Optimizing Neural Network Performance in Game Playing Using Simulated Annealing and Reinforcement Learning

Henrique Coutinho Layber[1], **Vitor Berger Bonella**[1], **Flávio Miguel Varejão**[1]

[1]Departamento de Informática – Universidade Federal do Espírito Santo (UFES)
Vitória – ES – Brazil

{henrique.layber,vitor.bonella}@edu.ufes.br,flavio.varejao@ufes.br

Abstract. *This paper proposes a method to optimize neural network parameters for game playing using Simulated Annealing (SA) and Reinforcement Learning (RL). The study focuses on the Dino Game, comparing the performance of the proposed neural network method against a baseline decision tree method. Experimental results demonstrate that the neural network outperforms the decision tree, achieving a higher mean score with greater consistency. Statistical tests confirm the performance improvements are statistically significant, indicating the effectiveness of the SA heuristic in optimizing neural network parameters.*

1. Introduction

Since the first time a machine beat a chess champion, there have been diverse efforts to create algorithms and methods that can play games as well as humans. One of these efforts has led to significant advancements in the realm of Artificial Intelligence (AI). Studies have demonstrated the potential of neural networks to tackle complex problems, including game playing, with notable success [Silver et al. 2016].

One of the ways a neural network can learn to play a game is through Reinforcement Learning (RL) [Sutton and Barto 2018]. RL is a type of Machine Learning where an agent learns to behave in an environment by performing actions and observing the results of those actions. A common implementation of RL involves using a search heuristic and a classifier. The search heuristic explores the environment, typically involving the neural network's parameters, while the classifier predicts the best action to take based on the current state of the environment.

This paper proposes the use of the Simulated Annealing (SA) [Burkard and Rendl 1984] search heuristic to optimize parameters of a neural network tailored to play the Dino Game [Kulkarni et al. 2023]. While the Dino Game was chosen for this study, the proposed method can be adapted for use in other games.

The Section 2 explains and discusses the proposed method to optimize the neural network. Section 3 details the experimental setup used to evaluate the performance of the proposed method. Section 4 presents the results of the experiments, highlighting the effectiveness of the approach. Finally, Section 5 summarizes the findings and outlines potential future work.

2. Proposed Method

In this section, we describe the methodology used to optimize the neural network for playing the Dino Game. The optimization process leverages the Simulated Annealing (SA) search heuristic to fine-tune the network's parameters.

2.1. Dino Game

Dino Game[1] is a runner game featuring a dinosaur as the player, the goal is to score as high as possible dodging obstacles. The game is very simple, and only need to choose when to jump or not, thus a good example of a game that can be played by a neural network. The game is simple enough that a neural network can learn to play it, but it is also complex enough that it is not trivial to play.

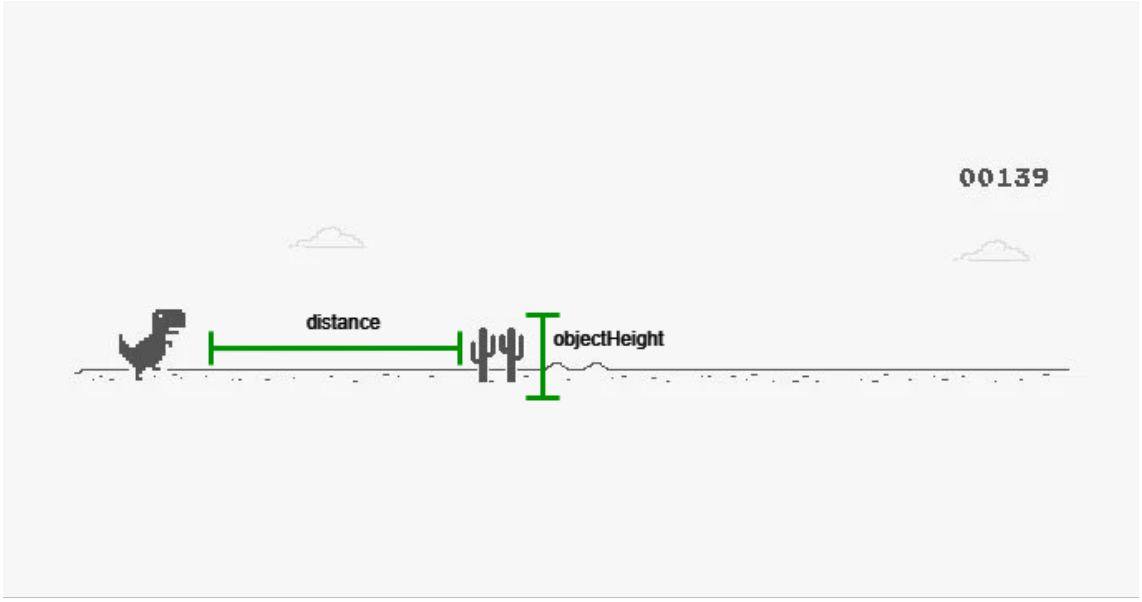

Figura 1. Dino Game Frame

The game has several variables that impact when you have to jump or not, including the actual speed of the game, the distance to the next object, and the height from the floor to the next object. Figure 1 shows a sample frame of the game, highlighting the variables *distance* and *objectHeight*. In the top right corner, the current score is displayed, the score increases every second alive.

2.2. Neural Network

Neural networks [McCulloch and Pitts 1943] are foundational to modern artificial intelligence, particularly in the context of Reinforcement Learning (RL). A neural network consists of interconnected layers of artificial neurons, each layer processing and transforming input data through weighted connections. These networks are trained to learn patterns and relationships within the input, enabling them to generalize and make predictions or decisions. In RL, neural networks act as function approximators that map states of an environment to actions, learning through interactions with the environment rather than explicit instructions. This capability is crucial for handling complex, uncertain environments where decisions must be adaptive and based on feedback received over time.

In this study, neural networks are used as decision-making agents to determine the optimal jump timing in the Dino Game. The proposed neural network architecture receives three inputs: the distance to the next object, the object's height, and the game's

[1]Officially available on `chrome://dino`, for Chromium browsers only

speed. These inputs are processed through an architecture that includes an input layer with three nodes, followed by a hidden layer with two nodes that use Rectified Linear Unit (ReLU) activation functions [Agarap 2018]. The ReLU functions introduce non-linearity to capture complex relationships in the data. The final layer consists of a single output node with a sigmoid activation function [Han and Moraga 1995], with a threshold set at 0.55. This threshold ensures that the agent only jumps when it is sufficiently certain about the decision, enhancing its performance and stability in gameplay scenarios. Notably, no biases were utilized in the network.

2.3. Simulated Annealing

Simulated Annealing (SA) is a probabilistic optimization technique inspired by the process of annealing in metallurgy, where materials are heated and slowly cooled to achieve a low-energy crystalline state. In computational contexts, SA is particularly useful for exploring complex search spaces and finding near-optimal solutions to combinatorial optimization problems.

In the field of neural networks and Reinforcement Learning (RL), SA can be employed to optimize the parameters of neural networks used as decision-making agents. In RL, agents learn to make decisions based on environmental feedback, aiming to maximize cumulative rewards over time. SA enhances this process by systematically adjusting the weights and biases of the neural network, akin to tuning the ”temperature”parameter in traditional SA, to explore the optimal configuration. By iteratively adjusting network parameters based on the rewards received from actions taken in different states of the environment, SA allows neural networks to adapt and improve their decision-making capabilities.

SA's ability to balance exploration (searching for new solutions) and exploitation (exploiting known solutions) makes it well-suited for training neural networks in RL applications. This approach helps neural networks navigate complex decision spaces, leading to improved performance in tasks such as game playing, where adaptive and strategic decision-making is crucial.

There are various methods to decrease the temperature in simulated annealing, each affecting how the optimization process unfolds. The temperature can be reduced by a fixed amount, a percentage of the current value, or by employing specific cooling functions. The cooling rate, a critical meta-parameter, dictates the speed at which the temperature decreases during the optimization process. Established cooling functions have been explored in simulated annealing for this paper, the Boltzmann schedule [Kirkpatrick et al. 1983], exponential cooling [Van Laarhoven et al. 1987], and geometric cooling [Černỳ 1985] strategies.

These cooling strategies play a pivotal role in balancing exploration and exploitation within the optimization process. A slower cooling rate allows for more extensive exploration of the solution space, potentially discovering new and better solutions. Conversely, a faster cooling rate prioritizes exploitation, focusing on refining known solutions to potentially achieve higher accuracy or performance in the final optimized configuration. The choice of cooling function significantly influences the convergence speed and the quality of solutions attained by simulated annealing algorithms applied to neural network optimization and reinforcement learning tasks. Figure 2 illustrates the variations

in temperature over time during the simulated annealing process.

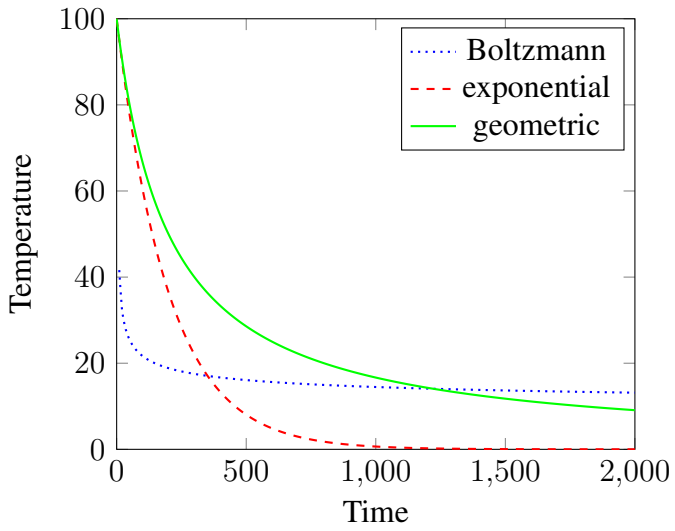

Figura 2. Temperature functions change over time

3. Experimental Setup

The experiments leverages in consideration two methods, the the proposed one, and a baseline which is a simple decision tree based on the same input variables as cited in section 2 and optimized by a gradient ascent search heuristic [Boyd and Vandenberghe 2004].

Python [Van Rossum and Drake Jr 1995] was used to implement the entire experimental setup. The neural network weights were adjusted using the Simulated Annealing (SA) algorithm during an extensive training session lasting eight hours. Figure 3 show the evolution of the agent score over the epochs.

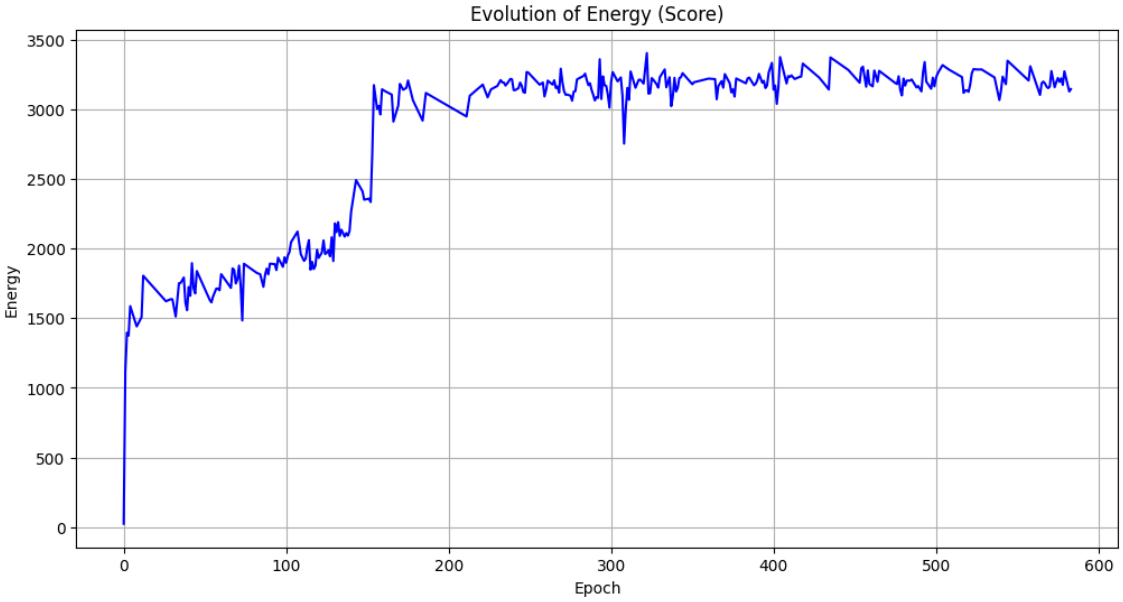

Figura 3. Energy evolution for each epoch

The SA settings included a starting temperature of 200 and a geometric cooling schedule with a rate of 0.003. Each solution was tested 15 times in the game to account for the randomness of the scenarios, ensuring solution stability. The solution final score was calculated as the mean of these 15 rounds minus the standard deviation.

To compare two methods, conventional practice involves using hypothesis tests to determine if there is a statistically significant difference between them. In this study, the paired t-test and the Wilcoxon test are employed to rigorously assess performance differences between the methods. The t-test assumes that the data follows a normal distribution; therefore, the best solution for each method was tested 30 times. In statistics, sample sizes of 30 or more are generally considered sufficient to achieve asymptotic normality.

4. Results and Discussions

The performance of the proposed neural network method and the baseline decision tree method was assessed based on experimental data. Table 1 summarizes the mean and standard deviation (Std) of the scores achieved by the two methods. From Table 1, it is evident that the neural network outperforms the decision tree, achieving a higher mean score with a lower standard deviation, indicating a consistent performance.

	Neural Network	**Decision Tree**
Mean	3381.99	1068.18
Std	246.54	304.04

Tabela 1. Mean and Standard Deviation

Figure 4 provides a visual comparison of the score distributions for both methods through boxplots. The boxplots illustrate that the neural network not only achieves higher median scores but also less variability in the results as opposed to the decision tree.

Table 2 presents the results of the paired t-test and the Wilcoxon test. Both tests yield p-values approximating zero, indicating that the differences in performance between the neural network and the decision tree are statistically significant at a 95% confidence level.

Test	**p-value**
t-test	≈ 0
Wilcoxon	≈ 0

Tabela 2. Statistical Test Results

Figure 3 reveals two notable performance jumps during the training process. The first significant increase occurs at the beginning, moving to around the 1200 score mark, indicating that the agent has successfully learned to execute basic actions such as jumping and ducking to avoid obstacles. This initial learning milestone demonstrates the agent's foundational ability to navigate the game environment proficiently. A second, more substantial jump in performance is observed between epochs 150 and 154. This leap can be attributed to the agent's refinement in timing its jumps, particularly at higher game speeds. By learning to anticipate and react to obstacles earlier, the agent significantly enhances its overall performance and stability.

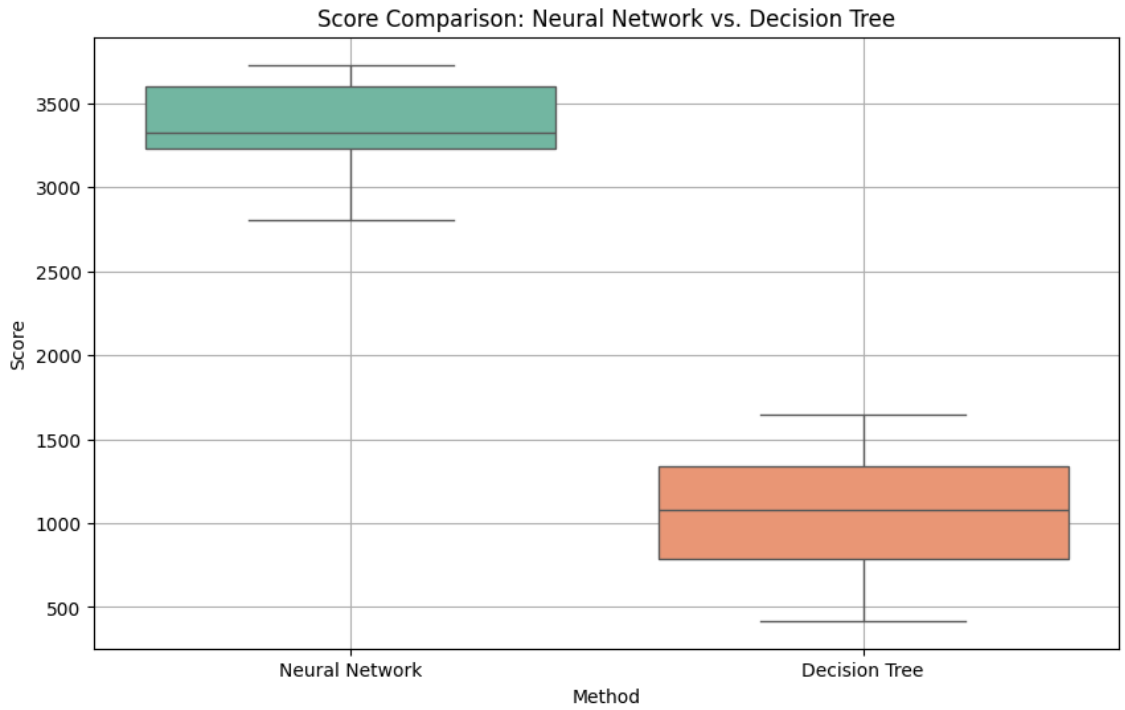

Figura 4. Score boxplots of the methods

5. Conclusion

In this paper, we have presented a novel method for optimizing neural network parameters using Simulated Annealing (SA) and Reinforcement Learning (RL) for game playing, specifically targeting the Dino Game. The results from our experiments show that the neural network outperforms a baseline decision tree method in terms of mean score and consistency. The statistical analysis, employing both the paired t-test and the Wilcoxon test, confirms that the performance improvements achieved by the neural network are statistically significant.

This study not only highlights the potential of using SA to fine-tune neural network parameters but also underscores the robustness and effectiveness of neural networks in complex game-playing scenarios. Future research could explore the application of this method to a broader range of games and investigate further enhancements to the optimization process, such as integrating different cooling schedules or combining SA with other meta-heuristic approaches.

By advancing the methodologies for neural network optimization, this work contributes to the ongoing efforts to leverage AI for game playing, opening new avenues for research and development in the field of Artificial Intelligence and Machine Learning.

Referências

Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Burkard, R. and Rendl, F. (1984). A thermodynamically motivated simulation procedure for combinatorial optimization problems. *European Journal of Operational Research*, 17(2):169–174.

Černỳ, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45:41–51.

Han, J. and Moraga, C. (1995). The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International workshop on artificial neural networks*, pages 195–201. Springer.

Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.

Kulkarni, A., Bapat, P., Kulkarni, T., and Pawar, R. (2023). Review of reinforcement learning in chrome dino game. In *2023 7th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, pages 1–5.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Van Laarhoven, P. J., Aarts, E. H., van Laarhoven, P. J., and Aarts, E. H. (1987). *Simulated annealing*. Springer.

Van Rossum, G. and Drake Jr, F. L. (1995). *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam.