Exploring Multi-Output Regression and Reinforcement Learning for Game Adaptivity

João Álvaro Ferreira
Mestrado Int. em Engenharia Informática e Computação
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal
up201605592@fe.up.pt

João Carlos Maduro
Mestrado Int. em Engenharia Informática e Computação
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal
up201605219@fe.up.pt

João Augusto Lima
Mestrado Int. em Engenharia Informática e Computação
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal
up201605314@fe.up.pt

Mariana Neto
Mestrado Int. em Engenharia Informática e Computação
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal
up201606791@fe.up.pt

***Abstract*—This document details research made into the subject of adaptive games, using reinforcement learning and supervised learning methods such as multi-output regression and individual regression. A prototype of the video game Breakout, along with simulated personalities, were the basis for data collection. The research developed produced promising results for the supervised learning methods, multi-output regression in particular, which has shown to be a viable method for implementations of game adaptivity.**

***Index Terms*—machine learning, data mining, game adaptivity, multi-output regression, reinforcement learning, supervised learning**

I. Introduction

The growth of the video game industry has made game adaptivity an increasingly relevant subject for study to prolong one's game longevity. One of the easiest forms of adaptivity is difficulty adjustment in levels[1]. However, this adjustment is manual and required the input from the developer and the player, often making a choice with incomplete information, even before playing the game to assess what each difficulty level entails, leading to sub-optimal experiences. This task can be made automatic by modifying the game to accommodate the different player's personalities and optimal game flow[2].

Ensuring the players have optimal satisfaction[3] is one of the most common goals for interactive media, being paramount for player retention, attracting new players and overall increasing or maintaining a player-base.

This report is an in-depth analysis of research made into the subject of game adaptivity, using multi-output regression and reinforcement learning. The techniques were implemented on data collected in a prototype of the video game Breakout developed by the researchers, along with diverse and complex simulated personalities[4]. The aim of this project was to explore the quality of the current state of the art regarding game adaptivity and compare the alternatives of multi-output regression and reinforcement learning.

II. State of the art and Objectives

A. Adaptivity

In the realm of game adaptivity, the standard is manually changing the different aspects of games to fit arbitrary difficulty levels. According to [3] there are many options to change a game, ranging from changing the world the player interacts with, to the mechanics of the game and music elements, to other artificial or simulated elements that the player may encounter. These changes made to the game can steer the user into performing according to what the developers expected [5], and getting the experience and feedback that was intended[6][7]. In this project, the changes to the parameters are only made at the start of each game, but in most scenarios these changes would occur during the game session.

Although options for users to manually adjust their game experience have also grown more advanced and detailed in recent years, automatic adaptation has been a constant presence in the industry with much success, such as the case of Mario Kart[8] and Left 4 Dead[9]. This project will explore how game adaptivity using data mining techniques can improve a player's experience and game flow.

By adapting a game, the players and users of the product can have different experiences, affecting the way they view, play and experiment with the game. Several applications of the adaptive game have been developed for different purposes and objectives, such as learning environments to teach[10][11] or to maintain a sustainable player base for the longevity of the game. These applications of adaptivity alter the different parameters and mechanics of the game to suit their needs. For example, in an educational computer game, the objective is for the player to acquire knowledge while paying, however, there needs to be a balance between promoting the learning processing and creating satisfaction and game flow[12].

The main objective of this work is to establish some performance benchmarks in the field of game adaptivity.

Specifically, the aim is to explore the performance of newer, less commonly used machine learning techniques such as multi-output regression and reinforcement learning, which is a more frequently used approach.

B. Machine learning

In terms of reinforcement learning, this type of machine learning is more commonly used (within the context of video games) to enhance the performance of enemies or "artificial players". Several approaches to reinforcement learning are available, and in this project, we will use the framework Sable Baselines3[13], which provides implementations of state-of-the-art reinforcement learning algorithm such as Proximal Policy Optimization[14].

The main focus of this paper is to utilize and compare multiple approaches of supervised learning algorithms with reinforcement learning. These approaches [15] include: building a single-target output regression model for each of the outputs and building a multi-target output that takes into account targets correlation and deliveries the multiple output variables needed[16]. The topic of multi-output regression is still a recent topic in terms of machine learning and the application on adaptivity in video games is still open to results and experiments.

C. Personalities

Each player demonstrates a certain personality and skill while playing a game. Consequently, one of the steps in-game adaptivity is identifying the player profile. Although a human personality may be very complex, some emotions, feeling[17] can be correlated with the game and, therefore, with game flow.

Simulation of an accurate human model is very expensive computationally and involves many different variables. In this paper, the simulated player has a static personality and actions while playing the game, with some of these personas being based on other work [4].

It is also important to refer that the scope of this project is limited to first time players and the adaptivity is meant to be decided early into the play session. Exploring the subject of game adaptivity as the game is played, or taking into account the same player's experience over multiple sessions (possibly learning the game and getting much better at it, or finding the mechanics repetitive) was deemed to be too complex of a situation to add onto an already complex endeavour.

III. Methodology

In a preliminary stage of this project, research was made to find a game suitable for our experiments that also provided users with data or open APIs to the detail and volume of data required. The game would also have to be simple, and ideally single-player, so as to not have to deal with the complexities of match-making (be that in 1v1 or, worse, with large teams) and with unfairness between different players.

As some our goals hinged on very fine details and the volume of data required to train a reinforcement learning algorithm is significant, no suitable game was found and instead, an in-house model of the popular arcade game Breakout was built, which fits all of the characteristics mentioned. This prototype was built in Unity, as building a pipeline that allows for data handling in Python is easy, it has a variety of tools that are useful for machine learning algorithms, and it is also a very simple engine to work with. The development of a prototype also facilitates manipulating the parameters that will then be used to "adapt" to a player, such as (in Breakout) ball size and speed.

Ideally, game session data for training and testing the algorithms would be collected with real players, publishing the game in an open platform and allowing users of all kinds to play it. Then, data would be collected on the game session and on the user's satisfaction with via a questionnaire. For this last step, we decided the Game Experience Questionnaire[18] would be a suitable reference for questions.

Due to the constraints of this work, a version of the prototype open to the public ended up not being suitable. The reasons for this are the low volume of data we'd acquire (lower than the required for data mining) due to the short time-span the project is developed in and the much larger workload than opening a refined version of the prototype to the public would entail. All of these combined meant that we'd have to do our data collection in-house as well, using static but complex personality types[4] in a simulation. These personality types were designed to emulate real humans: their inputs are regulated with accurate-to-life APM and reaction time values, among other parameters, within specific ranges for different kinds of players[19]. Alongside this, each player type will also have their own heuristic for how to play (leading to various playing styles) and different standards for satisfaction that are logged via questions from the Game Experience Questionnaire.

The data collection is ran simultaneously with a reinforcement learning model that is built to alter the game parameters mentioned and then learn from the resulting session. The same method is used for testing, with a filter for good performances applied so that the algorithm only learns the correct paths to take. Once that model gets tested, two separate multi-output regression models are ran on the reinforcement learning's test data to verify how close to its good predictions it is. These models are different and will be described in the following sections. Using this method, the multi-output regression may be limited to the data provided by the reinforcement learning.

A. Prototype

As was mentioned previously, the base for this project was a prototype of the game Breakout developed in Unity. The goal of the game is to destroy all of the bricks via bouncing the ball with the paddle while preventing the ball from touching the bottom of the screen (which results in an instant loss).

Although some of its characteristics are adaptable, there are many aspects which are fixed for every game session. These aspects include the number of bricks being 30, arranged in 3 lines of 10, with a fixed size that fits the game space and covers an entire horizontal row of the screen. The paddle is 1

unit in height and can only be moved across a horizontal line 3 units below the centre of the game area, which itself covers a fixed 1920x1080 space.

In turn, the adaptable aspects are the height at which the bricks are placed (1 to 5 units above the centre of the game space), the paddle's speed (15 to 35 speed units), the paddle's length (10 to 30 units), the ball's speed (2 to 12 speed units) and the ball's size (2 to 6 units of diameter).

The game cycles with this prototype function in a simple way. Before the game starts, it loads a list of templates of the personality types that it will generate for each game. A connection is established with the reinforcement learning model which then provides values for the aforementioned parameters, a player is generated from one of the templates (iterating through the personalities for each round) and the game starts.

Once each game ends, data about the game gets collected by the model and logged onto a file: from the parameters used, to game values such as "time" or "amount of ball bounces", to the player's simulated responses to the Game Experience Questionnaire and the player's personal values recorded throughout the game (reaction time, actions per minute, personality type and paddle safety distance). After that, a new game starts until all of the simulations are ran.

The cycles themselves are structured with episodes with each episode having 10 rounds each.

B. Personality System

As the goal of this project is to create a game that can adapt its characteristics (ball size, paddle length, etc...) in order to find the best possible fit for the player, a personality system was created to simulate certain traits that different people may have. The first part of the system consists in distinguishing the personalities according to their play style, in accordance with their physical capabilities. This includes differences in:

- Maximum and Minimum APM (actions per minute)
- Maximum and Minimum reaction time (the difference between eye and hand)
- Maximum and Minimum paddle safety distance (how close to the edge of the paddle the player feels comfortable playing with)
- Movement heuristic (a personalized heuristic that dictates how each personality analyzes and makes decisions within the game system)

These characteristics are integrated into the prototype via the player's input system. The player only has three input options, move the paddle to either left or right, or remain in place. To simulate a realistic human, the movement heuristic is run at a constant interval of 60/APM seconds, and there's a delay corresponding to the reaction time between the decision of the movement heuristic and the action taking place. The purpose of the "paddle safety distance" is to simulate how comfortable a player feels with the paddle hits. This introduces a buffer to the length of the paddle. For example, new players might not feel as safe playing with the edges as there's a bigger risk of the ball passing through, so they'll have a bigger "paddle safety distance" which means they'll consider a smaller area at the centre of the paddle as safe to hit the ball with.

There were three-movement heuristics: newbie, experienced and risky. Newbie's heuristic is very greedy, it always attempts to position the paddle directly below the ball as much as possible and does not consider possible bounces the ball might take, or aiming the ball in any specific direction. By contrast, the experienced heuristic calculates where the ball will cross the paddle's horizontal axis once the ball starts a downward descent. Finally, the risky heuristic is specific to one personality type ("Edgy") as instead of trying to play with an area in the centre of the paddle, it tries as much as possible to hit the ball with the edges of the paddle, leading to more bounces.

Furthermore, as a way to evaluate a players opinion of a game and its chosen characteristic, a satisfaction heuristic was created with the goal of quantifying a player's overall enjoyment. To fit a system which we thought was already becoming too complex, we decided to pick four questions from the "game core" section of the Game Experience Questionnaire that we believed to be most relevant to "Breakout". These questions are represented by the satisfaction heuristic's four main characteristics: content, skilful, occupied and difficulty, the values of which vary between 0 and 5. The overall satisfaction is calculated using the following equation:

$$satisfaction = content * C1 + skillful * C2$$

$$+occupied * C3 + hard * C4$$

C1, C2, C3, C4 - coefficients that attribute different weights to the characteristics Since the highest possible score is 20, and we have four characteristics, the coefficients values vary between [0.25,1.75], considering that

$$C1 + C2 + C3 + C4 = 4$$

always has to hold.

The satisfaction heuristics are unique to each personality, even though there may be some similarities (such as with "Newbie" and "Gifted Newbie"). Each personality values different attributes to calculate each variable (for example, Competitive values winning or losing more for the Content variable, while Newbie places more importance on the time played). The coefficients also vary in each personality to emulate how different kinds of players value different things in their game experiences. For example, a Competitive player values a game's difficulty being fitting to his skills more than a Newbie does.

Personalities	Physical Attributes	Movement Heuristic
Newbie	Slow	Newbie
Gifted Newbie	Fast	Newbie
Competitive	Fast	Experienced
Edgy	Fast	Risky
Experienced	Slow	Experienced
Fast Learner	Adaptive*	Newbie

*Fast Learner improves their reaction time and APM as the game timer increases.

C. Pipeline

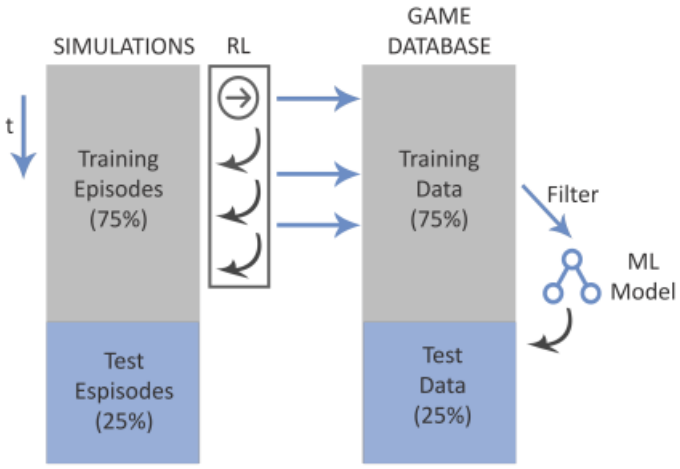

Figure 1. Pipeline scheme

The figure above is a visual representation of the pipeline of this experiment.

- This process will run a total of t simulation episodes, where 75% of them are used for training and the remaining 25% are used for testing purposes.
- For every training episode, a Reinforcement Learning Agent will receive the parameters of the player's personality in order to predict the best game parameters and submit them to run the simulation. After finishing the episode the reward will be based on the satisfaction of the player. The objective of the training is to learn the best parameters to maximize the satisfaction of all types of players.
- Parallel to the previous step, all the game data is being stored in a database including the characteristics of the player's personality, the game's parameters and the player's satisfaction.
- After the ending of the training stage, the data collected will be filtered to retrieve only the episodes with the right characteristics to train the Supervised Learning Models. The following graph represents how the filter works. To better visualize the logic behind the filter in 2 dimensions, we will assume only two characteristics **Characteristic 1** and **Characteristic 2**.

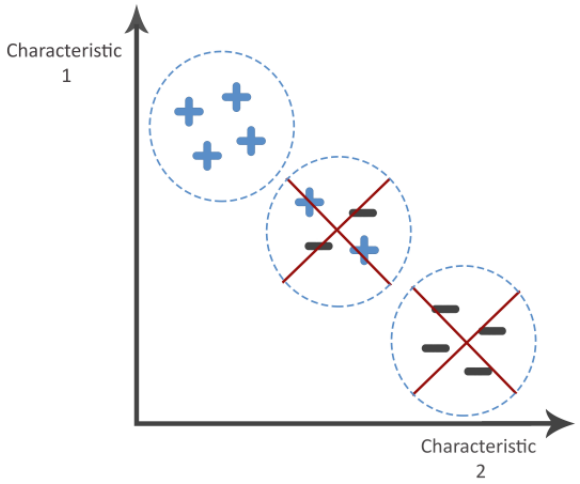

Figure 2. Game Data Filter Representation

To optimize the supervised learning models' pattern detection, we only intend to select the game episodes where the satisfaction is higher or equal to 14, including its neighbours in a distance equal or lower than 10. If the episode has an ideal value of satisfaction but is surrounded by episodes with low satisfaction, then it will be excluded.

- After filtered, two different supervised learning models will train with it. These models include a multi-output regression random forest and a group of multiple single-output random forest models to predict individually every single game parameter.
- In the testing stage, the remaining 25% of the total number of simulations (t) will resume. The RL Agent this time will only predict the game parameters using the player's personality, but won't train anymore. This data will be collected too but in a separate database.
- Then both supervised learning models will make their predictions to all the testing episodes, using only the player's personality as features, predicting new game parameters as an alternative to the ones predicted by the RL Agent.
- To evaluate the performance of the supervised learning models, we then measure the value of RMSE to every label and the global RMSE, using the values predicted by the reinforcement learning agent as a reference.
  The episodes used for evaluation are the test episodes where the RL Agent was able to achieve a satisfaction higher or equal to 14. These are the episodes where the reinforcement learning agent was able to converge in better policies and are the base reference values to evaluate the supervised learning models. If in an episode, the agent did not achieve this condition, the episode won't be used in the evaluation.

D. Reinforcement Learning

Reinforcement learning has shown success at resolving tasks that requires the understanding of the policies and rules of the world, to try to maximize a reward function. One of these successful algorithms is the proximal policy optimization (PPO), that can be used in training game's artificial players and enemies. The proximal policy optimization is an algorithm that uses policy gradient methods to find an optimal policy for a given reward. However, these type of algorithms requires a lot of time steps and are sensitive to step size.

For this project, the reinforcement learning algorithm was adapted to fit the need of the game. The action that can be taken is at the start of each game in the form of the game parameters. Then the game is played and observations are made at the end of the game. These observations are delivered to the algorithm, as well as with the reward value. The next set of parameters are calculated and delivered to the next game.

E. Supervised Learning

- **Multi-Output Regression**
  In this experiment, it will be used only one model to predict the values of multiple labels, while taking into consideration that there is a pattern correlating them.
  The chosen algorithm to predict the labels was the Random Forest, which supports native multi-output regression. For these scenarios, the model stores multiple output values in leaves, instead of only one and uses a splitting criterion that computes the average reduction across all the outputs. In general, all tree-based methods make use of this logic.[20]
  To measure the performance of the multi-output prediction, the calculation of the Root-mean-square deviation (RMSE) of every predicted label is used, along with the mean of all of the RMSEs to obtain a global RMSE.
- **Wrapped Individual Regression**
  Alternatively, multiple individuals supervised learning models are wrapped together to receive the same features as input, but each one of them will calculate every target label, in an independently, assuming that there is no pattern correlating them.
  This strategy is expected to be more intensive in memory, while the previous one to be more optimized.
  To measure the performance of this strategy, the strategy explained above will be used in this scenario too. If there isn't a pattern correlating the multiple labels, then both strategies will have similar values of RMSE.

IV. Results

The following tables are a small analysis of the results obtained in the experiments, divided according to the personality type they involved. There is also a table of the overall results and one that describes the RMSE for both supervised learning algorithms, detailing the RMSE for each variable and the global RMSE. These tables are a result of an extensive data analysis conducted after multiple experiments, with 7004 game sessions for training and 1593 game sessions for testing, within the reinforcement learning algorithm. The full data collected can be found at the group's repository[21]. Values in the tables are presented with their average value (avg) and the standard deviation (dev). Due to an error with the splitting of game rounds by personality type, the "fast learner" and "edgy" personality types had half as many game rounds as every other personality type.

Newbie Results

	Training	Test
Brick Height	2.9 avg, 1.5 dev	2.2 avg, 1.3 dev
Paddle Speed	17.7 avg, 5.1 dev	16.2 avg, 3.2 dev
Paddle Length	28.3 avg, 4.6 dev,	28.4 avg, 4.8 dev
Ball Speed	4 avg, 3.4 dev	2.9 avg, 2.6 dev
Ball Size	5 avg, 1.3 dev	5.1 avg, 1.1 dev
Wins	3/1401	107/314
Time	19.2 avg, 21.2 dev	55.6 avg, 36.8 dev
Satisfaction	9.4 avg, 4.3 dev	15.2 avg, 3.6 dev

Competitive Results

	Training	Test
Brick Height	3 avg, 1.6 dev	2.2 avg, 1.4 dev
Paddle Speed	17.6 avg, 5 dev	16.4 avg, 3.3 dev
Paddle Length	28.2 avg, 4.7 dev,	28.4 avg, 5.1 dev
Ball Speed	3.9 avg, 3.4 dev	2.7 avg, 2.4 dev
Ball Size	5 avg, 1.3 dev	4.8 avg, 3.5 dev
Wins	30/1401	256/319
Time	25.4 avg, 23 dev	94.7 avg, 55.5 dev
Satisfaction	8.4 avg, 2.4 dev	12.5 avg, 2.7 dev

Experienced Results

	Training	Test
Brick Height	2.9 avg, 1.5 dev	2.2 avg, 1.3 dev
Paddle Speed	17.6 avg, 4.9 dev	16.8 avg, 3.9 dev
Paddle Length	28.3 avg, 4.6 dev,	28.4 avg, 5.0 dev
Ball Speed	4.2 avg, 3.5 dev	2.9 avg, 2.6 dev
Ball Size	5 avg, 1.3 dev	5.2 avg, 1.1 dev
Wins	8/1401	95/320
Time	18 avg, 1.3 dev	51.4 avg, 39.4 dev
Satisfaction	10.1 avg, 1.9 dev	12.3 avg, 3.1 dev

Edgy Results

	Training	Test
Brick Height	2.8 avg, 1.5 dev	2.2 avg, 1.4 dev
Paddle Speed	17.4 avg, 4.8 dev	16.7 avg, 3.6 dev
Paddle Length	28.4 avg, 4.4 dev,	28.5 avg, 5.0 dev
Ball Speed	4.2 avg, 3.5 dev	2.9 avg, 2.6 dev
Ball Size	5.0 avg, 1.3 dev	5.2 avg, 1.1 dev
Wins	0/700	15/160
Time	18.5 avg, 1.3 dev	34.2 avg, 25.9 dev
Satisfaction	9.1 avg, 2.0 dev	11.0 avg, 2.7 dev

Gifted Newbie Results

	Training	Test
Brick Height	2.9 avg, 1.5 dev	2.0 avg, 1.3 dev
Paddle Speed	17.7 avg, 5.0 dev	16.3 avg, 3.2 dev
Paddle Length	28.2 avg, 4.8 dev,	28.5 avg, 4.8 dev
Ball Speed	3.9 avg, 3.4 dev	2.8 avg, 2.5 dev
Ball Size	5.0 avg, 1.3 dev	5.2 avg, 1.1 dev
Wins	32/1401	273/319
Time	26.6 avg, 32.7 dev	90.0 avg, 47.2 dev
Satisfaction	10.0 avg, 2.9 dev	14.8 avg, 2.5 dev

Fast Learner Results

	Training	Test
Brick Height	2.8 avg, 1.5 dev	2.2 avg, 1.4 dev
Paddle Speed	17.5 avg, 7.8 dev	16.4 avg, 3.5 dev
Paddle Length	28.0 avg, 5.0 dev,	28.6 avg, 4.6 dev
Ball Speed	3.9 avg, 3.4 dev	2.9 avg, 2.7 dev
Ball Size	5.0 avg, 1.3 dev	5.4 avg, 0.9 dev
Wins	32/700	132/160
Time	32.4 avg, 27.9 dev	81.1 avg, 44.1 dev
Satisfaction	11.6 avg, 3.2 dev	14.8 avg, 2.6 dev

Overall Results

	Training	Test
Brick Height	2.9 avg, 1.5 dev	2.2 avg, 1.3 dev
Paddle Speed	17.6 avg, 7.8 dev	16.5 avg, 3.5 dev
Paddle Length	28.2 avg, 4.7 dev,	28.5 avg, 4.9 dev
Ball Speed	4 avg, 3.4 dev	2.8 avg, 2.6 dev
Ball Size	4.9 avg, 1.3 dev	5.3 avg, 1.1 dev
Wins	105/7004	878/1593
Satisfaction	9.6 avg, 3.1 dev	13.5 avg, 3.3 dev

Multi Output and Individual Regression Results

	Multi Output RMSE	Individual Regression RMSE
Brick Height	1.293	1.277
Paddle Speed	2.836	2.799
Paddle Length	1.092	1.076
Ball Speed	1.415	1.575
Ball Size	1.039	1.038
Global	1.67	1.68

V. Result Discussion

- Due to the similarities of the values of RMSE obtained in both strategies, we can conclude that their performance in this problem is similar. It is then preferable to use the multi-output regression due to being more lightweight in its use of machine resources.
- It is essential to note that the RMSE values for ball speed and paddle speed are the ones with the most variation. This might be the importance of object speed related to the other variables and how there might be more variation due to the different movement heuristics by each personality type.
- It is clear from a small overview of the results that the tendency of the reinforcement learning algorithm was towards smaller values on brick height, paddle speed, and ball speed and higher values on paddle length and ball size. This means that the "strategy" the reinforcement learning algorithm maximizes satisfaction in Breakout within the constraints and feedback of our model. Furthermore, it seems the algorithm has determined what is the best compromise in a way to maximize a sum of the maximization algorithms, and thus ended up fitting the parameters to the preference of personalities that are more easily pleased (Newbie, Gifted Newbie, Fast Learner) at the detriment of a personality like "Competitive" that has a higher baseline for expectations but requires higher depth and difficulty to get pleased.
- Cross-referencing the results tables with the information listed in table 1, and comparing the win ratios across "Newbie", "Experienced", "Gifted Newbie" and "Competitive", it is possible to infer that the player's physical traits ended up being more relevant than their movement heuristic. This can also be noticed by analyzing the average times for each personality type mentioned, with "Competitive" and "Gifted Newbie" having more in common than "Competitive" and "Experienced", for example.
- We can infer that there is little correlation between the labels and only small patterns, due to the multi-output regressor having slightly lower values of RMSE.
- Through the analysis of the preferable game parameters to each type of personality, we can notice that the values of each table don't differ as much as we expected. It is possible that the reinforcement learning agent, instead of discovering a policy that maximizes each personality's satisfaction, tried to find a policy that could consensually satisfy all of the personalities. A possible strategy to force the maximization of the satisfaction of each personality type, instead of the maximization of the overall satisfaction is the division of the player data into clusters representing the personality type and application of individual reinforcement learning models onto each cluster; then, new players would be classified into each of the clusters naturally, and the appropriate model for defining the parameters would be used.
- The RMSE values for each variable are always smaller than the respective deviation values. As such, the method's results could be considered comparable to the ones obtained with reinforcement learning.

VI. Conclusions

In this study, we approach the topic of the recent and rapidly developing field concerning the use of machine learning to enhance a user's experience while playing video games. Our goal was to analyze methodologies like multi-output supervised learning models and individualized single-output models compared to reinforcement learning.

The basis for our experiment was the game Breakout, slightly modified to better fit our criteria. Reinforcement Learning seems to be a valuable tool when searching for an optimal solution in a changing environment despite some training difficulties. Developing a game with an RL Agent is a challenging and promising prospect, and its use could even begin to extend past the reality of video games. Our analysis led to satisfying conclusions in regards to its effectiveness. Still, some changes like a massively increased amount of data, further experiments with hyperparameter tuning, alternative implementations, or raising the minimum satisfaction required for a game to be considered for testing (thus having a higher quality filter) could lead to better results.

Our results show that both supervised learning methods analyzed are viable and comparable to reinforcement learning, with multi-output regression having a very slight advantage, which was unexpected. This can be explained due to the number of episodes being too small. Future exploration of this subject will, ideally, feature research methods and metrics that allow for the evaluation of the supervised learning models'

results in absolute terms. That way, we could make a more in-depth comparison of their performance with the reinforcement learning agent. Due to time constraints, we could not implement metrics that would better analyze this, such as separately simulating the games with the parameters chosen by the supervised learning methods.

Finally, in future work, it is imperative to use data from real humans instead of a simulation. The flaws of our personality system as it compares to real people put a caveat on definite conclusions derived from our results.

ACKNOWLEDGMENT

We would like to show our acknowledgement to some of our professors that accompanied the development of this project, namely professors João Jacob, Zafeiris Kokkinogenis and Carlos Soares from Faculty of Engineering of the University of Porto. They were responsible for the guidelines that helped kick-start this project and helped orient its course, so their collaboration is appreciated.

REFERENCES

[1] R. Hunicke and V. Chapman. AI for dynamic difficulty adjustment in games. In *AAAI Workshop - Technical Report*, volume WS-04-04, pages 91–96, 2004.

[2] A. Streicher and J.D. Smeddinck. *Personalized and adaptive serious games*, volume 9970 LNCS. 2016.

[3] S. Bakkes, C.T. Tan, and Y. Pisan. Personalised gaming: A motivation and overview of literature. In *ACM International Conference Proceeding Series*, 2012.

[4] X. Fang and F. Zhao. Personality and enjoyment of computer game play. *Computers in Industry*, 61(4):342–349, 2010.

[5] R. Lopes and R. Bidarra. Adaptivity challenges in games and simulations: A survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(2):85–99, 2011.

[6] J. Frommel, F. Fischbach, K. Rogers, and M. Weber. Emotion-based Dynamic Difficulty Adjustment Using Parameterized Difficulty and Self-Reports of Emotion. In *CHI PLAY 2018 - Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*, pages 173–185, 2018.

[7] T.J.W. Tijs, D. Brokken, and W.A. Ijsselsteijn. *Dynamic game balancing by recognizing affect*, volume 5294 LNCS. 2008.

[8] Nintendo EAD. Mario Kart Wii, 2008.

[9] Lars Jensvold. Left 4 Dead, 2011.

[10] B. Monterrat, E. Lavoué, and S. George. Motivation for learning: Adaptive gamification for web-based learning environments. In *CSEDU 2014 - Proceedings of the 6th International Conference on Computer Supported Education*, volume 1, pages 117–125, 2014.

[11] B. Magerko, C. Heeter, B. Medler, and J. Fitzgerald. Intelligent adaptation of digital game-based learning. In *ACM Future Play 2008 International Academic Conference on the Future of Game Design and Technology, Future Play: Research, Play, Share*, pages 200–203, 2008.

[12] D. Hooshyar, L. Malva, Y. Yang, M. Pedaste, M. Wang, and H. Lim. An adaptive educational computer game: Effects on students' knowledge and learning attitude in computational thinking. *Computers in Human Behavior*, 114, 2021.

[13] Noah Raffin, Antonin and Hill, Ashley and Ernestus, Maximilian and Gleave, Adam and Kanervisto, Anssi and Dormann. Stable Baselines3, 2019.

[14] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, jul 2017.

[15] G. Melki, A. Cano, V. Kecman, and S. Ventura. Multi-target support vector regression via correlation regressor chains. *Information Sciences*, 415-416:53–69, 2017.

[16] Willem Waegeman, Krzysztof Dembczyński, and Eyke Hüllermeier. Multi-target prediction: a unifying view on problems and methods. *Data Mining and Knowledge Discovery*, 33(2):293–324, 2019.

[17] L.E. Nacke, M.N. Grimshaw, and C.A. Lindley. More than a feeling: Measurement of sonic user experience and psychophysiology in a first-person shooter game. *Interacting with Computers*, 22(5):336–343, 2010.

[18] W A Ijsselsteijn, De Kort, and Poels. GAME EXPERIENCE QUESTIONNAIRE. Technical report, 2013.

[19] S.C.J. Bakkes, P.H.M. Spronck, and G. van Lankveld. Player behavioural modelling for video games. *Entertainment Computing*, 3(3):71–79, 2012.

[20] Dragi Kocev, Celine Vens, Jan Struyf, and Sašo Džeroski. Tree ensembles for predicting structured outputs. *Pattern Recognition*, 46(3):817–833, 2013.

[21] João Carlos Maduro Mariana Neto João Álvaro Ferreira, João Augusto Lima. Source Code Repository. https://github.com/JoaoAlvaroFerreira/BreakoutMining/, 2021.