Check for updates

Computer Animation and Virtual Worlds

WILEY

RESEARCH ARTICLE OPEN ACCESS

Learning to Play Like Humans: A Deep Neural Framework for Believable Agent Development in Virtual Worlds

Naveed Anwer Butt[1] | Kiran Shahzadi[1] | Mian Muhammad Awais[2] | Julien Brito Ballester[3,4,5] | Carmen Lili Rodriguez Velasco[3,6,7] | Nagwan Abdel Samee[8] | Imran Ashraf[9]

[1]Department of Computer Science, Faculty of Computing and Information Technology, University of Gujrat, Gujrat, Pakistan | [2]Department of Computer Science, SBASSE, LUMS Lahore, Opposite Sector U D.H.A Lahore Cantt, Lahore, Pakistan | [3]Universidad Europea del Atlantico, Santander, Spain | [4]Universidad Internacional Iberoamericana, Arecibo, Puerto Rico, USA | [5]Universidad de La Romana, La Romana, Dominican Republic | [6]Universidad Internacional Iberoamericana, Campeche, Mexico | [7]Universidade Internacional do Cuanza, Cuito, Bie, Angola | [8]Department of Information Technology, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia | [9]School of Computer Science and Engineering, Yeungnam University, Gyeongsan, Republic of Korea

Correspondence: Imran Ashraf (imranashraf@yu.ac.kr)

Received: 28 July 2026 | **Revised:** 27 October 2026 | **Accepted:** 31 October 2026

Keywords: deep learning | gaming robots | human behavior imitation | robots | virtual world

ABSTRACT

Many existing intelligent agents still behave in a mechanical way leading to user dissatisfaction and lack of engagement. To create more realistic and accurate imitations, this study proposes an imitation learning framework that explores deep learning (DL) and traditional machine learning, to build a computational model. The approach is tested using two case studies: Pac-Man to predict human behavior, and Player Unknown's Battlegrounds (PUBG) to evaluate how well the model could imitate it. Recurrent neural networks (RNN), gated recurrent units, long short-term memory networks (LSTM), random forest, extreme gradient boosting (XGBoost), etc. To assess model performance, mean squared error, root mean squared error, and accuracy are used. Among the employed models, RNN and LSTM achieved near-identical and highest accuracy on Pac-Man (99.98% and 99.96% on the combined dataset), clearly ahead of GRU, random forest and XGBoost, while XGBoost performed best on PUBG with 97.37% accuracy. A dedicated discussion explains why RNN performs well in the low-dimensional, short-horizon Pac-Man setting and lists its advantages and limitations. The Mann–Whitney U Test and the Cosine similarity tests are also applied to measure how believable the imitated behaviors were. Neither analysis detected a statistically significant difference between human and model traces, and cosine similarity was approximately 0.98.

1 | Introduction

The goal of developing virtual worlds that inspire real human interaction is at the forefront of interactive computing and artificial intelligence (AI). When in-game characters act in ways, consistent with human expectations, user immersion is greatly increased [1–4]. The believability of non-player characters (NPCs) or AI agents has a significant impact on user engagement and pleasure in these environments [2, 5]. As virtual worlds progress from organized games to wide metaverses, the necessity for intelligent agents grows, particularly for those capable of simulating human-like behavior. These agents are essential for educational applications, therapeutic simulations, and human-AI cooperation frameworks in addition to entertainment [3–7]. Despite developments, a large number of the currently used AI-driven agents, display mechanical or predictable actions, which reduces user engagement and satisfaction [4, 5, 8]. The gap highlights how urgently sophisticated computational frameworks are needed for believability.

This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2026 The Author(s). *Computer Animation and Virtual Worlds* published by John Wiley & Sons Ltd.

It seems plausible to refer to the idea of believability as "human-like." Nonetheless, it remains a broader concept with no established definition. One of the first definitions, proposed by Thomas and Johnston, argues that believability is about producing an illusion of life [6]. Likewise, this effect is referred to as a suspension of disbelief by Joseph Bates [7], in which the user should be able to think that the virtual environment is real. A clear conclusion is that believability is often defined based on agent behavior. As previously mentioned, for Tencé et al., it comes down to creating an agent that can perceive its surroundings; make decisions according to these perceptions and its own state; and perform such decisions [8]. The requirements are streamlined in a previous work [7], which states that a believable agent should not strive for perfection but rather for not being wholly unreal. Its advantages and complexity have prompted several contributions to the development of human-like AI. Numerous approaches are being used in both industry and academia to try to address this challenging idea [7, 9, 10].

We observe many approaches from both industry and academia that try to address this challenging idea. Some achieve this by hard-coding particular behaviors [9, 11], while others directly imitate other players' gameplay [7, 8]. Some approach the problem through player modeling [12, 13], while others employ a combination of existing approaches [14, 15]. An approach to believability has been the application of subjective computing methods [16], notably player modeling. According to [16–18], player modeling is connected to effective computing, but more particularly to video games. It is about creating models from player data. These models are built to better understand how individuals perceive and respond to the virtual environment. Machine learning algorithms [19–21] are used to create models by analyzing player-environment interactions and assigning labels based on experience or impact.

Imitation and prediction are two main game player modeling sub-tasks that have been achieved mainly via supervised and unsupervised learning methods in the domain of robotics and games. These are mainly artificial neural networks (ANN), reinforcement learning (RL), Fuzzy logic, genetic algorithms (GA), hidden Markov models (HMM), support vector machines (SVM), etc. [22–24]. Supervised learning and RL have been prominent in modeling player behavior and believable agents. Neuroevolution, scripting techniques, finite state machines, etc., are also in use. These approaches exhibit some limitations; unable to generalize unseen circumstances [24], dependence on fitness function either during the evolution of human-like or AI agents. This research aims to take those theories and build an equivalent model in the domain of Game AI.

Recently, there has been a lot of interest in developing AI models that can imitate human behavior, notably through imitation learning. Imitation learning enables AI agents to learn directly from human players' demonstrations, eliminating the need for manually designed incentive schemes, which are frequently necessary for reinforcement learning. This method is increasingly being applied in gaming settings after displaying outstanding performance in tasks ranging from robotic manipulation [25, 26] to autonomous driving [27].

A dynamic paradigm for modeling high-dimensional, complicated data, deep learning (DL) is especially well-suited for simulating human behavior in virtual settings. Because DL-based techniques can learn directly from human demonstrations, they can capture subtle behavioral patterns that are hard to manually encode, in contrast to classic rule-based or RL approaches. Chen et al. [28] demonstrated how neural networks can capture nuanced human interactions by using a DL framework that analyzes multimodal behavioral inputs to successfully assess presence in virtual reality. DL uses hierarchical feature learning, as opposed to rule-based systems, to imitate human demonstrations directly, including subtleties like contextual adaptability and strategic improvisation [29].

The ability of DL models to analyze intricate data patterns provides a number of significant benefits for simulating human-like behavior in virtual settings. By immediately processing raw inputs like game states and player actions, the end-to-end learning strategy removes the need for human feature building and allows for more realistic behavior replication [30]. In complicated games like Player Unknown's Battlegrounds (PUBG), advanced architectures such as long short-term memory (LSTM) models and transformers are excellent at temporal modeling, capturing long-term dependencies that are crucial for multi-step tactics [31]. Moreover, multimodal data streams may be efficiently integrated and analyzed by DL frameworks, which combine visual, kinematic, and semantic data to produce behavior models that are more thorough and accurate [28]. The development of virtual agents that can replicate human decision-making processes with previously unheard-of precision and subtlety is made possible by these qualities taken together.

One of the main challenges for human-like behavior is evaluating how believable agents, characters, and simulated actors are. There are two sorts of agent believability: user (player) and character (non-player) [32]. Although the literature offers many different strategies, there isn't a single, widely accepted method for evaluation. The most often used method for evaluating believability is the Turing Test [33] for evaluating non-player behavior [32, 34]. The Turing Test has been criticized in the past and is inappropriate for evaluating the believability of a game. Turing tests are subjective, frequently erroneous, and time-consuming due to the several individuals involved in evaluating agents.

Agents can also be evaluated quantitatively. When believability efficacy is the primary objective, this style has received greater attention. This appears to be the approach used when quantitative scores and error calculations can be computed as in [35], where the score might indicate the player's or agent's progress. In order to "quantify" how well a driver was driving, Abbeel and Ng [36] used a driving simulator to count the number of crashes. In the end, these techniques measure the degree to which values match specific standards. Statistical validation through the Mann–Whitney U test and Cosine Similarity test ensures that the imitated behaviors are not only accurate but also perceived as realistic.

This research is based upon two of our previous works in the subject of believable agent design [37, 38]. In the prior work, we established a case-based reasoning (CBR) framework for reproducing human-like characteristics in virtual contexts by replaying

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

human traces [37]. We then proposed a hybrid model that further improves behavioral realism and adaptability in intelligent agents by combining neural architectures with an adaptive neuro-fuzzy inference system (ANFIS) [38]. To develop a scalable and generalized imitation learning framework, the current study broadens this research trajectory by investigating DL approaches. More expressive and reliable agent modeling is achieved by using recurrent architectures like recurrent neural network (RNN), gated recurrent unit (GRU), and LSTM to characterize temporal relationships in human gaming behavior. A significant advancement in creating human-like behavior in virtual agents has been made with the switch from symbolic and neuro-fuzzy approaches to deep neural networks.

The current work develops a human-like AI agent that can effectively imitate the actual behavior of individual players in a gaming environment that is considered believable. The objective is to solve the issues of training agents in the Pac-Man domain, since it is a valuable setting for collecting data and experimenting with agents. To achieve this, the study builds AI agents by learning from how specific players win or lose games, using their recorded gameplay data. An automated method based on a collection of these game records was used to make sure the results are accurate. The model was trained separately on winning and losing traces, allowing comparison of predictive patterns. No causal attribution is attempted.

The present work develops a framework for determining a measure of human behavior imitation using likelihood ratios by exploring two learning approaches, DL and traditional machine learning (ML). This study focused on RNN, GRU, LSTM, random forest (RF), and extreme gradient boosting (XGBoost). Statistical validation using the Cosine similarity and Mann–Whitney U tests guarantees that the imitated behaviors are regarded as realistic and correct.

This study presents an empirical investigation of DL and traditional machine learning approaches for predicting human gameplay behavior from recorded demonstrations. The main contributions of this work are summarized as follows:

- Coordinate-Based Gameplay Trace Representation and Processing Pipeline: A structured data-processing pipeline is implemented to transform recorded human gameplay traces into coordinate-based state representations for supervised learning. The framework uses the current spatial positions of Pac-Man and two surrounding ghosts to construct input–output pairs for predicting Pac-Man's subsequent position. This representation provides a consistent basis for training and evaluating machine learning and DL models using recorded human gameplay data.
- Comparative Evaluation of Recurrent and Tree-Based Learning Models: A comparative experimental study is conducted using five established AI/ML architectures: Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Random Forest (RF), and Extreme Gradient Boosting (XGBoost). Their predictive performance is assessed using accuracy, Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) in Pac-Man and PUBG experimental settings. The study examines differences in the observed performance of recurrent and tree-based approaches under the respective evaluation conditions.
- Player- and Gameplay-Outcome-Dependent Performance Analysis: The study investigates variations in model performance across individual players, winning and losing gameplay scenarios, and aggregated gameplay datasets. This analysis provides empirical observations regarding the influence of player-specific behavior, gameplay outcomes, and dataset aggregation on prediction performance. It also highlights differences in the observed strengths and limitations of the evaluated models within the investigated settings.
- Quantitative Evaluation of Human–Agent Behavioral Similarity: A complementary evaluation approach is applied using predictive performance metrics, the Mann–Whitney U test, and cosine similarity to examine agreement between recorded human gameplay and model-generated outputs. These analyses assess selected distributional and vector-based characteristics of gameplay behavior, providing exploratory evidence of behavioral similarity. The evaluation distinguishes predictive accuracy and automatic similarity measures from human-perceived believability, which requires additional validation.
- Collectively, these contributions provide an empirical assessment of established DL and machine learning methods for human gameplay imitation. The study characterizes differences in reported model performance across the evaluated settings and identifies methodological considerations for developing and assessing human-like behavior in virtual agents.

To achieve these objectives and verify the proposed framework, this study uses two case studies: Pac-Man (emphasis on low-dimensional action prediction inside a traditional grid setting) and PUBG (scalability in highly complicated circumstances that require real-time adaptation and strategic planning). By bridging important gaps in agent design, this study makes it possible for virtual characters to interact with users in a way that is authentically human, thereby leading to increased engagement.

After the introduction, related work is given in Section 2. The methodology part is explained in Section 3, followed by results and discussion in Section 4 (including the new Sections 4.6 and 4.7 on the performance of RNN and on the advantages and disadvantages of the best model), while the conclusion is given in Section 5.

2 | Related Works

Human behavior modeling has found practical use in the development of intelligent agents for video games over the previous decade. The purpose of the agents' development is to play like humans. Recently, there has been a growing focus on developing video games that can simulate or replace human players. This player-focused design approach helps create better and more enjoyable gaming experiences for users [39, 40]. When the purpose of AI is to entertain the human player rather than defeat the human player, behavioral modeling of players is practically

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

required. To enhance realism and unpredictable gameplay, AI may imitate human behavior. Human behavior modeling has been described as translating observations into actions based on demonstrations, to complete a task involving the surrounding entities. To test the game's quality, it needed a lot of human testers, technical skills, and domain expertise [41, 42]. Most often, ad-hoc authoring, tree search, evolutionary algorithms, supervised learning, reinforcement learning (RL), and unsupervised learning are adopted for autonomous and assisted game players [43–45]. The investigations [43–45] are classified as indirect or direct imitation [46]. Developers mostly employ supervised learning in direct imitation to teach the agent to behave in the same way as the human player in the same circumstance. The human gameplay traces are used as input to train the created AI. The output is the player's action at the same point in the game, and the input parameters are chosen based on the player character's surroundings. A significant issue is the generalization to unseen scenarios: the agent may act in ways that differ substantially from what the human player would have done in the same situation. The primary cause is that the goal of this training is to have the player repeat the same choices in the same situation while doing poorly in terms of game objectives [46]. To solve the generalization problem, it was proposed to combine RL with indirect imitation. An agent's behavior is evaluated for human-likeness using an optimum reward function.

Very recent studies confirm that human-likeness remains an open and actively researched problem. Świechowski and Ślęzak [5] survey 13 conceptual and technical challenges in building human-like agents for games and discuss how human-likeness can be measured. Lin [47] argues in a 2025 dissertation that defining and measuring human-likeness is at least as important as the choice of the imitation technique itself.

2.1 | Machine Learning Approaches

Recent developments in supervised learning have shown impressive potential for predicting human behavior. Supervised learning algorithms, such as RF and support vector machines (SVMs), have been extensively used in educational robots to categorize student involvement levels and predict learning outcomes [48]. In some implementations, these systems achieve classification accuracy of over 85% by analyzing multimodal data from robot interactions, such as gaze patterns, facial expressions, and performance metrics [48]. For processing data on visual activity, convolutional neural networks (CNNs) have emerged as the most popular design. By using CNNs to analyze student interactions with augmented reality (AR)-enhanced robotic systems, robotics in education has made it possible to modify teaching tactics in real time [49]. According to recent research, as compared to conventional techniques, AR-robotic systems that use CNN-based visual analysis can boost student engagement by 30%–40% [49].

A recent large-scale benchmark of 20 models on 111 tabular datasets reports that DL models frequently do not outperform traditional methods such as gradient boosting machines on tabular data, and it characterizes the conditions under which deep models do excel [50]. This is relevant to the present study because the Pac-Man state is a low-dimensional tabular record of six coordinates, so tree ensembles such as RF and XGBoost are natural competitors to recurrent networks.

In many applications, transformer topologies have substantially replaced conventional RNNs for sequential behavior modeling. The Centaur model, which is based on Llama 3.1 70B, shows how human decision-making may be predicted and simulated in a variety of circumstances using big language models that have been fine-tuned using human behavioral data [51]. At the same time, recurrent models are being revisited: Feng et al. [52] show that simplified LSTM and GRU variants with fewer parameters can reach competitive performance on a range of sequence tasks, rivaling recent models including Transformers. Yang [53] reports a systematic efficiency–accuracy trade-off between GRU and LSTM, with GRU training 30%–40% faster using 25%–26% fewer parameters, while LSTM gave lower prediction error in the forecasting tasks studied.

Modern unsupervised learning approaches have evolved significantly, particularly in the analysis of social media behavior. Contrastive learning frameworks now enable models to extract meaningful representations from unlabeled behavioral data [54]. These techniques have proven valuable in mental health detection, where models can identify patterns indicative of depression or anxiety from social media activity without explicit labeling. Variational autoencoders (VAEs) have shown particular promise in robotics applications, enabling robots to learn compact representations of human movement patterns. This capability is crucial for socially assistive robots that need to adapt to individual users' physical behaviors. Recent implementations demonstrate that VAEs can reduce the dimensionality of motion capture data while preserving 95% of the variance, facilitating more efficient behavior modeling [55].

In human behavior modeling, RL has made significant strides, especially with the use of DL. Robotic path planning in dynamic situations has been transformed by deep RL techniques such as proximal policy optimization (PPO) and soft actor-critic [56]. With success rates over 90% in congested settings, these methods allow robots to acquire navigation tactics that take into consideration the erratic motions of people [56]. Similarly, hierarchical RL systems have made it possible for more complex behavior imitation in gaming applications. High-level strategic planning and low-level reactive controls are combined in modern implementations to closely resemble human decision-making in intricate games [51]. The Centaur model serves as an example of how RL may be used with big language models to generate exploration techniques in decision-making tasks that are similar to those of humans [51]. In addition, with models like PPO [57], asynchronous advantage actor-critic (A3C) [58], and deep Q-networks (DQN), deep DRL has made great strides. Despite the fact that training is still sample-inefficient, they have allowed agents to perform well in arcade-style games.

Hybrid imitation learning (IL)-RL systems suggest integrating online learning via reinforcement signals with expert demonstrations. By bootstrapping using IL, these techniques offer quicker convergence while mitigating the drawbacks of sparse reward situations. A 2025 example is the hierarchical framework of Sun et al. [59], which learns basic behaviors and intentions of human players at the lower level through imitation learning and

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

generalized policies at the higher level through reinforcement learning, in order to obtain human-like strategies in modern 3D games. Deep deterministic policy gradient and actor-critic with experience replay have also demonstrated potential in continuous control environments such as racing simulations (e.g., TORCS).

2.2 | Vision-Based Deep Learning for Game Environments

CNNs are frequently employed in gaming contexts to handle input that is based on images. End-to-end systems that learn straight from raw pixel input include DeepMind's DQN. Modular pipelines, on the other hand, improve interpretability by separating the perception and policy modules. The incorporation of vision encoders with IL [60] (e.g., Visual Imitation with encoders) enhances sample efficiency and behavior accuracy in contemporary 3D games, like CS:GO and Minecraft, hence facilitating human-like gaming. Reinforcement-guided fine-tuning and extensive pre-training are beneficial for these models. Spick et al. [61] trained agents by behavioral cloning to play Doom 2 from pixel data and compared IL with RL in terms of humanness using camera movement and trajectory data; they report that the IL agents behave more human-like than traditional AI, whereas RL performs better on pure performance. Justesen et al. [62] train neural networks on human trajectory data to play a VALORANT-like 2v2 tactical shooter using a small set of ray-cast sensors instead of pixels, and report through human evaluation that the agents provide human-like gameplay under modest CPU budgets.

2.3 | Multi-Agent and Multi-Action Policies

IL in concurrent-action and multi-agent scenarios has been studied recently. In first-person shooter and multiplayer online battle arena (MOBA) games, policies that are taught to manage many simultaneous activities (such as movement and firing) more closely resemble human gaming. Gao et al. [63], for instance, presented a centralized critic architecture that facilitates improved agent coordination for complicated behaviors. Similar to this, Baker et al. [64] used imitation-based initialization in multi-agent reinforcement learning setups to illustrate emergent team play. Furthermore, emergent team coordination through imitation-based initialization is demonstrated by collaborative multi-agent frameworks (e.g., hierarchical imitation of soccer play in humanoid agents). To balance individual and group behavior realism, these systems include central critics or graph-based interaction models, allowing for realistic training, research, and entertainment simulations.

2.4 | Deep Learning Advances

DL for behavior modeling has seen a number of revolutionary advancements in recent years, including:

i. Multimodal Fusion Architectures: Visual, aural, and kinematic data streams are being combined in modern systems to describe human behavior more comprehensively. To customize treatment plans in robotic rehabilitation, hybrid CNN-RNN systems evaluate physiological signals and movement data [55]. The results of these systems are 15%–25% better than those of unimodal methods [55].

ii. Foundation Models for Behavior Prediction: A paradigm change has occurred with the advent of large-scale behavioral models such as Centaur. Developed on Psych-101, a dataset with 10 million human selections from 160 tests, Centaur has hitherto unheard-of generalization skills. It correctly forecasts how people will behave in new situations, such as altered task structures and completely uncharted territory [51].

iii. Explainable AI (XAI) in Behavior Modeling: Recent research has concentrated on improving the interpretability of behavior models in order to address the "black box" issue. Saliency maps and attention processes are being used to assist in explaining why models predict particular behaviors, which is important for therapeutic and instructional purposes [55]. Decision tree-based explainable models in career recommendation systems provide transparent reasoning and attain 85% accuracy [65].

iv. Edge AI for Real-time Behavior Analysis: Real-time behavior modeling without reliance on the cloud is made possible by the implementation of lightweight neural networks on robotic systems. It is possible to run sophisticated models on embedded devices while preserving over 90% of the initial accuracy by using quantization and knowledge distillation techniques [55]. In commercial games, where most computation is reserved for rendering, compute-efficient human-like agents are likewise a stated design requirement [62].

2.5 | Identified Research Gaps

The following is a list of the primary concerns raised by the body of current literature:

- Limited Realism in Agent Behavior: Although many AI agents demonstrate high performance in decision-making tasks, their behavior often lacks believability or naturalness. Most models optimize for performance rather than mimicking human play-styles, which reduces immersion in game environments [39, 40].
- Underutilization of Sequential and Temporal Dependencies: Many existing imitation learning experiments, particularly in real-time gaming settings, fail to take advantage of temporal patterns in human behavior employing deep sequential models such as RNNs, GRUs, and LSTMs. This causes behavior that does not represent the progression and decision rhythm of human players [42, 66].
- Unable to Generalize Across Environments and Games: The majority of imitation learning models are domain-specific and have poor generalization across game genres (e.g., from complicated first-person shooter games like PUBG to predictable landscapes like Pac-Man). According to [43, 44], cross-domain validation is still an unresolved topic.
- Inadequate Contextual Awareness Integration: Existing agents frequently act on raw input data, lacking higher-level

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

semantic knowledge such as player goals, team strategy, and contextual architecture. This inhibits their capacity to make judgments consistent with human purpose [41, 45].

- Lack of flexibility and customization: Many agents do not adapt to particular playing styles since they are trained on general datasets. The development of models capable of learning and reproducing distinct behavioral characteristics for player-specific customization is severely lacking [46].
- Metrics for Evaluation That Only Consider Accuracy: The majority of research evaluates agent performance based on quantitative standards like accuracy or score, disregarding qualitative assessments of behavioral similarity or plausibility. For behavioral comparison, few studies use statistical methods like cosine similarity or the Mann–Whitney U Test.

2.6 | Positioning of the Present Study With Respect to Recent Works (2024–2026)

Recent studies on human-like game agents mainly rely on high-dimensional inputs (pixels or ray-cast sensors) and on human or subjective evaluation of human-likeness [61, 62], or combine IL with RL for generalization [59]. Surveys and theses emphasize that the measurement of human-likeness is still an open problem [5, 47]. In contrast, the present study:

i. uses a compact, fully observable coordinate representation,

ii. compares three recurrent architectures with two tree-based learners under the same protocol, which connects to the ongoing discussion of when deep sequential models are, and are not, preferable to tree ensembles or to lighter recurrent cells [50, 52, 53], and

iii. validates believability with two automatic statistical measures (Mann–Whitney U and cosine similarity) instead of human judges.

We do not claim superiority over the vision-based or hybrid IL–RL systems above, since they address different input modalities and tasks.

3 | Proposed Framework

This work introduces a unique framework that uses DL and ML approaches to develop believable AI agents in virtual worlds by imitating human game players. The method is based on using human trace data, which is gathered during game replays, to simulate and forecast human-like behavior. Figure 1 shows the Game AI Data Generation Pipeline in detail and Figure 2 illustrate End-to-end workflow of the proposed framework for human-like gameplay modeling and believability validation.

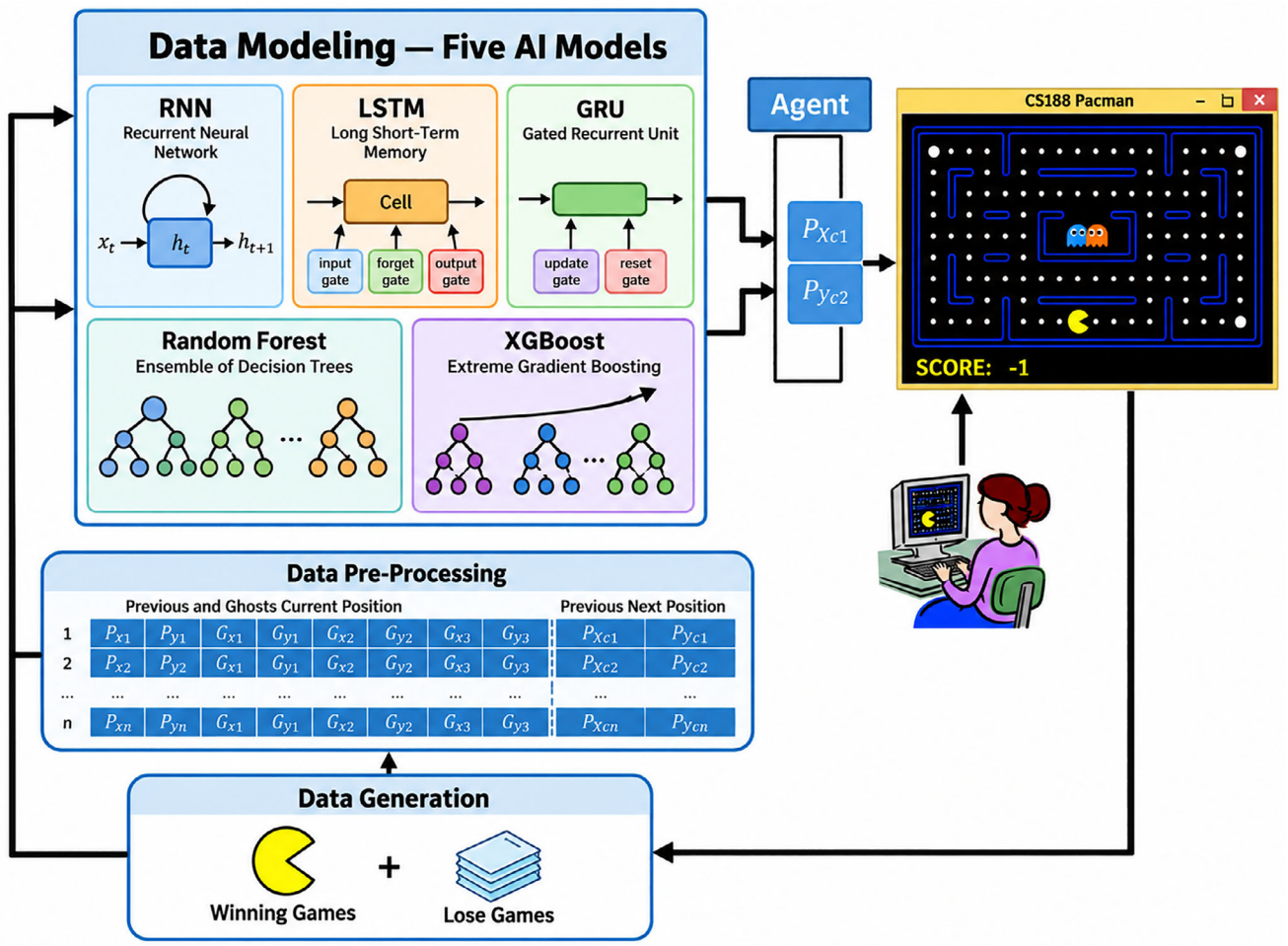


FIGURE 1 | Workflow of the proposed approach.

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

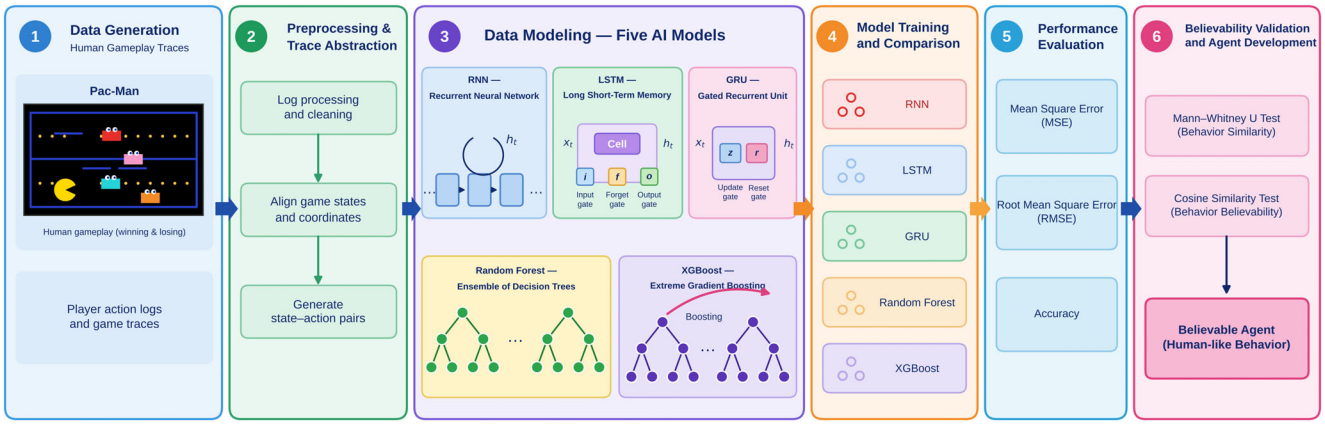


FIGURE 2 | End-to-end workflow of the proposed framework for human-like gameplay modeling and believability validation.

ALGORITHM 1 | Human Player Imitation Using Deep Learning.

Require: Current positions of Pac-Man (P_x, P_y), Ghost1 $(G1_x, G1_y)$, Ghost2 $(G2_x, G2_y)$

Ensure: Predicted next position of Pac-Man (P'_x, P'_y)

Step 1: Data Acquisition

Capture real-time gameplay data from human player interactions of winning and losing game patterns.

Step 2: Preprocessing

Normalize and format data as input-output pairs:

$$[P_x, P_y, G1_x, G1_y, G2_x, G2_y] \rightarrow [P'_x, P'_y]$$

Step 3: Dataset Split

Divide data into training (70%) and testing (30%).

Step 4: Model Selection

Choose models: RNN, LSTM, GRU, Random Forest, XGBoost.

Step 5: Model Training

Train models using training data and backpropagation/tree boosting.

Step 6: Hyperparameter Tuning

Optimize parameters using grid search or cross-validation.

Step 7: Performance Evaluation

Test on unseen data using MSE, RMSE, Accuracy, Mann-Whitney U Test, Cosine Similarity.

Step 8: Model Selection

Choose model with best performance (lowest error, highest realism).

Step 9: Deployment

Deploy model to predict Pac-Man's next move (P'_x, P'_y).

The proposed model, learning techniques, and experimental setting are covered in detail in this section.

Algorithm 1 depicts the whole process, including data acquisition, abstraction, modeling, and training, and Figure 2 summarizes the end-to-end workflow. This research uses two fundamental frameworks to model human behavior:

- Deep Learning: RNN, LSTM, and GRU are examples of DL techniques that are well-known for their capacity to recognize sequential decision-making patterns and temporal connections.
- Machine learning: RF and XGBoost are two techniques that were selected because of their interpretability and potent results on structured data.

The human-generated game traces from two case studies, Pac-Man and PUBG, were used to train and evaluate these models, with Pac-Man serving as the primary case study for controlled testing in this section.

3.1 | Stage 1: Data Generation From Human Gameplay

To accurately imitate human behavior, authentic gaming data is necessary. In the Pac-Man environment (UC Berkeley's CS188 AI course project [67], http://ai.berkeley.edu/projectoverview.html), a specially designed game-logging module was put into place using Python 2.7. The game was modified to log player and ghost status data without using third-party tools. Pac-Man and two ghosts are agents in a deterministic setting with predefined game dynamics. In order to gather event logs that reflect agent placements and state changes over time, the game engine was modified. A raw case basis was created using these logs, which were then converted into abstract behavioral traces for training.

This study used a large amount of game data including both winning and losing situations in order to accurately examine player behavior patterns. A total of 12,713 behavioral recordings from various game outcomes were collected and analyzed. This dataset serves as the foundation for our experimental architecture, enabling the development of prediction models with two crucial components:

i. Outcome Analysis: finding certain in-game behaviors that are associated with successful (winning) vs. unsuccessful (losing) results;

ii. Movement Prediction: predicting the path and choices made by the player while they are playing.

By processing this behavioral data, the model architecture creates measurable connections between player actions and the

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

outcomes of their following games, offering important insights into the best ways to play the game.

3.1.1 | Case Study: Pac-Man Simulation

This study uses a simulation environment based on a game, specifically a version of the Pac-Man project originally developed for the CS188 AI introductory course at UC Berkeley, and is available at http://ai.berkeley.edu/projectoverview.html. The game includes one level, where the main goal of the Pac-Man agent is to collect as many pills as possible and finish the level in the shortest time possible. The general layout of the Pac-Man game interface is seen in Figure 3.

3.1.2 | Environment Representation

Using the concept of a board separated into tiles, the Pac-Man project's maze design (Figure 4) was created to better understand the behavior of the Pac-Man ghost. Figure 5 shows the absolute coordinates of hurdles and valid movements. An 8 by 8 pixel square region on a screen is called a "tile." This gives us an 18×9 tile size overall. The gray shade tiles are inaccessible to Pac-Man and the ghosts, as seen in Figure 5. The mirrored tiles indicate that a ghost snatched the Pac-Man while occupying the same tile.

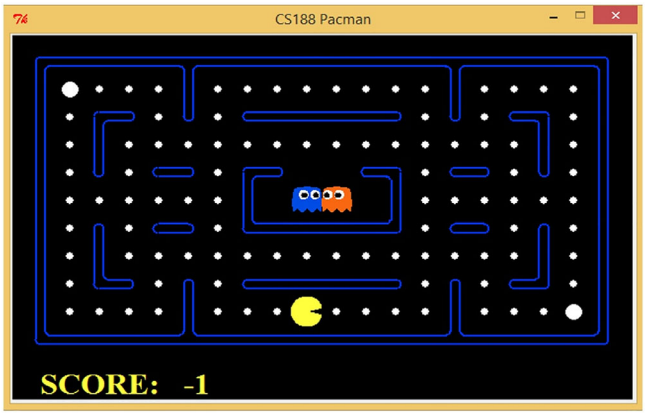


FIGURE 3 | Pacman game standard (http://ai.berkeley.edu/project_overview.html).

The search algorithms used to create and manage the actions of the game's opponents (Ghosts) include depth-first search (DFS), breadth-first search (BFS), uniform cost search (UCS), and A*.

3.2 | Stage 2: Abstraction of Human Traces

Raw gameplay logs were preprocessed into abstract traces to capture essential decision-making cues. An abstract trace R is defined as a sequence of entries. Preprocessing of raw gaming logs into abstract traces was done in order to capture key clues for making decisions. A series of entries is called an abstract trace R:

$$X_t = \left\{\text{PacMan}(x_c, y_c),\ \ \text{Ghost1}(x_1, y_1),\ \ \text{Ghost2}(x_2, y_2)\right\} \quad (1)$$

$$Y_t = \text{PacMan next move} = \left(x_{c+1},\ \ y_{c+1}\right) \quad (2)$$

where X_t is the game state at time t, consisting of the spatial coordinates of the agents, and Y_t is the target action, that is, the next move made by the human player.

The training dataset for DL/ML models is made up of these abstracted traces. The mapping of raw logs into abstract behavioral sequences to assist in localizing player decision points, as shown in Figure 5.

The designed agents are trained on six inputs (Px_c, Py_c, Gx_1, Gy_1, Gx_2, Gy_2), that is, Pac-Man's current location and the positions of two nearby ghosts on the grid, while Pac-Man's next move (Pxc_1, Pyc_2) is treated as the output used for the learning purpose and is shown in Table 1. Originally, the actual gameplay data was stored in two distinct log files: one that tracked the locations of both ghosts and another that recorded Pac-Man's timestamped coordinates. During the preparation phase, we merged winning game files into unified game state records by eliminating superfluous logs and system artifacts, and did the same with the lost games data. With the use of supervised learning techniques, this organized dataset allows thorough examination of movement patterns, result prediction, and strategic decision-making. By preserving exact coordinate alignment and eliminating missing frames, the preprocessing pipeline guarantees data integrity and offers a strong basis for behavioral modeling and agent training.

9	(9,1)	(9,2)	(9,3)	(9,4)	(9,5)	(9,6)	(9,7)	(9,8)	(9,9)	(9,10)	(9,11)	(9,12)	(9,13)	(9,14)	(9,15)	(9,16)	(9,17)
8	(8,1)	(8,2)	(8,3)	(8,4)	(8,5)	(8,6)	(8,7)	(8,8)	(8,9)	(8,10)	(8,11)	(8,12)	(8,13)	(8,14)	(8,15)	(8,16)	(8,17)
7	(7,1)	(7,2)	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)	(7,8)	(7,9)	(7,10)	(7,11)	(7,12)	(7,13)	(7,14)	(7,15)	(7,16)	(7,17)
6	(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)	(6,8)	(6,9)	(6,10)	(6,11)	(6,12)	(6,13)	(6,14)	(6,15)	(6,16)	(6,17)
5	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)	(5,8)	(5,9)	(5,10)	(5,11)	(5,12)	(5,13)	(5,14)	(5,15)	(5,16)	(5,17)
4	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)	(4,8)	(4,9)	(4,10)	(4,11)	(4,12)	(4,13)	(4,14)	(4,15)	(4,16)	(4,17)
3	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)	(3,8)	(3,9)	(3,10)	(3,11)	(3,12)	(3,13)	(3,14)	(3,15)	(3,16)	(3,17)
2	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)	(2,8)	(2,9)	(2,10)	(2,11)	(2,12)	(2,13)	(2,14)	(2,15)	(2,16)	(2,17)
1	(1,1)	(2,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	(1,8)	(1,9)	(1,10)	(1,11)	(1,12)	(1,13)	(1,14)	(1,15)	(1,16)	(1,17)
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

FIGURE 4 | (X, Y) coordinates of hurdles and valid movements.

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

9	9.055385	9.219544	9.486833	9.848858	10.29563	10.81665	11.40175	12.04159	12.72792	13.45362	14.21267	15	15.81139	16.64332	17.49286	18.35756	19.23538
8	8.062258	8.246211	8.544004	8.944272	9.433981	10	10.63015	11.31371	12.04159	12.80625	13.60147	14.42221	15.26434	16.12452	17	17.88854	18.78829
7	7.071068	7.28011	7.615773	8.062258	8.602325	9.219544	9.899495	10.63015	11.40175	12.20656	13.0384	13.89244	14.76482	15.65248	16.55295	17.46425	18.38478
6	6.082763	6.324555	6.708204	7.211103	7.81025	8.485281	9.219544	10	10.81665	11.6619	12.52996	13.41641	14.31782	15.23155	16.15549	17.08801	18.02776
5	5.09902	5.385165	5.830952	6.403124	7.071068	7.81025	8.602325	9.433981	10.29563	11.18034	12.08305	13	13.92839	14.86607	15.81139	16.76305	17.72005
4	4.123106	4.472136	5	5.656854	6.403124	7.211103	8.062258	8.944272	9.848858	10.77033	11.7047	12.64911	13.60147	14.56022	15.52417	16.49242	17.46425
3	3.162278	3.605551	4.242641	5	5.830952	6.708204	7.615773	8.544004	9.486833	10.44031	11.40175	12.36932	13.34166	14.31782	15.29706	16.27882	17.26268
2	2.236068	2.828427	3.605551	4.472136	5.385165	6.324555	7.28011	8.246211	9.219544	10.19804	11.18034	12.16553	13.15295	14.14214	15.13275	16.12452	17.11724
1	1.414214	2.236068	3.162278	4.123106	5.09902	6.082763	7.071068	8.062258	9.055385	10.04988	11.04536	12.04159	13.0384	14.03567	15.0333	16.03122	17.02939
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

FIGURE 5 | Absolute coordinates of hurdles and valid movements.

TABLE 1 | Log data generated by Pac-Man and two ghosts.

Pac-Man current position		Ghost 1 position		Ghost 2 position		Pac-Man next position	
Px_c	Py_c	Gx_1	Gy_1	Gx_2	Gy_2	Pxc_1	Pyc_2
9	1	8	5	11	5	10	1
10	1	9	5	10	5	11	1
11	1	9	6	9	5	12	1
12	1	10	6	8	5	13	1
13	1	10	5	9	5	13	1

3.3 | Training/Model Building

We split the final dataset into a 70/30 percentage for training and testing the models. For experiments, RNN, GRU, LSTM, RF, and XGBoost models are used.

3.3.1 | Recurrent Neural Network

Sequential information is used by recurrent neural networks. Every input is considered independent in a traditional neural network, this is not the case for all actions or problems. When the output at time t depends on earlier inputs, RNNs may simulate sequential data. The hidden state is computed recursively as:

$$h_t = f\left(h_{t-1}, x_t; \theta\right) \tag{3}$$

where x_t is the input at time t, h_t is the hidden state and θ denotes the trainable parameters. This makes it possible for the network to preserve a kind of memory, which is necessary for recording dynamic gaming decision-making processes [68].

3.3.2 | Long Short-Term Memory

An improved form of RNNs called LSTMs is made to recognize long-term dependencies [69]. Information flow is controlled by the gating methods (input, forget, and output gates): LSTM cell operations are defined by the input, forget, and output gates. It also includes candidate's memory, cell state update, and hidden state update.

3.3.3 | Gated Recurrent Unit

GRU is a type of RNN designed to handle sequential data. It uses gating mechanisms, reset and update gates, to control information flow, enabling efficient learning of long-term dependencies while reducing computational complexity compared to traditional RNNs and LSTMs. It reduces training time and computational load while maintaining performance. The GRU is defined by the update gate, the reset gate, the candidate activation and the final activation:

$$\text{Update gate}: \quad z_t = \sigma\left(W_z \cdot \left[h_{t-1}, x_t\right]\right) \tag{4}$$

$$\text{Reset gate}: \quad r_t = \sigma\left(W_r \cdot \left[h_{t-1}, x_t\right]\right) \tag{5}$$

$$\text{Candidate activation}: \quad \tilde{h}_t = \tanh\left(W \cdot \left[r_t \odot h_{t-1}, x_t\right]\right) \tag{6}$$

$$\text{Final activation}: \quad h_t = \left(1 - z_t\right) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{7}$$

where σ is the sigmoid function and $\odot$ denotes element-wise multiplication. GRUs were especially helpful when there were shorter sequences and a need for faster convergence [70].

3.3.4 | Random Forest

An ensemble learning approach called RF constructs multiple decision trees during training and returns the mean prediction for regression or the majority vote for classification. It is appropriate for structured behavioral trace data due to its great interpretability and resilience to overfitting [71]. Let $h_1(x), \ldots, h_T(x)$ be the

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

individual decision trees, where x is the input feature vector and T is the number of trees. For regression the prediction is:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} h_t(x) \tag{8}$$

and for classification:

$$\hat{y} = \text{mode}\{h_1(x), \ldots, h_T(x)\} \tag{9}$$

3.3.5 | Extreme Gradient Boosting

A scalable, regularized gradient boosting system called XGBoost creates trees one after the other to fix the residuals of earlier forecasts. XGBoost was employed as a baseline machine learning model because of its excellent efficiency and predictive ability [72]. The objective at iteration t is:

$$\mathcal{L}^{(t)} = \sum_i \left[g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i) \right] + \gamma T + \frac{1}{2}\lambda \parallel w \parallel^2 \tag{10}$$

where g_i and h_i are the first and second derivatives of the loss, T is the number of leaves, w are the leaf weights and γ, λ are regularization parameters. The optimal leaf weight of leaf j and the split gain are:

$$w_j^* = -\frac{G_j}{H_j + \lambda}, \qquad G_j = \sum_{i \in I_j} g_i, \qquad H_j = \sum_{i \in I_j} h_i \tag{11}$$

$$\text{Gain} = \frac{1}{2}\left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma \tag{12}$$

3.4 | Stage 4: Performance Measures

In this study, we calculate mean square error (MSE), RMSE, and accuracy. Details are discussed in the subsequent sections.

3.4.1 | Mean Squared Error

The error is measured as the MSE between the real performance and the desired performance. The MSE between observed and predicted output was computed for each run, to compare models and select the better one. The models were trained to make future behavioral predictions, using the metrics described above We saw the error (mean square error) for each test to compare the results of these models and to select the best of them. The models have been built with a number of different instances and separate neurons. The discrepancy between the observed and estimated value is the MSE:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{13}$$

where y_i is the observed value, $\hat{y}_i$ the estimated value and n the number of observations. MSE was used to select the best-performing architecture and training configuration.

3.4.2 | Root Mean Squared Error

The residuals are the gap between the actual values and the predicted values. The square of residuals, the average number of squares, and the square root give us the root mean squared (RMS) error:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (\hat{y}_i - y_i)^2} \tag{14}$$

where y_i is the observed value for the i-th observation and $\hat{y}_i$ is the predicted value.

3.5 | Evaluating Playing Style Identification

To check how believable the AI agents are, their accuracy is tested using statistics. The agent's behavior is compared to the real player's by looking at their game actions. Instead of asking humans to judge, two automatic methods are used: the Mann–Whitney U test and cosine similarity.

3.5.1 | Mann–Whitney U Test

The Mann–Whitney U test is a non-parametric statistical method, and is often used to find a significant difference between two independent groups. For example, in the case of AI agents, it can be used to find differences between human and AI players, without assuming data uniformity. It compares the distributions (medians) of two separate groups and helps understand whether two behaviors are statistically different. The null hypothesis H_0 states that the two populations are equal, and H_1 that they are not equal (two-tailed). The statistics are:

$$U_1 = n_1 n_2 + \frac{n_1(n_1+1)}{2} - R_1 \tag{15}$$

$$U_2 = n_1 n_2 + \frac{n_2(n_2+1)}{2} - R_2 \tag{16}$$

where R_1 and R_2 are the sums of ranks of Group 1 and Group 2, and n_1, n_2 are the sample sizes.

3.5.2 | Cosine Similarity Metric

Cosine similarity is a widely used metric to compare how similar two vectors are. It is simple, efficient, capable of handling high-dimensional data and less sensitive to the magnitudes of the vectors, which makes it suitable for evaluating AI agents that act in complex situations. By measuring how closely the AI agents' activities mimic those of humans, it helps assess how credible the agents are. The cosine similarity between two vectors is:

$$CS(V_h, V_c) = \frac{V_h \cdot V_c}{\parallel V_h \parallel \parallel V_c \parallel} \tag{17}$$

where $V_h = \{x_i, y_i\}$ indicates an individual human player with a set of real Pac-Man locations, and $V_c = \{x_i, y_i\}$ indicates the AI controller with a set of Pac-Man positions.

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

4 | Experiments and Results

This section includes the data collection procedure, experimental design, model evaluation findings, and comparison analysis. The Pac-Man and PUBG game datasets were used as case studies. The goal was to assess how well different DL and machine learning models performed at imitating human behavior.

4.1 | Experiment Design

In this study, the effectiveness of three DL models, RNN, LSTM, and GRU, and two ML models, RF and XGBoost, in imitating human gameplay in Pac-Man and PUBG is evaluated. The primary objective was to analyze win/loss patterns to identify the model that most closely reflects human decision-making. To compare model performance, we used a likelihood-ratio-based evaluation, concentrating on MSE, RMSE, and prediction accuracy. The goal was to choose the model with the highest accuracy and lowest error that best reflects human behavior.

4.2 | Dataset Description

A modified version of the Pac-Man game was used to gather 30 gameplay sessions (15 wins and 15 losses) from five different players to train our models. To this goal, data from several human players, particularly Pac-Man, with varying degrees of expertise, were gathered. Keep in mind that players with different levels of ability and play styles might play in a variety of ways at the same stage. As a result, imitation learning performance will differ among players to better depict winning and losing patterns, laying a solid foundation for imitation learning. Table 2 presents the summary of the dataset statistics.

TABLE 2 | Players' traces of winning and losing game play.

	Sample size	
Players	**Winning instances**	**Losing instances**
Player 1	2295	1236
Player 2	1236	825
Player 3	2531	789
Player 4	832	1143
Player 5	779	1047
Total	**7673**	**5040**

Note: The bold values shows the total instances.

Every player's model of the two patterns determines whose actions cause the player to lose the game and which move contributes to a win. It may be claimed that five participants is a very limited selection. However, every player has played 30 games, so each imitation-based model focused on trained in winning and losing game trends separately, and it is a non-trivial exercise.

4.3 | Model Prediction Evaluation

4.3.1 | RMSE Evaluation on Win/Loss Games

To measure how well models predicted player movements, the RMSE was employed. The findings show that in both the win and loss datasets, LSTM continuously produced the lowest RMSE values. LSTM substantially outperformed RNN and GRU, with average RMSE values of 0.019 for losses and 0.030 for wins, as shown in Table 3. This reveals the ability of LSTM to recognize subtle patterns in human movement behavior as well as sequential dependencies. RNN performed second-best in win games (0.078), and in loss games its RMSE was about three times that of LSTM (0.061, the mean of the five per-player values in Table 3), with a noticeably high value for Player 4 (0.128), indicating less stable generalization under different gameplay outcomes (The previously printed loss average of 0.189 did not match the per-player values and has been corrected). Even though GRU is a simpler RNN variation, it performed less consistently, with RMSE averages of 11.10 and 11.63 in win and loss scenarios, respectively. This suggests that it could be difficult to replicate the complex temporal dynamics of Pac-Man activities or, as discussed in Section 4.6, that the GRU configuration did not converge properly. The following observations are made.

- XGBoost demonstrated outstanding performance, often surpassing even DL models, with an average RMSE of 0.032 (win) and 0.024 (loss).
- With RMSE values of 0.064 (win) and 0.070 (loss), RF demonstrated competitive performance, demonstrating its strong generalization capabilities even though it is a non-sequential model.

Although LSTM is still the most reliable and accurate DL model, XGBoost shows promise as an ML candidate, particularly in settings with structured data or less complicated temporal patterns.

TABLE 3 | RMSE comparison across individual players' Win-lost games.

	Win games			Lost games		
Player	**LSTM**	**RNN**	**GRU**	**LSTM**	**RNN**	**GRU**
Player 1	0.030	0.031	10.820	0.021	0.032	13.960
Player 2	0.018	0.070	14.620	0.025	0.040	9.742
Player 3	0.062	0.066	10.000	0.020	0.032	9.615
Player 4	0.021	0.085	10.450	0.016	0.128	12.080
Player 5	0.020	0.141	9.668	0.016	0.074	12.780
Averages	**0.030**	**0.078**	**11.100**	**0.019**	**0.061**	**11.630**

Note: The bolds values show averaged values for various columns.

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 4 | MSE comparison across individual players' win-lost games.

	Win games			Lost games		
Player	**LSTM**	**RNN**	**GRU**	**LSTM**	**RNN**	**GRU**
Player 1	0.009	0.003	10.700	0.005	0.001	13.920
Player 2	0.003	0.005	14.590	0.007	0.002	9.719
Player 3	0.003	0.004	9.903	0.004	0.001	9.584
Player 4	0.004	0.007	10.430	0.003	0.016	12.040
Player 5	0.004	0.019	9.641	0.003	0.005	12.770
Averages	**0.004**	**0.007**	**11.050**	**0.004**	**0.005**	**11.600**

Note: The bolds values show averaged values for various columns.

TABLE 5 | Accuracy (%) of different models for the Win-games and Loss-games.

	Win-games					Loss-games				
Player	**LSTM**	**RNN**	**GRU**	**RF**	**XGB**	**LSTM**	**RNN**	**GRU**	**RF**	**XGB**
Player 1	99.89	99.89	60.13	89.67	94.23	98.78	98.98	97.97	91.56	95.63
Player 2	99.59	84.09	70.96	86.13	93.00	59.39	52.12	27.87	61.23	89.52
Player 3	99.60	99.60	59.76	88.12	92.41	90.38	62.82	23.07	76.13	91.00
Player 4	87.27	99.19	87.27	89.72	94.73	54.58	97.38	92.14	75.00	95.29
Player 5	96.79	59.82	36.53	80.15	91.63	74.76	70.47	99.04	82.00	94.85
Averages	**96.62**	**88.51**	**62.93**	**86.76**	**93.20**	**75.58**	**76.35**	**68.01**	**77.58**	**93.26**

Note: The bolds values show averaged values for various columns.

4.3.2 | MSE Evaluation on Win/Loss Games

To validate the RMSE results and offer further information on error size, the MSE was also calculated. The supremacy of LSTM on win datasets and the performance of RNN on loss datasets are further supported by MSE findings, as shown in Table 4. In both win and loss datasets, LSTM once again takes the lead with the lowest average MSE of 0.004, demonstrating its ability to lower prediction error in both successful and unsuccessful human games. The robustness and dependability of XGBoost in imitating human-like behaviors in structured datasets were confirmed by the fact that it had the lowest MSE among the machine learning models in both win (0.0012) and loss (0.0010) games.

4.3.3 | Accuracy Evaluation of Individual Players

The examination of accuracy for each player (Table 5) offers a thorough understanding of how effectively each model represents and predicts human gameplay behaviors in both winning and losing situations. For win datasets, LSTM showed the best average accuracy (96.62%) among the deep models, whereas in loss datasets the averages of LSTM (75.58%) and RNN (76.35%) were similar and XGBoost was clearly more stable (93.26%). LSTM performs better than others even in less ideal circumstances and exhibits great generalization across a variety of player strategies.

Overall, XGBoost has the second-highest accuracy, with 93.60% (Win Dataset) and 93.26% (Loss Dataset). With outstanding generalization and believability, XGBoost outperformed DL models in certain individual loss scenarios (such as Player 2 and Player 4). With gradient boosting and fine-grained decision limits to make up for its lack of sequence modeling capabilities like LSTM, its performance is very remarkable.

TABLE 6 | Aggregated gameplay data.

Dataset	**Instances**
Win	7,673
Loss	5,040
Combined	12,713

4.3.4 | Overall Model Accuracy

An aggregate assessment was carried out over complete gaming datasets to evaluate the generalization and prediction performance of all five models: LSTM, RNN, GRU, RF, and XGBoost. Win-game traces, loss-game traces, and the aggregate dataset of all cases were all evaluated separately. The aggregated players' gameplay data is shown in Table 6 (7673 win, 5040 loss, and 12,713 total instances).

In general, LSTM and RNN are dominant, particularly where sequential dependencies are crucial, as shown in Table 7 and Figure 6. They can adjust to both predictable (win) and unpredictable (loss) gaming due to their memory-based design. A convincing trade-off between accuracy and processing efficiency is provided by XGBoost. When there are computing limitations, its performance on all datasets, particularly the combined one, makes it a good lightweight substitute for DL. The depth and adaptability required for fine-grained behavioral imitation in complex contexts are lacking in RF, despite its stability. Despite its straightforward design, GRU continuously fared poorly on

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 7 | Accuracy (%) across all models and datasets.

Dataset	LSTM	RNN	GRU	RF	XGBoost
Pacman — Win	99.99	99.99	60.58	70.79	96.56
Pacman — Loss	99.91	99.91	64.80	81.36	94.37
Pacman — Combined	99.96	99.98	58.43	70.79	98.00

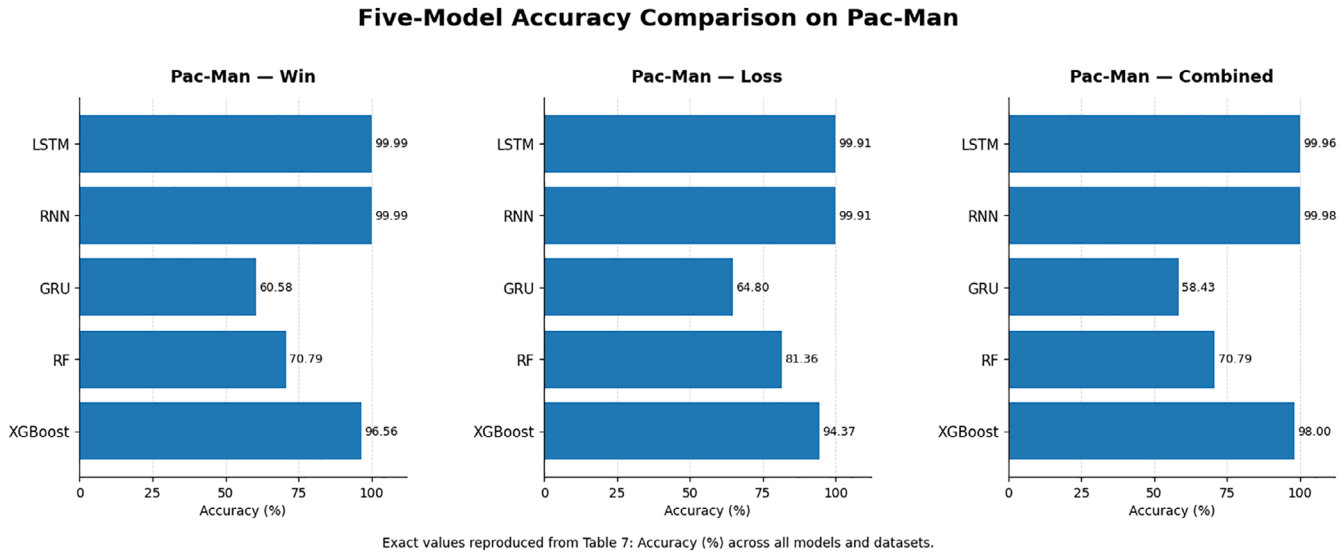


FIGURE 6 | Comparison of prediction accuracy among five machine learning models on the Pac-Man winning, losing, and combined datasets.

TABLE 8 | Mann–Whitney test *p*-values comparing each player with the AI agents.

	LSTM		RNN		GRU	
	Win	Loss	Win	Loss	Win	Loss
Players			***p***			
Player 1	0.94	1.00	0.94	0.38	0.97	0.38
Player 2	0.86	0.95	0.72	0.77	0.77	0.40
Player 3	0.91	0.65	0.97	0.83	0.92	0.82
Player 4	0.96	0.95	0.96	0.75	0.64	0.96
Player 5	1.00	0.93	0.83	0.93	0.65	0.54
Averages	**0.93**	**0.89**	**0.88**	**0.73**	**0.79**	**0.62**

Note: The bold values show averaged values.

all datasets, which makes it less appropriate for imitation tasks requiring intricate decision-making.

4.4 | Statistical Validation of Believability of Playing Style

We used automatic methods rather than human judgment to determine the believability of the acquired AI controllers in an arcade Pac-Man game. Since human evaluations are subjective, they are always inexact and require time since multiple people are involved in evaluating agents. However, the number of individual testers needed for acceptable evaluation results is a major challenge [10, 32]. To address this, Mann–Whitney U and Cosine methods were utilized as statistical tools. The ultimate judge of the resulting quality of game character believability remains human perception [32].

4.4.1 | Mann–Whitney U Test

This experiment was conducted to analyze the resemblance of AI controllers with human players by comparing human player game logs with those of the three best-performing AI agents. The null hypothesis (H_0) postulates that the distributions of human and AI-generated actions do not differ significantly, indicating that the AI has been successful in simulating human behavior. It is generally accepted that two distributions are statistically comparable when the *p*-value is large (usually > 0.05).

The results of applying this test to winning and losing playing style patterns are shown in Table 8 and Figure 7. The test setup used input samples of action traces of human players and model-generated predictions (position or movement vectors), the Mann–Whitney U test, and a significance level of $\alpha = 0.05$. The

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

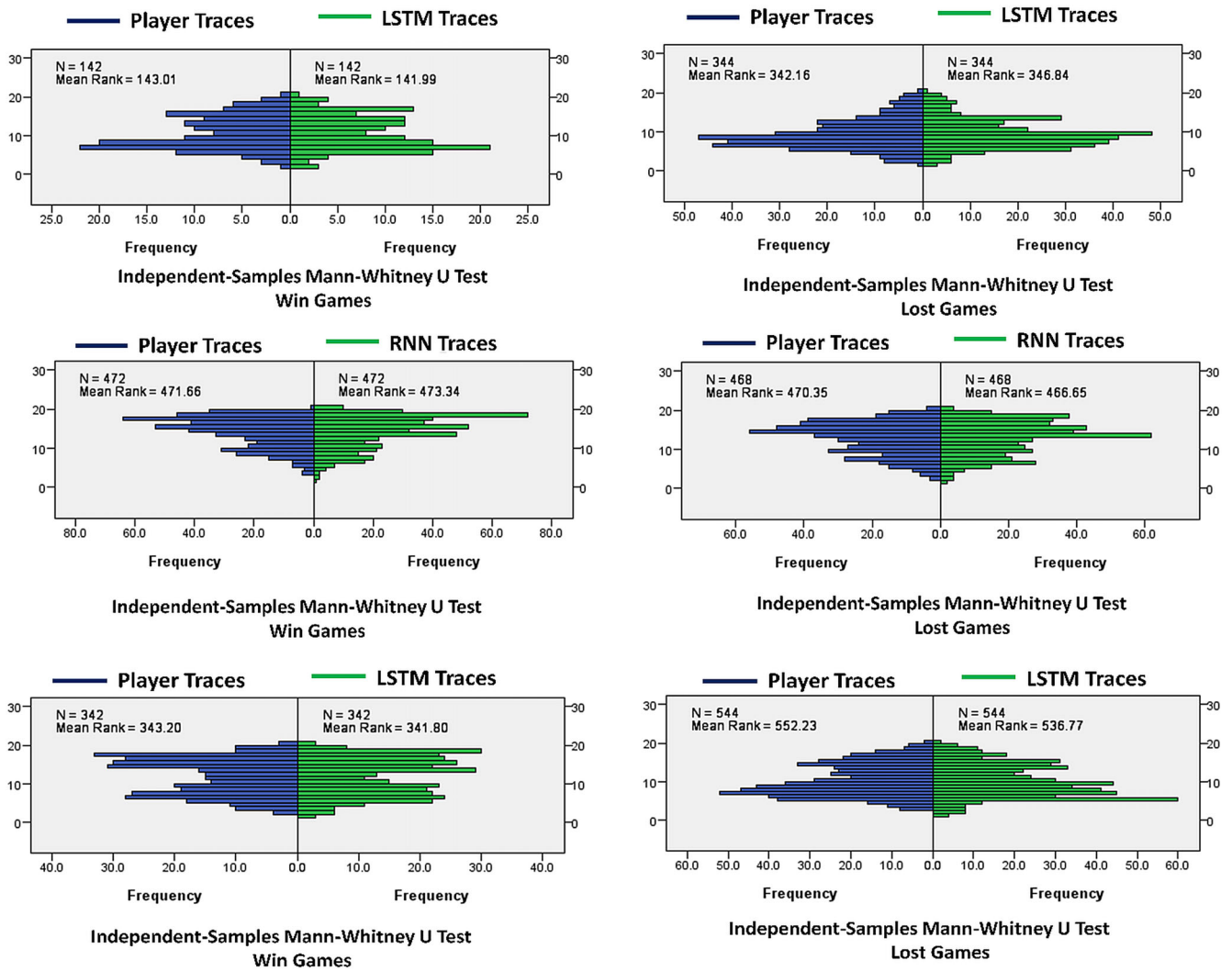


FIGURE 7 | Graphs comparing results from the independent-sample Mann–Whitney U test show how well RNN, LSTM, and GRU models mimic human players based on their behavior traces.

best average *p*-values were obtained by LSTM, suggesting that there is no discernible difference between the actions it created and actual human behaviors. RNN scored admirably, with average *p*-values over 0.88 in win games, confirming its ability to replicate games even if it was somewhat less effective than LSTM. GRU demonstrated a moderate degree of human action alignment. Their *p*-values were comparatively lower, particularly in loss-game scenarios, although they were still above the essential threshold (0.05).

4.4.2 | Cosine Similarity Analysis

Cosine similarity was used to assess how well the AI agents imitated the behaviors of human players. Cosine similarity, in contrast to conventional error-based metrics, calculates the angular difference between two vectors, giving information on the direction and orientation of actions rather than just their magnitude. Because of this, it is especially useful in gaming settings where realistic imitation depends on movement patterns and action direction. Cosine similarity values lie in $[-1, 1]$: -1 means opposite, 1 means identical, while 0 indicates no relation. Table 9 demonstrates the playing-style similarity computed by using a cosine-similarity value for vectors of actual playing behavior and agents.

All three recurrent models reproduced the direction of human movement with very high and nearly identical similarity: the averages over the five players are 0.978 (win) and 0.977 (loss) for LSTM, 0.983 (win) and 0.980 (loss) for GRU, and 0.978 (win) and 0.980 (loss) for RNN. The differences between the models are in the third decimal place and therefore should not be interpreted as a meaningful ranking (In the earlier version, the RNN win average was printed as 0.997, which does not match the per-player values of the same table and has been corrected). This shows that even though GRU struggles with precise prediction accuracy, it captures the general action structure. Despite not being intended for temporal modeling, XGBoost and RF also managed to obtain high cosine similarity scores, with XGBoost (0.959) indicating excellent alignment with human action flow in both win and loss scenarios and RF (0.9375) showing acceptable directional coherence. This supports the believability of the proposed imitation learning framework, which shows that the agents' actions were

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 9 | Cosine similarity between each AI agent and the corresponding human player.

	LSTM		GRU		RNN	
Players	**Win**	**Loss**	**Win**	**Loss**	**Win**	**Loss**
Player 1	0.973	0.984	0.988	0.990	0.970	0.983
Player 2	0.983	0.974	0.986	0.978	0.983	0.972
Player 3	0.973	0.985	0.988	0.985	0.972	0.985
Player 4	0.975	0.968	0.974	0.966	0.982	0.996
Player 5	0.984	0.975	0.977	0.981	0.982	0.966
Average	**0.978**	**0.977**	**0.982**	**0.980**	**0.978**	**0.980**

Note: The bold values show averaged values.

TABLE 10 | Results of PUBG games dataset.

PUBG	GRU	LSTM	RNN	RF	XGBoost
Root mean square error	0.1608	0.1594	0.1661	0.87	0.01
Mean square error	0.0263	0.0260	0.0270	0.03	0.01
Accuracy (%)	97.157	97.143	97.146	97.17	97.37

directionally and statistically comparable to those of human players (as determined by the Mann–Whitney U test).

4.5 | Cross-Game Validation Using PUBG

To validate the model, we apply it to another game dataset named PUBG. The trained models were tested using gameplay data from PUBG, a contemporary, intricate multiplayer game, to determine the ability to generalize of the proposed imitation learning framework regardless of the strictly controlled Pac-Man context. PUBG is a perfect benchmark for confirming the scalability and resilience of the framework since it has high-dimensional state spaces, real-time multiplayer interactions, and strategic variety, in contrast to Pac-Man, which runs in a deterministic and simplified grid setting. To investigate cross-domain adaptation, the same models employed in the Pac-Man experiments, LSTM, RNN, GRU, RF, and XGBoost, were applied to this dataset with no significant architectural modifications. Table 10 shows results using the PUBG game dataset.

XGBoost surpassed all models in terms of RMSE, MSE, and accuracy, with the lowest RMSE of 0.0100 and best accuracy of 97.37%. This demonstrates the ability to predict highly ordered spatial activity in large-scale systems. LSTM and GRU, despite their power in sequence modeling, had slightly larger errors (RMSE ~ 0.16). This might be attributed to the increased noise and unpredictability in PUBG's dynamic game setting, where judgments are impacted by a variety of real-time elements not found in simpler situations. RNN, like LSTM, performed well (97.15% accuracy), but suffered significantly more with generalization than in Pac-Man. The following key insights are observed.

- The results from PUBG demonstrate that the suggested paradigm could be applied to more complicated, real-world gaming situations.
- The most scalable model was found to be XGBoost, which demonstrated exceptional flexibility in handling a wide range of data distributions while preserving high accuracy and low prediction error.
- While DL models, especially LSTM and GRU, are still useful for temporal modeling, they might need more context information (such as tactical status or environmental metadata) to function better in high-variability games like PUBG.

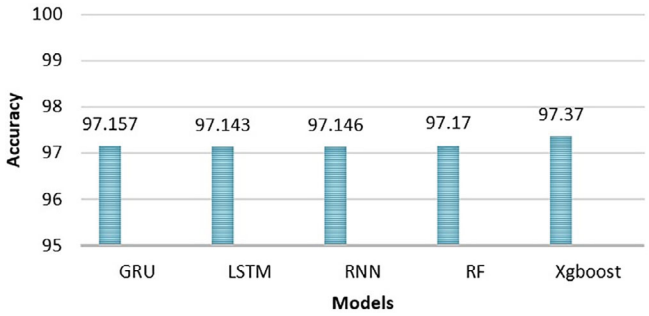


FIGURE 8 | Accuracy graph for PUBG game.

Table 10 shows that all models provide good accuracy on PUBG, with the deep models giving RMSE of about 0.16. XGBoost gives 97.37% accuracy with a minimum RMS error of 0.01 and mean square error of 0.01. The accuracy graph for all models on the PUBG game dataset is shown in Figure 8.

4.6 | Discussion

4.6.1 | Why Does the RNN Achieve Such High Performance?

i. What the results actually show?
   - Pac-Man, pooled data (Table 7). RNN reaches 99.99% (win), 99.91% (loss) and 99.98% (combined), which is practically identical to LSTM (99.99%, 99.91%, 99.96%) and far above XGBoost (96.56%, 94.37%, 98.00%), RF

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

(70.79%, 81.36%, 70.79%) and GRU (60.58%, 64.80%, 58.43%). The RNN advantage is therefore large over GRU, RF and, to a smaller extent, XGBoost, but it is not an advantage over LSTM.
- Pac-Man, per player (Table 5). The picture is more nuanced: the RNN averages 88.51% (win) and 76.35% (loss), with a wide range across players (from 52.12% to 99.89%), while XGBoost is steadier (93.20% and 93.26%). RNN accuracy rises sharply when all players' data are pooled.
- Believability (Tables 8 and 9). The Mann–Whitney *p*-values of all models are above 0.05, with LSTM highest on average (0.93/0.89) and RNN at 0.88/0.73. Cosine similarities of LSTM, GRU and RNN are all about 0.98, so no model can be ranked on this measure.
- PUBG (Table 10). The RNN is not the best model there: its RMSE (0.1661) is the highest among the three deep models, and XGBoost is best.

ii. Reasons for the strong Pac-Man performance of the RNN
- The task is short-horizon and nearly Markovian: The target is Pac-Man's next tile given the current positions of Pac-Man and two ghosts on a small, deterministic 18 × 9 grid. In the logged data (Table 1) Pac-Man moves at most one tile per step, so the current position already determines most of the next position. A simple recurrent network is sufficient to represent such short dependencies. The main weakness of the plain RNN, the decay of the error signal over long time lags, was the motivation for the LSTM [69] and matters mainly when long-range memory is needed; in this task it is rarely exercised. LSTM and RNN therefore end up with almost the same accuracy, which is what this explanation predicts.
- Model capacity matches data size: The dataset contains 12,713 instances from five players (Table 6), and per-player and per-outcome models are trained on only 779 to 2531 instances (Table 2). A plain RNN has fewer trainable parameters than an LSTM (one recurrent weight block instead of four) or a GRU (three), which lowers the risk of over-fitting on small datasets. This is consistent with recent evidence that lighter recurrent cells with fewer parameters can remain competitive with heavier ones [52], and that the GRU/LSTM choice involves an efficiency–accuracy trade-off rather than a uniform winner [53]. It is also consistent with our own observation that RNN accuracy is much higher on the pooled data than on single-player data (Table 5 versus Table 7).
- Sequential structure is preserved, which the tree models cannot exploit: RF and XGBoost treat the six coordinates as independent tabular features. Benchmarks on tabular data show that tree ensembles are often very strong competitors to deep networks [50], and XGBoost indeed performs well here (94%–98%), but in Pac-Man the information about movement direction lies in the ordering of successive positions, which the recurrent models receive and the tree models do not. This helps explain why RNN and LSTM exceed XGBoost on Pac-Man, whereas in PUBG, where the data are noisier and less temporally clean, the ranking is reversed.
- The low GRU accuracy is probably a configuration issue, not an inherent weakness: GRU RMSE values of 9.6 to 14.6 (Table 3) are two to three orders of magnitude larger than those of the other deep models, while the literature reports GRU and LSTM to be comparable on shorter sequences [52, 53]. An error of this size suggests that the GRU runs did not converge (e.g., because of the learning rate, number of epochs, input scaling or initialization). The comparison "RNN versus GRU" should therefore not be read as evidence that a plain RNN is architecturally superior to a GRU; a re-tuned GRU is included in our future work.

iii. Caveats that limit the interpretation
- Possible near-duplicate leakage. The 70/30 split was applied to consecutive game frames. Because successive states differ by at most one tile, randomly assigned frames of the same game can be almost identical, which inflates accuracy of flexible models (accuracy above 99.9% should be read with this in mind). A game-wise or player-wise hold-out split would provide a stricter test.
- No significance test between RNN and LSTM. The 0.02-point accuracy difference (99.98 vs. 99.96) is far below what can be attributed to the architecture without repeated runs with different seeds and a paired test.
- Small number of players and one low-dimensional game. Generalization of the RNN result beyond Pac-Man is not established; the PUBG results point in the opposite direction.

4.7 | Advantages and Disadvantages of the Best Model (RNN)

Why the RNN is considered the best model here. On the primary case study (Pac-Man) the RNN obtained the highest pooled accuracy (99.98%, Table 7), a mean Mann–Whitney *p*-value above 0.05 in both outcomes (Table 8), and cosine similarity of about 0.98 (Table 9). It is tied with LSTM on accuracy and is the lighter of the two models. This designation applies to Pac-Man only; on PUBG, XGBoost is the best model (Table 10). Table 11 summarizes the advantages and disadvantages together with the justification for each.

Overall, the RNN is a justified choice when the state space is low-dimensional, the dynamics are deterministic and the prediction horizon is short, as in Pac-Man, and when computational cost matters. It is not a universal solution: for noisy, high-variability settings such as PUBG a tree ensemble (XGBoost) was more accurate, and for tasks that require long memory a gated model such as LSTM is the safer choice. We therefore recommend selecting the imitation model according to the environment, and, in future work, testing hybrid designs that combine recurrent state with gradient-boosted decision heads.

4.8 | Future Research Directions

These positive findings suggest a number of future research projects to broaden the scope and depth of the proposed imitation learning framework.

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

TABLE 11 | Advantages and disadvantages of the RNN as the best model on Pac-Man, with justification.

	Point	Justification (evidence)
Advantage	Very high accuracy and statistically human-like output	99.98% pooled accuracy (Table 7); Mann–Whitney $p > 0.05$ for all players (Table 8); cosine similarity ≈0.98 (Table 9).
Advantage	Light-weight and fast	Fewer parameters than LSTM and GRU; suitable when most of a game's computing budget is reserved for rendering, as emphasized by recent work on compute-efficient human-like bots [62], and consistent with the efficiency–accuracy trade-off reported for gated cells [53].
Advantage	Models temporal order with a simple design	Hidden-state recurrence (Equation 3) captures the sequence of positions that tree models ignore; simple to implement, train and deploy.
Advantage	Competitive with more complex models on this task	Matches LSTM (99.98 vs. 99.96) and exceeds GRU, RF and XGBoost on pooled Pac-Man data (Table 7).
Disadvantage	Short memory	Plain RNNs suffer from decaying error signals over long lags [69]; they are unsuitable for long-horizon tactics such as those of PUBG.
Disadvantage	Unstable across players and outcomes when data are small	Per-player accuracy ranges from 52.12% to 99.89% (Table 5) and loss-game RMSE is about three times that of LSTM (0.061 vs. 0.019, Table 3); XGBoost is much steadier (93.2%–93.3%).
Disadvantage	Weaker in noisy, high-variability games	On PUBG the RNN has the highest RMSE of the deep models (0.1661) and is overtaken by XGBoost (Table 10).
Disadvantage	Low interpretability	The hidden state is a black box, whereas tree ensembles offer feature-based explanations; relevant for design and debugging of NPC behavior.
Disadvantage	Results may be optimistic	Random frame-level splits can leak near-duplicate states into testing (Section 4.6 (c)); evaluation on unseen games and players is still required.

i. Agents would be able to dynamically modify their behavior in reaction to changing player strategies or in-game circumstances by combining real-time learning and adaptability using online learning or reinforcement learning techniques. This ability is necessary to increase agent response and realism in uncertain situations.

ii. The agent's situational knowledge and decision-making accuracy may be greatly enhanced by integrating context-enriched modeling by embedding higher-level semantic information, such as map awareness, goal orientation, team coordination, or environmental dynamics.

iii. The architecture may also be expanded to provide co-learning and human-agent interaction, in which AI can cooperate or compete with human participants in addition to being able to imitate.

iv. Creating agents that adjust to the distinct behavioral patterns, preferences, or cognitive types of particular users might significantly boost immersion by enabling player-specific customization, particularly in serious gaming, education, or therapy.

v. Methodological extensions: game-wise and player-wise hold-out evaluation, repeated runs with multiple random seeds and paired significance tests between RNN and LSTM, a re-tuned GRU baseline, and complementary human evaluation of believability.

5 | Conclusion

In order to create convincing, human-like entities in virtual gaming settings, this study developed and verified a thorough imitation learning framework. By using deep learning and conventional machine learning techniques on human gaming traces, the main goal was to close the gap between mechanical artificial intelligence (AI) behaviors and natural human decision-making patterns. The framework used models like recurrent neural networks (RNN), long short-term memory (LSTM), gated recurrent units (GRU), random forest (RF), and extreme gradient boosting (XGBoost). These models were trained on rich behavioral data gathered from a modified version of Pac-Man and then verified on the intricate modern game PlayerUnknown's Battlegrounds (PUBG). A structured pipeline comprised the following steps: generating data through human trace logging, transforming it into abstract state-action representations, training and testing the model, and assessing it using statistical validation techniques (Mann–Whitney U Test and Cosine similarity) and quantitative performance metrics such as RMSE, MSE, and accuracy. RNN and LSTM were the most successful models for the structured and predictable Pac-Man setting, with pooled accuracy of 99.98% and 99.96%, respectively, and the lowest errors were obtained by LSTM. The strong RNN result is attributed mainly to the short-horizon, nearly Markovian structure of the task and to the small data size; it comes with the limitations listed in Sections 4.6 and 4.7. XGBoost outperformed all other models in the PUBG case study, exhibiting higher generalization and cross-environment adaptability. Its ability to handle high-dimensional, nonlinear, and noisy situations is demonstrated by its achievement of the greatest accuracy (97.37%) and lowest error (RMSE = 0.010). The Mann–Whitney U Test gave p-values above 0.05 for all recurrent models (highest for LSTM), and cosine similarity of the recurrent models was about 0.98, supporting that the generated behaviors are statistically comparable to human behavior.

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Despite being computationally economical, GRU was the least consistent across all datasets and had trouble generalizing in both complex (PUBG) and structured (Pac-Man) contexts, especially in loss-game circumstances; its very large RMSE suggests that its configuration should be re-tuned before firm conclusions are drawn. These results confirm that no single method consistently performs better than others in all gameplay conditions and that model efficacy is largely context-dependent. While gradient boosting models like XGBoost provide outstanding performance in dynamic and high-noise scenarios, deep recurrent models perform best in structured and sequential tasks.

Author Contributions

N.A.B. conceptualization, data curation, writing manuscript. K.S. formal analysis, conceptualization, writing manuscript. M.M.A. supervision, methodology, formal analysis. J.B.B. funding acquisition, methodology, investigation. C.L.R.V. investigation, software, project administration. N.A.S. software, visualization, data curation. I.A. supervision, validation, writing and editing manuscript. All authors reviewed the manuscript.

Funding

This study is funded by the European University of Atlantic and Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2026R746), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Ethics Statement

The authors have nothing to report.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. F. Kuijk, S. Van Broeck, C. Dareau, et al., "A Framework for Human-Like Behavior in an Immersive Virtual World," in *2013 18th International Conference on Digital Signal Processing (DSP)* (IEEE, 2013), 1–7.

2. J. Zhang, Y. Li, X. Wu, and Q. Zhao, "Human-Like Behavior Imitation for Gaming AI Using Generative Adversarial Networks," (2024), arXiv Preprint arXiv:2404.07656.

3. J. R. J. Neo, A. S. Won, and M. M. Shepley, "Designing Immersive Virtual Environments for Human Behavior Research," *Frontiers in Virtual Reality* 2 (2021): 603750, https://doi.org/10.3389/frvir.2021.603750.

4. S. T. Jesso, W. G. Kennedy, and E. Wiese, "Behavioral Cues of Humanness in Complex Environments: How People Engage With Human and Artificially Intelligent Agents in a Multiplayer Videogame," *Frontiers in Robotics and AI* 7 (2020): 531805, https://doi.org/10.3389/frobt.2020.531805.

5. M. Świechowski and D. Ślęzak, "The Many Challenges of Human-Like Agents in Virtual Game Environments," in *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025)* (IFAAMAS, 2025), 1996–2005, arXiv:2505.20011.

6. F. Thomas, *The Illusion of Life* (Disney Publishers, 1995).

7. J. Bates, *The Nature of Characters in Interactive Worlds and the Oz Project* (School of Computer Science, Carnegie Mellon University, 1992).

8. F. Tencé, C. Buche, P. De Loor, and O. Marc, "The Challenge of Believability in Video Games: Definitions, Agents Models and Imitation Learning," (2010), arXiv Preprint arXiv:1009.0451.

9. Y. A. Sekhavat, "Behavior Trees for Computer Games," *International Journal on Artificial Intelligence Tools* 26, no. 2 (2017): 1730001, https://doi.org/10.1142/S0218213017300010.

10. P. Hingston, *Believable Bots: Can Computers Play Like People?* (Springer, 2012), https://doi.org/10.1007/978-3-642-32323-2.

11. A. Gill, "Introduction to the Theory of Finite-State Machines," (1962).

12. J. Gratch and S. Marsella, "Evaluating a Computational Model of Emotion," *Autonomous Agents and Multi-Agent Systems* 11, no. 1 (2005): 23–43, https://doi.org/10.1007/s10458-005-1081-1.

13. D. Melhart, G. N. Yannakakis, and A. Liapis, "I Feel i Feel You: A Theory of Mind Experiment in Games," *KI-Künstliche Intelligenz* 34, no. 1 (2020): 45–55, https://doi.org/10.1007/s13218-020-00641-2.

14. S. Devlin, A. Anspoka, N. Sephton, P. I. Cowling, and J. Rollason, "Combining Gameplay Data With Monte Carlo Tree Search to Emulate Human Play," (2016).

15. K. Kimura, Y. Tu, R. Tanji, and M. Mozgovoy, "Creating Adjustable Human-Like Ai Behavior in a 3d Tennis Game With Monte-Carlo Tree Search," in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)* (IEEE, 2021), 1–6.

16. G. N. Yannakakis and J. Togelius, *Artificial Intelligence and Games*, vol. 2 (Springer, 2018).

17. G. N. Yannakakis and J. Togelius, "A Panorama of Artificial and Computational Intelligence in Games," *IEEE Transactions on Computational Intelligence and AI in Games* 7, no. 4 (2015): 317–335, https://doi.org/10.1109/TCIAIG.2014.2339221.

18. D. Hooshyar, M. Yousefi, and H. Lim, "Data-Driven Approaches to Game Player Modeling: A Systematic Literature Review," *ACM Computing Surveys (CSUR)* 50, no. 6 (2018): 1–19.

19. S. Badillo, B. Banfai, F. Birzele, et al., "An Introduction to Machine Learning," *Clinical Pharmacology & Therapeutics* 107, no. 4 (2020): 871–885, https://doi.org/10.1002/cpt.1796.

20. C. Robert, *Machine Learning, a Probabilistic Perspective* (Taylor & Francis, 2014).

21. S. Marsland, *Machine Learning: An Algorithmic Perspective* (Chapman and Hall/CRC, 2011).

22. N. Justesen and S. Risi, "Continual Online Evolutionary Planning for In-Game Build Order Adaptation in StarCraft," *Proceedings of the Genetic and Evolutionary Computation Conference* (2017): 187–194.

23. J. Zhu, J. Villareale, N. Javvaji, et al., Player-AI Interaction: What Neural Network Games Reveal About AI as Play (2021), arXiv Preprint arXiv:2101.06220.

24. J. Ortega, N. Shaker, J. Togelius, and G. N. Yannakakis, "Imitating Human Playing Styles in Super Mario Bros," *Entertainment Computing* 4, no. 2 (2013): 93–104, https://doi.org/10.1016/j.entcom.2012.10.001.

25. N. Li, C. Gao, M. Li, Y. Li, and Q. Liao, "Econagent: Large Language Model-Empowered Agents for Simulating Macroeconomic Activities," (2023), arXiv Preprint arXiv:2310.10436.

26. B. Gao, W. Xue, and F. L. Lewis, "Heterogeneous Multi Player Imitation Learning," *Control Theory and Technology* 21 (2023): 281–291, https://doi.org/10.1007/s11768-023-00171-w.

27. Y. Luo, J. Huang, Y. Han, et al., "Beyond Believability: Accurate Human Behavior Simulation With Fine-Tuned LLMs," *ArXiv* (2024).

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

28. L. Chen, H. Wang, M. A. Rahman, and A. Seneviratne, "A Deep Learning Framework for Automatic Assessment of Presence in Virtual Reality," *Neural Computing and Applications* 37, no. 8 (2025): 6283–6303, https://doi.org/10.1007/s00521-024-10943-3.

29. J. Ho and S. Ermon, "Generative Adversarial Imitation Learning," in *Advances in Neural Information Processing Systems*, vol. 29 (Curran Associates, Inc, 2016), 4565–4573.

30. D. Silver, J. Schrittwieser, K. Simonyan, et al., "Mastering the Game of go Without Human Knowledge," *Nature* 550, no. 7676 (2017): 354–359, https://doi.org/10.1038/nature24270.

31. A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention Is All You Need," *Advances in Neural Information Processing Systems* 30 (2017): 5998–6008.

32. J. Togelius, G. N. Yannakakis, S. Karakovskiy, and N. Shaker, "Assessing Believability," in *Believable Bots* (Springer, 2013), 215–230.

33. A. M. Turing, "Computing Machinery and Intelligence," in *Parsing the Turing Test* (Springer, 2009), 23–65.

34. P. Hingston, "A Turing Test for Computer Game Bots," *IEEE Transactions on Computational Intelligence and AI in Games* 1, no. 3 (2009): 169–186, https://doi.org/10.1109/TCIAIG.2009.2032534.

35. S. Ross and D. Bagnell, "Efficient Reductions for Imitation Learning," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)* (PMLR, 2010), 661–668.

36. P. Abbeel and A. Y. Ng, "Apprenticeship Learning via Inverse Reinforcement Learning," in *Proceedings of the Twenty-First International Conference on Machine Learning* (ACM, 2004), 1.

37. N. A. Butt, Z. Mahmood, G. U. Rehman, et al., "The Development of Intelligent Agents: A Case-Based Reasoning Approach to Achieve Human-Like Peculiarities via Playback of Human Traces," *IEEE Access* 11 (2023): 78693–78712, https://doi.org/10.1109/ACCESS.2023.3274740.

38. N. A. Butt, M. M. Awais, S. Shahzadi, T. Kim, and I. Ashraf, "Towards the Development of Believable Agents: Adopting Neural Architectures and Adaptive Neuro-Fuzzy Inference System via Playback of Human Traces," *Journal of King Saud University, Computer and Information Sciences* 36, no. 8 (2024): 102182, https://doi.org/10.1016/j.jksuci.2024.102182.

39. P. Skočir, I. Bojić, and G. Ježić, "Implementation of Agent-Based Games Recommendation System on Mobile Platforms," in *Agent and Multi-Agent Systems: Technologies and Applications* (Springer, 2014), 67–76.

40. A. Mora, F. Aisa, P. García-Sánchez, P. Castillo, and J. Merelo, "Modelling a Human-Like Bot in a First Person Shooter Game," *International Journal of Creative Interfaces and Computer Graphics* 6, no. 1 (2015): 21–37, https://doi.org/10.4018/ijcicg.2015010102.

41. Y. Lin, X. Yu, and W. Glaubitt, "Trusted Behavior Identification Model for Distributed Node," *Journal of Intelligent & Fuzzy Systems* 37, no. 4 (2019): 4621–4631, https://doi.org/10.3233/JIFS-179296.

42. S. Ontañón, J. Montaña, and A. Gonzalez, "A Dynamic-Bayesian Network Framework for Modeling and Evaluating Learning From Observation," *Expert Systems With Applications* 41, no. 11 (2014): 5212–5226, https://doi.org/10.1016/j.eswa.2014.02.049.

43. G. Synnaeve, N. Nardelli, A. Auvolat, et al., "Torchcraft: A Library for Machine Learning Research on Real-Time Strategy Games," (2016), arXiv Preprint arXiv:1611.00625.

44. A. Summerville, S. Snodgrass, M. Guzdial, et al., "Procedural Content Generation via Machine Learning (PCGML)," *IEEE Transactions on Games* 10, no. 3 (2018): 257–270, https://doi.org/10.1109/TG.2018.2846639.

45. R. Baumgarten, S. Colton, and M. Morris, "Combining AI Methods for Learning Bots in a Real-Time Strategy Game," *International Journal of Computer Games Technology* 2009 (2009): 129075, https://doi.org/10.1155/2009/129075.

46. J. Togelius, R. De Nardi, and S. Lucas, "Towards Automatic Personalised Content Creation for Racing Games," in *2007 IEEE Symposium on Computational Intelligence and Games* (IEEE, 2007), 252–259.

47. C. C. Lin, *Playstyle and Artificial Intelligence: An Initial Blueprint Through the Lens of Video Games* (National Yang Ming Chiao Tung University, 2025).

48. J. Fombona, J. M. Sáez, and S. Sánchez, "Artificial Intelligence and Robotics in Education: Advances, Challenges, and Future Perspectives," *Social Sciences & Humanities Open* 11 (2025): 101533, https://doi.org/10.1016/j.ssaho.2025.101533.

49. G. Lampropoulos, E. Keramopoulos, K. Diamantaras, and G. Evangelidis, "Augmented Reality and Gamification in Education: A Systematic Literature Review of Research, Applications, and Empirical Studies," *Applied Sciences* 12, no. 13 (2022): 6809, https://doi.org/10.3390/app12136809.

50. A. Shmuel, O. Glickman, and T. Lazebnik, "A Comprehensive Benchmark of Machine and Deep Learning Across Diverse Tabular Datasets," (2024), arXiv Preprint arXiv:2408.14817, https://doi.org/10.48550/arXiv.2408.14817.

51. M. Binz, E. Akata, M. Bethge, et al., "A Foundation Model to Predict and Capture Human Cognition," *Nature* 644 (2025): 1–8.

52. L. Feng, F. Tung, M. O. Ahmed, Y. Bengio, and H. Hajimirsadeghi, "Were RNNs All we Needed?," (2024), arXiv Preprint arXiv:2410.01201, https://doi.org/10.48550/arXiv.2410.01201.

53. B. Yang, "Trade-Off Analysis of Efficiency and Accuracy in GRU vs LSTM," *ITM Web of Conferences* 80 (2025): 01017, https://doi.org/10.1051/itmconf/20258001017.

54. A. H. Rion, "Understanding Human Behavior on Social Media Through Machine Learning: A Literature-Guided Approach," (2025).

55. G. Nicora, S. Pe, G. Santangelo, et al., "Systematic Review of AI/ML Applications in Multi-Domain Robotic Rehabilitation: Trends, Gaps, and Future Directions," *Journal of Neuroengineering and Rehabilitation* 22, no. 1 (2025): 79, https://doi.org/10.1186/s12984-025-01605-z.

56. R. Hoseinnezhad, "A Comprehensive Review of Deep Learning Techniques in Mobile Robot Path Planning: Categorization and Analysis," *Applied Sciences* 15, no. 4 (2025): 2179, https://doi.org/10.3390/app15042179.

57. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," (2017), arXiv Preprint arXiv:1707.06347.

58. V. Mnih, A. P. Badia, M. Mirza, et al., "Asynchronous Methods for Deep Reinforcement Learning," (2016), arXiv Preprint arXiv:1602.01783.

59. Z. Sun, S. Qi, X. Huang, et al., "Towards Building Human-Like Smart Agents in Modern 3D Video Games (Student Abstract)," *Proceedings of the AAAI Conference on Artificial Intelligence* 39, no. 28 (2025): 29506–29508, https://doi.org/10.1609/aaai.v39i28.35305.

60. L. Schäfer, L. Jones, A. Kanervisto, et al., "Visual Encoders for Imitation Learning in Modern Video Games," in *The Seventeenth Workshop on Adaptive and Learning Agents* (Adaptive and Learning Agents (ALA) Workshop, 2025).

61. R. Spick, T. Bradley, A. Raina, P. V. Amadori, and G. Moss, "Behavioural Cloning in VizDoom," (2024), arXiv Preprint arXiv:2401.03993.

62. N. Justesen, M. Kaselimi, S. Snodgrass, et al., Human-Like Bots for Tactical Shooters Using Compute-Efficient Sensors (2024), arXiv Preprint arXiv:2501.00078, https://doi.org/10.48550/arXiv.2501.00078.

63. Y. Gao, F. Liu, L. Wang, et al., "Towards Effective and Interpretable Human-Agent Collaboration in Moba Games: A Communication Perspective," (2023), arXiv Preprint arXiv:2304.11632.

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

64. B. Baker, I. Kanitscheider, T. Markov, et al., "Emergent Tool Use From Multi-Agent Autocurricula," in *International Conference on Learning Representations* (ICLR, 2019).

65. F. Trujillo, M. Pozo, and G. Suntaxi, "Artificial Intelligence in Education: A Systematic Literature Review of Machine Learning Approaches in Student Career Prediction," *Journal of Technology and Science Education* 15, no. 1 (2025): 162–185.

66. A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation Learning: A Survey of Learning Methods," *ACM Computing Surveys (CSUR)* 50, no. 2 (2017): 1–35.

67. Project UBA, "Pac-Man Projects—UC Berkeley AI Course," (2025), http://ai.berkeley.edu/projectoverview.html.

68. J. L. Elman, "Finding Structure in Time," *Cognitive Science* 14, no. 2 (1990): 179–211, https://doi.org/10.1207/s15516709cog1402_1.

69. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation* 9, no. 8 (1997): 1735–1780, https://doi.org/10.1162/neco.1997.9.8.1735.

70. K. Cho, v B. Merriënboer, C. Gulcehre, et al., "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation," (2014), arXiv Preprint arXiv:1406.1078.

71. L. Breiman, "Random Forests," *Machine Learning* 45, no. 1 (2001): 5–32, https://doi.org/10.1023/A:1010933404324.

72. T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (ACM, 2016), 785–794.

1546427x, 2026, 6, Downloaded from https://onlinelibrary.wiley.com/doi/10.1002/cav.70185 by Spanish Cochrane National Provision (Ministerio de Sanidad), Wiley Online Library on [26/11/2026]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License