derivative trick into a useful update rule. Since an infinite number of samples would be needed to determine the true expected value, the algorithm approximates the gradient by taking the average over a finite sample size (N). Each effect of the solution on the gradient is determined by its fitness score in the weighted sum computed by the formula. Essentially, this guides the algorithm to adjust the parameters more heavily towards choices with high scores and away from those with poor performance, gradually directing the process towards achieving greater rewards.

The approximate gradient calculation is shown below:

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i=1}^{N} F(z^i)\ \nabla_\theta \log \pi(z^i, \theta) \tag{24}$$

3.2. Block Diagram

Several neuro-evolutionary methods were used in order to test which would be the best for training the biped to walk. They were randomly chosen and tested to come to the conclusion of which method would be the best to use. The NEAT was best suited for implementation because it is faster and easier to implement on the real robot.

As shown in Figure 1 for the purpose of creating artificial neural networks (ANNs) for uses such as robotic control, the NEAT method is a neuro-evolutionary approach. In contrast to conventional ANNs that have a set structure, NEAT begins with a collection of simple networks like direct input-output connections and progressively adds neurons and connections over several generations to increase the network complexity. This approach helps to discover new and more complex solutions.

The following are crucial steps that the NEAT algorithm undertakes:

1. **Initialization**: A set of fundamental neural networks is formed, usually consisting of input neurons that are directly linked to output neurons.
2. **Assessment**: Each network is assessed based on its performance in a specific task (e.g., managing a biped robot's walking pattern and balance) and assigned a fitness score.
3. **Speciation**: To prevent new network designs from being dominated by more intricate, and potentially superior, networks, NEAT groups comparable networks into "species". This classification employs a compatibility distance metric that considers both topology and weights.
4. **Selection**: The best-performing networks for each species are selected. This step ensures that individuals from diverse evolutionary backgrounds are selected for reproduction.
5. **Reproduction (Crossover and Mutation)**: Offspring networks are created by the following:
    - **Crossover**: Genetic components (connections and neurons) from two parent networks, potentially from different species, are combined to create a new offspring network.
    - **Mutation**: Offspring networks undergo mutations such as the following:
        - **Weight Perturbation**: Slight modifications to existing connection weights.
        - **Add Connection**: Introduction of a new link between previously unconnected neurons.
        - **Add Neuron**: Splitting an existing link to include an additional neuron in the route.
6. **New Generation**: The reproduced and mutated networks form a new generation that replaces the old one. Species are re-evaluated, and the cycle continues.
7. **Termination**: This process is repeated until a specified termination condition is satisfied, such as achieving a target number of generations or an acceptable fitness level.

This iterative approach enables NEAT to effectively explore both the weight and the topological spaces, often resulting in more sophisticated and resilient solutions compared to fixed-topology neuro-evolution methods.

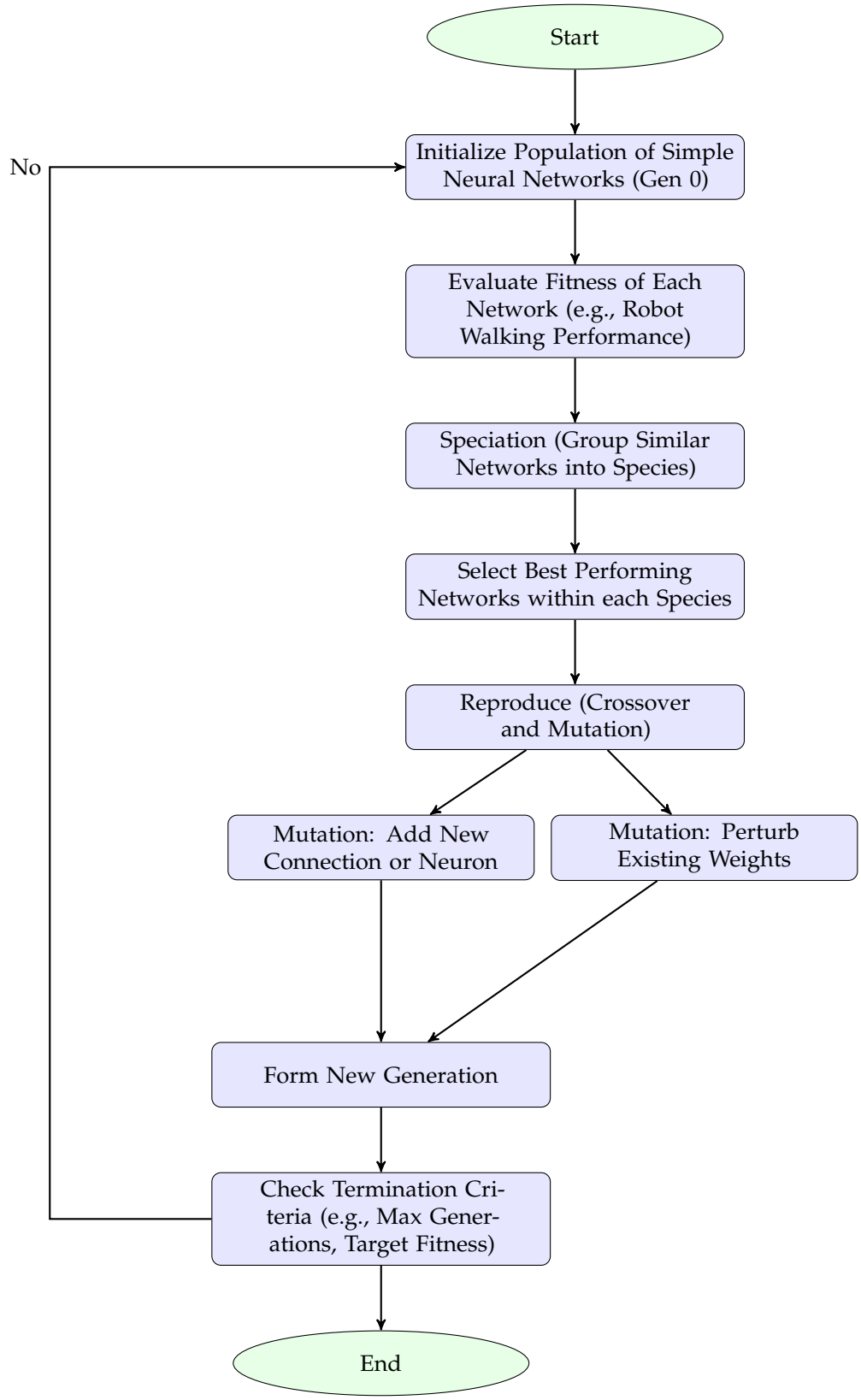


Figure 1. Diagram illustrating the process of the NeuroEvolution of Augmenting Topologies (NEAT) algorithm. NEAT concurrently develops the connection weights and the network's topology, starting with simple networks and gradually enhancing their complexity. Key factors involve speciation to protect novel innovations and unique structural mutation operators.

3.3. Practical Results

For the simulated biped, four implementations were performed: Deep Q-Learning (DQN), NeuroEvolution of Augmenting Topologies (NEAT), Deep Deterministic Policy Gradients (DDPG), and Augmented Random Search (ARS). However, for the real biped, only the NeuroEvolution of Augmenting Topologies (NEAT) method was used because it was simpler to implement and was optimal since, contrary to the simulated biped, the real biped had quite good stability.

For ARS it was trained as a simulated biped for four hours; for DDPG it was trained for six hours on a standard personal computer, which is not a supercomputer. The parameters of the computer used for tests are the following: 16 GM RAM (Random Access Memory), 12th Generation Intel Core i5 with an 8 core @ 2 GHz CPU (Central Processing Unit), 128 MB Intel graphics, and 500 GB Samsung SSD (Solid State Drive). The result was empirical; it was only a visual result that walking training was better (more natural walking) with ARS than with DDPG. For DQN, an acceptable result was not obtained (the robot did not really learn to walk; it fell a lot), so this is why it was not illustrated in the simulation. NEAT was not illustrated also because it had similar results as DDPG, but it was the fastest one and easiest to implement, so that is why it was tested on the real robot.

No quantitative metrics were obtained; only an empirical evaluation was performed. The empirical evaluation was performed as follows: visually, it was observed in the simulation that the biped using the ARS algorithm walked better than with the other algorithms. The NEAT algorithm produced similar results to the ARS algorithm, but was easier to implement on the real robot. For walking, it is difficult to present quantitative metrics; therefore, visual observation is the most practical approach. This includes the quality of walking, whether or not the biped falls, how well it balances, and how straight it walks.

Figure 2 shows the performance of Reinforcement Learning (RL) training. The graph shows a performance tendency of the Q-Learning method used in training the biped.

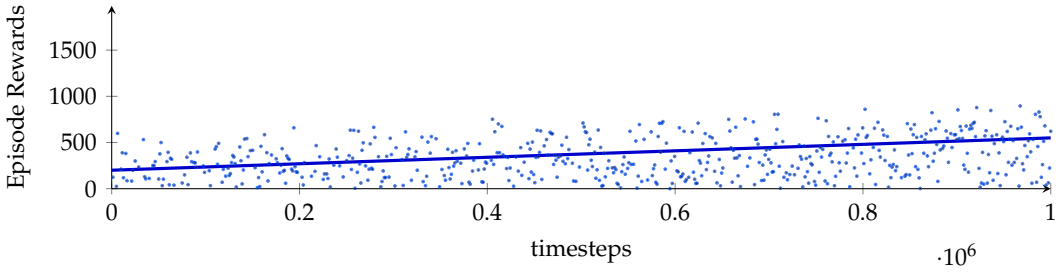


Figure 2. Performance of Reinforcement Learning training – episode rewards in relation to timesteps (smoothed moving average).

Initial tests were performed in a simulated environment to not destroy the real biped due to falling. The simulated biped is the Bipedal Walker from the Box2D environment from OpenAI's Gymnasium toolkit. The Bipedal Walker has 24 inputs and 4 outputs.

The 24 inputs are as follows: hull_angle, hull_angularVelocity, vel_x, vel_y, hip_joint_1_angle, hip_joint_1_speed, knee_joint_1_angle, knee_joint_1_speed, leg_1_ground_contact_flag, hip_joint_2_angle, hip_joint_2_speed, knee_joint_2_angle, knee_joint_2_speed, leg_2_ground_contact_flag, and 10 lidar readings.

The four outputs are as follows: Hip_1 (Torque/Velocity), Knee_1 (Torque/Velocity), Hip_2 (Torque/Velocity), Knee_2 (Torque/Velocity).

Figure 3 shows a trained simulated biped with the DDPG algorithm. Here, it can be seen that the walker did not have the best walking steps and training took six hours.

Figure 4 shows a trained simulated biped with the ARS algorithm. Here, it can be seen that the walker has a more balanced walk and the training took four hours. It can be seen visually that the ARS algorithm has better balance after the biped was trained to walk.

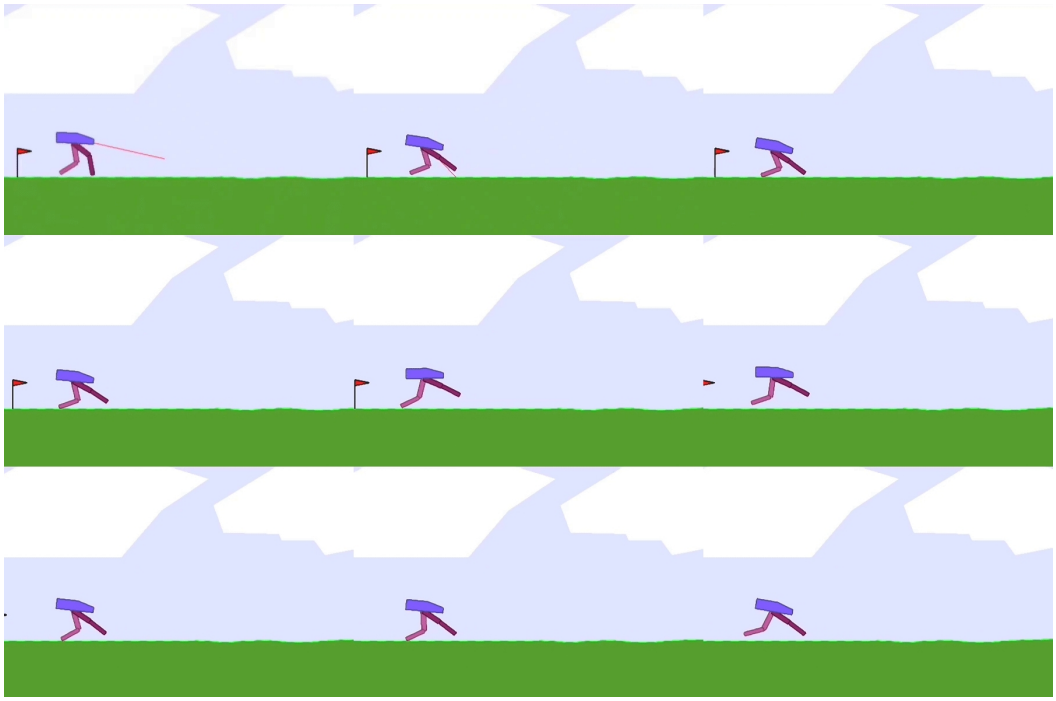


Figure 3. Trained simulated biped with the DDPG algorithm.

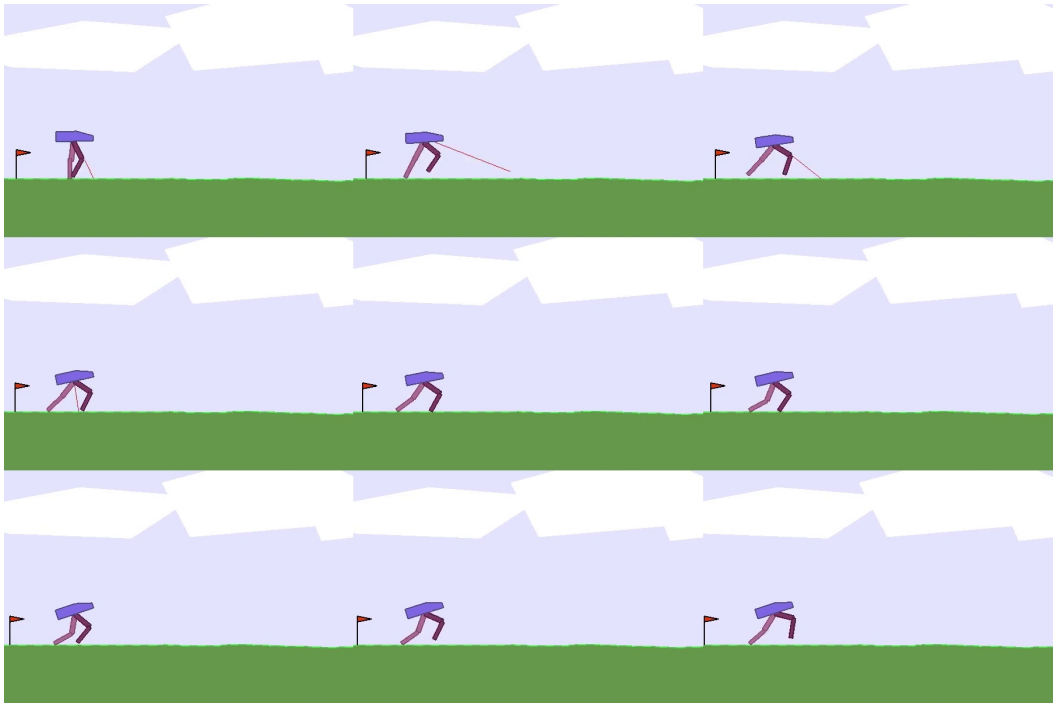


Figure 4. Trained simulated biped with the ARS algorithm.

Some of the parameters of the simulated biped NEAT algorithm are the following: number of nodes = 28, number of hidden layers = 0, number of inputs = 24, number of outputs = 4, number of connections = 96, mutation rate = 0.01.

Figure 5 shows the block diagram of the biped used.

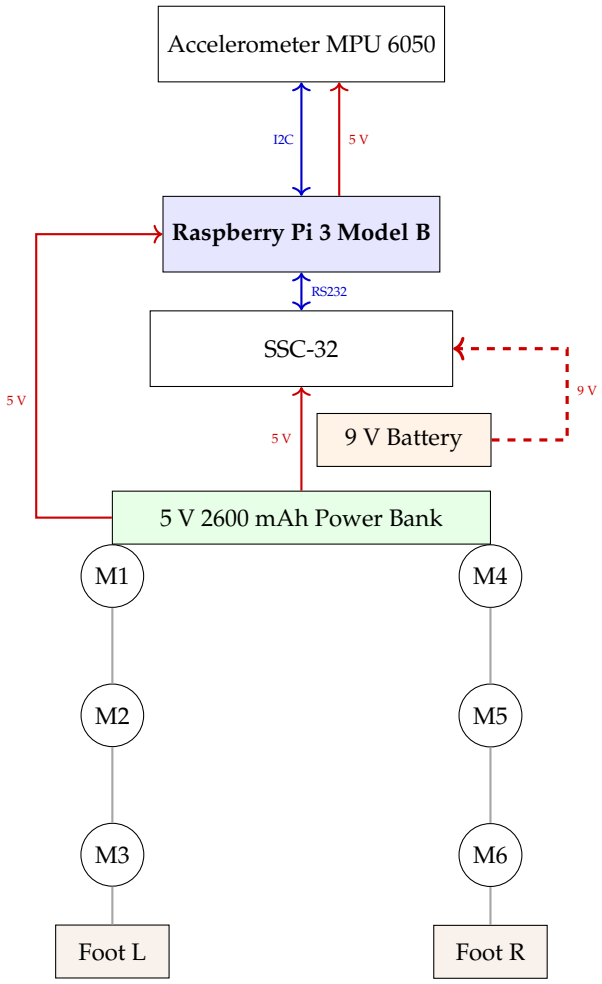


Figure 5. Diagram of the wiring for the biped robot's control system, illustrating the primary electrical and data linkages among the microcontroller (Raspberry Pi), sensor (MPU 6050), motor controller (SSC-32), two power supplies (5 V and 9 V), and six motors (M1, . . .,M6) at the joints of the biped.

The central focus of this project is the Lynxmotion BRAT bipedal robot, which features a minimalist design dedicated to locomotion, as it does not include an upper body, arms, or head. The Lynxmotion BRAT biped has three inputs, accelerometer values for the three axes—Ax, Ay, and Az—and has six outputs. The angles for each joint of the legs are as follows: left hip, left knee, left ankle, right hip, right knee, and right ankle. The robot's structure incorporates six degrees of freedom (DOFs) powered by six Hitec HS-645 servomotors, allocated as two per hip, knee, and ankle, making it well-suited for learning to walk. The internal gears and control electronics are features of the electromechanical Hitec HS-645 servos. They run on a 5 V power supply and are connected by three wires—a black wire for the ground, a red wire for power (V_{CC}), and a yellow wire for the PWM input signal that determines position. The 180° movement span of these motors is mechanically limited.

Figure 6 shows a trained real biped with the NEAT algorithm. Here, it can be seen that the real biped really learned to walk. The NEAT algorithm is not the best, but based on the fact that the biped has quite good balance, this algorithm trained the biped robot quite well.

The BRAT biped robot is manufactured by the Lynxmotion company, which is part of RobotShop, which has its headquarters in Mirabel, Quebec, Canada. The robot was shipped unassembled, and the biped was assembled and programmed by the author. Then a 9 V battery, a 5 V and 2660 mAh power bank, and a Raspberry Pi 3 Model B (which was manufactured in the Sony UK Technology Center in Pencoed, Wales, UK) were added. An

MPU 6050 accelerometer was also added, which was manufactured by TDK InvenSense in, Shenzhen, China.

The Lynmotion BRAT biped does not have a learning algorithm designed by the manufacturer to experiment with. The learning algorithms were developed by the author and implemented on the Raspberry Pi, which controls the Lynxmotion BRAT biped.

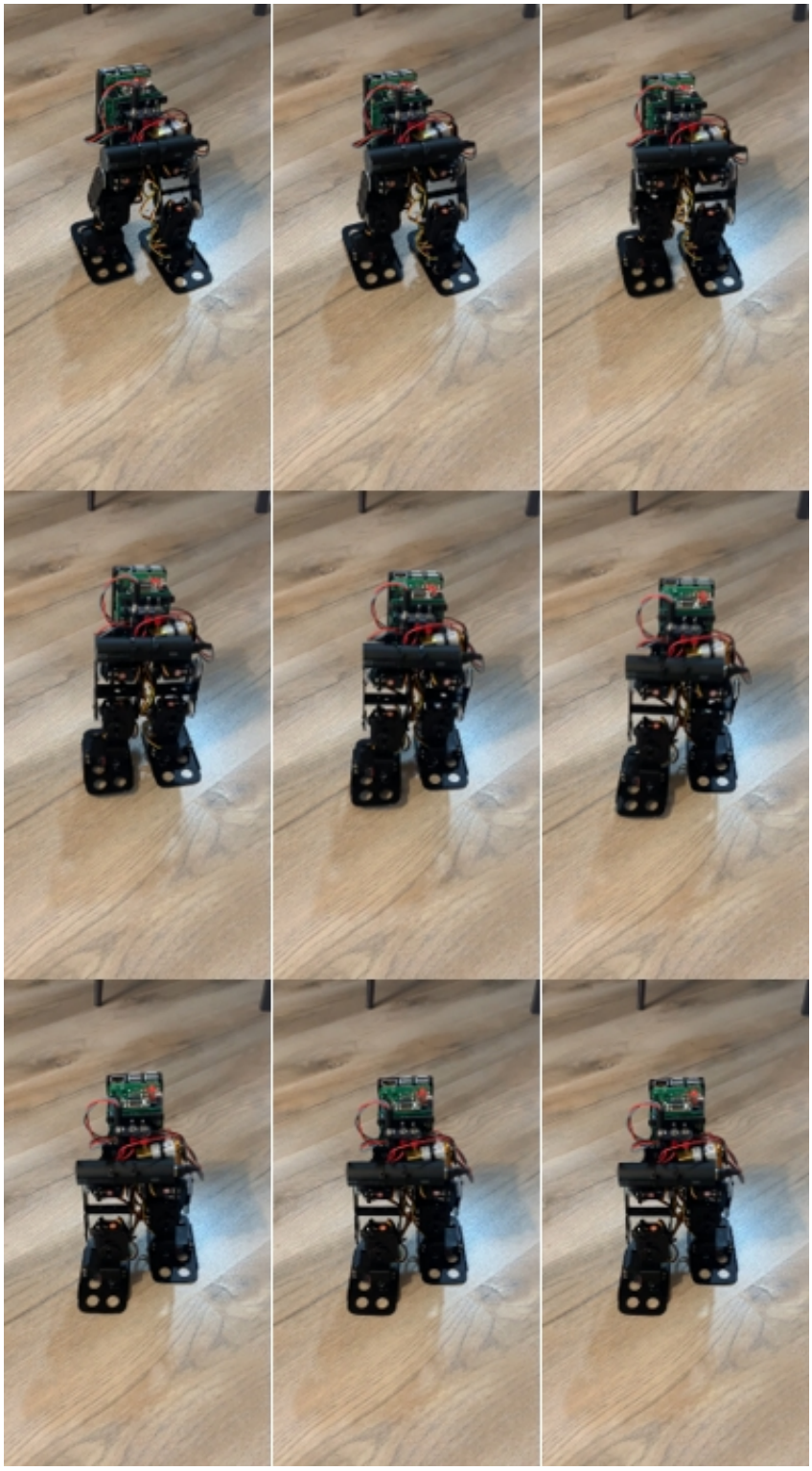


Figure 6. Trained real biped with NEAT algorithm.

For the control system, the Raspberry Pi 3 Model B is the main processing unit. Due to its superior power efficiency and built-in Wi-Fi capability, this particular model was

selected over the more recent versions of B+ and Model 4. In contrast to the B+s 350 mA (1.9 W) and the Model 4s 540 mA (2.7 W) while idle, Model B only uses 260 mA (1.4 W). Model B uses 730 mA (3.7 W) of power even under heavy loads, which is much less than Model 4s 1280 mA (6.4 W). Because the robot must operate independently with the smallest and lightest batteries possible, this reduced power consumption is essential to its design.

Three crucial circuit boards make up the final electronic assembly: the MPU 6050 accelerometer, the Lynxmotion SSC-32 servo controller board, and the Raspberry Pi 3 Model B. Two separate battery packs provide power: a 5 V, 2600 mAh power bank is used for both the Raspberry Pi and the servo motors, while a traditional 9 V battery is necessary for the electronics of the SSC-32 board, as 5 V does not suffice for its operation. The Raspberry Pi runs a genetic algorithm to produce movement data, which is transformed into motor command strings. These strings are transmitted to the SSC-32 servo controller via the RS-232 serial interface. The SSC-32 functions as an interpreter, decoding the commands and converting them into the required PWM signals to mechanically drive the six connected servo motors.

4. Discussion

The goal of performing different tests on a simulated biped and porting it to a real biped that can learn to walk was achieved. The tests were performed in a simulated environment using the DDPG and ARS algorithms. These were too complex on a real biped, because the real biped would fall too much. This way the real biped with NEAT algorithm was used. The NEAT algorithm is not the best if the biped's balance is not good, but the real biped had a much better balance than the simulated biped. It was easier to code, and it even fell a few times; it could be used to teach the real biped to walk, without destroying the biped.

The best environment to train the biped to walk is the laminate flooring, not the tile floor, because it is too hard and the biped can get stuck in the edges. The carpet floor is not good because it is not as smooth as the laminate flooring, and the biped can get stuck in the carpet.

The most difficult part is the construction of the biped, which must be built to withstand falls. Half a case was added on the back of the biped to protect the PCBs from being destroyed due to falls.

The other issue was the power supply, which must be the most powerful possible to withstand powering the six motors of the biped SSC-32 servo controller and the Raspberry Pi. A dual power supply with 5 V was used for the Raspberry Pi and for the six motors of the biped, and 9 V for the logic of the SSC-32 servo controller. The 9 V was provided by a simple 9 V battery because it does not need a high current. The 5 V was provided by a 2600 mAh power bank, because it needs a high current for the Raspberry Pi and the six motors. The 2600 mAh battery is one of the smallest power banks, which also makes it lightweight enough for the biped. The Raspberry Pi 3 Model B was chosen because it is a model with WiFi and lowest power consumption. The Raspberry Pi 3 Model B+ has a higher power consumption, which matters here. In standalone autonomous robotics, devices with low power consumption and lightweight batteries with high power have to be used. The higher the power of the battery, the higher the weight of it, so we need to achieve a soft middle, where the battery has enough high power, but is still light enough for the biped to carry.

The power supply problem is well-known in mobile robotics. The power consumption data presented in this document are taken from the manufacturer's website. It can be said that the power supply issue was largely addressed by using the lightest batteries with the highest possible capacity, as well as low-power-consumption electronics. Ultimately, the

robot was able to learn to walk, carry the batteries, and had sufficient power to operate all of its electronics.

5. Conclusions

This paper presents a comparison and implementation of the following four neuro-evolution methods: Deep Q-Learning (DQN), NeuroEvolution of Augmenting Topologies (NEAT), Deep Deterministic Policy Gradients (DDPG), and Augmented Random Search (ARS). The best results in simulation were obtained with the ARS method; however, the NEAT method was implemented on the real biped due to its simplicity of implementation and faster training speed.

The final goal was to train a real biped based on tests performed on a simulated biped. The easiest neuro-evolution method to implement, NEAT, was transferred to the real biped and trained to walk.

For the simulated biped, ARS appeared to perform better than DDPG and also had a much faster training time.

The main contribution of this paper is that walking learning was first tested on simulated bipeds and then transferred to a real biped. In this way, the real biped was protected from damage by reducing the number of falls.

If simulation had not been possible, the real biped would have had to be tested using all four neuroevolution methods (ARS, DDPG, NEAT, and DQN). However, since simulations were performed and conclusions were drawn regarding the fastest and most effective method, the real biped was tested using only the NEAT method. In this way, the number of falls during training was reduced by at least 75%.

Future work will include training additional simulated robots, such as Walker2D or Humanoid from the MuJoCo environment in OpenAI's Gymnasium toolkit.

The goal is to extend neuroevolution-based walking training to full humanoid robots with an upper body, such as Pete from Lynxmotion and other more advanced humanoid platforms.

Funding: This research was funded by Politehnica University Timisoara, Romania.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data are available on request from the corresponding author.

Acknowledgments: The author thanks the Politehnica University Timisoara, Romania, for the support given.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Yu, H.; Guan, S.; Li, X.; Feng, H.; Zhang, S.; Fu, Y. Whole-Body Motion Generation for Wheeled Biped Robots Based on Hierarchical MPC. *IEEE Trans. Ind. Electron.* **2025**, *72*, 8301–8311. [CrossRef]
2. Dallard, A.; Benallegue, M.; Scianca, N.; Kanehiro, F.; Kheddar, A. Robust Bipedal Walking with Closed-Loop MPC: Adios Stabilizers. *IEEE Trans. Robot.* **2025**, *41*, 4679–4698. [CrossRef]
3. Li, J.; Nguyen, Q. Dynamic Walking of Bipedal Robots on Uneven Stepping Stones via Adaptive-Frequency MPC. *IEEE Control Syst. Lett.* **2023**, *7*, 1279–1284. [CrossRef]
4. Choe, J.; Kim, J.-H.; Hong, S.; Lee, J.; Park, H.-W. Seamless Reaction Strategy for Bipedal Locomotion Exploiting Real-Time Nonlinear Model Predictive Control. *IEEE Robot. Autom. Lett.* **2023**, *8*, 5031–5038. [CrossRef]
5. Gu, Z.; Zhao, Y.; Chen, Y.; Guo, R.; Leestma, J.K.; Sawicki, G.S.; Zhao, Y. Robust-Locomotion-By-Logic: Perturbation-Resilient Bipedal Locomotion via Signal Temporal Logic Guided Model Predictive Control. *IEEE Trans. Robot.* **2025**, *41*, 4300–4321. [CrossRef]

6. Daneshm, E.; Khadiv, M.; Grimminger, F.; Righetti, L. Variable Horizon MPC with Swing Foot Dynamics for Bipedal Walking Control. *IEEE Robot. Autom. Lett.* **2021**, *6*, 2349–2356. [CrossRef]
7. Shamsah, A.; Agarwal, K.; Katta, N.; Raju, A.; Kousik, S.; Zhao, Y. Socially Acceptable Bipedal Robot Navigation via Social Zonotope Network Model Predictive Control. *IEEE Trans. Autom. Sci. Eng.* **2025**, *22*, 10130–10148. [CrossRef]
8. Challa, S.K.; Kumar, A.; Semwal, V.B.; Dua, N. An Optimized-LSTM and RGB-D Sensor-Based Human Gait Trajectory Generator for Bipedal Robot Walking. *IEEE Sens. J.* **2022**, *22*, 24352–24363. [CrossRef]
9. Beranek, R.; Karimi, M.; Ahmadi, M. A Behavior-Based Reinforcement Learning Approach to Control Walking Bipedal Robots Under Unknown Disturbances. *IEEE/ASME Trans. Mechatronics* **2022**, *27*, 2710–2720. [CrossRef]
10. Krishna, L.; Castillo, G.A.; Mishra, U.A.; Hereid, A.; Kolathaya, S. Linear Policies are Sufficient to Realize Robust Bipedal Walking on Challenging Terrains. *IEEE Robot. Autom. Lett.* **2022**, *7*, 2047–2054. [CrossRef]
11. Ch, ; P.; Veer, S.; Poulakakis, I. Interactive Dynamic Walking: Learning Gait Switching Policies with Generalization Guarantees. *IEEE Robot. Autom. Lett.* **2022**, *7*, 4149–4156. [CrossRef]
12. Yao, D.; Yang, L.; Xiao, X.; Zhou, M. Velocity-Based Gait Planning for Underactuated Bipedal Robot on Uneven and Compliant Terrain. *IEEE Trans. Ind. Electron.* **2022**, *69*, 11414–11424. [CrossRef]
13. Hong, Y.-D.; Lee, B. Real-Time Feasible Footstep Planning for Bipedal Robots in Three-Dimensional Environments Using Particle Swarm Optimization. *IEEE/ASME Trans. Mechatronics* **2020**, *25*, 429–437. [CrossRef]
14. Acosta, B.; Posa, M. Perceptive Mixed-Integer Footstep Control for Underactuated Bipedal Walking on Rough Terrain. *IEEE Trans. Robot.* **2025**, *41*, 4518–4537. [CrossRef]
15. Caron, S.; Escande, A.; Lanari, L.; Mallein, B. Capturability-Based Pattern Generation for Walking with Variable Height. *IEEE Trans. Robot.* **2020**, *36*, 517–536. [CrossRef]
16. Crews, S.; Travers, M. Energy Management Through Footstep Selection For Bipedal Robots. *IEEE Robot. Autom. Lett.* **2020**, *5*, 5485–5493. [CrossRef]
17. Xiong, X.; Ames, A. 3-D Underactuated Bipedal Walking via H-LIP Based Gait Synthesis and Stepping Stabilization. *IEEE Trans. Robot.* **2022**, *38*, 2405–2425. [CrossRef]
18. Park, H.Y.; Kim, J.H.; Yamamoto, K. A New Stability Framework for Trajectory Tracking Control of Biped Walking Robots. *IEEE Trans. Ind. Informatics* **2022**, *18*, 6767–6777. [CrossRef]
19. Mihalec, M.; Yi, J. Balance Gait Controller for a Bipedal Robotic Walker with Foot Slip. *IEEE/ASME Trans. Mechatronics* **2023**, *28*, 2012–2019. [CrossRef]
20. Pi, M.; Kang, Y.; Xu, C.; Li, G.; Li, Z. Adaptive Time-Delay Balance Control of Biped Robots. *IEEE Trans. Ind. Electron.* **2020**, *67*, 2936–2944. [CrossRef]
21. Ahmadi, M.; Xiong, X.; Ames, A.D. Risk-Averse Control via CVaR Barrier Functions: Application to Bipedal Robot Locomotion. *IEEE Control Syst. Lett.* **2022**, *6*, 878–883. [CrossRef]
22. Horn, J.C.; Mohammadi, A.; Hamed, K.A.; Gregg, R.D. Nonholonomic Virtual Constraint Design for Variable-Incline Bipedal Robotic Walking. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3691–3698. [CrossRef]
23. Akbari Hamed, K.; Ames, A.D. Nonholonomic Hybrid Zero Dynamics for the Stabilization of Periodic Orbits: Application to Underactuated Robotic Walking. *IEEE Trans. Control Syst. Technol.* **2020**, *28*, 2689–2696. [CrossRef]
24. Khadiv, M.; Herzog, A.; Moosavian, S.A.A.; Righetti, L. Walking Control Based on Step Timing Adaptation. *IEEE Trans. Robot.* **2020**, *36*, 629–643. [CrossRef]
25. Wang, R.; Lu, Z.; Xiao, Y.; Zhao, Y.; Jiang, Q.; Shi, X. Design and Control of a Multi-Locomotion Parallel-Legged Bipedal Robot. *IEEE Robot. Autom. Lett.* **2024**, *9*, 1993–2000. [CrossRef]
26. Zhang, Q.; Chen, W.; Zhang, H.; Lin, S.; Xiong, C. A Bipedal Walking Model Considering Trunk Pitch Angle for Estimating the Influence of Suspension Load on Human Biomechanics. *IEEE Trans. Biomed. Eng.* **2025**, *72*, 1097–1107. [CrossRef] [PubMed]
27. Zhao, H.; Yu, L.; Qin, S.; Jin, G.; Chen, Y. Design and Control of a Bio-Inspired Wheeled Bipedal Robot. *IEEE/ASME Trans. Mechatronics* **2025**, *30*, 2461–2472. [CrossRef]
28. Lee, Y.; Lee, H.; Lee, J.; Park, J. Toward Reactive Walking: Control of Biped Robots Exploiting an Event-Based FSM. *IEEE Trans. Robot.* **2022**, *38*, 683–698. [CrossRef]
29. Guadarrama-Olvera, J.R.; Kajita, S.; Cheng, G. Preemptive Foot Compliance to Lower Impact During Biped Robot Walking Over Unknown Terrain. *IEEE Robot. Autom. Lett.* **2022**, *7*, 8006–8011. [CrossRef]
30. Soliman, A.F.; Ugurlu, B. Robust Locomotion Control of a Self-Balancing and Underactuated Bipedal Exoskeleton: Task Prioritization and Feedback Control. *IEEE Robot. Autom. Lett.* **2021**, *6*, 5626–5633. [CrossRef]
31. Li, Z.; Zhao, K.; Zhang, L.; Wu, X.; Zhang, T.; Li, Q.; Li, X. Human-in-the-Loop Control of a Wearable Lower Limb Exoskeleton for Stable Dynamic Walking. *IEEE/ASME Trans. Mechatronics* **2021**, *26*, 2700–2711. [CrossRef]
32. Lee, H.; Rosen, J. Lower Limb Exoskeleton—Energy Optimization of Bipedal Walking with Energy Recycling—Modeling and Simulation. *IEEE Robot. Autom. Lett.* **2023**, *8*, 1579–1586. [CrossRef]

33. Li, K.; Tucker, M.; Gehlhar, R.; Yue, Y.; Ames, A.D. Natural Multicontact Walking for Robotic Assistive Devices via Musculoskeletal Models and Hybrid Zero Dynamics. *IEEE Robot. Autom. Lett.* **2022**, *7*, 4283–4290. [CrossRef]
34. Zhu, C.; Yi, J. Knee Exoskeleton-Enabled Balance Control of Human Walking Gait with Unexpected Foot Slip. *IEEE Robot. Autom. Lett.* **2023**, *8*, 7751–7758. [CrossRef]
35. Aller, F.; Pinto-Fernandez, D.; Torricelli, D.; Pons, J.L.; Mombaur, K. From the State of the Art of Assessment Metrics Toward Novel Concepts for Humanoid Robot Locomotion Benchmarking. *IEEE Robot. Autom. Lett.* **2020**, *5*, 914–920. [CrossRef]
36. Alemayoh, T.T.; Lee, J.H.; Okamoto, S. A Deep Learning Approach for Biped Robot Locomotion Interface Using a Single Inertial Sensor. *Sensors* **2023**, *23*, 9841. [CrossRef] [PubMed]
37. Arena, P.; Li Noce, A.; Patanè, L. Stability and Safety Learning Methods for Legged Robots. *Robotics* **2024**, *13*, 17. [CrossRef]
38. Wu, Y.; Tang, B.; Qiao, S.; Pang, X. Bionic Walking Control of a Biped Robot Based on CPG Using an Improved Particle Swarm Algorithm. *Actuators* **2024**, *13*, 393. [CrossRef]
39. Schumacher, P.; Geijtenbeek, T.; Caggiano, V.; Kumar, V.; Schmitt, S.; Martius, G.; Haeufle, D.F.B. Emergence of natural and robust bipedal walking by learning from biologically plausible objectives. *iScience* **2025**, *28*, 112203. [CrossRef] [PubMed]
40. Naya-Varela, M.; Faina, A.; Duro, R.J. Learning Bipedal Walking Through Morphological Development. In Proceedings of the 16th International Conference on Hybrid Artificial Intelligence Systems (HAIS), Bilbao, Spain, 22–24 September 2021; pp. 184–195. [CrossRef]
41. Yamano, J.; Kurokawa, M.; Sakai, Y.; Hashimoto, K. Walking Motion Generation of Bipedal Robot Based on Planar Covariation Using Deep Reinforcement Learning. In Proceedings of the 6th International Conference on Synergetic Cooperation between Robots and Humans (SCR), Tokyo, Japan, 15–17 May 2024; pp. 217–228. [CrossRef]
42. Peng, T.; Bao, L.; Humphreys, J.; Delfaki, A.M.; Kanoulas, D.; Zhou, C. Learning Bipedal Walking on a Quadruped Robot via Adversarial Motion Prior. In Proceedings of the 26th Annual Conference on Towards Autonomous Robotic Systems (TAROS), London, UK, 3–5 September 2025; pp. 118–129. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.