An Intelligent Agent Playing Generic Action Games based on Deep Reinforcement Learning with Memory Restrictions

Lucas Antunes de Almeida
Consulting Services
Create IT
Lisbon, Portugal
lucas.almeida@create.pt

Marcelo Resende Thielo
Dept. of Computer Science
Federal University of Pampa
Alegrete, Brazil
marcelothielo@unipampa.edu.br

***Abstract*—Among the topics that increasingly gained special attention in Computer Science recently, the evolution of Artificial Intelligence has been one of the most prominent subjects, especially when related to games. In this work we developed an intelligent agent with memory restrictions so to investigate its ability to learn playing multiple, different games without the need of being provided with specific details for each of the games. As a measure of quality of the agent, we used the difference between its score and the scores obtained by casual human players. Aiming to address the possibilities of using Deep Learning for General Game Playing in less powerful devices, we explicitly limited the amount of memory available for the agent, apart from the commonly used physical memory limit for most works in the area. For the abstraction of machine learning and image processing stages, we used the Keras and Gym libraries. As a result, we obtained an agent capable of playing multiple games without the need to provide rules in advance, but receiving at each moment only the game video frame, the current score and whether the current state represents an endgame. To assess the agent effectiveness, we submitted it to a set of Atari 2600™ games, where the scores obtained were compared to casual human players and discussed. In the conclusion, we show that promising results were obtained for these games even with memory limitations and finally a few improvements are proposed.**

***Index Terms*—General Game Playing, Artificial Intelligence, Atari 2600™**

I. Introduction

The development of an algorithm capable of playing games properly in a similar way to an experienced human player is still a challenge for today's state-of-the-art in the field of Artificial Intelligence (AI). In the last decades, the approaches used so that an agent could determine its next step have shown a steady evolution, with most of this growth occurring due to the focus on the seek for adequate test environments. Within the area of AI, games could represent test environments for the models developed, making it possible to carry out tests in numerous and different situations just by changing the selected game.

The need to provide specific, hard-coded rules and characteristics of a game to an algorithm that should become able to play it can prevent its direct reuse with other games requiring different rules. Even if the game style changes just slightly, games like Chess and Checkers although being visually similar since they occur on the same board and have a similar number of pieces, possess differences between the appearance of the pieces and their respective movement rules that make them completely different games. So, providing an algorithm with the ability to adapt to a game which the rules have never been presented to it is a very complex and challenging task.

An agent must determine which elements of the environment are to be taken into account at each moment, thus becoming able to select the action that will maximize its chances of victory or progress in the game in the long term. To bridge the gap in gaming performance between human players and AI agents, many professionals in the field have been making constant efforts to develop heuristic optimizations and Machine Learning (ML) models capable of achieving satisfactory results.

The *DeepMind Group* [1] develops programs that can learn and solve many complex problems without having to be explicitly taught about it. By implementing their research in the field of games, an useful and flexible training ground, the agent they developed was able to learn to play more than 40 completely different Atari 2600™ titles, just being provided with raw input pixels for the agent. In the article published by the *AlphaStar* Group [2], members of the DeepMind group, a program is presented to play *StarCraft II* with the first AI able to defeat a top professional player. Although before that there were significant successes in AI playing video games so far, AI techniques have struggled to cope with some strategy games.

The general objective of this work is, focused on the theme of General Game Playing (GGP) and using machine learning techniques and algorithms, to build an agent that should be capable of determining the context of the game to which it is submitted and develop skills to playing as good as it can. In order to figure out its actions, this agent should receive just the frame corresponding to the current game image and also the numeric score, eliminating the need to develop specific metrics for the agent to determine the core elements of the game, so that the agent's focus remains on how to play the game properly and not to extract the features.

This work is strongly inspired by the results of Mnih et al. [3], where they present an agent model capable of obtaining scores similar to that of an experienced human in many of the tested games. As a specific objective, we aim at developing an agent that demonstrates a performance similar to that of a human in multiple games, but with more restrictions in terms of available memory for storing its past experiences.

II. Theoretical Reference

A. General Game Playing

The term GGP refers to algorithms that are able to play more than one game successfully. In games like Chess, specific algorithms are generally employed, with heuristics designed to maximize the chances of winning using specific characteristics of the game, thus preventing the use of such algorithms in any other games. Unless the game is some variation of Chess, the heuristic algorithm tends not to be able to play it properly.

A heuristic algorithm developed to play Chess does not have the ability to properly play the Othello game, even if both occur in similar environments. An algorithm proposed in the context of GGP should display the ability of efficiently playing similar games. Algorithms for GGP competitions are usually tested in different games, thus proving the algorithm's ability to adapt efficiently to the environment in which it was submitted.

According to Browne et al. [4], throughout most of the history, game rules were transmitted verbally. One of the aspects that can contribute to proving the veracity of a set of rules for a game is the ease with which it can be explained and understood. Games with extremely long and detailed rule descriptions are less likely to have been played throughout history than games with rules that can be explained easily.

According to the authors of [5], an agent for GGP is designed to cover a wide variety of games, including single-player games, two-player games, multiplayer games, shift or simultaneous games, competitive or cooperative games. Agents must automatically adapt not only to play the games properly, but they must play them efficiently, usually exceeding the capacity of a human player.

The authors of [6] present the development of an agent for GGP capable of dealing with multiple different types of problems, properly interpreting the environment and automatically determining which actions should be selected considering the information available about the environment that are at its disposal.

B. Artificial Intelligence

The domain of AI mainly consists of problems in which the usual, deterministic algorithms are not generally effective at solving, such as recognizing faces, speaking a language, or being creative. An early example for the use of AI in action games is Sega's 1989 game Golden Axe™, which for its time represented a major advance in the use of AI for that type of game. The game featured enemies running past the player and then attacking him/her from behind.

According to Galway et al. [7], the AI area for games represents the implementation of a set of algorithms and techniques of traditional and modern AI, with the aim of providing solutions for a series of problems depending on the game environment.

According to Millington et al. [8], AI in most modern games addresses three fundamental topics: the ability to move characters, the ability to make decisions about where to move, and the ability to think tactically or strategic in relation to the environment in which they find themselves.

According to Yannakakis et al. [9], when most people hear the term AI in games, the image of robots performing actions in games is the first thought that comes to mind. This is because the AI area is largely connected with concepts of autonomous decisions, which make it possible to generalize information, but it is important to emphasize that AI can also be used to generate content, and not just to interact with it.

C. Machine Learning

Machine learning(ML) is one of many applications in the area of AI that provide systems with the ability to automatically learn and improve themselves from iteration cycles. Through these iterations and data analysis that are supplied to the application, it starts self-adjusting, acquiring the ability to automatically determine which set of actions are most appropriate to be taken at each moment.

According to Fürnkranz et al. [10], ML has become one of the main areas of AI where the algorithms that use ML techniques inevitably rival and exceed human capabilities in most games related to strategy or motor coordination.

D. Deep Learning

The definition of Deep Learning (DL) consists of an emerging theme within the field of AI. It consists of a subcategory of ML that emphasizes the possibility of learning using specially designed Neural Networks (NN). DL performs the training of a computational model so that it can detect patterns in the data to which it has been submitted, thus aiming to approximate solutions based on the generalization of these data.

According to Bengio [11], techniques linked to some model of DL aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower level features. Automatically learning features at multiple levels of abstraction allow a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features.

According to Justesen et al. [12], the rapid evolution of DL methods observed is due to the convention of comparing results against publicly available data sets. A similar convention in the area of AI is to use game environments to compare game algorithms, in which methods are classified based on their ability to score points and win in games.

E. Reinforcement Learning

Reinforcement Learning (RL) techniques allow the agent to have the ability to change its behavior based on responses obtained from the environment. In the ideal cases, the agent can reach the global minimum for a given problem, that is, there is the possibility to minimize the number of errors made in a given game environment. A simple diagram of how a RL agent works can be seen in Fig. 1.

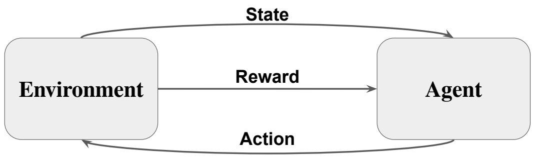

Fig. 1. Simplified scheme of a Reinforcement Learning model.

The Fig. 1 presents a classic scheme of RL, adapted for digital games, where the agent is provided with a status and a reward, which can be represented by a screen image and a score, respectively. In response, the agent returns an action for the game.

The authors of [8] present that RL is the name given to a variety of learning techniques based on experience. A reinforcement-based algorithm has three fundamental components, which are an exploration strategy to test different actions linked to the game, a reinforcement function that determines how much is the reward for a given action, and a learning rule that unites the first two characteristics.

As shown by the authors of [13], RL is a class of ML models where the learning process is based on evaluative responses without any supervised signals. RL aims to create human-like agents who learn by trial and error, using only rewards or punishments to develop successful strategies that eventually lead to the greatest long-term rewards.

F. Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) models consist of combinations of traditional models such as RL and DL, providing networks with DL capabilities, but with scoring characteristics. Fig. 2 shows that the only difference between a traditional DRL and RL agent is in the presence of a network used in the agent's own learning.

According to Torrado et al. [14] an agent of DRL learns through interactions with a dynamic environment and balances the exchange of rewards between long-term and short-term planning, that is, it determines which actions are advantageous in the short term and long term. DRL agents are essentially made up of the combination of RL and DL models.

According to Lample et al. [15], DRL allows the agent to interact autonomously with the environment. At each interaction with the environment, the agent seeks to observe what

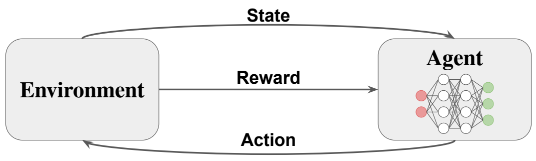

Fig. 2. Simplified scheme of a Deep Reinforcement Learning model.

is happening at the moment, and through this observation, according to its reward policy, the next movement that it will execute is decided, always aiming to maximize the best result.

According to Narasimhan et al. [16], DRL is generally used to infer the meaning of a state of the game, so that it can learn to play the games in which it is being submitted. Therefore, DRL is often used as a form of training for agents.

G. Q-Learning

The model of Q-Learning (QL) is a reinforcement model enhanced with policies that seek to find the best action to be taken in relation to the current state of the provided environment. The learning function uses random actions and therefore looks for actions that maximize the total reward. QL models value the quality of the solution, even if the time spent searching to obtain higher quality solution is greater. A QL agent stores the states found and their best response in a table so to allow comparisons between states, which can be seen in the Fig. 3.

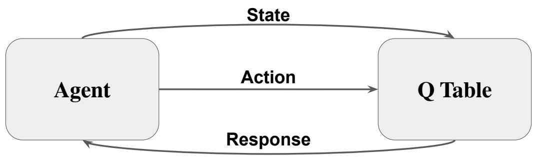

Fig. 3. Functioning of a Q-Learning agent.

According to Millington et al. [8], QL techniques depend on having the problem represented in a particular way. With that representation, it could store and update relevant information as it explores possible actions. QL models treat the game world as a state machine. At any time, the algorithm is in some state, which must encode all relevant details about the environment and the character's internal data. Thus, if the character's health is significant for learning, and if the character is in two identical situations with two different health levels, then it will consider them as different states. Any information not included in the state cannot be learned.

According to Hu et al. [17], a problem where the agent does not have full knowledge of the possible states or possible rewards of the environment, where the agent has to learn based on observations, gives rise to a model-free reinforcement learning system, being QL one example of these situations.

According to Hasselt et al. [18], interesting problems are usually large and extensive, making learning all states of the environment a complex task. But these problems can usually be described using parameterizable functions. A QL based agent uses the available parameters of the functions to maximize the reward it may receive.

H. Games and Machine Learning

Determining the effectiveness of a specific AI model is a complex task since the degree of difficulty involved in creating and validating tests is not trivial. Video games provide an environment free of external influence, being promising for the validation of the effectiveness of a model by allowing the developed agent to be submitted to multiple, isolated test environments with different contexts and objectives. Performing benchmarks with games make it possible to detect points of improvement or corrections that may be necessary for a machine learning model.

According to Bowling et al. [19], games are created for entertainment, simulation, or education, although they provide great opportunities for ML algorithm applications. The variety of possible contexts and the problems arising from them are limited only by the implementation of each game, and it is clear that the games industry is in a state of constant evolution. Consequently, games using ML techniques would attract attention to the field.

According to Serafim et al. [20], games are often used as environments to test different algorithms and techniques. Some characteristics that contribute to the popularity of these environments are linked to complexity, determinism, limited data entry, the number of pixels present on the screen and the great impact when doing a public presentation, resulting in an increase of interest and importance of research using digital games as testing environments.

According to the *AlphaStar Group* [2], games have been used for decades as an important way to test and evaluate the performance of artificial intelligence systems. As the algorithms' capacities grow, the academic community seeks games with increasing complexity that incorporate more challenging aspects such as artificial intelligence elements of greater complexity, necessary to solve scientific and real-world problems.

I. Gym

According to the authors of [21], Gym is a toolkit for developing and comparing RL algorithms. It makes no assumptions about its agent's structure and is compatible with any numerical computing library, such as TensorFlow or Theano. The library is a collection of test problems and environments that one can use to design his or her own RL algorithms. These environments have a shared interface, allowing the writing of general algorithms.

The Gym library has emulation environments that allow developers to test their RL algorithms. Among these environments, there is an Atari 2600™ emulator. Fig. 4 shows the Space Invaders™ and Kung Fu Master™ games as emulated in the Gym library.

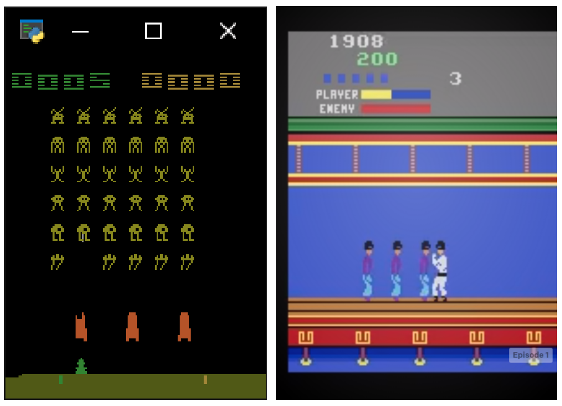

Fig. 4. Space Invaders™ and Kung Fu Master™ games running in the Gym library.

Because it is a library focused on testing RL algorithms, the Gym library makes it easy to obtain information that is present in multiple games. Whenever an agent submits an action to the library, it receives the screen image, the reward, confirmation that the game is not over, and specific information about the game in progress.

J. Keras

For the implementation of the necessary networks in this work, the Keras library was selected based on the ease of use and diversity of networks available for use, enabling tests with multiple types of models and frameworks without major changes in agent code.

Keras is a high-level open-source neural network API, written in Python, and able to run on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling rapid experimentation. Being able to get from idea to result with the shortest possible delay is the key to doing good research according to the authors of [22].

III. Development

A. Operation of the Proposed Agent

As previously mentioned, the main objective of this work is to develop a software agent within the context of GGP that could be further improved and used with different ML algorithms. For this, an agent was developed using Gym communication and ML methods from Keras, only receiving the game frames and score. In Fig. 5, the position of the agent with relation to the other tools is shown.

A considerable part of the related works used techniques based on Deep Reinforcement Learning and Double Q-learning (DRL and DQL), suggesting the effectiveness of these two models in the context of GGP. As it can be seen in Fig. 6, the proposed agent uses the Gym library to obtain the frames

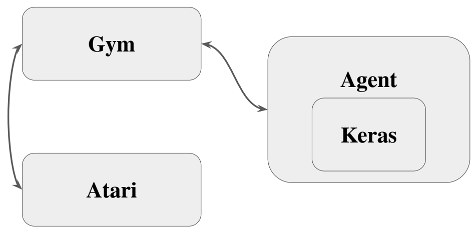

Fig. 5. Location of the proposed agent.

and their respective information and, in response, provides the library with the action it decided to take.

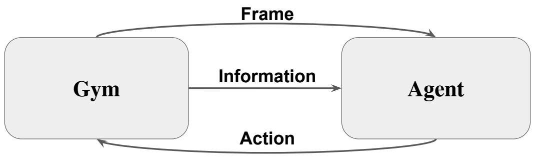

Fig. 6. Communication between the agent and the Gym library.

The communication model presented in the Fig. 6 provides the agent with the ability to play any game, as long as the correct information is provided. Therefore, the standardization of the data format received by the agent allows testing with games without the need to adapt the agent's source code.

B. Definition of the Neural Network used in the Agent

The Neural Network(NN) model implemented is based on the fact that the input provided is a frame previously prepared, that is, the capture of the current visual output of the game, which is delivered to the network as a gray scale image. In addition to the current frame of the game, the network is provided also with information obtained from the Gym library, these being the number of lives remaining and whether the current frame represents a state of the game over or not.

The proposed and developed agent consists primarily of the insertion of processing layers from the Keras library. The layers used were essentially:

- **Convolution layers:** responsible for creating a convolution Kernel that is convoluted with the layer input to produce an output tensor
- **Leveling layers:** which are responsible for leveling the input, without affecting its size
- **Fully connected layers:** all neurons in these layers are connected with all neurons in the next layers

The representation of these layers can be seen in Fig. 7, which presents the layers with the same names found in the Keras library.

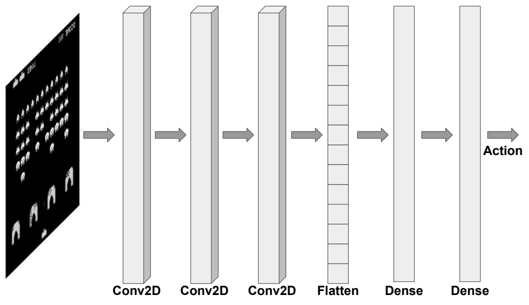

Fig. 7. Definition of the neural network used.

The origin of this layer configuration is based on the system used by the authors of [3], with the difference that, in this work, adaptations were used allowing the implementation of the network in the Keras library, together with the possibility of changing the number of neurons present in the output layer. That allows one to dynamically control the amount of movement of each game.

C. Agent Phases

In order to offer the developed agent experiences that allow it to understand how the game it was subjected to works, the agent's training function has a system of phases, enabling training without defining game rules. The system consists of three phases, which are:

- **Note:** The agent performs random actions to observe its consequences, filling its memory with the results of its first learned experiences.
- **Exploration:** After carrying out the observation phase, the agent starts to explore actions that are not in memory, aiming to fill the gaps created by bifurcations in the choices of past matches.
- **Selection:** In the last stage, the agent starts to select the actions better, aiming at those that offer the best reward.

D. Agent Training

For the developed network to have any real impact on the way the agent interacts with the environment, a training phase is necessary, which can be seen in the Algorithm 1.

The agent training phase consists of a cycle of iterations in which random actions are initially tested. As the number of episodes progresses and the number of states saved in memory increases, the agent's decisions start to present a more intelligent and consistent behavior.

E. Memory Optimization

Agent training requires a history of actions and results, enabling the agent to learn from mistakes and achievements made in previous iterations. However, the problem appears when the memory spent by this history starts to represent the highest cost of the agent. To solve this problem, a parameter was added to allow controlling the number of iterations that

Algorithm 1: Training phase.

```
number of episodes ← zero;
maximum number of episodes ← n;
while number of episodes < the maximum number of episodes do
    restart the game environment;
    while did not die in the game do
        check for possible action;
        send action to the game;
        check the result of this action;
        save the result of the action in memory;
    end
    increase the number of episodes;
end
```

the agent can store in its memory. We have chosen a scrolling windows approach, as can be seen in the Algorithm 2.

Algorithm 2: Scrolling window memory optimization.

```
if history size == maximum history size then
    remove the oldest state from history;
    add the new state to the history;
end
```

Using this optimization during the training, we expect that only the most recent of the advantageous actions will remain in memory. As the agent selects its actions, the memory replaces the oldest actions. Therefore, when the agent starts to choose the best actions based in the maximum score that can be obtained, the worst ones are erased from memory freeing up more space.

F. Agent Operation

After defining how each part of the agent should work and how all of the parts connect together, an agent with the structure seen in Algorithm 3 was developed.

The agent was developed aiming at the context of GGP, that is, the possibility of submitting it to learn to play any game, as long as the correct information is provided. To submit the agent to any test environment, it is only necessary to provide the current frame of the game in the form of an image represented by shades of gray and the score. To optimize the agent's learning process, the number of possible actions of the game is provided, allowing the narrowing of the agent's decision options.

IV. Games Selected for Testing

Using the Gym Library, four Atari 2600™ games with distinct mechanics were selected based on their characteristics, namely Space Invaders™, Breakout™, Demon Attack™, and Kung Fu Master™. The main reason for the selection of those games was the standardization of the information provided by the emulation environment, wherein each game it was possible to obtain the screen image, the current score, and the information corresponding to the game over state.

Algorithm 3: Agent operation.

```
receive the training environment;
extract the number of actions;
declare the neural network;
receive the maximum number of episodes;
episode number ← zero;
while number of episodes < maximum number of episodes do
    restart the game environment;
    define the current phase;
    while did not die in the game do
        get the screen image;
        get the reward;
        provide the screen image and the reward for the neural network;
        save the screen image and the reward in memory;
        if current observation phase then
            select a random action;
        else if current exploration phase then
            select an action that has not yet been explored;
        else
            select the best memory-based action;
        end
        send action to the game;
    end
    get the score;
    display the score;
    increment episode number;
end
save neural network parameters;
```

The proposed agent aims at the possibility of running in a simple computational environment, such as a home computer or smartphone. What the current state of the art corresponds is that most of the existing agents end up undergoing previous training, which makes it possible to obtain excellent scores in the games. However, this work aims at implementing an agent capable of autonomously learning game rules in less robust environments, enabling the use of the same agent in the role of the main player, or even the opponent of the player for different types of games.

V. Results and Discussion

For the simulations we made tests with a laptop equipped with an Intel i5-5200U processor with 2.20 GHz of clock and 8GB of DDR3L RAM. In order to measure improvements throughout the training of the proposed agent the obtained scores for each game were saved, generating a set of time series. Considering that the agent behaves randomly at the beginning of the training, and at the end it chooses its actions more accurately, the average initial and final scores obtained at each of the games are presented in the table I.

TABLE I
COMPARISON OF AVERAGE SCORES BETWEEN THE AGENT AND HUMAN PLAYERS.

Game	Initial Score	Final Score	Human Player
SpaceInvaders™	75	580	1200
Breakout™	2	24	26
Demon Attack™	100	640	1440
Kung Fu Master™	400	7500	3400

The data presented in Table I were obtained by submitting the agent to training of 1000 episodes in each game, with the size of the memory vector set to 80000 positions, where each position occupies the space of 88 bytes. It took approximately 20 hours of training for each of the games. To properly compare the performance of the agent in each of the games, Fig. 8 shows the percentage of performance gain at each of the games, where the human score is represented by the value of 100% and the initial and final scores are represented by the percentage improvement in relation to the human score.

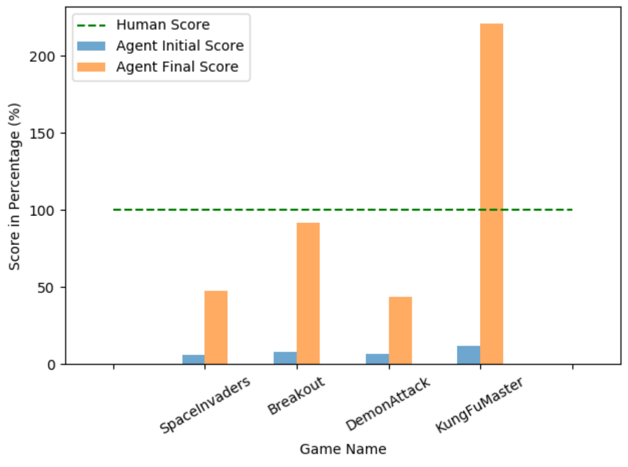

Fig. 8. Comparison of scores obtained for the agent and human players.

The score of the human player presented in the Table I was obtained from fifteen casual players who allowed the collection of their scores and use of it in this work. The value in the table is the average score obtained by the players in each of the selected games. All human player scores were obtained using the MAME™ [23] emulator.

Fig. 8 shows an interesting pattern, considering the selection criteria of each game, observing the agent's score in the Space Invaders™ and Demon Attack™ games, which have similar mechanics. These are games in which the player controls a ship and must destroy the opponents found at the top of the screen. In these games, if we compare the score obtained by the agent to that of a human, we will see that both games reached a similar percentage in the scores. In the Breakout™ game in which the proposal is different from previous games, the agent almost reached the human score. Finally, in the game Kung Fu Master™, the game selected for almost not sharing similarities with the other games, the score reached by the agent was higher than that of a human player.

The scores obtained in all games in this work could be compared with those obtained by the authors of [3]. However, unlike the present work, the work of the DeepMind group does not have a low memory limiter, which would require the proposed agent demanding a machine with approximately 80GB of RAM to be able to store the full history containing the states and rewards experienced by our agent. In addition to the large amount of memory needed, it would be necessary to increase the number of episodes in the training phase, which would result in a considerably larger amount of training time.

According to the information presented in the Table I, even with such restrictions the agent was able to determine what actions it should take in each game and learn to improve its score. Fig. 8 shows that in all games tested, the scores obtained at the end of the training improved significantly compared to the beginning of the training. Even though it did not obtain the same score as a human player in most tested games, the agent was able to infer useful information about the environment in which it is found.

The evolution of the agent was observed by selecting the game Kung Fu Master™ for a more detailed analysis, as it is the most complex of the tested games, and raw score data was logged along the playing episodes, as it can be seen in Fig. 9.

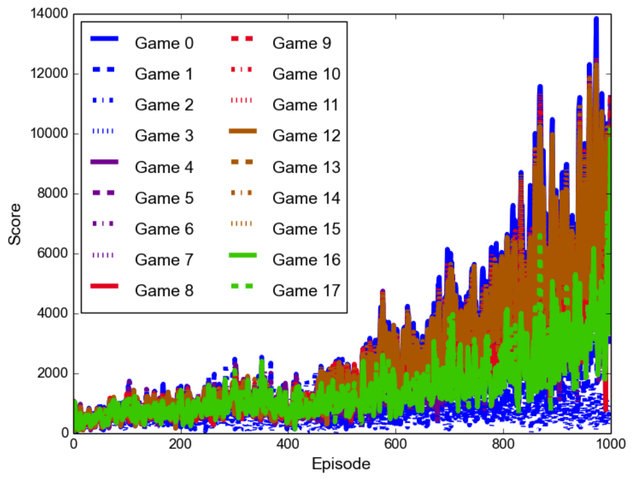

Fig. 9. Evolution of the scores in the game Kung Fu Master™. Raw score data from 17 games played shown. A tendency for improvement is observable, but not much more information can be obtained by visually inspecting the raw data.

Because the raw scores data is too noisy for visual interpretation, we filtered it with a sliding average for high frequency noise removal and better visual analysis, as it can be seen in Fig. 10.

From a closer analysis of Fig. 10, it is possible to notice that sixteen of the eighteen runs show the scores obtained by

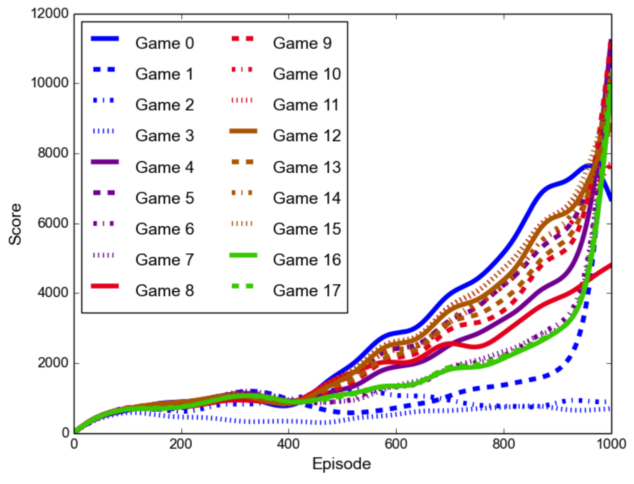

Fig. 10. Evolution of the agent scores in the game Kung Fu Master™, smoothed using sliding averages with first neighbours for better visualization, 1000 passes.

the agent growing up in a similar way along the episodes. What provides this feature is one of the key points of the game Kung Fu Master™: as the game consists of attacking opponents that emerge from the sides of the screen, strikes in the wrong direction do not affect the opponents and also leave the player vulnerable.

In the two executions of the training phase (Games 2 and 3 in Fig. 10) where the agent failed to realize that it should not deliver blows with its back to the opponent, the scores did not grow as much as it did in the other executions. When this characteristic was captured by the agent, the scores progressed in each subsequent episode. An important characteristic present in this game is that, at the ending of each stage it is necessary to defeat a boss, but instead the agent learned that staying at the beginning of the game just attacking opponents and obtaining points, without actually advancing in the game, still allows the obtaining of high scores without the need to face bigger challenges during the current match. That is a problem that could be addressed by specifying and properly weighting a global objective of advancing whenever possible and going through all stages and bosses, but doing so for such specific game would violate the main purpose of this work that is allowing the agent to learn as much as possible about the game by itself.

One important highlight is that, despite the apparent variation in the final scores for the curves presented in Fig. 10, there is no difference in initial conditions for each simulation other than the random initialization of the game parameters (these are hard-coded) as well as the weights in the neural network. So, multiple instances of the simulation could be executed in parallel and at the end the best outcome could be chosen as the final result.

VI. CONCLUSION

In agreement with the reviewed literature, we found that the combination of ML techniques such as Reinforcement Learning, Q-learning and Deep neural networks for the implementation of agents intended for GGP is feasible and provides promising results. We also noticed in the literature the constant appearance of agents linked to Reinforcement Learning techniques, so it suggests that this is an important topic to keep being investigated.

We also noticed that the number of games (runs) played in each game directly impacts its learned ability to play. However, the factor that had the greatest impact on game performance was the size of the memory available for storing the events. The greater the number of past events that the agent can recall, the sooner it reached a high score in the game. The problem with this approach is that, by largely increasing the size of the available memory, we start demanding too much of the space used in the temporary memory of the computer on which the agent is running and this could slow down the training. The optimization of memory allows the use of the agent in environments that need a considerable part of its computing power destined to the execution of the game.

By using a less robust environment than the one used by Mnih et al. [3], we noticed that it is still possible to obtain satisfactory results even with restricted resources. This suggests the possibility of using the proposed agent in a wider range of scenarios such as games on portable devices, where the processing capacity is usually more limited. From this, we can foresee a practical use for the combination of IA and ML techniques for GGP since many users own portable devices as platforms for personal entertainment through games. Although the playing skills learned by our agent don't seem to be as good as the ones in the aforementioned work, it is interesting to observe that the environment used in that work is comprised of machines with resources way beyond the reality of the common user. So, a model using a large amount of physical memory as the upper limit for training would make it impossible for the common user to train agents in casual games, where the player in many cases only has a smartphone or a personal computer.

Due to the fact that the agent was developed with a focus on playing multiple games with the ability to identify which is the one it is submitted to, the environment in which it finds itself may eventually change after the learning process and the agent showed the ability to learn the characteristics of the new game.

As it was seen in the presented results, each agent obtained scores superior to those of a casual human player in all of the tested games. However these agents faced situations in which they were unable to properly understand the games and consequently were unable to obtain satisfactory results. Without being forced to pursue a global objective, we can consider that the agent, although obtaining a better score, "cheated" to get these scores and isn't yet acceptable as a good opponent for a real game scenario.

So, it might be important providing the agent with ways to avoid just keeping increasing the score by making points locally (as it happened with the Kung Fu Master™ game), but instead seeking for a global objective that will also provide a higher score. This should be implemented keeping in mind that is should count for all games whether a global or more intermediate objectives exist or not.

For a future work, it is also intended to adapt the agent to perform as part of an actual game, such as an opponent or second player, instead of just the main player. In this way, the agent will have the opportunity to learn not only how to overcome the algorithms of the game agents, but the style of the human player with whom it is interacting.

Acknowledgment

The authors would like to thank Universidade Federal do Pampa through the AGP financial support for part of this work.

References

[1] *DeepMind*, 2019. [Online]. Available: https://deepmind.com/about/
[2] AlphaStar, "Alphastar: Mastering the real-time strategy game starcraft ii," *DeepMind*, 2019. [Online]. Available: https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/
[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529 EP –, Feb 2015. [Online]. Available: https://doi.org/10.1038/nature14236
[4] C. Browne, D. Soemers, E. Piette, M. Stephenson, M. Conrad, W. Crist III, T. Depaulis, E. Duggan, F. Horn, S. Kelk, S. Lucas, J. Neto, D. Parlett, A. Saffidine, U. Schadler, J. Silva, A. Voogt, and M. Winands, "Foundations of digital archæoludology," May 2019. [Online]. Available: https://arxiv.org/pdf/1905.13516.pdf
[5] X. Sheng and D. Thuente, "Contextual decision making in general game playing," in *2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, Nov 2011, pp. 679–684.
[6] A. Dockhorn and D. Apeldoorn, "Forward model approximation for general video game learning," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, Aug 2018, pp. 1–8.
[7] L. Galway, D. Charles, and M. Black, "Machine learning in digital games: a survey," *Artificial Intelligence Review*, vol. 29, no. 2, pp. 123–161, Apr 2008. [Online]. Available: https://doi.org/10.1007/s10462-009-9112-y
[8] I. Millington and J. Funge, *Artificial Intelligence for Games, Second Edition*, 2nd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009.
[9] G. N. Yannakakis and J. Togelius, *Artificial intelligence and games*. Springer, 2018, vol. 2.
[10] J. Fürnkranz, "Machine learning in games: A survey," *Machines that learn to play games*, pp. 11–59, 2001.
[11] Y. Bengio, "Learning deep architectures for ai," *Foundations and trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.
[12] N. Justesen, P. Bontrager, J. Togelius, and S. Risi, "Deep learning for video game playing," *IEEE Transactions on Games*, pp. 1–1, 2019.
[13] A. Jeerige, D. Bein, and A. Verma, "Comparison of deep reinforcement learning approaches for intelligent game playing," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan 2019, pp. 0366–0371.
[14] R. R. Torrado, P. Bontrager, J. Togelius, J. Liu, and D. Perez-Liebana, "Deep reinforcement learning for general video game ai," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, Aug 2018, pp. 1–8.
[15] G. Lample and D. S. Chaplot, "Playing fps games with deep reinforcement learning," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI'17. AAAI Press, 2017, p. 2140–2146.
[16] K. Narasimhan, T. Kulkarni, and R. Barzilay, "Language understanding for text-based games using deep reinforcement learning," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1–11. [Online]. Available: https://www.aclweb.org/anthology/D15-1001
[17] J. Hu and M. P. Wellman, "Nash q-learning for general-sum stochastic games," *Journal of machine learning research*, vol. 4, no. Nov, pp. 1039–1069, 2003.
[18] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, p. 2094–2100.
[19] M. Bowling, J. Fürnkranz, T. Graepel, and R. Musick, "Machine learning and games," *Machine Learning*, vol. 63, no. 3, pp. 211–215, Jun 2006. [Online]. Available: https://doi.org/10.1007/s10994-006-8919-x
[20] P. B. S. Serafim, Y. L. B. Nogueira, C. A. Vidal, and J. B. Cavalcante-Neto, "Towards playing a 3d first-person shooter game using a classification deep neural network architecture," in *2017 19th Symposium on Virtual and Augmented Reality (SVR)*, Nov 2017, pp. 120–126.
[21] *Gym Documentation*, 2019. [Online]. Available: https://gym.openai.com/docs/
[22] *Keras Documentation*, 2019. [Online]. Available: https://keras.io/
[23] *Multi Arcade Machine Emulator*, 2020. [Online]. Available: https://www.mamedev.org/