Amrita School of Engineering, Bengaluru

Amrita Vishwa Vidyapeetham

Course: 21AIE445 – Application of Robotics

Course Instructor: Nippun Kumaar A. A.

Course Project: Rock, Paper, and, Scissors game robot using Reinforcement Learning

Team Members:

1.) BL.EN.U4AIE190**13** - **CHARAN TEJ K.**

2.) BL.EN.U4AIE190**14** - **CHAVALI SURYA TEJA**

3.) BL.EN.U4AIE190**53** - **RAMA SAI ABHISHEK PODILA**

Course Problem Statement:

To Design and implement a robotic hand that can play identify the pattern of moves played by an opponent in the game of rock papers, scissors, and defeat the human opponent in real time.

INTRODUCTION

In this project, we are going to implement a 3 – D printed robotic hand that can play the games of rock, paper, and scissors (RPS) against a human being, in a sequential manner using reinforcement learning techniques.

Before moving on to our detailed explanation, it is important for us to talk about the background for the design plan of our project as the primary step. So, let's look at Robotic hands on a higher level.

3D Printed Robotic hands/arms:

Robotic hands and arms that are manufactured in 3D are showing great promise for use in manufacturing, assistive technologies, and rehabilitation. These tools provide a flexible, affordable alternative for jobs that need dexterity and accurate manipulation. We will examine the design, production, and present uses of 3D printed robotic hands and arms in this paper.

From rehabilitation to manufacturing to assistive devices, 3D printed robotic hands and arms have the potential to change a wide range of disciplines and businesses. Robotic hands and arms that are 3D printed will probably become more common and have a bigger impact on society as technology develops. For instance, in rehabilitation, patients with hand and arm disabilities can benefit from using 3D printed robotic hands and arms to assist them to restore mobility and function. In comparison to conventional prostheses, these devices can be tailored to the patient's unique demands and requirements, making them a more efficient and individualized alternative.

In contrast, 3D printed robotic hands and arms can be utilized in manufacturing to carry out activities like assembly, inspection, and testing that call for precise manipulation. They are easily adaptable to

new jobs and may be modified to match the requirements of the manufacturing process. They can be used in conjunction with other automation systems to increase efficiency and accuracy. Additionally, 3D printed robotic arms and hands can be utilized in assistive devices to improve the capabilities of people with disabilities, such as by offering a prosthetic hand or arm. In comparison to conventional prostheses, these devices may be tailored to the user's unique requirements and preferences, making them a more pleasant and useful option.

Broadly speaking, 3D printed robotic hands and arms have a huge, varied potential. These devices are probably going to have a big impact on a lot of different fields and industries as technology develops and spreads.

One of the most important stages in the creation of these gadgets is the design of 3D-printed robotic hands and arms. A 3D model of the hand or arm is created during the design process, including the location of the actuators and sensors, the kind and arrangement of the joints, the size and shape of the component parts, and the type of joints. Finding a balance between utility, precision, and affordability while building 3D printed robotic hands and arms is one of the key issues. The design must be capable of satisfying the application's unique needs and specifications while also being affordable to produce.

To make the robot learn the strategy in the RPS game, we employed the traditional Q – learning algorithm, as part of reinforcement learning (RL) to design and implement an RL agent.

Reinforcement learning (RL) is a type of machine learning in which an agent learns to take actions in an environment in order to maximize a reward signal. RL involves a feedback loop in which the agent takes an action, receives a reward or punishment, and uses this feedback to adjust its behavior. The core idea behind RL is that the agent should learn to take actions that lead to the most reward over time. This is often referred to as the "reward maximization" principle.

Environment, Agent, Action, and Reward – the four essential elements that make up the RL process:

- Environment: The RL agent's operating space is known as the environment. It may be a real-world setting (such as a robot navigating a maze) or a computer-generated one (e.g., a computer game).
- Agent: An RL system that adapts to act in the environment is referred to as an agent. It takes information from the surroundings into account while determining what to do.
- Decisions made by an agent in response to environmental input are referred to as actions. For instance, the actions would be rock, paper, or scissors in the event of a robotic hand playing rock-paper-scissors.
- Reward: The response signal that the agent acquires after performing an action is known as the reward. The reward informs the agent whether and how much its action was successful or unsuccessful. The outcome of a game of rock-paper-scissors, for instance, could be a win, a loss, or a tie.

The agent requires a means to assess the relative merits of various acts in order to learn to choose those that will maximize reward. A value function is frequently used for this, as it predicts the long-term reward that will come from performing a specific action in a specific state. By choosing the action with the highest estimated value, the value function directs the agent's decision-making process.

Various RL algorithms have been created, each with a set of distinctive properties. RL algorithms that are frequently used include Q-learning, SARSA, and Deep Q-Networks (DQN). In our project, we train the RL agent for our 3D-printed robotic hand using the Q - learning algorithm The learning of the agent in completely in real–time, i.e., each episode is a move played by the robot and the opponent.

PROJECT IMPLEMENTATION

The entire project execution is carried as a 3 – step process:

1) Developing a deep learning hand gesture recognition module
2) 3 – D printing and assembling the robot
3) Designing the RL agent to play the game

Hand – Gesture recognition software based on deep learning:

In order to build a 3D-printed robotic hand that could play rock-paper-scissors (RPS) against a human opponent, we first needed to develop a hand gesture recognition system that could identify the hand gestures made by the human opponent. To do this, we used the Teachable Machine software by Google. With the help of Teachable Machine software, users can quickly train a model to detect particular objects or behaviors from camera feeds. We trained a model to detect the rock, paper, and scissors' hand motions using Teachable Machine.

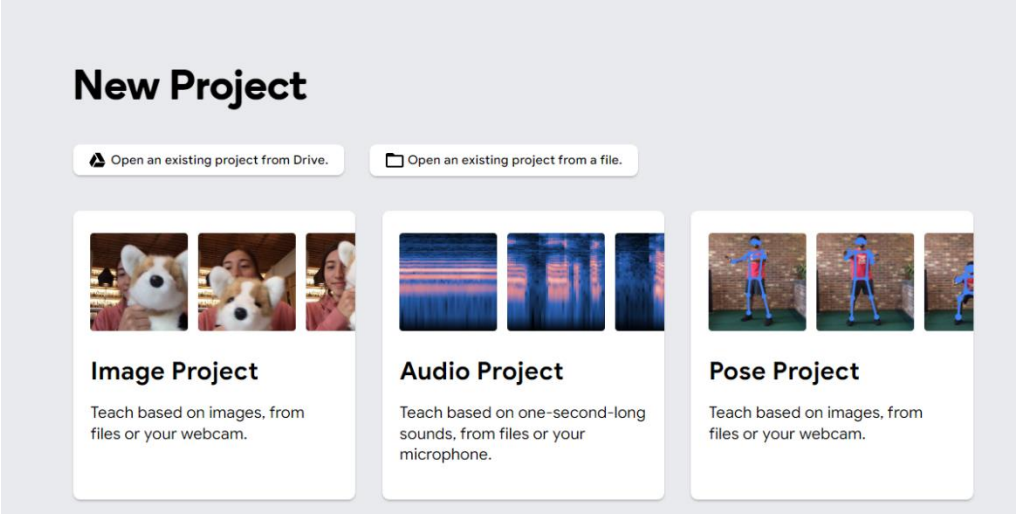


Fig.1 – Teachable Machine Layout

The website open's up, as illustrated in Figure 1, and lets us choose any one of the three projects that we desire to build.

We chose the Image project and created a custom model based on the images that we provide for it to learn and train. In order to achieve this, we used Google's Media pipe library, which detects and estimates the pose of each finger on our hand, that we show the camera input. We gathered around 400 images of rock, paper, and scissors gestures, each having different poses of a hand.

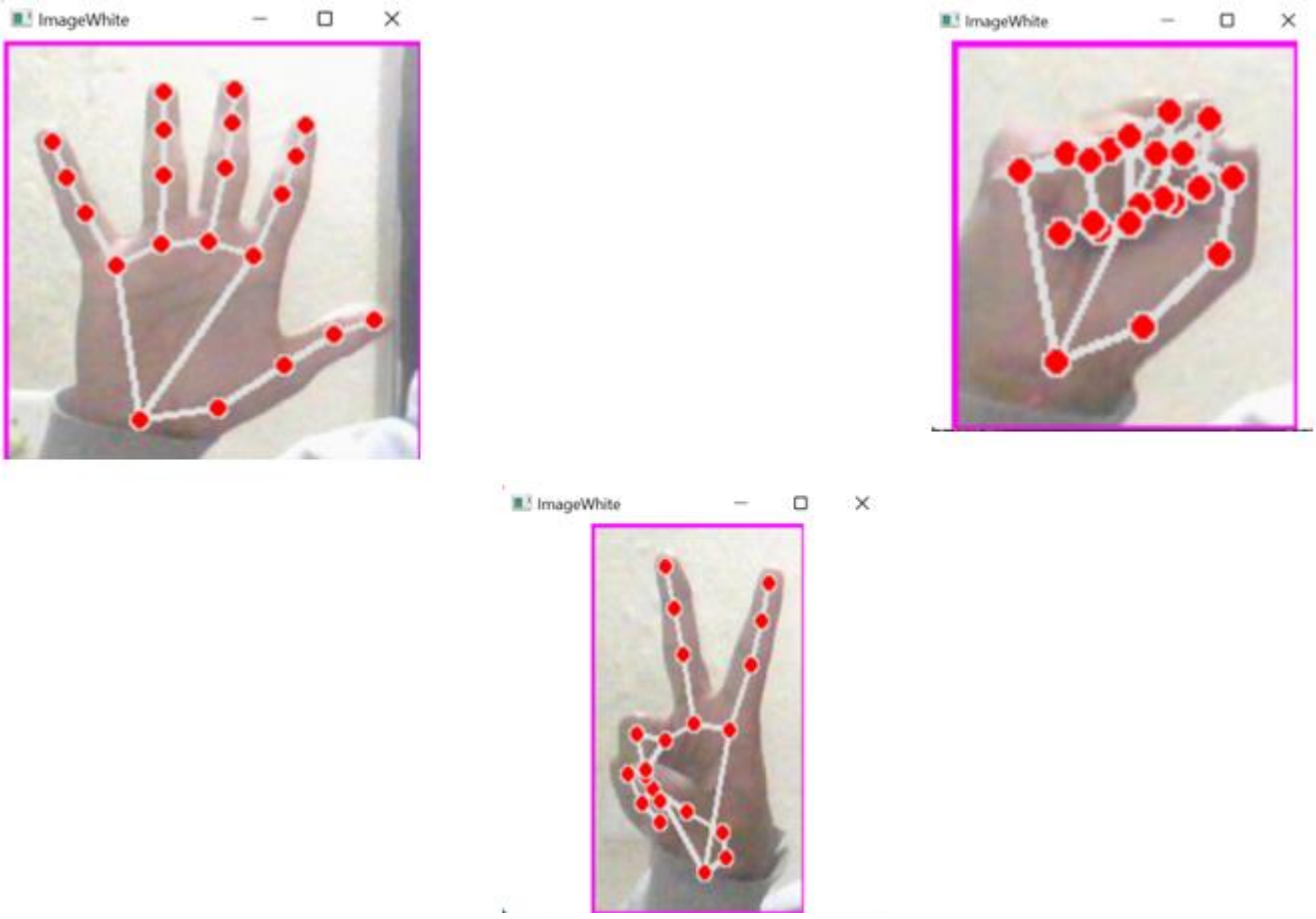


Fig.2 – Pose estimations for each gesture of R,P,S

As visualized in figure 2, and curated dataset of 1200 image samples from each class (R, P, S) have been provided for training into the Teachable Machine module. After training the model, we put it to the test by exposing it to fresh hand gesture images and videos and evaluating how well it recognized the movements. We observed that the model had a high degree of accuracy in the hand motions.

The software directly provided us with a working keras working model for further use.

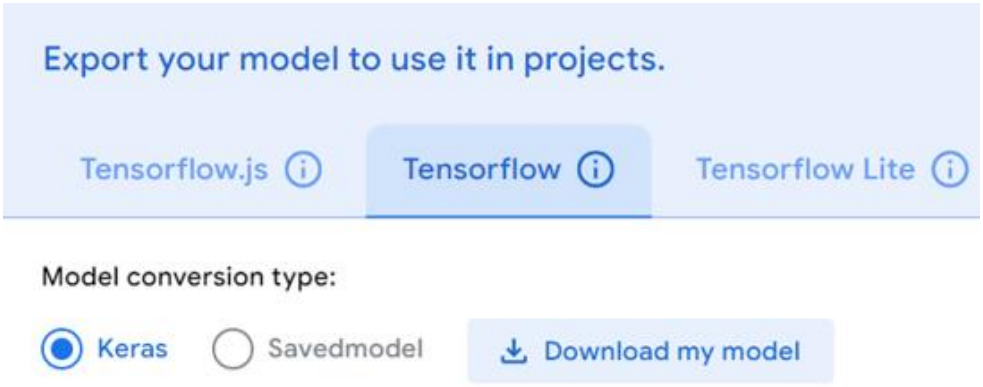


Fig.3 – Exporting the model

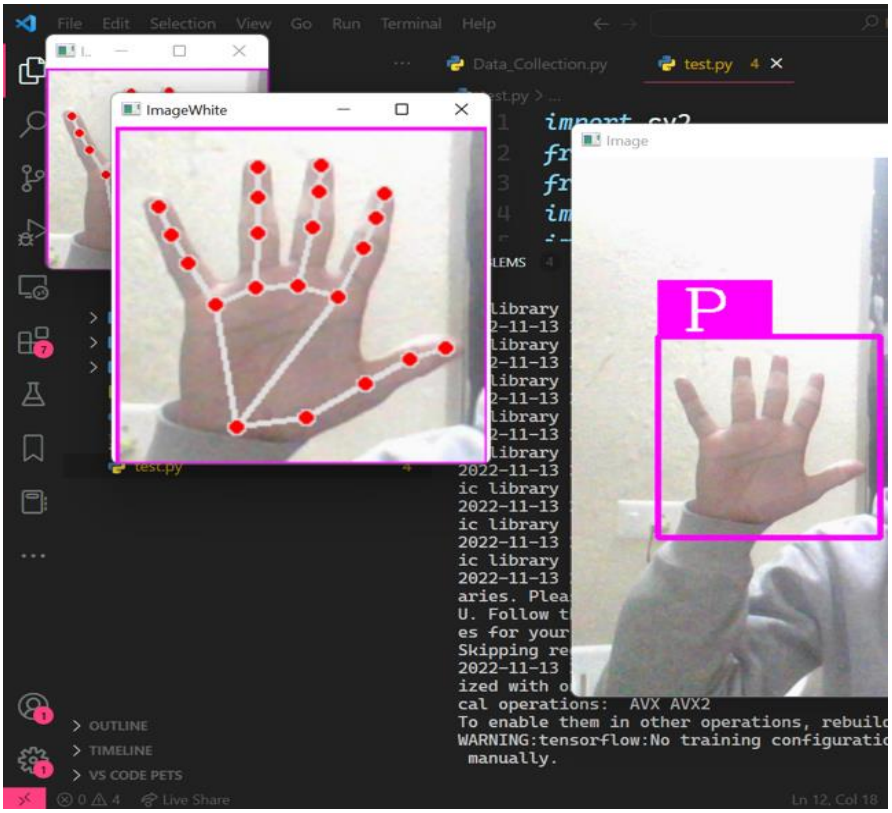


Fig.4 – Identifying the hand gesture

Upon completing this module, we jump onto creating and implementing the RL agent for effectively identifying the gameplay strategies used by an opponent.

RL agent formulation:

As discussed in the *Introduction* section, we formulated the RL agent as follows:

- **Environment** – In our particular case the opponent itself is the Environment, as the agent has to ultimately observe the user and identify the moves of R, P, S.
- **Agent** – Player 1 (Bot) tries to understand the pattern with which the opponent is playing.
- **State** – Win, Tie, Lost. This basically indicates if the robot has won/lost against the player, or if the game simply ended in a draw for that particular episode.
- **Action Space** – The three basic actions, which are: rock, paper, and scissors (encoded as 0,1,2, respectively for the robot to understand the move).
- **Reward** – **+1** for a win, **-1** for a defeat, and **0** for a tie.
- **Value** – The total reward in each episode serves as our value (or in words the return for that episode).
- **Policy** – During exploration, if the epsilon (ε) value is greater than generated random value, the agent randomly chooses any action. Whereas, if epsilon is less than generated random value the agent chooses greedy policy to choose the best action, i.e., it goes for **"GREEDY POLICY"**.

Q-Learning, which is an RL algorithm will determine the subsequent optimal course of action given the current situation. It randomly selects this action with the intention of maximizing the payout. Because the revised policy differs from the behaviour policy, Q-learning is also known as off-policy learning. In other words, it adds a value to the new state and calculates the reward for upcoming actions.

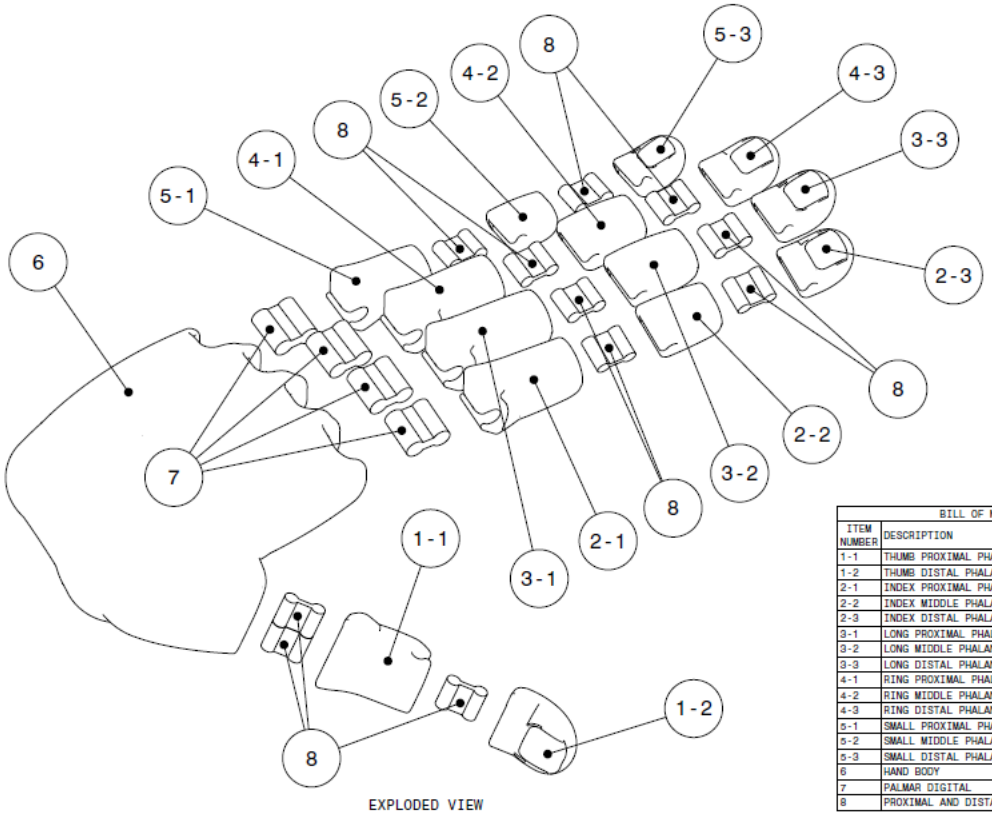
The Q-function uses the Bellman equation and takes two inputs: state (s) and action:

$$Q(S, A) = Q(S, A) + \propto [R(S, A) + \gamma (\max (Q' (S', A') - Q(S, A)\ ]$$

The entire process is executed as follows:

- Create an initial Q-table with all Q-values initialized to zero.
- Choose an action and perform it to update values in the table.
- Get the value of the reward and calculate the value, Q-value by using Bellman Equation.
- Then continue the same until the table is filled or an episode ends.

Printing and assembling the hardware components:



BILL OF MATERIALS

ITEM NUMBER	DESCRIPTION	NUMBER OFF
1-1	THUMB PROXIMAL PHALANX	1
1-2	THUMB DISTAL PHALANX	1
2-1	INDEX PROXIMAL PHALANX	1
2-2	INDEX MIDDLE PHALANX	1
2-3	INDEX DISTAL PHALANX	1
3-1	LONG PROXIMAL PHALANX	1
3-2	LONG MIDDLE PHALANX	1
3-3	LONG DISTAL PHALANX	1
4-1	RING PROXIMAL PHALANX	1
4-2	RING MIDDLE PHALANX	1
4-3	RING DISTAL PHALANX	1
5-1	SMALL PROXIMAL PHALANX	1
5-2	SMALL MIDDLE PHALANX	1
5-3	SMALL DISTAL PHALANX	1
6	HAND BODY	1
7	PALMAR DIGITAL	4
8	PROXIMAL AND DISTAL INTERPHALANGEAL	11

Fig.5 – Deign of the robotic hand

We employed the ‘Wanhao Duplicator i3 mini’ 3D printer to carry out the 3D printing task. The entire design, as shown in figure 5, is printed separately and later assembled. To print the materials part by part, we have considered the PLA filament as the printing material.

A thermoplastic polymer called polylactic acid (PLA) is frequently used as a filament in 3D printing. It is a popular choice for 3D printing due to its inexpensive cost, quick availability, and outstanding print quality. The fact that PLA is produced using renewable materials like sugarcane and cornflour is one of its key advantages. Due to this, it is a greener option to other 3D printing materials like polymers derived from petroleum. PLA has a smooth, glossy appearance and may generate prints that are incredibly accurate and detailed. With its low melting point and lack of a heated bed, it is also rather simple to print with. Because of this, beginners and educators alike should consider using it.

However, if we notice that parts 7, 8 are joint holders for the fingers and act as connecting parts, and are hard plastic after printing is done. Thus, they will restrict the movement of each finger near the joint separation. Therefore, to achieve realistic human finger like motion between the finger joints, we taken small parts of cycle tubes into consideration. This allows flexible motion and assist the overall finger to bend forward and then retract back whenever possible. To pull the finger down, we have used badminton racket strings, which are made of Nylon material.

Since nylon strings are not particularly heavy, they won't significantly increase the weight of the robotic hand. This is crucial to maintaining the hand's flexibility and ability to move fast. Nylon strings can endure the frequent bending and flexing needed to operate the fingers of a robotic hand since they are strong and long-lasting. Because nylon strings have little friction, they can pass easily and smoothly through pulleys and other mechanical parts. As a result, less force is needed to move the hand's fingers, which can enhance the hand's functionality as a whole.

Nylon strings are a practical material to utilize in a robotic hand because they are affordable and widely accessible.

Also, we have 3D printed only 3 fingers to denote the notation of R, P, S, put by the final assembly of robot:

- If 3 fingers are upright – it's a **Paper** symbol.
- If only 2 fingers are upright – it's a **Scissors** symbol.
- If none of the fingers are upright – it's a **Rock** symbol.

In order to achieve this, we used a pair of MG995 servo motors, which offer 180° motion. The strings from two fingers have been winded to one motor. And another motor has been used to wind the string from the remaining 1 finger. The entire hardware is powered by an Arduino UNO board, which is connected to the laptop. Two 5V pins have been chosen to pass current to the motors, and the command is received by the digital pins, for each of the two motors.

To enable Python support on the Arduino software, a python – based package called **'PyFirmata'** is taken in consideration. This offers seamless integration of a python script to easily run on the Arduino board, thereby giving commands to the motors, for specific degree of rotation directly from the laptop itself. It provides a set of easy-to-use functions for sending and receiving data to and from the Arduino, which can save time and effort when building projects. PyFirmata is compatible with a wide range of Arduino boards and sensors, making it a flexible choice for projects that require Arduino integration. Because PyFirmata is open-source and actively maintained, it is continuously updated to support new features and platforms.

RESULTS

We have experimented many 2 – 3 length patterns based seqeunces on the RL agent thereby giving commands to the robot.

- Sequence Pattern – P, R, S (abbreviates to Paper, Rock, Scissors)

```
Total Lost count: 6 || Lost rate: 2.9850746268656714
Exploiting....
choose  PAPER
Player move : ROCK, bot: PAPER, reward: 1, result: WIN, total_reward: 180
[[-0.67281915  1.25        0.36219638]
 [ 0.69747875 -0.66650496  1.25      ]
 [ 1.25        0.45616554 -0.60114273]]
Total Win count: 187 || Win rate: 93.03482587064677
Total Tie count: 7 || Tie rate: 3.482587064676617
Total Lost count: 6 || Lost rate: 2.9850746268656714
Exploiting....
choose  ROCK
Player move : SCISSORS, bot: ROCK, reward: 1, result: WIN, total_reward: 181
[[-0.67281915  1.25        0.36219638]
 [ 0.69747875 -0.66650496  1.25      ]
 [ 1.25        0.45616554 -0.60114273]]
Total Win count: 188 || Win rate: 93.53233830845771
Total Tie count: 7 || Tie rate: 3.482587064676617
Total Lost count: 6 || Lost rate: 2.9850746268656714
```

Fig.6 – RL agent performance for the pattern PRS

- Sequence Pattern – P, S, R (abbreviates to Paper, Scissors, Rock)

```
Total Lost count: 5 || Lost rate: 2.4875621890547266
Exploiting....
choose  ROCK
Player move : SCISSORS, bot: ROCK, reward: 1, result: WIN, total_reward: 173
[[ 0.09015329  1.25        0.96702657]
 [-0.48660394 -0.44029269  1.25      ]
 [ 1.25        0.21782755 -0.37171033]]
Total Win count: 184 || Win rate: 91.54228855721394
Total Tie count: 11 || Tie rate: 5.472636815920398
Total Lost count: 5 || Lost rate: 2.4875621890547266
Exploiting....
choose  PAPER
Player move : ROCK, bot: PAPER, reward: 1, result: WIN, total_reward: 174
[[ 0.09015329  1.25        0.96702657]
 [-0.48660394 -0.44029269  1.25      ]
 [ 1.25        0.21782755 -0.37171033]]
Total Win count: 185 || Win rate: 92.03980099502488
Total Tie count: 11 || Tie rate: 5.472636815920398
Total Lost count: 5 || Lost rate: 2.4875621890547266
```

Fig.7 – RL agent performance for the pattern PSR

- Sequence Pattern – P, S (abbreviates to Paper, Scissors)

```
Total Lost count: 7 || Lost rate: 3.482587064676617
Exploiting....
choose  SCISSORS
Player move : PAPER, bot: SCISSORS, reward: 1, result: WIN, total_reward: 170
[[ 3.34011095  3.62087154 -0.95268142]
 [ 1.21880688 -0.66601085  1.34483486]
 [ 1.72417431  0.52100395 -0.44140869]]
Total Win count: 181 || Win rate: 90.04975124378109
Total Tie count: 11 || Tie rate: 5.472636815920398
Total Lost count: 7 || Lost rate: 3.482587064676617
Exploiting....
choose  ROCK
Player move : SCISSORS, bot: ROCK, reward: 1, result: WIN, total_reward: 171
[[ 3.34011095  3.62087154 -0.95268142]
 [ 1.21880688 -0.66601085  1.34483486]
 [ 1.72417431  0.52100395 -0.44140869]]
Total Win count: 182 || Win rate: 90.54726368159204
Total Tie count: 11 || Tie rate: 5.472636815920398
Total Lost count: 7 || Lost rate: 3.482587064676617
```

Fig.8 – RL agent performance for the pattern PSR

As we can clearly see that for all the experimental patterns that we have tried, the win rate of the bot has been over 90 % in all the trials and the bot has been trained on that particular pattern for over 200 episodes. Moreover, tie rate and lost rates aren't exceeding 6 %, which the robustness the accuracy of the model in guessing the next moves of the human opponent.

Fig. 8 - The robot showing Rock as the symbol

Fig. 9 - The robot showing Paper as the symbol

Fig. 8 - The robot showing Scissors as the symbol

As shows in figures 7, 8, 9, we observe that the robot responds to the input given by the user and return back a move as a counter move by identifying the pattern in the moves put the opponent.

CONCLUSION

To the best of our knowledge, we have successfully developed a 3D printed robotic hand that can play rock-paper-scissors using reinforcement learning. The hand was able to learn the pattern in which the opponent was playing and adapt its strategy accordingly, resulting in consistently defeating the human opponent after 20-25 moves. This project demonstrates the potential of reinforcement learning in developing intelligent robotic systems.

In creating intelligent robotic systems that can adapt to and learn from their surroundings, reinforcement learning has a lot of potential, as demonstrated by this study. It also exemplifies the advantages of adopting 3D printing technology in the creation of robotic hands and arms, including its affordability, adaptability, and capacity for producing intricate and personalized designs.

We as team, gained a valuable learning experience by implementing this project, and it has opened up new avenues for further research and development in the field of intelligent robotics. We believe that our approach has the potential to be applied to other games and tasks that require learning and adaptability, and we hope to continue exploring the possibilities of reinforcement learning in the future.

REFERENCES

- https://towardsdatascience.com/a-beginners-guide-to-reinforcement-learning-using-rock-paper-scissors-and-tensorflow-js-37d42b6197b5
- https://www.researchgate.net/publication/360889680_A_Rock_Paper_Scissors_Game_using_Reinforcement_Learning_and_Q-Tables
- https://www.cs.toronto.edu/~guerzhoy/411_2018/lec/week10/REINFORCE.pdf
- https://www.bbc.com/news/technology-24803751
- http://ishikawa-vision.org/fusion/Janken/index-e.html