FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION FOR HIGHER PROFESSIONAL EDUCATION NATIONAL RESEARCH UNIVERSITY

«HIGHER SCHOOL OF ECONOMICS»

1.1.1.1.1.1 Faculty of Computer Science

Anwar, Ibrahim

Интерпретируемые методы обучения с подкреплением

Methods of Interpretable Reinforcement Learning

Qualification paper – Master of Science Dissertation

Field of study 01.04.02 «Applied Mathematics and Informatics»

Program: Data Science

Student

_____

Anwar Ibrahim

Supervisor

_____

Кузнецов Сергей Олегович

Moscow, 2022

Abstract:

Deep learning “DL” applications have dominated most of daily life’s activities, but they still face a lot of criticisms because people are skeptical of why they should trust them, especially since these applications started solving critical problems, like autonomous driving and deciding who could get an organ transplant, this is why researchers and DL developers are now faced with a new challenge, to convince users that DL applications are trustworthy and this idea was the main reason why the concept of interpretability was introduced to raise peoples to believe in these applications, and in this work, we will go over the latest literature in the domain of interpretable Reinforcement learning, and we will try to create an interpretable model that will solve the Atari game “Pong”.

Contents

2 Table of Figures

3 Introduction:

Deep learning “DL” has held its position for the last years as one on the most important and fast-raising domains in the 20th century, because it is proving each day that it is a great tool that can make our life a lot easier, but as it is becoming more integrated with our lives, people are growing more doubtful of why we should trust DL applications especially when most of these applications are viewed as black boxes that take an input and magically spits out an answer, so researchers are now working to understand how do these black boxes do it, by introducing the concept of interpretability, which means creating humanely-understandable Deep learning applications, so in this work, we will in the third chapter define the concept of interpretability and explain why it is an important concept, then we will show how interpretability can view and how to measure that an application is interpretable. Reinforcement learning “RL” is a branch of machine learning that has made it possible to solve untraditionally DL solvable problems, like problems that need extremely huge data sets like driving a car or problems that can be solved heuristically like playing heuristic games “Go game” and they have proven that they can achieve astonishing results, so in the fourth chapter we will introduce the concept of Reinforcement learning, and we will define the RL problem, then we will combine these two concepts to introduce interpretable Reinforcement learning “IRL”, and list the proposed methods in the latest literature to achieve IRL, in the fifth chapter we will introduce self-attention models “SAM” as one of the approaches to achieve IRL, then we will introduce a self-attention model and use this model to solve one of the OpenAI gym environment “Pong”, finally we will conclude our report with the ideas of how we will improve this model to achieve the desired results.

4 Interpretability

4.1 What is interpretability

We now live in a world, where artificial intelligence is used in every concept of our lives. With the increasing dependency on artificial, comes the necessity to increase its credibility. One way to do so is by introducing the concept of Interpretability.

Interpretability was defined by Miller (2017) as[1]: "the degree to which a human can understand the cause of a decision." In other words, it is the level of accuracy on which users can predict and understand the model's decisions.

4.2 Importance of interpretability

Why is it an important concept[2], if our model gives us good results, we should still look for the reason behind the predictions because: (Molnar, 2022)

- Interpretability helps open our models and extract valuable knowledge, which leads to a better understanding of tasks: most machine learning models are black boxes that are trained using huge datasets and their accuracy can only be judged using metrics such as classification accuracy. So, using interpretability can help describe real-world problems more accurately.
- Learning the reason behind our predictions can help us understand more about the problem, especially when some models can provide results better than humans "for example models that play games like Go and Chess have already surpassed human players and introduced new unknown strategies in said games".
- Interpretability can help check if our model follows an ethical code because when designing an autonomous system, it is crucial to ensure that this system follows some pre-agreed-upon ethical code.
- , indeed,t some tasks are not that critical like movies recommendation systems where we don't care if the results are a hundred percent accurate, but other tasks "like finding the best candidate to receive an organ transplant" are very critical because they not only can save a person's life but the reason behind their decision is very important, we can even say that it is a part of the solution.
- Humans are curious, the human mind works in a way where we try to find relations between different concepts like if a person got sick, he or she would ask "why did I get sick?" or if my mobile stopped working so we would ask "why is my mobile stopped working?" because finding the reason is our way of understanding the world around us.
- Since most machine learning models are trained on datasets, it is a huge possibility that these models will be biased because it is very difficult to find a dataset with equal representations for all classes and interpretability can provide a debugging tool that helps us detect this bias.

[1] Miller, Tim. "Explanation in artificial intelligence: Insights from the social sciences." arXiv Preprint arXiv:1706.07269. (2017)

[2] Interpretable machine learning, the book and the chapter "importance of interpretability".

- Interpretable models are more user-friendly because they are interactive and can give the end user explanations to justify the outcome.

Those are a few reasons to show the importance of interpretability, but there are more reasons like making artificial intelligence more accepted by society and creating more fair and reliable models.

4.3 Classifying Interpretability

We can classify interpretable methods according to various criteria: (Molnar, 2022)

1- *Intrinsic/transparency or post hoc:*
   a. Intrinsic methods: are methods that aim to achieve interpretability by restricting the complexity of the model, or in general it is achieved by understanding the mechanism by which the model works, like short decision trees, and linear models. According to the interpretable part of the model, we can divide Transparency/ intrinsic models into three levels,
      i. Simulatibility: is when a human can perform all the calculations that the model does for a given input, in a resealable time.
      ii. Decomposability: when there is an intuitive explanation, for each part of the model "inputs, calculations…"
      iii. Algorithmic Transparency: this is when the used algorithm has a theoretical explanation and a guarantee of how this algorithm is going to perform.
   b. Post-hoc methods aim to achieve interpretability by applying methods to analyze the model after the training phase is completed, and by extracting information from this trained model to understand what it has learned exactly, and these methods can be categorized according to the explanations:
      i. Text explanations: this is when the explanations are presented in a text, basically like how humans justify their decisions verbally.
      ii. *Visualization: when a quotative determination is presented to show how the model is learning.*
      iii. *Local explanations: this is used when it is difficult to get a global explanation of the model's performance, so a local visualization of what the network is considering at each step to make its decision is presented.*
      iv. *Explanation by example: this is when the model outputs the nearest training example to justify its performance.*

2- *Classifying methods depending on the results of the interpretation method:*
   a. Feature summary statistic: this refers to the interpretable methods that provide summary statistics for each feature, for example, some methods will give a number for each feature to show its importance.
   b. Feature summary visualization: in this category, the feature summary is visualized as curves that show the features and the average predicted outcome.

c. Model internals (e.g., learning weights): intrinsic models fall under this category, for example, the weights in linear models, or features and thresholds used for splits in decision trees.
d. Datapoints: this refers to methods that return data points to make the model interpretable, and the idea behind these methods is to explain one data instance, the method finds similar data points by changing some features for which the predicted outcome changes in a relevant way.
e. Intrinsically interpretable methods: this category includes methods that try to approximate black box models with an interpretable model. The new model is interpretable by either using Feature summary statistic methods or by model internals methods.

3- Model specific or model-agnostic:
   a. Model-specific interpretation tools are limited to specific model classes, for example, when interpreting regression weights in a linear model is a model-specific interpretation.
   b. Model-agnostic tools are not limited to any machine learning model, and they are applied after the model is already trained "post-hoc" and these methods don't have access to the model internals "weights or structural information" the interpretability is achieved by analyzing input-output pairs.
4- Local or global:
   a. Local: is when our method explains an individual prediction.
   b. Global: is when the interpretation method explains the whole model.

4.4 Evaluating Interpretable methods

Evaluating interpretable methods depends on how useful the given explanations are and do they provide insights to help humans achieve some tasks, so to evaluate interpretable methods there should be a direct measure of the change in human performance on a given task before and after obtaining the explanations, and this measure will give a definitive conclusion about interpretability, but we can easily see that this method is not easily generalized and it is expensive. When considering the relationship between the exactness of task-specific user evaluations and their costs, (Doshi-Velez & Kim, 2017) presented a classification of interpretability evaluation methods, and it goes from most to least specific methods as follows:

1- Application-Grounded Evaluation: the user tests are performed when expert humans are performing the specific real-world task.
2- Human-Grounded Evaluation: the user tests are performed with humans and simplified tasks.

3- Functionally-Grounded Evaluations: the tests are not performed on humans, instead some proxy measures of interpretability "e.g. Local fidelity…" are calculated for the method on several datasets.

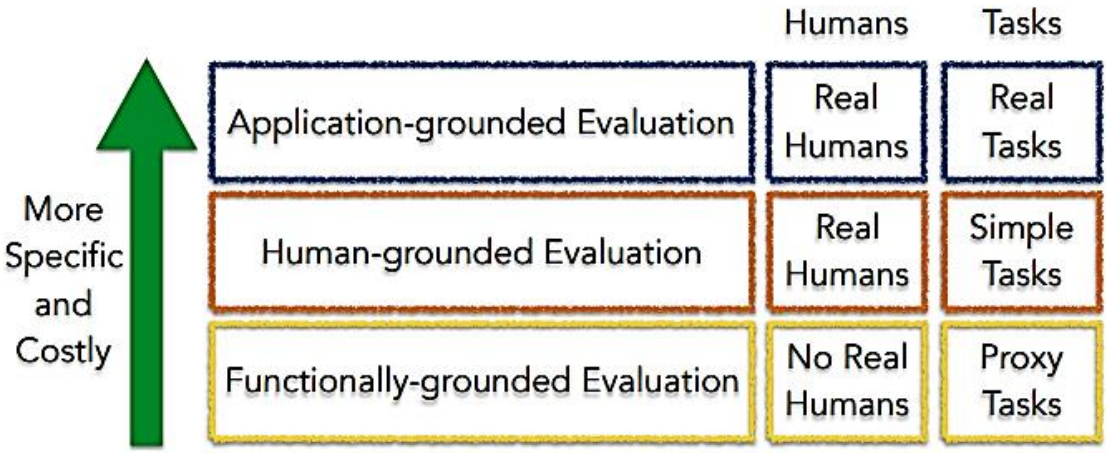


Figure 1: Taxonomy of evaluation approaches for interpretability. (Doshi-Velez & Kim, 2017)

4.5 How to achieve interpretability

There are several ways to achieve interpretability (Glanois, et al., 2021) the easiest one is to use only algorithms that create interpretable models, such as linear regression, logistic regression, decision trees, RuleFit, Naïve Bayes, and K-nearest neighbors, so the developer can choose algorithms that best suit the task that he/she is trying to solve. But for our work, we will try to achieve interpretability by using attention models, and we will explain our approach more in detail in the following chapters.

5 Reinforcement Learning

5.1 What is reinforcement learning

Deep learning algorithms can indeed be used to solve many different types of problems like image classification and prediction… but several types of tasks can't be automated using the traditional method of training an AI "artificially intelligent" model because it might be extremely difficult to find suitable datasets to train our models on, or because some tasks depend only on human heuristics and there is no logical explanation on how to solve them, it is impossible to create datasets to describe these type of tasks like "playing the game of Go (Silver, Schrittwieser, & Simonyan, 2017), or driving a car which includes several tasks from image processing to finding the best action to take in a specific situation" and from this idea, RL "reinforcement learning" was introduced,

Reinforcement learning is a branch of Machine learning where a model/an agent is created to learn what to do and how to map observations to actions, to achieve the goal of maximizing a numerical reward signal, and the agent learns this decision-making process by interacting with

an environment and performing some actions to optimize the resulting feedback from this environment. (Sutton & Barto, 2014,2015) The following figure can explain an RL problem:

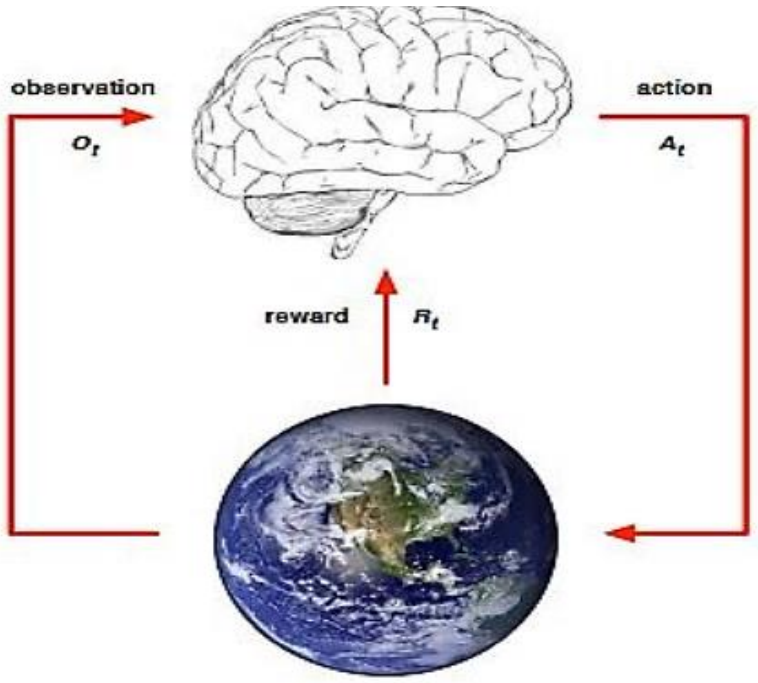


Figure 2: the RL problem is a closed-loop problem that has several elements, the agent here is represented by a brain, and the environment in this figure is represented by the earth, there are also several interactions between these two elements which will be explained in the following chapter.

The RL problem consists of several elements, which are:

- The agent: receives "observations" that explain the environment's status, and according to these observations it chooses and performs the best action, the performed actions will change the environment's status, which will invoke the environment to give the agent a reward and a new observation to describe the new status, the agent will use the reward as a metric to measure the efficiency of the performed action… and this loop will continue until we find the best action for every observation, and this is how we train our agent.
- The environment: will give the agent observations to describe its status, it will also be affected by the agent's actions, leading it to produce rewards that will help train the agent.

To further explain the relationship between the agent and the environment we will show the following figure:

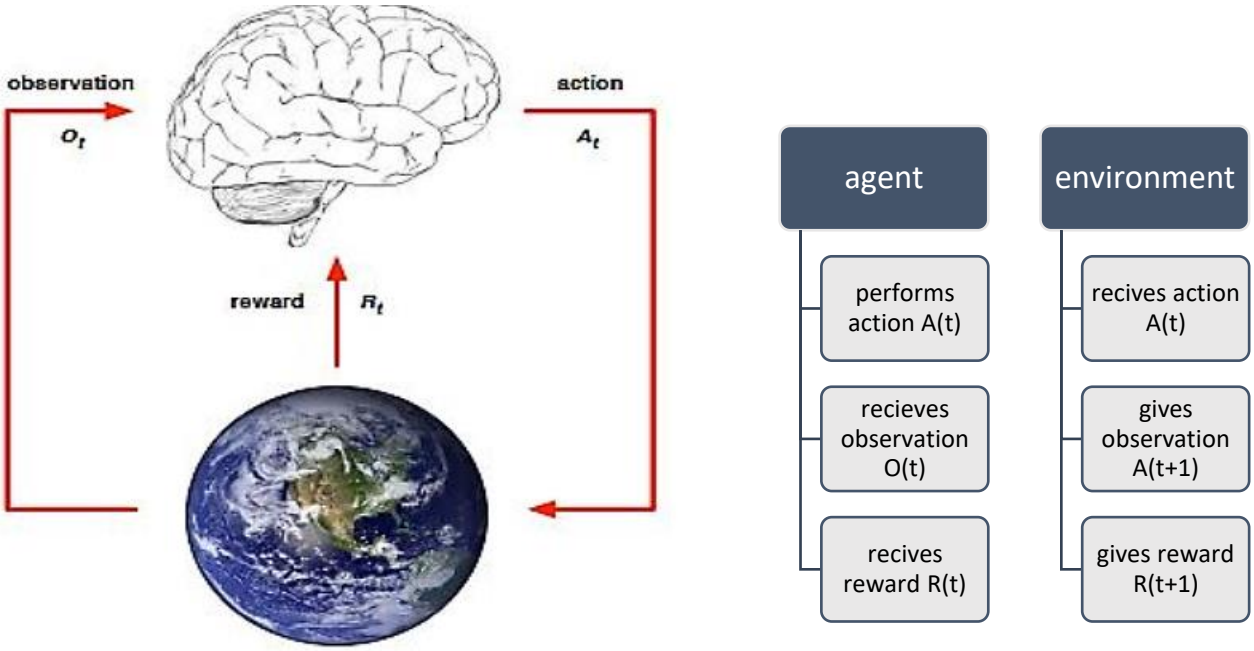


Figure 3: the relationship between the agent and the environment at a time step (t).

We will not go further in explaining RL for Now but for more information, you can see “Using RL to solve Atari games”

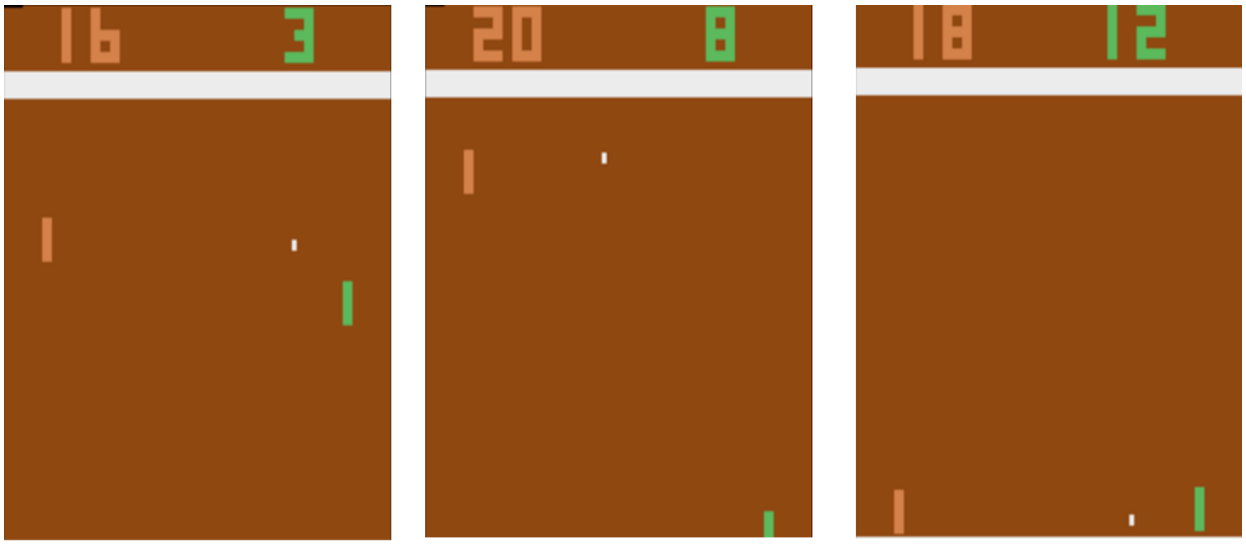
Deeper dive into the RL problem:

Our job is to create an agent “the brain in figure 2”, able to understand the input, which can be as simple as an array “like frozen lake game of (Frozen Lake, n.d.) gym” or as complicated as a high-dimensional image, then learns from this input using the previously accumulated experience, to create knowledge representation and reasoning, which will finally enable the agent to create a model of this environment.

Generally, the RL problem is modeled as a Markov Decision Process “MDP”, and an MDP is defined as a tuple, as such:

$$< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$$

$\mathcal{S}$: is the set of states and it could be either a finite or infinite set, and the states can be either discrete or continuous states, and a State is a summarization of the chronological record, containing all the useful information, for example when talking about Atari games a state could



Frame (1) Frame (2) Frame (3)

Figure 4: This figure shows examples of different frames of the game Pong of OpenAI gym (Pong, n.d.), where the frame a multi-dimensional image of the game board.

be the screen frames or the last four frames, and figure 3 shows what we mean by a frame.

$\mathcal{A}$: is the set of available actions, for example, if we are talking about an Atari game like Pong, the available moves are sliding the paddle up or down and you can get more information about this game from the documentation (Pong, n.d.)

$\mathcal{P}$: is the transactions array that shows the probabilities of moving from a state "s" to another one "s'" by performing some action "a", and the relationship at a timestep "t" is as follows:

$$\mathcal{P}_{ss'}^{a} = \mathbb{P}[s_{t+1} = s' | s_t = s, A_t = a]$$

$\mathcal{R}$: is the reward function, and it expresses the reward at a timestep "t" that the environment gives the agent after taking an action "a", given a specific state "s".

$$\mathcal{R}_{s}^{a} = \mathbb{E}[R_{t+1} | s_t = s, A_t = a]$$

γ: is the discount rate, and it is used to discount the expected future rewards, and its values are in the range [0,1].

On the other hand, the environment could be either a fully observed environment, or a partially observed environment and the following table shows the difference between the two types:

Fully observed environment	Partially observed environment
• The agent can see everything in the environment. • The environment state and the agent's state are the same. • This type of problem is a Markov Decision Process (MDP)	• The agent monitors the environment indirectly, for example, the poker player can only see the cards facing up. • The environment state is different from the agent's state. • This problem is a partially observable Markov Decision Process (POMDP)

From the previous explanations, we can notice that the challenge that faces solving the RL problem is that the agent doesn't know the transition function and the reward function, and we need to create an agent to choose the best action for each state to maximize the expected discounted sum of current and future rewards.

5.2 Reinforcement Learning accomplishment

Deep reinforcement learning (DRL) helped solve tasks that were previously considered to be the exclusive domain of humans. DRL has already proved to be one of the strongest techniques that can be used to solve decision-making problems, such as autonomous driving (Kiran, et al., 2021), and beating the world's best players in several board games like the game of Go, Chess, shogi and Atari games (Alphago, n.d.), DRL algorithms have also achieved impressive results when it comes to resource optimization, by learning about the vacancy and allocating resources to the waiting jobs, resulting in less delay (Hongzi, Alizadeh, Menache, & Kandula, 2016), and other applications like personalized recommendations, and real-time bidding and advertising.

5.3 Interpretable Reinforcement Learning

Now that we are convinced about the importance of RL and DRL, and the importance of interpretability, we can easily justify the need and the importance of interpretable Reinforcement Learning, and in the last years, several techniques were proposed to achieve that, for example (Glanois, et al., 2021):

1- trying to get interpretable input for the model, and there exist several approaches:
   a. Structured Approaches: the idea is assuming that there exists a pre-given structured representation of the environment, that can be modeled through objects and their relations, for example, relational MDP and Relational RL…
   b. Learning symbolic representations: if the input is high-dimensional raw data, we can try to extract some information and use this new information as the new input, for example using object recognition, or relational representations…
   c. Hierarchical approach: instead of removing the whole input data like in symbolic representations, we can try to incorporate the symbolic knowledge, with sub-

symbolic components, for example, the modularity method and, symbolic planning.

2- trying to use interpretable Agents to do the decision-making process, and there are several approaches:
   a. Direct approach: where we attempt to direct search for the policy, for example, Decision Trees, Formulas, fuzzy controllers, programs, or graphical models.
   b. Indirect approach: we first train a non-interpretable model, then try to convert this model into an interpretable one, for example, Decision trees and variants...
   c. Architectural inductive bias: in this approach we chose interpretable architectures to model the policy network or value function, for example, relational inductive bias. Logical inductive bias, or attention-based inductive bias "which is what we tried to use"
3- trying to make the preference step interpretable "the step that shows the preferred policy", for example, transition model, probabilistic models, or Neural network-based models…
4- mixing any of the previous three options: a good example is a work done by (Custode & Iacca, 2022), where they also solved the pong environment, by combining the interpretable input approach, and the interpretable decision-making approach. They used and the following figure explains their approach:

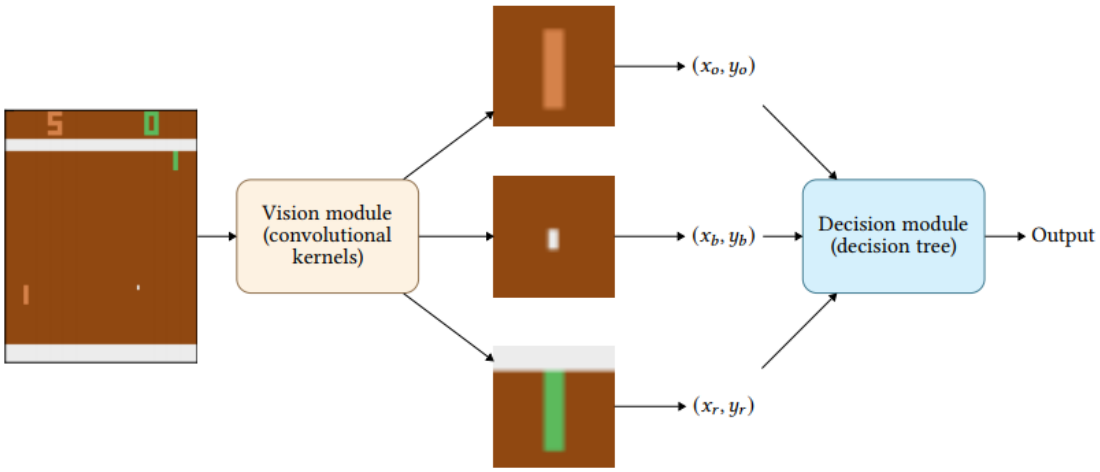


Figure 5: (Custode & Iacca, 2022) proposed pipelines to solve the Pong environment.

5.4 Our environment "Pong-v4"

To study the problem of interpretable Reinforcement Learning, we need first to decide on a suitable environment where we can test our interpretable model, and for that purpose, we will check the environments that are available in the gym Library (Gym Documentation, n.d.) and specifically we will choose Pong Atari environments (Pong, n.d.), the following figure, shows the environment's frame:

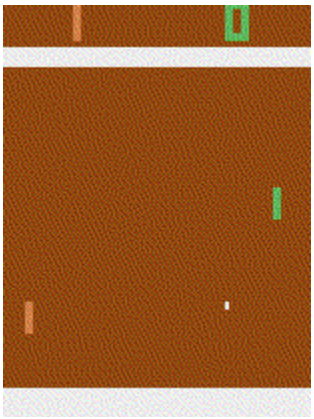


Figure 6: Pong-v4 frame

Since we know that the environment is determined by several attributes, we mentioned before, that the environment gives observations, and a reward to the agent and it can receive actions, so in the following, we will show these attributes:

The attribute	The value
Observation	Frames with the dimensions (210,160,3) ==(height, width, 3 colors)
Action Space	Discrete(6) ['NOOP', 'FIRE', 'RIGHT', 'LEFT', 'RIGHTFIRE', 'LEFTFIRE']
Reward	Scalar value (1) for a winning rally Scalar value (-1) for a losing rally
Episode terminating	The player reaches a score of 21 Episode length > 400000
Solved Requirement:	The average score of 17 over 100 consecutive trials

Finally, we need to note that both actions 2,4 or “RIGHT, RIGHTFIRE” and the actions (3,5) or “LEFT, LEFTFIRE” Are the same movements with different amplitudes.

6 Self-attention Models/Our model

Graph neural networks are a type of neural network that operates on the graph structure, and the graph structure, can be used to represent a wild variety of real-world examples like “social networks, people are nodes and friendships are the edges between them or raw images pixels are nodes and an edge will exist if two pixels are adjacent, or any other relational model, where objects are nodes and the relations are represented by edges”. In this chapter, we will introduce Self-attention models (SAMs), which will be used to create a graph neural network “GNN”, that will be used to understand the raw images “frames” and convert it into a graph structure, and the last component will be interpretable. We are hoping that if we train our model on pictures of people playing football, the model will be able to connect the players with the ball, and the ball with the basket…

Attention models were first introduced and developed for machine translation applications, their use has been generalized and spread to many other applications, and to understand the general idea behind how SAMs work let’s consider the following example where we want to translate a phrase from French to English:

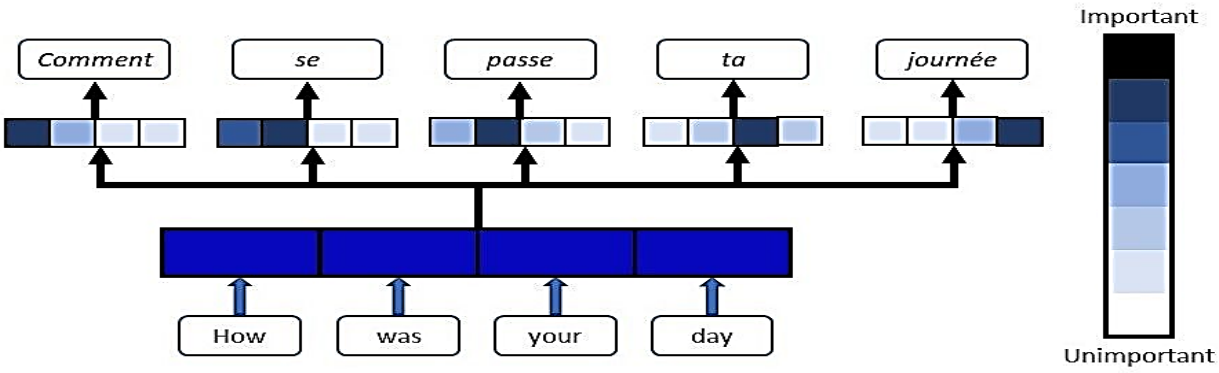


Figure 7:Weights are assigned to input words at each step of the translation. (Loye, 2019)

So the model will take this sentence as an input and we will calculate the activations of this input, and the inputs for the second layer are the sum of the activations of the previous layer, multiplied by an operator “alpha in our example”, and this operator will add weights to each activation or for each word of this sentence, and this operator will show how much does the output depends on this word in particular to get the output translation, or in other words the amount of attention that was paid for each world of the input sentence to get the output sentence, the following figure 6 explains this process:

Another idea is how can we calculate the operator alpha, and to do so we have to create a small neural network usually only one neuron and this NN will take as input the bigger network’s hidden state of the previous time step S<t-1> “shown in the following ”, and the features from

the time step t A<t>, and this will give us all we need to calculate the operator alpha (Bahdanau, Cho, & Bengio, 2015)

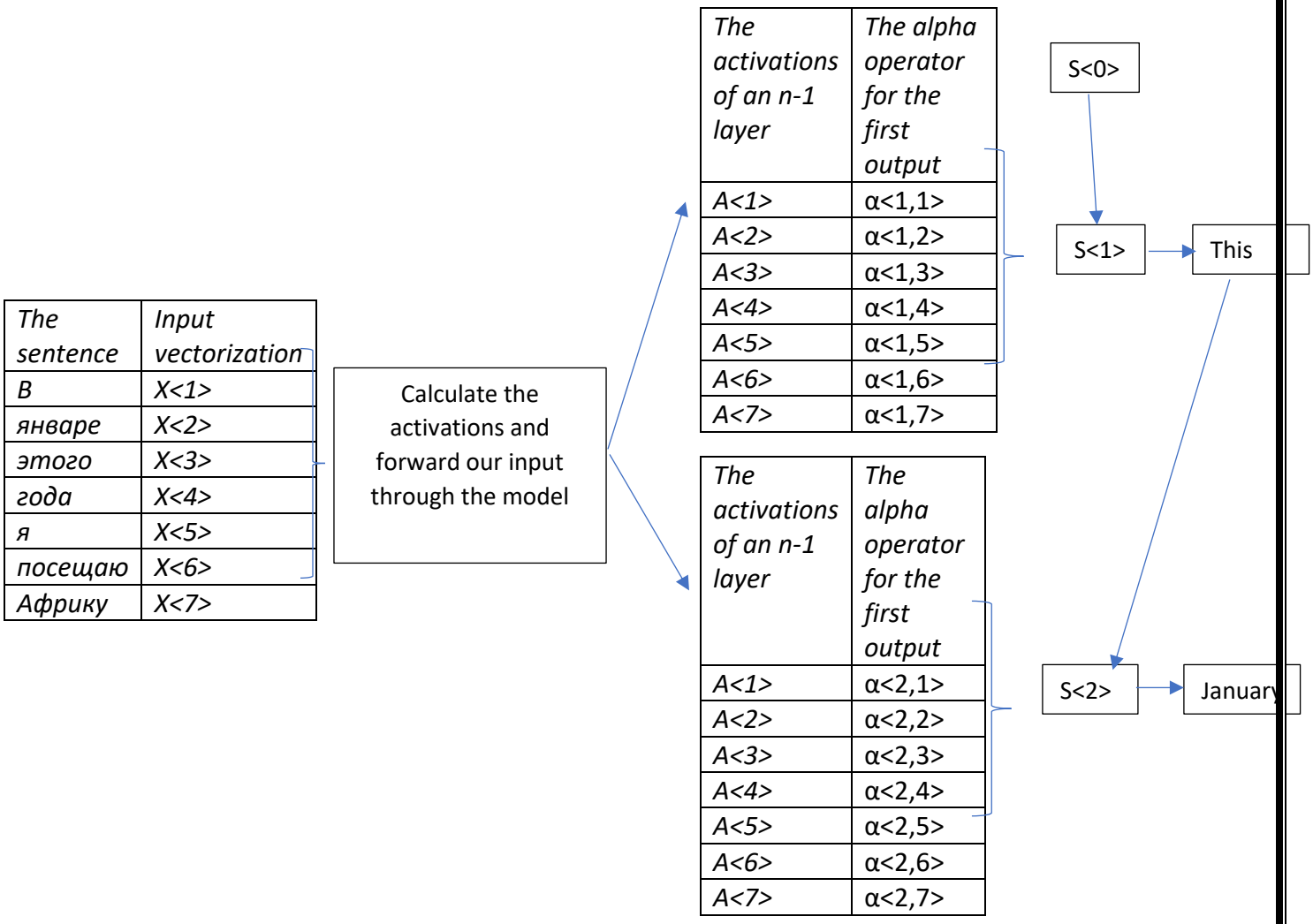


Figure 8: a simplified figure to show how self-attention works. The figure only shows the output of the first two words, but the operation will keep going until the model gets the translations of every word in this sentence.

There are several implementations of self-attention models, but in our model, we will depend on the implementation that was presented in the paper "Deep reinforcement learning with relational inductive biases" (Zambaldi, Raposo, Santoro, Bapst, & Li, 2018) , and I will specifically describe the model that we will use to try to solve our environment:

Our model is specifically called the multi-head Relational model, which is one type of relational model but with a few differences:

6.1 The Input

We have seen figured from our environment's frames that "pong-v4" observations lack a lot of important information like which direction is the ball headed, and there are some useless pixels that will just make our computations more expensive without yielding any benefits, so it is important to define some functions that will perform some preprocessing

Preprocessing(frame):

1- ***Gray= Convert image to gray image (frame)***
2- ***Cropped= Crop useless pixels(Gray)***
3- ***Normalized= Normalize(Cropped)***
4- ***Final= resize the image(Normalized)***

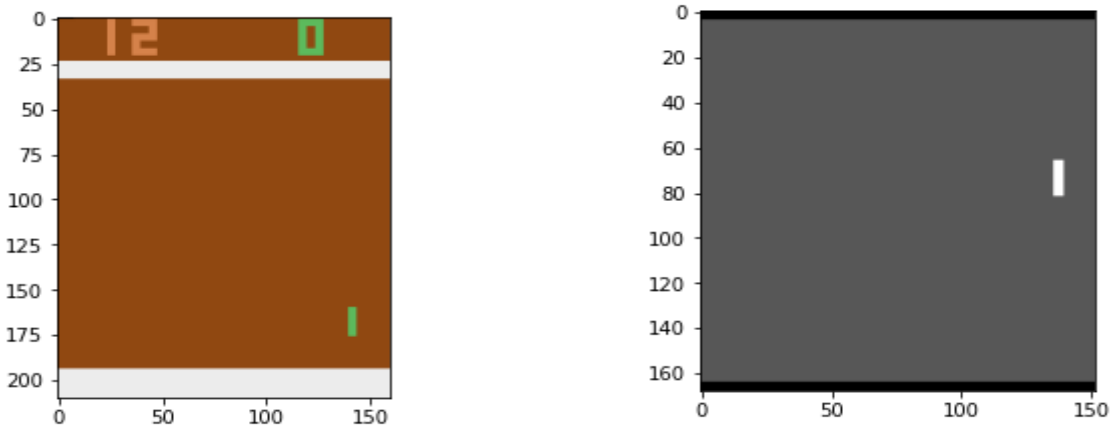


Figure 9: the figure on the left shows the observations before preprocessing, and the one on the right shows the result after preprocessing.

To solve the problem of finding the direction of the ball, we will take 4 consecutive frames instead of one, as our state, so in conclusion, our model's input is shown in the following figure:

We can easily see that with this type of state, the agent/model will be able to determine the direction of the ball.

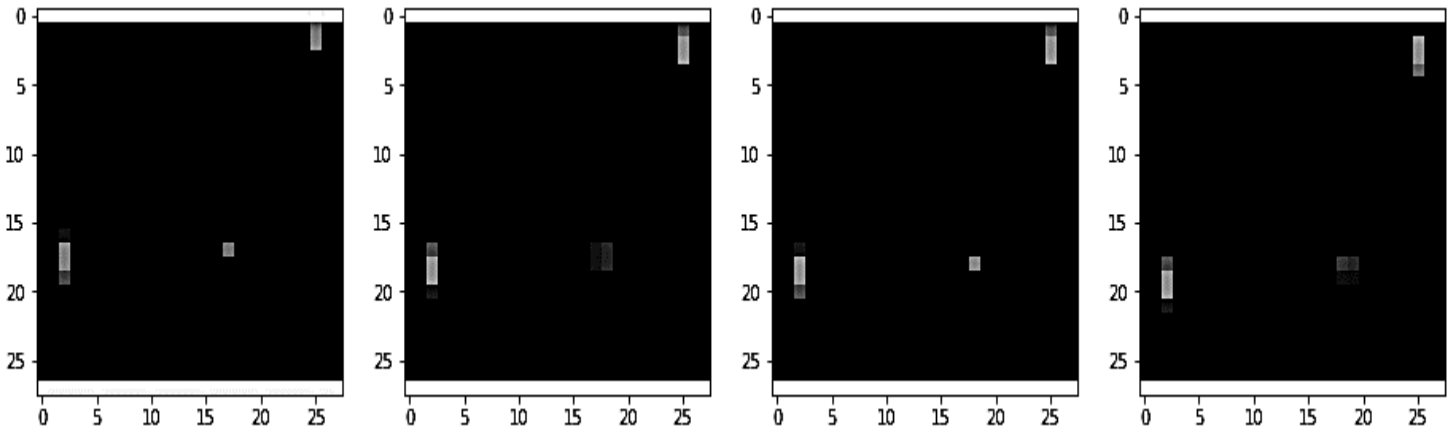


Figure 10: Our Relational Agent's input state.

6.2 The model/Agent

Our model will take the previously defined input, then it will go through the layers, the following figure shows the structure of the multi-head attention model, and we can easily see that the transformer has three separate linear layers, each with a special function, and they are the Query, the key and the Value, and to understand the relationship between this three layers let's compare them to information retrieval systems "for example google", we know that each information retrieval system has this three components, so let's assume we want to search for the sentence "harry potter" in a dataset of documents:

1- The Query: represents the sentence that we are trying to look for "harry potter".
2- The Key: represents the documents that exist in the datasets.

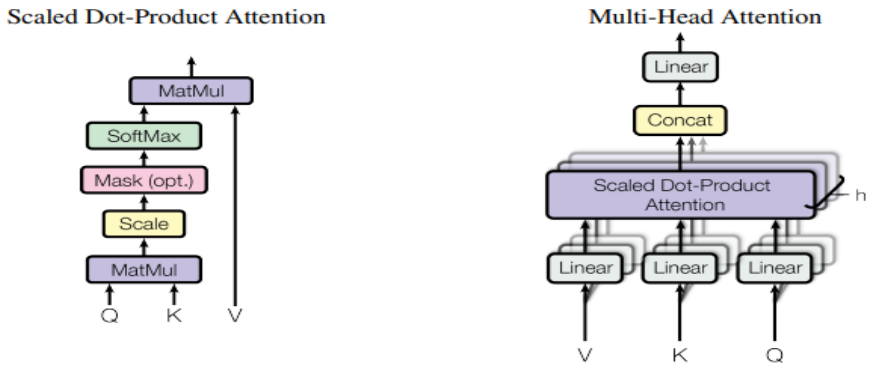


Figure 11: the (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel. (Vaswani, et al., 2017)

3- The Value: represent a search result from the keys that are most similar to the value.

For the scaled Dot-product Attention, the results of the query and key layers will be fed to a similarity measure "matMul in the previous figure" then we will perform some operations on the result to get attention filters and the formula is:

$$attention\ filter = sofmax(\frac{Q * K^T}{\sqrt{d_k}})$$

After that we will calculate the attention using the formula:

$$attention(Q, K, V) = sofmax\left(\frac{Q * K^T}{\sqrt{d_k}}\right) * V$$

Where:

Q: is the result of the query linear layer.

K: is the result of the key linear layer.

d_k:is the diminution of the key vector, this value is similar to the paper (Vaswani, et al., 2017), but it can be different.

V: is the result of a key linear layer.

And if we want to convert to multi-head Attention we will perform the previous computations h times, where h is the number of heads, and each head will give us different representation subspaces at different positions, so we will get the formula:

$$MultiHead(Q, K, V) = Concat(head_1, head_2, \dots head_h)W^O$$

Where: $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$

where the projections are parameters matrices $W_i^Q \in \mathbb{R}^{d_{model}\, X\, d_k}, W_i^K \in \mathbb{R}^{d_{model}\, X\, d_k}, W_i^V \in \mathbb{R}^{d_{model}\, X\, d_k}$, and $W^O \in \mathbb{R}^{\,hd_k X d_{model}}$, and the computations for the heads will be done parallelly.

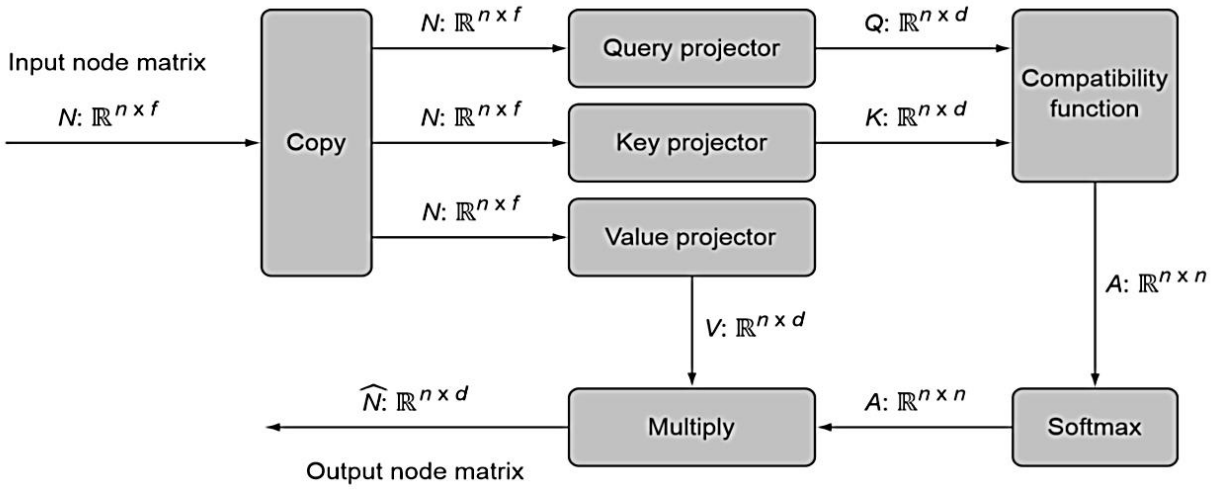


Figure 12: the model that we will try to implement.

As we already know the Compatibility function is the function that measures the similarities, between the Query and the keys, and we already mentioned that we can use “dot product” as a compatibility function, but since it is not stable, we can replace it with “additive attention” which is more stable.

6.3 Training the agent

First, let us differentiate between sparse and dense awards, sparse rewards are those given for only a small handful of states/events. Whereas dense rewards are given to evaluate the agent's actions in many different states. So, it is noticeable that having a dense reward is far better, but the "Pong" environment has a sparse reward, which means our agent could get only one single reward for approximately 10 back-and-forth rallies, each of which contains closer to 10 actions/moves.

To train the model, we will use Double Q-Learning approaches (Hasselt, Guez , & Silver, 2015), which means we will have two networks, one the value network to choose the appropriate action, and the other is the target network for outputting the actual Q-values, the update function is:

$$Q_{new} = r_t + \gamma \, max \, (Q`(S_{t+1}))$$

Where:

Q': is the target function.

r_t: the reward from the previous timestep.

6.4 The results

Our results using the previously defined model, were not good, because we need to depend more on a stochastic policy at the beginning of the play, and from the stochastic policy we can reach the perfect score. Another reason is the output of the Agent is of the shape (6), where we can see from chapter (2.3 our environment pong-4) the actions are redundant, so it is possible to make the shape of the output two number one shows the probability of sliding the paddle down, and another one to show, and the other represents the probability of sliding the paddle up.

Another possible improvement is to introduce discounted rewards instead of just taking the reward given by the environment as it is, this will help add more importance/ weight to the recent awards by multiplying with a discounting factor.

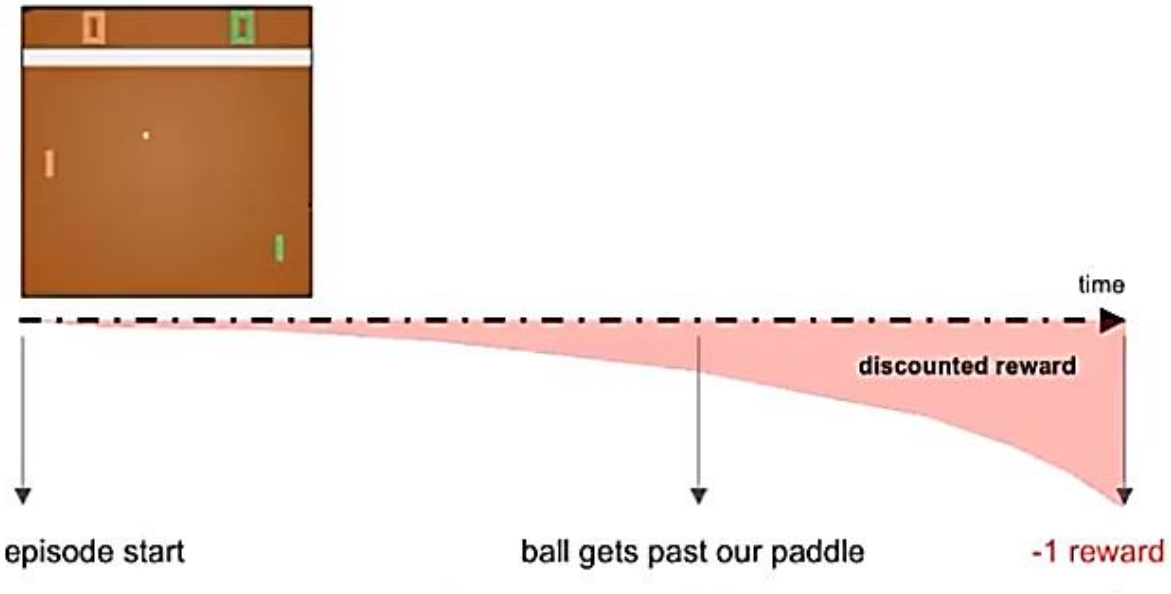


Figure 13:The effect of discounting rewards — the -1 reward is received by the agent because it lost the game is applied to actions later in time to a greater extent (Karpathy, 2016)

Another improvement is instead of using a batch of 4 frames as our input, we can just take the difference between the current frame and the previous ones to show the direction in which the ball is moving, and this will also reduce the computations cost.

7 Conclusion:

In this report we defined interpretability for Machine learning models as the degree to which a human can understand the cause of a decision made by this model, then we explained why interpretability is important and how it helps humans in trusting ML models, helps to gain knowledge that can raise human understanding of the problems solved using ML models, it also helps us debug these black boxes, so we can build better models in the future. Then we classified interpretability according to several criteria, so the interpretable method could be either *transparency or post hoc* according to if the used model is interpretable by itself or if some computations were applied to the already trained model to achieve interpretability, another proposed criteria is according to how *the results of the interpretation method* are viewed, the third proposed criteria depends on wither the interpretable method explains only one local example or if it explains the whole model, and it classifies interpretability to local or global. Then we explained how to evaluate our interpretability by measuring the improvement in humans' performance in imitating the ML model before and after achieving interpretability. After that, we talked about RL and we defined the RL problem, as a problem with an agent and an environment where the agent learns a decision-making process by interacting with the environment. Then we proposed self-attention models as a tool that will be used to achieve interpretable RL in solving the Pong environment, and we described the model's architecture, and how some ideas for how we will get our model to work better in the future.

‘

8 References

[1] *Alphago*. (n.d.). Retrieved from Deepmind: https://www.deepmind.com/research/highlighted-research/alphago

[2] Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural Machine translation by Jointly Learning to align and translate. *ICLR.*

[3] Custode, L. L., & Iacca, G. (2022). Interpretable pipelines with evolutionarily optimized modules for reinforcement learning tasks with visual inputs. *cs.LG.*

[4] Doshi-Velez, F., & Kim, B. (2017). Towards A Rigorous Science of Interpretable Machine Learning. *stat. ML.*

[5] *Frozen Lake*. (n.d.). Retrieved from OpenAI: https://www.gymlibrary.dev/environments/toy_text/frozen_lake/?highlight=frozen+lake

[6] Glanois, C., Weng, P., Zimmer, M., Li, D., Yang, T., Hao, J., & Liu, W. (2021). *A Survey on Interpretable Reinforcement Learning.*

[7] *Gym Documentation*. (n.d.). Retrieved from OpenAI gym: https://www.gymlibrary.dev

[8] Hasselt, H. v., Guez , A., & Silver, D. (2015). Deep Reinforcement Learning with Double Q-learning. *cs.LG.*

[9] Hongzi, M., Alizadeh, M., Menache, I., & Kandula, S. (2016). Resource Management with Deep Reinforcement Learning. *the 15th ACM workshop on hot topics in networks*, (pp. 50-56).

[10] Karpathy, A. (2016). *Deep Reinforcement Learning: Pong from Pixels*. Retrieved from http://karpathy.github.io/2016/05/31/rl/

[11] Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A. A., Yogamani, S., & Pérez, P. (2021). Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Transactions on Intelligent Transportation Systems.*

[12] Loye, G. (2019, September 15). *attention-mechanism*. Retrieved from https://blog.floydhub.com/attention-mechanism/

[13] Molnar, C. (2022). *Interpretable Machine Learning.*

[14] *Pong*. (n.d.). Retrieved from OpenAI: https://www.gymlibrary.dev/environments/atari/pong/?highlight=pong

[15] Silver, D., Schrittwieser, J., & Simonyan, K. (2017). Mastering the game of GO without human knowledge. *Nature*, 354-359.

[16] Sutton, R. S., & Barto, A. G. (2014,2015). *Reinforcement Learning: An introduction.* London, England: The MIT Press.

[17] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). Attention Is All You Need. *cs.CL.*

[18]Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., & Li, Y. (2018). Deep reinforcement learning with relational inductive biases. *International conference on learning representations.*