Deep Reinforcement Learning usage in games playing

Andrew Zakhary
Electronics Engineering
Hochschule Hamm-Lippstadt
Hamm, Germany
2210009

***Abstract*—**

***Index Terms*—Deep learning, Reinforcement learning, artificial intelligence, video games**

I. Introduction

Games have been a large part of our human society and history from the beginning. Evidence was found that early humans used $Talusbones$ as rudimentary dice as early as 10,000 years B.C [1]. As human society developed so did the games they played. The first board game found was in the Levant, dating back to around 7,500 years ago [2] with a couple of rows of holes carved into limestone. from then on board games developed more complexity from the relative simplicity of games as $Mancala$ in the Mediterranean to $Senet$ in Egypt to Go in chine, which is the oldest board game of mental skill in the world that is still being played [3]. Games like Go and $Chess$ (previously $chatarunga$) captivated the minds of humans for centuries because they are so complex that no one human could ever master them. This is due to the vast number of possible variations in a single game. In chess for example a typical game is around 40 moves, this could lead 10^{120} possible positions according to Claude Shanon [4].

In the last decades though, games have found their renaissance with the new technological revolution. The invention of the computer allowed games to take on a new shape. It's highly argued what could count as the first $videogame$ because the exact definition of the word $game$ is contested. what could be said with certainty though is that video games started around the 1950s in research facilities. After that video games became commercially available such as $ComputerSpace$ in 1971 and $Pong$ in 1971. $Pong$ Specially was important for the fact that the user could play against a computer system (commonly known as $Game - AI$ afterwards). From the late 80s and early 90s video games established themselves as a mean of entertainment and art. The growing capabilities of computers allowed the creation of more complex games till our current day.

With the growing computational power available also came the need for more powerful AI systems. AI systems developments proved to be essential for developing solutions that would have been very difficult using normal computational methods. In recent history multiple methods for developing such systems as supervised learning, unsupervised learning, reinforcement learning, deep learning and transfer learning among many others. A lot of effort has been put recently in the scope of developing AI systems that can play video games by themselves. This is due to the fact that video games can be a stepping stone in generating AI systems that have ability to work alongside humans in their tasks. Similar to the real world, many video games have an environment where the player is free to do a set of actions and only at the end of a set of time would he know if he was successful or not. The AI model would then need to learn the patterns that succeed and and unlearn the patterns that don't. This is why methods as Reinforcement Learning and Deep Reinforcement Learning are ideal in many video games playing situations. In this paper the workings of a Deep Reinforcement Learning model is explained as well as showing an implementation of such a model using a $Python$ script.

II. Basic Concepts

A. Reinforcement learning

Unlike other approaches of creating AI models where there's a correct and wrong answers, Reinforcement learning is useful where answers are not binary but rather can lead do different levels of success. This can be exemplified by the different tasks of identifying a cat from a dog and playing a game like Snake. In the former, for each image there's a correct and incorrect solution while in the later, there's various degrees of success for each attempt. In Reinforcement Learning, the system can be split into two parts Agent and Environment. The environment produces all the information about the state of the system. This collection of information is called $state$. The agent takes in the $state$, process it and depending on the internal model generates a corresponding output called $action$, The environment is also responsible for returning a $reward$ depending on the $action$ produced by the agent. The $reward$ evaluated by the agent in order to know whether the action was good or bad. Depending on this reward adjustments to the internal model of the agent in order to $reinforce$ actions that bring high rewards and inhibit actions that lead to bad or negative rewards. In RL the agent's internal model for generating actions is called $policy$, change from one $state$ to another is called $transition$. This could be visualized by the figure 1 The $transition$ from one state to another could be thought of as a $transitionfunction$. This transition function

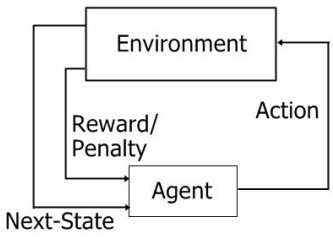

Fig. 1. Overview of an RL model [5]

is formulated as a Makrov Decision Process (MDP) [6]. The general formula for the transition function 1. This function means that the state at time $t+1$ is sampled from a probability distribution P. This is dependant on all previous states and the actions performed by the agent during these states.

$$s_{t+1}\ P(s_{t+1}|(s_0,a_0),(s_1,a_1),...,(s_t,a_t)) \tag{1}$$

$$R(\tau) = r_0 + \gamma r_1 + \gamma^2 r_2 + ... + \gamma^T r_T \tag{2}$$

The reward function could be summarized as in equation 2 where γ represents the $discount\ rate$. This factor is used to weigh the importance of recent actions and reduce the importance of earlier actions. For example if a model is to be trained to only care about the last action performed γ would be set to zero.

B. Deep Learning

Deep learning is a method of machine learning that depends on the use of neural networks [7]. Neural network consist mainly of 3 parts or layers, Input layer, hidden layer(s) and an output layer. The number of hidden layers can vary depending on the complexity and parameters of the problem that is to be solved with the model. An example of such a network can be shown in figure 2. Each point in the network is called a $node$ and the lines between each node is called a connection. Each connection has a specific $weight$. Similar to how our brains learn where with each repetition of a piece of information or performing a specific action, the connection between the neurons firing together gets stronger, so do the weights in a neural network. Given an example of a labelling task for photos of resolution 640*480 photos of traffic lights, in this case the input layer would have $640 * 480 = 307200$ nodes representing each pixel for each image and only 1 output node. If the output node has a value of more 0.5 then the system is guessing it's a traffic light. In the middle all that's happening is that each pixel is transformed into a value (usually [0,1]) and then a series of matrices multiplications is performed by the values of the weights which sum to the final value at the output node.

A deep learning model is usually first trained on a set of data where the answer is already known. In the training phase the model would be experimenting with various weights and comparing its outputs with the correct answers of the data set. The degree of correctness of the model is calculated by what's usually called a *loss function* [**?**]. After the prediction is made

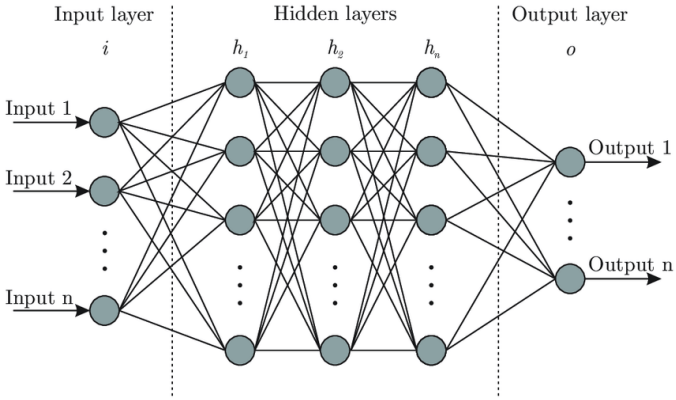

Fig. 2. Overview of a DL model [8]

and measured the *loss function* the model needs to adjust itself in order to be more accurate. This is where the *optimizer* plays a role. This is also a function that reworks the model (or rather the weights) in order to make it more probable to get a less *loss* next test.

C. Deep Reinforcement Learning

DRL is a method of machine learning which combines both RL and DL. The model in essence is an RL model with all the same properties, i.e rewards,states and actions,,etc. but the different thing is that the "learning" is happening on a neural network level. This means that each input is also mapped to a node where weights are created between it and the output. In this case however, the loss functions is replaced by the reward action from equation 2.

III. Implementation

A. Methodology

B. Example

IV. Conclusion

References

[1] Koichi Masukawa. The origins of board games and ancient game boards. In Toshiyuki Kaneda, Hidehiko Kanegae, Yusuke Toyoda, and Paola Rizzi, editors, *Simulation and Gaming in the Network Society*, pages 3–11, Singapore, 2016. Springer Singapore.

[2] SJ Simpson and ILE Finkel. The earliest board games in the middle east, 2007.

[3] Peter Shotwell. The game of go: Speculations on its origins and symbolism in ancient china. *Changes*, 2008:1–62, 1994.

[4] Claude E Shannon. Xxii. programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 41(314):256–275, 1950.

[5] Snehalika Lall, Arup Kumar Sadhu, Amit Konar, Kalyan Mallik, and Sanchita Ghosh. Multi-agent reinfocement learning for stochastic power management in cognitive radio network. 01 2016.

[6] L. Graesser and W.L. Keng. *Foundations of Deep Reinforcement Learning: Theory and Practice in Python*. Addison-Wesley Data & Analytics Series. Pearson Education, 2019.

[7] John D Kelleher. *Deep learning*. MIT press, 2019.

[8] Facundo Bre, Juan Gimenez, and Víctor Fachinotti. Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings*, 158, 11 2017.