Neuroevolution of Self-interpretable agents on procgen game

July 4, 2023

Alessio Ferrone 1751859

Abstract

With this paper, I aim to demonstrate how attention can be a viable solution in reinforcement learning tasks that rely solely on visual information (RGB images). Specifically, the objective is to create an agent that utilizes attention and the simple models presented in the original paper titled "Neuroevolution of Self-interpretable agents"(Yujin Tang, 2020) to play one of the procgen games (Karl Cobbe, 2020). Furthermore, I will present my own proposal that leverages attention principles in a slightly different manner, leading the agent to exhibit distinct behaviors. The following link provides the GitHub repository of the project and a demo of StarPilot game https://github.com/Pnlalessio/DLAI_project.

1. Introduction

During gameplay, humans exhibit a natural ability to shift their attention across the screen, focusing on relevant information based on the task at hand. Our adeptness at identifying and focusing on pertinent details amidst a wealth of information greatly influences our decision-making. So, as shown in the paper (Yujin Tang, 2020), this crucial insight holds the potential to enhance the performance of Reinforcement Learning agents.

Attention bottleneck mechanism. The core concept integrates attention into image processing by dividing them into N patches. A self-attention mechanism determines patch importance and information richness. To focus, a trainable module selects k patches from N, creating an attention bottleneck. This prioritizes significant patches, ignoring less impactful ones. Training uses evolutionary algorithms due to non-differentiable selection module.

Email: Alessio Ferrone 1751859 <ferrone.1751859@studenti.uniroma1.it>.

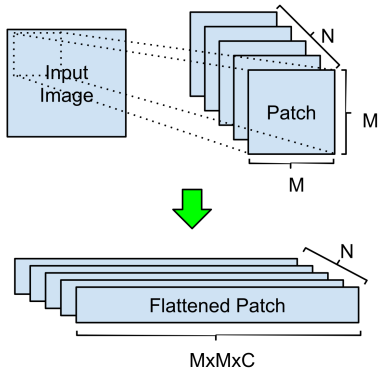
Deep Learning and Applied AI 2023, Sapienza University of Rome, 2nd semester a.y. 2022/2023.

Few parameters. This approach enables a significant reduction in the number of trainable parameters to just a few thousand (less than 10,000 parameters). This makes it feasible to employ an evolutionary algorithm for training our agent's network.

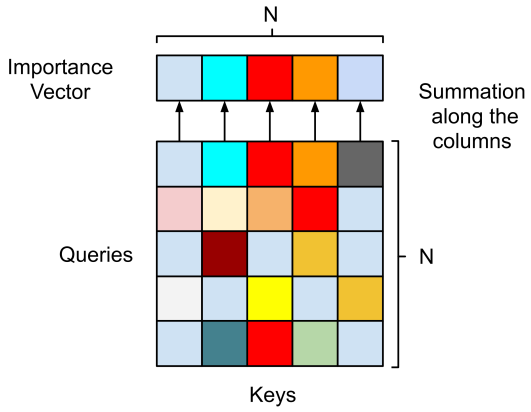
2. Method

The agent engages in one of the procgen games, where at each step of a game episode, it reviews the observations (sequential RGB images depicting the game's progress) and selects the optimal action to maximize the total average reward. The envisioned base network consists of three phases:

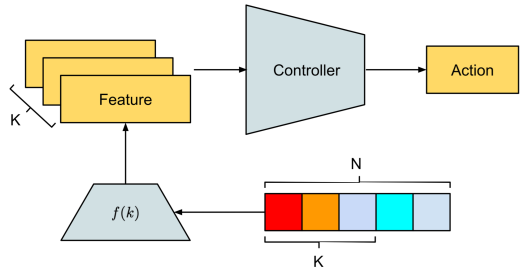
Input Transformation. Following initial image preprocessing, we employ a sliding window approach with dimensions MxM and stride S to extract patches from the input image. This results in an output shape of (N, M, M, C), where N represents the number of patches, M denotes the height/width of each patch, and C signifies the number of channels in the input image. Subsequently, we flatten the data.



Self-attention to compute importance voting. By projecting the flattened patches into Keys and Queries, we derive an attention matrix with dimensions (N, N). To determine the patch importance vector, we apply softmax to each row of the matrix and subsequently sum it along the columns.



Patch Selection and Feature retrieval. Sorting the patch importance vector, we extract the indices of the K highest-ranking patches. Next, we associate these indices with their corresponding features, which are then fed into the controller module to determine the action. Once the top k patches are selected, the controller can be fed with two distinct types of features: the coordinates of the k best patches' centers (as suggested in the original paper), or (as a novel approach) both the coordinates of the k patches and the average color values (in RGB) of the pixels within each selected patch. The controller resembles the one presented in the paper. It utilizes an LSTM to enable the agent to take actions based on previous steps, with an additional linear layer after the LSTM and a ReLU activation function before outputting the result.



Pick K patches from sorted importance vector

3. Double Self Attention

The bottleneck of self-attention renders the agent blind to less relevant patches compared to the entire current observation. The idea is to obtain the K most relevant patches with the initial self-attention, representing what the agent actually sees. Then, considering these K best patches as a new observation (while preserving their original positions), a new self-attention (referred to as "Double") is applied. This "Double" self-attention selects patches that are more important among the previously identified relevant ones, giving greater emphasis to patches in close proximity.

4. CMA-ES and Exploration

The network was trained using the Covariance-Matrix Adaptation Evolution Strategy (CMA-ES(Hansen, 2016)), a genetic algorithm. This choice was made because obtaining the top K sorted patches is a non-differentiable problem. Genetic algorithms are well-suited for reinforcement learning problems where rewards are not available at each timestep but are instead cumulative at the end of the task. However, for this task, it was necessary to train networks with several thousand parameters, with the largest one having approximately 7,061 parameters. To mitigate the risk of getting trapped in local minima, a mechanism was introduced to allow the agent to explore more extensively during the early stages of training. Whenever the agent repeatedly performs the same action for multiple consecutive timesteps, that action receives a penalty in terms of the probability of it being chosen in the next timestep. Meanwhile, the probabilities of selecting other possible actions as the next action to execute are increased.

5. Results and conclusion

The network was trained in three modes on the StarPilot game from procgen (The network's versatility allows application to other procgen games with minor adjustments.). In the first mode, the controller received only the coordinates of the top K patches' centers as features. The second mode incorporated average color values of the RGB channels, and the third mode used the Double Self Attention mechanism. After training for approximately a week per network on a GPU-less MacBook, the network was tested in these modes. In the first two modes, agents exhibited similar behavior, moving towards the left screen edge, adjusting vertical position based on enemy ships, and shooting continuously. With the Double Self Attention, the strategy focused on the left edge and evading enemies at close range. Prioritizing nearby threats resulted in less continuous but more agile movement. The agent moves only when enemies are very close, but with a quicker dash. As observed in the table, the three agents achieve similar average rewards during testing across 100 random episodes. However, it is noteworthy that the Double Self Attention agent reaches higher peaks in terms of the maximum reward obtained among the 100 episodes.

Table 1. Performance comparison.

#Test reward	Test 1 mean max	Test 2 mean max	Test 3 mean max	Test 4 mean max	Test 5 mean max
Only-Pos	7.4/36	6.2/20	7.3/43	6.7/31	5.8/18
Pos+Col	6.2/22	6.3/21	6.0/22	6.2/23	6.8/27
Double	8.7/56	7.1/31	7.0/32	7.2/34	8.0/39

References

Hansen, N. The cma evolution strategy: A tutorial. 2016.

Karl Cobbe, Christopher Hesse, J. H. J. S. Leveraging procedural generation to benchmark reinforcement learning. 2020.

Yujin Tang, Duong Nguyen, D. H. Neuroevolution of self interpretable agents. 2020.