CS4242 Machine Learning for Interactive Systems Final Project

AI & ML in Gaming

Oscar Gutierrez Roman

University of Limerick, 23365978@studentmail.ul.ie

Donncha Mitchell

University of Limerick, 23389559@studentmail.ul.ie

1 INTRODUCTION

The integration of Artificial Intelligence (AI) and Machine Learning (ML) techniques into the gaming industry has marked the start of a new era of innovation and immersion. This report presents an exploration of the use of AI and ML in gaming. Later, we will be focusing particularly on the development of a Rock Paper Scissors game employing gesture recognition for the interaction between the user and the game.

A brief overview of the project's objectives highlights the history of integration of AI and ML in gaming, the main methods used and more. The primary aim is to develop a Rock Paper Scissors game utilizing HandPoseOSC for gesture recognition, enhancing player interaction. Other key objectives are the choice of an appropriate ML algorithm, effective training, feature extraction, and interactive gameplay design.

This project aims to provide valuable insights into the effectiveness of AI and ML in gaming, while also addressing the challenges and opportunities associated with their implementation. Furthermore, the conclusion will discuss lessons learned, successful outcomes, challenges faced, and opportunities for future refinement and expansion of the developed system.

2 RESEARCH: AI & ML IN GAMING

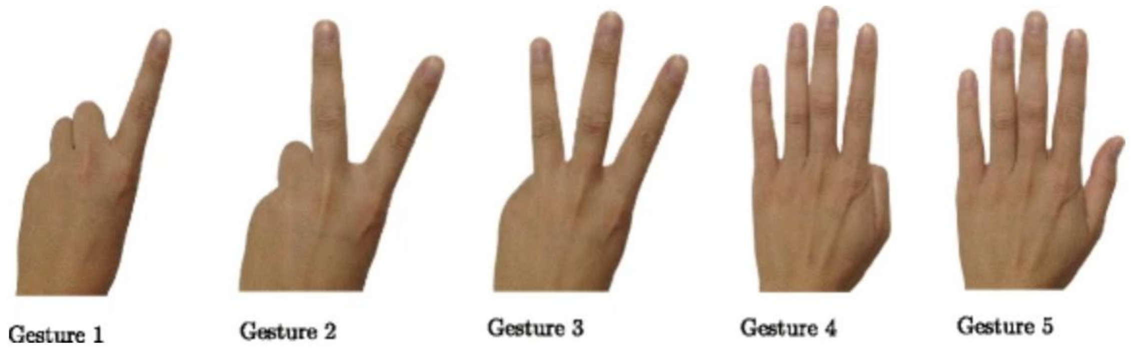
The history of AI in games traces back several decades, evolving side by side with advancements in computing power and algorithmic sophistication. Early implementations of AI in games primarily focused on rule-based systems, where predefined algorithms dictated the behaviour of non-player characters (NPCs). These NPCs followed scripted routines or decision trees, offering limited variability and adaptability in gameplay [1]. For instance, one of the first programs using AI was designed to play chess, it was named MANIAC in 1956 and it became the first computer to defeat a human in a chess-like game [2]. Playing with the simplified Los Alamos rules, it defeated a novice in 23 moves.

However, with the advent of machine learning techniques, the landscape of AI in gaming underwent a paradigm shift. ML algorithms empowered NPCs to learn from player interactions and adapt their behaviour dynamically. Games like "F.E.A.R." and "Black & White" showcased AI agents capable of learning and evolving their strategies based on player actions, enhancing immersion and challenge [3].

Many ML models and AI techniques have been prominently utilized in gaming to enhance player experiences and optimize game mechanics. Among these, reinforcement learning (RL) stands out as a fundamental approach in training AI agents to make sequential decisions in dynamic environments. RL algorithms, such as Q-learning and Deep Q-Networks (DQN), have been applied extensively in gaming scenarios to enable AI agents to learn optimal strategies through trial and error [4]. One prominent application of AI in gaming is the enhancement of non-player character (NPC) behaviour to create more immersive and challenging gameplay experiences. RL algorithms have been instrumental in this regard, enabling NPCs to learn and adapt their strategies based on interactions with the game environment and player actions [5].

Given our project's focus on utilizing machine learning for gesture recognition in gaming, it is pertinent to explore the landscape of gesture recognition techniques within this context. Gesture recognition using machine learning techniques has gained significant traction in the realm of gaming, offering intuitive and immersive interaction methods for players. Various ML algorithms have been explored for gesture recognition, with convolutional neural networks (CNNs) emerging as one of the most effective approaches. CNNs are well-suited for analysing spatial relationships in image data [6], making them ideal for capturing intricate hand gestures. By training CNN models on labelled gesture datasets, developers can achieve high accuracy in recognizing and classifying different hand movements, enabling players to control in-game actions through natural gestures.

In addition to CNNs, k-nearest neighbors (k-NN) algorithm is another viable option for gesture recognition in gaming applications. K-NN is a simple yet powerful supervised learning algorithm that classifies input data points based on their similarity to neighbouring instances [7]. In the context of gesture recognition, k-NN can be employed to classify hand gestures by comparing their feature vectors with those of known gestures in the training dataset. While k-NN may not offer the same level of complexity and feature extraction capabilities as CNNs, its simplicity and efficiency make it a practical choice for real-time gesture recognition tasks in gaming environments.



Hand gestures

Now we will discuss the psychological factors that influence a player's decision-making in rock-paper-scissors just to get more insights about how people play this game. According to a comprehensive study conducted by The Psychology of Rock Paper Scissors in 2021 [8], most players tend to adopt a strategy known as 'conditional response.' This strategy entails players mimicking their previous move if they win a round, while switching to the next gesture in the sequence (rock, paper, scissors) if they lose.

Interestingly, certain psychological associations influence the choice of gestures. For instance, players often initiate with rock, considering it as the most aggressive play due to its association with strength and weaponry. Similarly, scissors, with its sharp and menacing connotations, is favoured by confident players or those who hold an advantageous position in the game. In contrast, paper is perceived as a more passive gesture, symbolizing vulnerability. Consequently, players are less inclined to use paper when they are behind, opting for more confident moves instead.

Analysing the probabilities of opponents' moves reveals a distribution where there is a 35% chance of encountering rock, a 35% chance of scissors, and a 30% chance of paper. Armed with this understanding of the psychological aspects of each gesture, devising an effective strategy becomes a relatively straightforward task.

3 OUR DEVELOPMENT PROCESS

3.1 Overview of the system

The goal of our development process was to create a Rock Paper Scissors game that allows players to interact with the game using hand gestures to play against the computer. The system would utilize a machine learning algorithm to recognize and interpret the gestures made by the player, enabling them to make their moves in the game without the need for traditional input devices like controllers. The main idea was to employ hand recognition software to detect hand positions, then use an ML algorithm to classify the gestures made by the user and produce a visual output in a graphical user interface (GUI).

3.2 Choice of the algorithm

We decided that to differentiate between the three different hand poses, the most appropriate and yet simple solution would be to use a classifier with three classes and one output. Since we already had learnt how to use Wekinator, we used it to process the input of the hand recognition software (HandPoseOSC). The choice of classification model or algorithm was now reduced to the following:

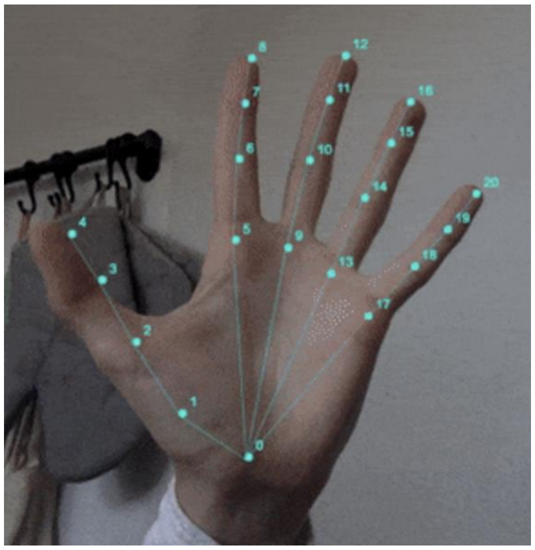
- K-Nearest Neighbour
- AdaBoost
- Decision tree (J48)
- Support vector machine
- Native Bayes
- Decision Stump

The ones that we used during labs, and we were more familiarised with were k-nearest neighbour and decision stump. We chose k-nearest neighbors (k-NN) over decision stump for this project due to its suitability for more complex gesture recognition tasks. Decision stump is not as suitable for this project because it tends to oversimplify the classification process and may struggle with hand gesture recognition. K-NN, on the other hand, excels at handling complex and nonlinear data patterns by classifying data points based on their similarity to neighbouring instances in the feature space. This makes k-NN a more appropriate choice for accurately classifying the diverse range of hand gestures required for the

Rock Paper Scissors game. Additionally, during our research, we already found evidence that K-NN could be a good option for hand pose recognition.

3.3 Training data, training process and feature extraction

The training data is the output from the HandPoseOSC program, which sends the [x, y, z] coordinates of 21 points spread around the hand, 5 on each finger and 1 on the palm of the hand via OSC. That is a total of 63 (21 x 3) inputs for Wekinator.



HandPoseOSC hand detection

We then recorded samples with different variants of the three hand poses (rock, paper, or scissors) and then trained the model initially with k = 1 (1 neighbour). After training, we made our first test run. It was not too bad although there was room to improve. Thus, we started to play around with the number of neighbours, we tried with a big number, even numbers, odd numbers… And based on our tests, we concluded that the optimal number of neighbours was 3.

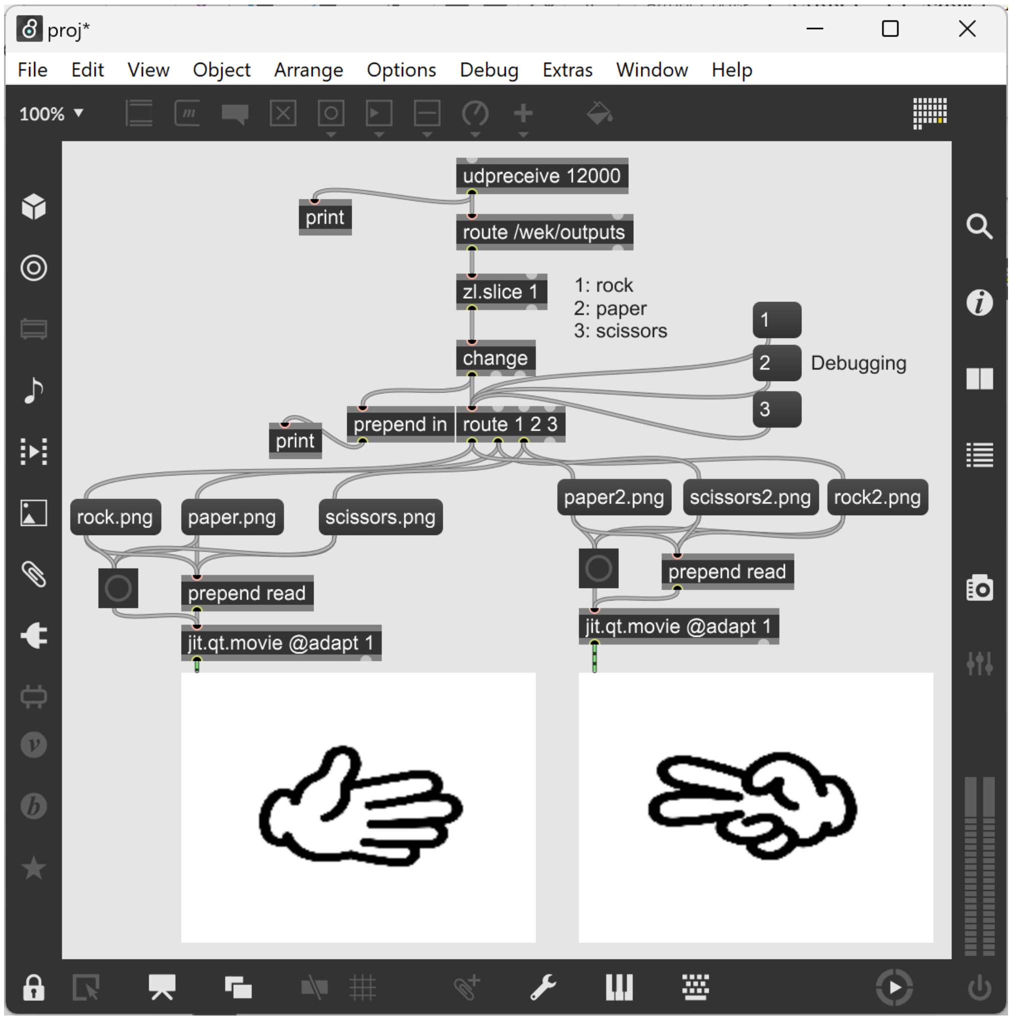
During the training process, we also observed that the model had difficulty distinguishing between hand poses, particularly when transitioning between rock and scissors, and vice versa. After conducting numerous tests and augmenting the dataset with additional samples, we identified two main factors contributing to these issues: the HandOSC program occasionally failing to detect the index and middle fingers, and the thumb's position being similar in both hand poses. After adding even more samples to the dataset, the model seems to have improved. At this stage, our classifier could identify the three hand poses, but only when they were well distinctly formed. To enhance its accuracy, we included samples of poses midway between these distinct positions.

That was our first attempt at training the algorithm using the specified format of input data. However, during the demo day, we realize that when the input data consists of [x, y, z] coordinates, the general position of the hand captured by the camera, regardless of the specific gesture made, significantly influences Wekinator's output. To address this issue, we implemented a solution that involved determining the position of the fingers relative to a reference point, specifically the hand palm. To implement this solution, we developed a Python script positioned between HandPoseOSC and Wekinator.

This script receives the raw [x, y, z] coordinates, calculates the new coordinates relative to the hand palm, and then forwards them to Wekinator. This adjustment resulted in noticeable improvements in Wekinator's ability to identify gestures accurately.

3.4 The development of the output

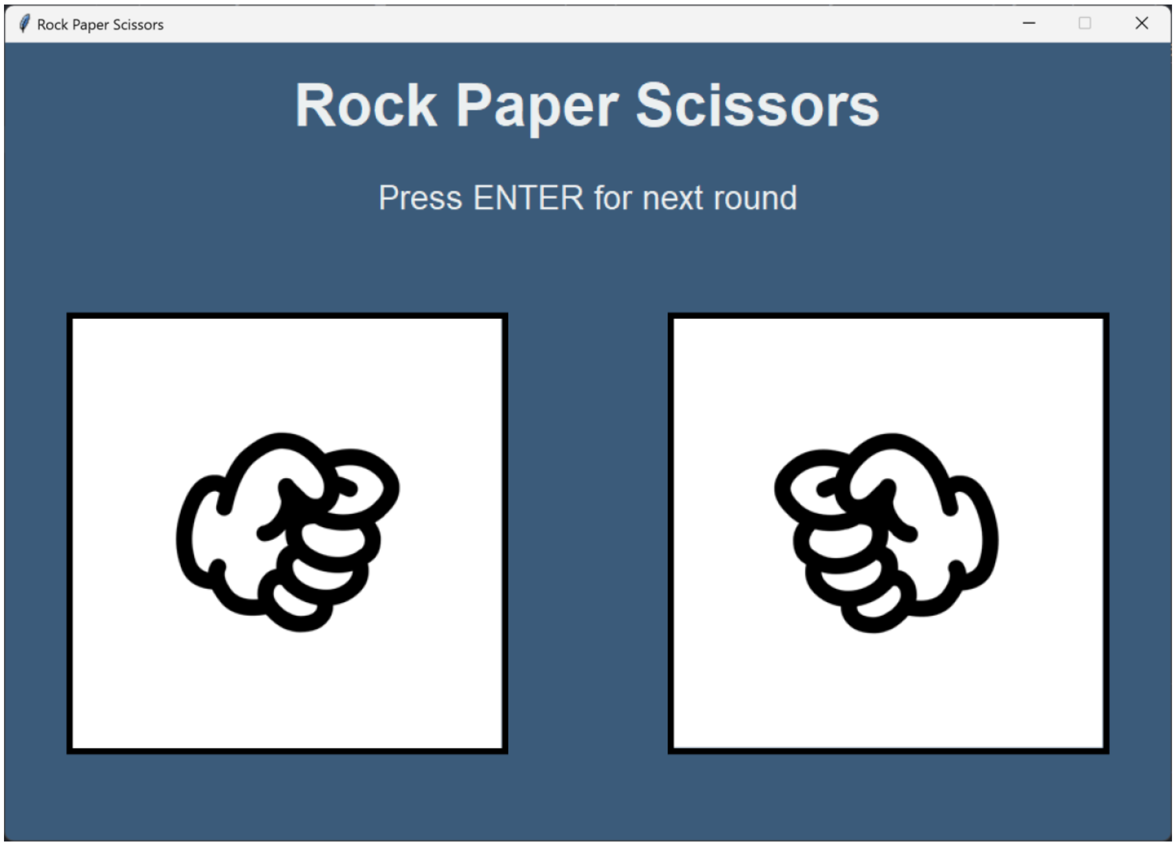
Initially, our approach for the output involved sending Wekinator's output to Max/MSP, where it was cleaned and routed to jit.qt.movie objects that would change images based on the input gesture.



MAX/MSP prototype

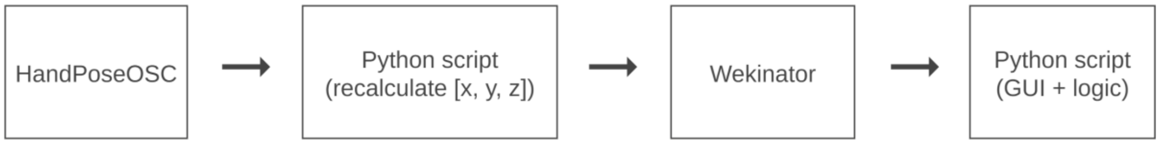
However, our ambition to do better led us to enhance the user experience by implementing a graphical user interface (GUI) using Python's Tkinter library. This Python script, apart from serving as the GUI, receives input from Wekinator via OSC, parses the message, and orchestrates the computer's response accordingly.

Throughout the development of the logic, we encountered an issue with the continuous input stream from Wekinator, causing rapid changes in the gesture pose that the GUI struggled to keep up with. To address this, we implemented a solution where the player needed to press a designated key on the keyboard to advance to the next round, allowing for a better interaction between the player and the GUI.



Last version of the Graphical User Interface in Python

When it comes to the response of the computer, in our first iteration of the project, we randomized it. However, the lab instructors gave us some feedback and concluded that it would be better if the response wasn't random. Hence, we coded a function for the computer to respond to the user's input. We also tried to train a neural network to respond but once we got it trained, the outcome was the same as with the hard coded function we had done. The computer would always win. Therefore, we decided to leave aside the idea of the neural network since it didn't change the gaming experience and it just needed more resources to run (a second Wekinator project).


HandPoseOSC
Python script (recalculate [x, y, z])
Wekinator
Python script (GUI + logic)

Workflow diagram

4 CONCLUSION

4.1 What was learned?

Throughout this project, we gained valuable insights into various aspects of AI and ML in gaming. Our research provided a comprehensive understanding of how AI and ML technologies have evolved and integrated into gaming environments, from the historical context to the main models, algorithms, and methods used. Additionally, we learnt about gesture recognition and even explored the psychological aspects of rock-paper-scissors. Moreover, we acquired practical knowledge on training ML models for hand gesture recognition, refining their accuracy through iterative experimentation and the acquisition of more diverse and plentiful training data. Specifically, we refined our skills in working with the K-nearest neighbors (KNN) model, leveraging its capabilities to improve gesture recognition accuracy.

4.2 What worked?

The connection between HandPoseOSC, the python script and Wekinator went better than expected, with no big problems. This smooth connection made it easy to send hand gesture data from a place to another for training and classification, which was vital for building our system.

4.3 What did not work as planned?

However, we faced challenges with the hand pose OSC software, which occasionally struggled to accurately detect finger positions, leading to inconsistencies in gesture recognition. Additionally, while developing the GUI, we encountered several bugs that impeded its functionality initially. Despite these setbacks, our troubleshooting efforts eventually resolved these issues, allowing us to proceed with the project successfully.

4.4 Future work

For future work, exploring alternative ML models for gesture recognition in gaming could enhance accuracy and versatility beyond the limitations of Wekinator. Additionally, developing custom hand pose recognition software aimed at our specific needs would offer greater control and flexibility, enabling recognition of two hands simultaneously and expanding the scope of interaction and engagement. HandPoseOSC can only track one hand at a time, limiting our ability to create a two-player rock-paper-scissors game within our project timeline. Despite debating more advanced game designs such as a first-person shooter where players use their hands to aim and shoot, we lacked the necessary skills and time to learn them. While our current exploration has been constrained, our shared interest in this field may lead to further development of our ideas in the future.

REFERENCES

[1] Westera, W., Prada, R., Mascarenhas, S., Santos, P. A., Dias, J., Guimarães, M., ... & Ruseti, S. (2020). Artificial intelligence moving serious gaming: Presenting reusable game AI components. *Education and Information Technologies*, *25*, 351-380.

[2] *Human–computer chess matches*. Wikipedia. (2024, March 18). Retrieved from https://en.wikipedia.org/wiki/Human%E2%80%93computer_chess_matches#:~:text=In%201956%20MANIAC%2C%20developed%20at,a%20novice%20in%2023%20moves.

[3] Yannakakis, G. N., & Togelius, J. (2018). *Artificial intelligence and games* (Vol. 2, pp. 2475-1502). New York: Springer.

[4] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *nature*, *518*(7540), 529-533.

[5] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

[6] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.

[7] Liu, Y., Wang, X., & Yan, K. (2018). Hand gesture recognition based on concentric circular scan lines and weighted K-nearest neighbour algorithm. *Multimedia Tools and Applications*, *77*, 209-223.

[8] *The psychology of Rock Paper Scissors* - World Rock Paper Scissors Association. (2021). Retrieved from https://wrpsa.com/the-psychology-of-rock-paper-scissors/

SOFTWARE REFERENCES

Rebecca Fiebrink (2009) Wekinator - http://www.wekinator.org/

Faiip (2020) HandPoseOSC - https://github.com/faaip/HandPose-OSC