IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

# INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

## IN COMPUTER & COMMUNICATION ENGINEERING

Volume 9, Issue 6, June 2021

INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA

Impact Factor: 7.542

☎ 9940 572 462 6381 907 438 ✉ ijircce@gmail.com @ www.ijircce.com

# Creating an Autonomous Agent using Machine Learning

P. Vasavi[1], G. Nikhila[2], A.Prasanna[3], M.Manideep[4], Dr.Niraj Upadhayaya[5]

B.Tech. Student, Department of Computer Science and Engineering, JB Institute of Engineering and Technology, Moinabad, Telangana, India[1,2,3,4]

Associate Professor, Department of Computer Science and Engineering, JB Institute of Engineering and Technology, Moinabad, Telangana, India[5]

**ABSTRACT:** An autonomous agent is a smart agent that acts on behalf of its owner without the interference of that owner's entity. Intelligent agents are software entities that perform a set of activities independently or autonomously on behalf of a user or another programme, and do so using some knowledge or representation of the user's goals or wants. A system in a technological or natural environment, which perceives any or some status of that environment and acts on it in pursuit of its own agenda, is an example of such an agent. Agent-based Reinforcement Learning (ARL) is a type of reinforcement learning that involves direct experimentation. It makes no assumptions about the presence of an instructor who can offer examples for students to learn a task. Experience is the sole instructor in Agent-based RL. Because of its theoretical significance and possible applications in disciplines as diverse as Operational Research, Robotics, and gaming, RL has attracted the attention of Engineers and Computer Scientists.

**KEYWORDS**: autonomous agent, Artificial intelligence (AI), Agent-based Reinforcement Learning, chess.

## I.INTRODUCTION

Napoleon Bonaparte was defeated by a chess computer in only 24 moves in 1809. Only a highly complicated mixture of many years, according to creator Wolfgang von Kempelen, would be the secret of success. A puppet dressed in the garb of a Turkish pasha sat behind a wooden box. Opening the box's two doors revealed a plethora of rollers, levers, and gears. Von Kempelen wound the puppet at the start of every chess match, and it played a game by itself. The role of the turk was still unknown at the time of von Kempelen's death. In 1840, the box's final owner, Edgar Allan Poe, disclosed the truth: there was enough room inside for one person, so the turk was only a ruse. Nonetheless, this anecdote demonstrates the interest that autonomous chess playing may arouse even many years ago. This is one of the reasons why chess is a great model for developing robots. The autonomous chess playing agent is a technology that is meant to play board games against human opponents automatically. The fact that a significant number of chess-playing automata have been envisioned or built in the previous three centuries implies that robot chess may be attractive as a kind of entertainment.

Finding someone who plays at the same level as you might be tough. The chess robot comes in useful to overcome this conundrum and improve playing skills. Sensor boards have been used in most prior work on autonomous chess playing robots to detect moves utilizing magnetic pieces and sensors. It is accomplished using magnetic pieces and sensors placed beneath the board to detect piece movement. Instead, an image processing approach may be used to identify moves, since any simple chess board can be used to play chess, and we can play chess in any place, at any time, and without any specific prerequisites.

Sensory boards are used by chess-playing robots to detect movement using magnetic pieces and sensors. It is accomplished with the use of magnetic pieces and sensors placed beneath the board to detect piece movement. Instead, an image processing approach may be used to identify moves, since any simple chess board can be used to play chess, and we can play chess in any place, at any time, and without any specific prerequisites. Picking up and placing pieces with robotic arms is highly imprecise and unreliable since the gripper and piece orientation can change and cannot be held exactly in the same position for each second of time, causing difficulties in picking up parts. The design is also unstable since the gripper, along with the arm, is highly heavy, destabilizing the system, which is grounded by a single base.

## II.RELATED WORK

John McCarthy co-wrote the text that originated the phrase "artificial intelligence," created the Lisp programming language family, impacted the design of the ALGOL programming language, popularized time-sharing, pioneered garbage collection, and was a key figure in the early development of AI.

Alan Turing created the Turing Test for evaluating intellect in 1950. As a search engine, Claude Shannon provided a thorough analysis of chess playing.

IBM developed Deep Blue, a chess-playing computer. It was the first computer to defeat a reigning world champion in both a chess game and a chess match under normal time constraints. Grandmaster Joel Benjamin was a member of the development team for Deep Blue, which began in 1985 with the ChipTest project at Carnegie Mellon University. When the project was given the moniker Deep Thought, IBM hired the development team.

## III.METHODOLOGY

**Existing System**:

Chess is currently available on computers, notably the top-level chess system known as Deep Blue, which was invented by IBM (IBM).

We can view 32 pieces and 64 places for each board position.

The Deep Blue computer software is based on a dataset. This dataset includes all conceivable pawn placements for each and every pawn. It's a massive data set that necessitates the use of a supercomputer.

**Proposed System:**

The computation requirement in the proposed system is rather straightforward. The amount of data required is lower than in a normal system.

The agent in this case does not move the pawns based on a dataset; rather, it learns how to play with a human opponent.

Reinforcement Learning is used in this system.

**Proposed algorithm**

A. *Description of the Proposed Algorithm:*

A Deep Reinforcement Learning model looks at each state in an environment and utilises a neural network to decide which action to take. The agent then performs this action in a modified environment, making a new observation of the new state and preparing the next action. Each action provides the agent with a reward, and the agent's aim is to select the action that maximises this value in each condition.

Step 1: The initial stage is to build a simulation environment in which the model may observe states and execute actions, similar to a chess board. I used a combination of Python packages for this: chess, gym, and gym chess. These libraries enabled me to recreate and monitor all of the moves on the chess board while simulating a game, as well as gain analytical scores and clear explanations of the states and movements, including check, checkmate, and potential movements, among other things.

Step 2: Selection Criteria:

The intelligence of ChessAI is built on three predetermined strategies:

0 — Human Long-Term Memory: Uses a series of plays from a database of 20,000 human-played games, filtered solely for checkmate games. Because it always ends in a checkmate for the algorithm, this alternative focuses on the long term.

1 — Human Short-Term Memory: Next Word Predictor with LSTM previously trained on a huge game database using a 10-move sequence. The model reads chess movements as words in a phrase, using chess motions as a string. Chess takes on the form of a dialogue between players. Because it is based on the previous 10 steps, this option focuses on the near term and creates the following one.

2 — Stockfish Engine: The usage of Stockfish Engine. Stockfish is a free and open source chess engine that analyses the game and calculates some moves ahead of time in order to determine the optimal move.

**System Architecture:**

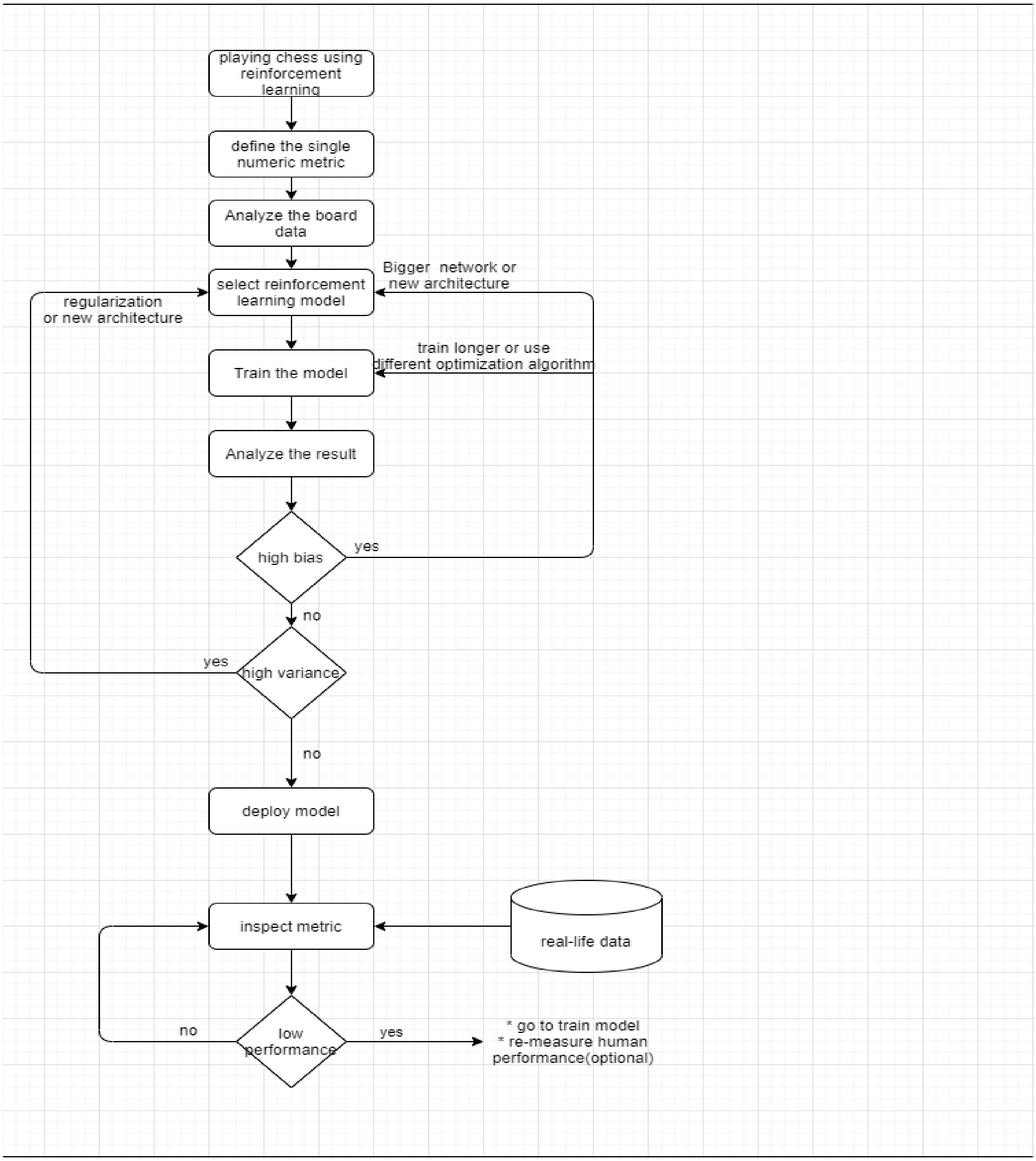
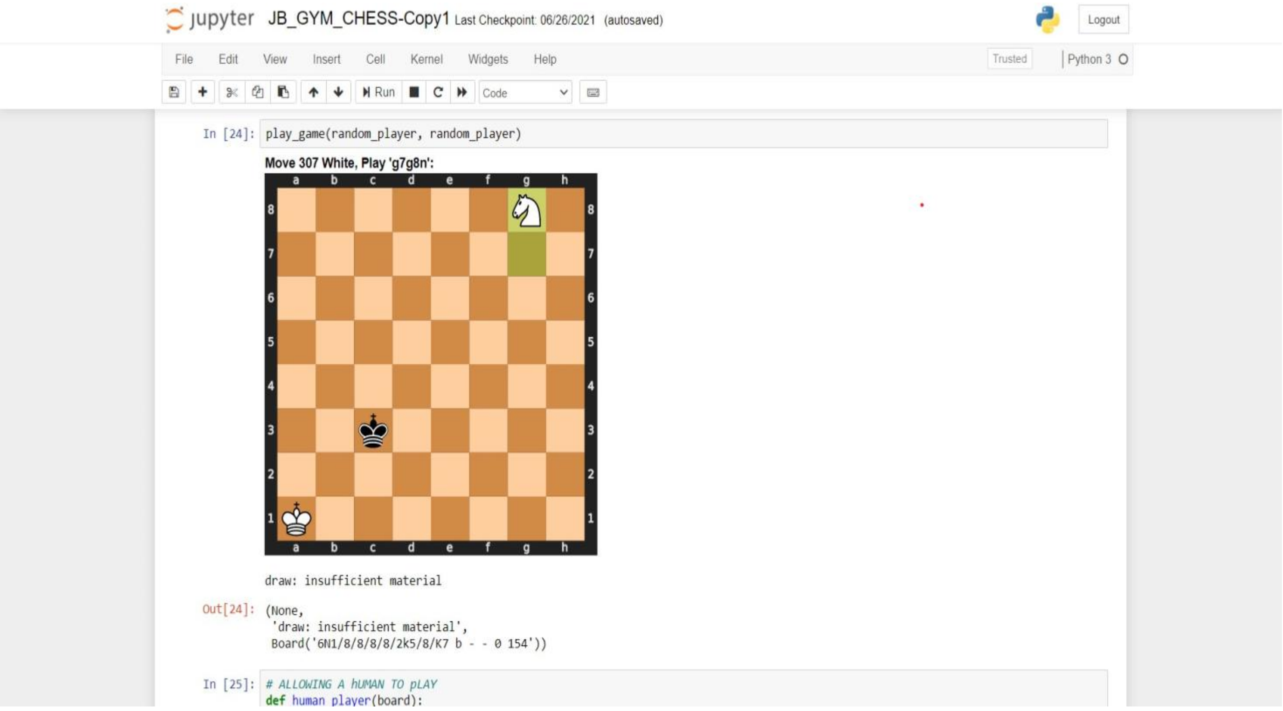

playing chess using reinforcement learning
define the single numeric metric
Analyze the board data
select reinforcement learning model
Bigger network or new architecture
regularization or new architecture
Train the model
train longer or use different optimization algorithm
Analyze the result
high bias
yes
no
high variance
yes
no
deploy model
inspect metric
real-life data
low performance
no
yes
* go to train model
* re-measure human performance(optional)

Fig.3.1: System Architecture

## IV. EXPERIMENTAL RESULTS


jupyter JB_GYM_CHESS-Copy1 Last Checkpoint: 06/26/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Move 307 White, Play 'g7g8n':
a b c d e f g h
8 7 6 5 4 3 2 1

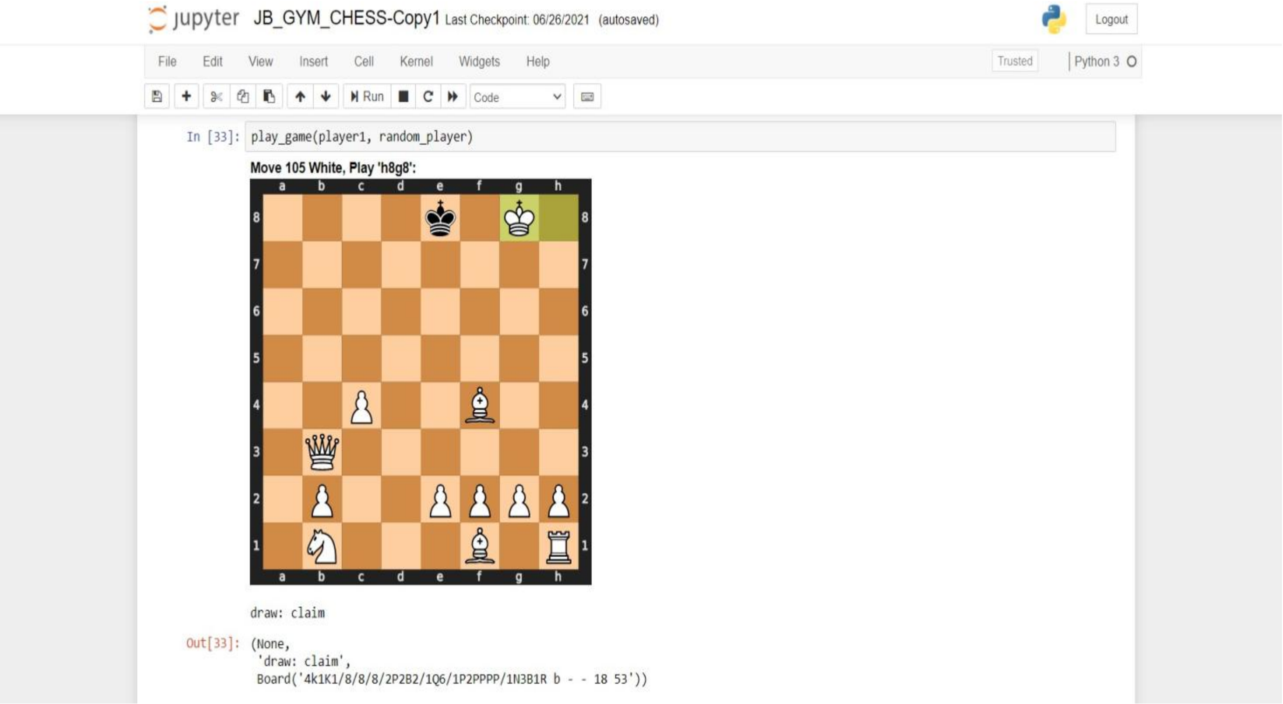
```
In [24]: play_game(random_player, random_player)

draw: insufficient material

Out[24]: (None,
          'draw: insufficient material',
          Board('6N1/8/8/8/8/2k5/8/K7 b - - 0 154'))

In [25]: # ALLOWING A hUMAN TO pLAY
         def human_player(board):
```

Fig.4.1:A Game played between Agent and a Random player, which resulted in a draw match.

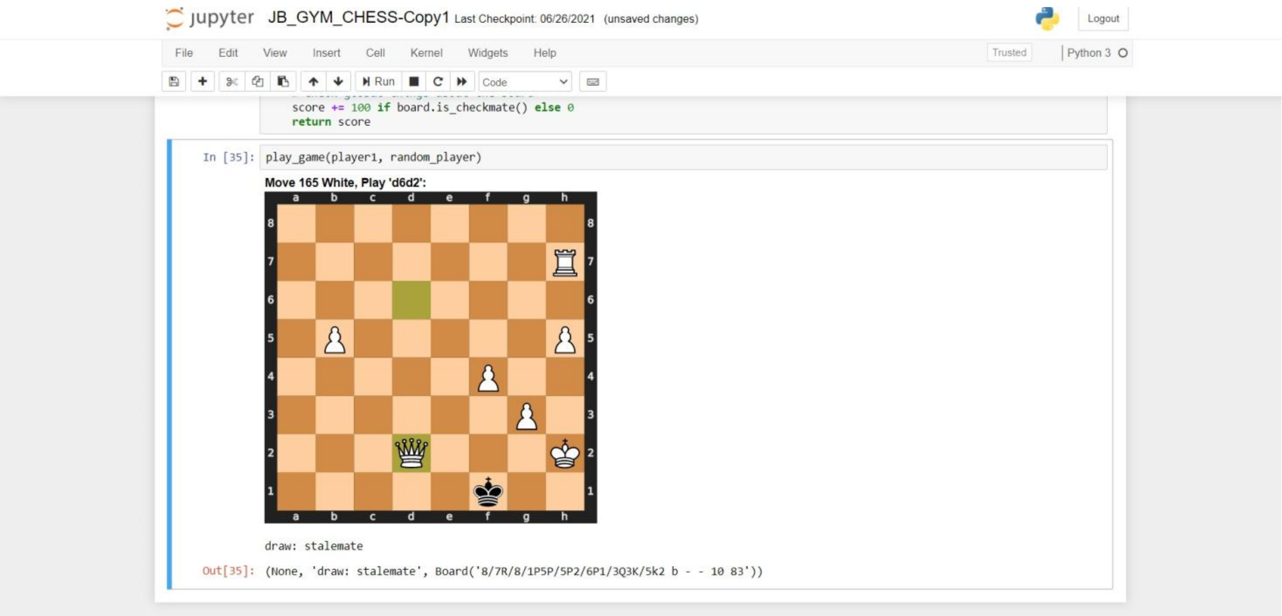

jupyter JB_GYM_CHESS-Copy1 Last Checkpoint: 06/26/2021 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Move 105 White, Play 'h8g8':
a b c d e f g h
8 7 6 5 4 3 2 1

```
In [33]: play_game(player1, random_player)

draw: claim

Out[33]: (None,
          'draw: claim',
          Board('4k1K1/8/8/8/2P2B2/1Q6/1P2PPPP/1N3B1R b - - 18 53'))
```

Fig.4.2:A Game played between Agent and a Random player, in which Agent won the match.


jupyter JB_GYM_CHESS-Copy1 Last Checkpoint: 06/26/2021 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
a b c d e f g h
8 7 6 5 4 3 2 1

```
    score += 100 if board.is_checkmate() else 0
    return score
```

```
In [35]: play_game(player1, random_player)
```

Move 165 White, Play 'd6d2':

```
draw: stalemate
Out[35]: (None, 'draw: stalemate', Board('8/7R/8/1P5P/5P2/6P1/3Q3K/5k2 b - - 10 83'))
```

Fig.4.3:A Game played between Agent and a Random player, which resulted in a draw match.

## V.CONCLUSION

The purpose of agent-based reinforcement learning (RL) is to determine how successfully or poorly it has performed the needed task. These signals are frequently linked to a dramatic circumstance, such as completing a subtask (reward) or failing completely (punishment), and the learner attempts to optimise its behaviour by utilising a performance measure.

As a result, it learns by direct exploration while also doing nothing.

## VI. FUTURE ENHANCEMENT

The principle of reinforcement learning (RL) is simple. Reinforcement learning is basically an agent learning to interact with an environment based on feedback signals it receives from the environment at a very high level. This distinguishes it from other machine learning methods in which a learning agent may observe a right answer during training. Reinforcement learning is a type of learning that allows you to learn from your mistakes.

## REFERENCES

1. Blakely, B. & Theron, P., 2018. Decision flow-based Agent Action Planning.
2. Amont, J.-P., Occello, M. &Lagrèze, A., 2010. A multiagent approach to manage communication in wireless instrumentation systems. Measurement, 43(4), pp. 489-503.
3. Kott, A. et al., 2018. Initial Reference Architecture of an Intelligent
4. Samuel, A. L. (1959). Some studies in machine learning using the game of checkers.
5. IBM Journal on Research and Development, 3:211-229. Sutton, R.S. (1988). Learning to predict by the method of temporal differences. Machine Learning, 3:9-44. Sutton, R.S., and Barto, A. G. (1998)

# INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details