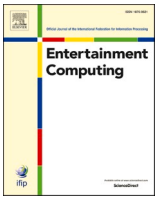
Entertainment Computing 55 (2025) 100987

Contents lists available at ScienceDirect

Entertainment Computing

journal homepage: www.elsevier.com/locate/entcom



Enhancing imitation learning training for non-player characters based on provenance data☆

Lauro V.R. Cavadas *, Esteban W.G. Clua, Troy C. Kohwalter, Sidney A. Melo

Universidade Federal Fluminense (UFF), Instituto de Computação, Rua Passo da Pátria, 156 - Bloco E, São Domingos, Niterói, RJ 24210-240, Brazil

ARTICLE INFO

Keywords:
Provenance
Non-Player Character
Imitation Learning
Games

ABSTRACT

Non-Player Characters (NPCs) play a crucial role in the immersive experience of a game world. When designed effectively, NPCs have unique personalities and react realistically to player actions. Meeting players' expectations for NPCs to resemble real individuals has become a major focus for game developers striving to enhance immersion.

In this work, we propose the use of data collected via provenance to create a model for training an NPC to act similarly to a human player using Imitation Learning. The main goal of this work is to improve the training efficiency of the agent, while preserving the high level of believability achieved in previous work. To this end, provenance is employed not only as a form of logging, but also as a means to guide and optimize the learning process — a contribution not previously explored in the literature.

To validate our model, we used the DodgeBall game within the Unity ML-Agents Toolkit for the Unity Engine. We compared our trained agent with an agent from previous work that used provenance solely for logging. Using win rate as a proxy for training efficiency, agents trained with our new model outperformed those trained with the previous approach, when evaluated after the same number of training steps.

Additionally, we created scenarios in which players participated in matches against both the new and previous agents, rating their believability. The results were promising in terms of both perceived believability and the efficiency of the training process.

In this work we propose to use data collected via provenance to create a model for training an NPC to act similarly to a human player using Imitation Learning. We use provenance not only as a form of log, but also to improve training efficiency, something that has not been presented in the literature until now. To validate our model, we used the DodgeBall game within the Unity ML-Agents Toolkit for Unity Engine. We compared our trained agent with an agent trained in previous work, which use provenance as a form of logging. Through matches between the two agents, those that were trained with our new model demonstrated greater efficiency. Additionally, we created scenarios of players participating in games against our current agent and our previous solutions, rating the believability of each. The results were quite promising, both in terms of believability and training efficiency.

1. Introduction

Artificial Intelligence (AI) plays a crucial role in game development. It holds a significant position across all genres of games, enabling developers to craft immersive worlds. By analyzing player actions, AI helps uncover unique properties within the game environment. Moreover, it simplifies enhancing the intelligence of non-player characters (NPCs), whether they are friendly or antagonistic. In many games, AI-controlled characters dynamically respond to real players' actions, often governed by intricate behavioral rules. The industry increasingly emphasizes creating NPCs that feel believable, enhancing player immersion and directly impacting enjoyment. Positive player reception translates to increased game sales, making AI an essential pillar for successful game design.

The advancement of AI research, along with the growth of the video game industry, has increasingly driven the development of believable

☆ This article is part of a special issue entitled: 'IFIP ICEC 2025' published in Entertainment Computing.

* Corresponding author at: Rua Passo da Pátria, 156 - Instituto de Computação - Sala 455, São Domingos, Niterói, RJ 24210-346, Brazil.
E-mail address: laurovrc@id.uff.br (L.V.R. Cavadas).

https://doi.org/10.1016/j.entcom.2025.100987
Received 5 February 2025; Received in revised form 24 May 2025; Accepted 28 June 2025
Available online 29 June 2025

non-player characters (NPCs). In recent years, Imitation Learning (IL) has been explored as an efficient and intuitive approach to programming autonomous behavior. Foundational perspectives on IL include algorithmic overviews and theoretical frameworks [1,2], as well as studies addressing challenges such as causal confusion in learning from demonstrations [3]. Other works provide broad insights into robot learning by demonstration and learning from human behavior [4–6].

A variety of IL methods have been developed, establishing the field as a prominent area of research. In the context of video games, several approaches have aimed to produce believable NPCs across different genres. For instance, IL has been applied to generate human-like bots using provenance data [7], to imitate player styles in platform games such as Super Mario Bros. [8], and to adapt Monte Carlo Tree Search (MCTS) for more human-like decision making [9]. Efforts have also been made to personalize content generation based on player behavior [10]. Broadly, these works can be categorized into direct and indirect behavior imitation [10]. Direct imitation involves the use of supervised learning algorithms trained on human play traces. In contrast, indirect imitation seeks to optimize a fitness function that evaluates how human-like an NPC's behavior appears.

In previous works [7,11], we proposed a method for training an NPC using imitation learning with the Generative Adversarial Imitation Learning (GAIL) framework to act similarly to a human player. We also changed the traditional method of rewarding the agent, which rewards correct actions and penalizes wrong actions, producing positive results. Some players competed against our agent and found that our NPC was believable by observing its actions and behaviors.

The use of provenance to improve agent training has not been explored in previous studies. In [7], we applied provenance strategies to manage data collected from gameplay sessions, but this information was not directly used in the learning process. We believe that provenance can offer valuable insights due to its ability to represent cause-and-effect relationships.

Based on this premise, the main objective of the present work is to improve the efficiency of NPC training by incorporating provenance data collected from previous gameplay. While training efficiency is the central focus of this work, we aim to preserve the level of believability already achieved in our earlier work. To support this approach, we also propose a new reward system that complements the use of provenance during the training phase.

We conducted a comparative analysis between two distinct models. Both models were used to train agents under identical conditions, employing the same neural network and an equal number of training steps. This approach ensures a fair comparison between the two models. The first model is the one developed in the current work, and the second is from our previous works [11]. To compare these two models we implemented one model interacting against the other through 30 matches, in order to verify which was better trained and had more victories.

Moreover, we invited a group of human participants to play against the agent trained using the new model. Their feedback, gathered through an evaluation process, was used to assess the believability of our proposed model. Specifically, we sought to determine whether the believability level improved, remained consistent or declined compared to the agent trained with the previous model. This feedback is crucial for guiding our ongoing efforts to refine and enhance the model introduced in this work.

The evaluation was made through the DodgeBall[1] game environment, which is part of the Unity ML-Agents Toolkit[2] for the Unity3D game engine.[3] Fig. 1 depicts the Dodgeball game, featuring a player's avatar within an environment with walls and obstacles.

The remainder of the paper is structured as follows: The second section introduces the theoretical foundation, the third section discusses related work, and the fourth section outlines our proposed model. The fifth section presents the results, and the final section concludes the work, highlighting conclusions about the proposal and future directions.

2. Theoretical foundation

In this section, we outline the fundamental components that underpin our approach. Our methodology is built upon three key concepts: Imitation Learning, Provenance, and Augmented Rewards (AR). Imitation Learning serves as the core training method for our agent, while provenance data is leveraged to enhance the agent's realism and optimize training efficiency. Provenance is increasingly recognized in Machine Learning (ML) systems for its ability to provide detailed workflow insights and support informed decision-making.

Provenance is a well-established concept in the domains of art and digital libraries, where it refers to the documented history of an artwork or the record of processes throughout a digital object's lifecycle. In 2006, during the International Provenance and Annotation Workshop, participants explored issues related to data provenance, documentation, derivation, and annotation. As a result, the Open Provenance Model (OPM) [12] emerged from the Provenance Challenge held at the workshop [13].

The OPM is a proposed framework for provenance designed to fulfill several key requirements [12]:

- Enable the exchange of provenance information across different systems;
- Facilitate the development and sharing of tools that operate on the provenance model;
- Provide a precise and technology-agnostic definition of provenance;
- Support the digital representation of provenance;
- Allow multiple levels of description to coexist;
- Establish a core set of rules to define valid inferences within provenance representations.

In OPM, it is assumed that the provenance of objects is represented by an annotated causality graph, which is a directed acyclic graph enriched with annotations capturing further information about its execution [13]. According to Moreau et al. [12], a provenance graph is a record of a past or current execution and not a description of something that could happen in the future.

The causality graph consists of nodes representing Artifacts, Processes, and Agents. Artifacts are immutable states that may represent either physical objects or digital entities within a computer system. Processes refer to actions or sequences of actions performed on or triggered by artifacts, leading to the creation of new artifacts. Agents are contextual entities that act as catalysts for processes, enabling, facilitating, controlling, or influencing their execution. The graph's edges denote causal dependencies, where the source represents the effect and the destination represents the cause [13].

Belhajjame et al. [14] introduced a conceptual data model that serves as the foundation for the W3C provenance (PROV) family of specifications, formalizing and superseding the model presented in [12]. PROV-DM differentiates between core structures, which encapsulate the fundamental aspects of provenance information, and extended structures, which accommodate more specialized use cases.

The PROV data model distinguishes core structures from extended structures: core structures form the essence of provenance information. They are commonly found in various domain-specific vocabularies that

[1] https://blog.unity.com/technology/ml-agents-plays-dodgeball. Last accessed: 15 Feb 2024.

[2] https://github.com/Unity-Technologies/ml-agents. Last accessed: 10 Jan 2024.

[3] https://unity.com. Last accessed: 18 Jan 2024.

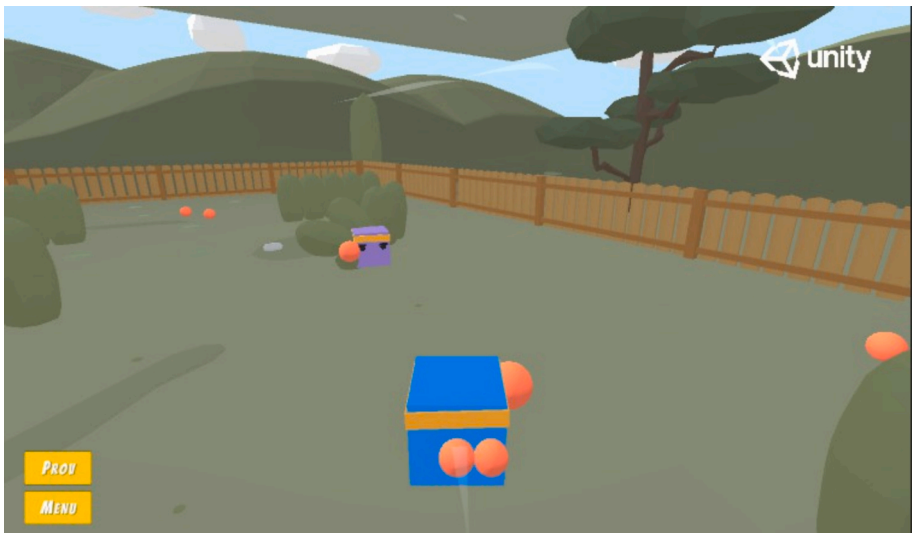


Fig. 1. DodgeBall game.

deal with provenance or similar kinds of information [Mappings]. Extended structures enhance and refine core structures with more expressive capabilities to cater more advanced uses of provenance [14].

Fig. 2 shows the PROV core structures. This specification presents the concepts of the PROV data model, and provenance types and relations, without specific concern for how they are applied. This makes possible to write useful provenance data and publish or embed it alongside the data it relates to [14].

Kohwalter et al. [13] introduced PinGU,[4] a novel approach designed to capture and store provenance data from game sessions, based on the Provenance in Games conceptual framework. The rich provenance data collected during gameplay is crucial for analyzing mistakes and reproducing results at a later stage.

In this approach, causal relationships between game elements are represented as edges connecting their respective nodes, forming a game provenance graph. Causality, in this context, describes the relationship between two events, where an earlier event influences a subsequent one. The provenance approach explicitly encodes these causal relationships as defined by the game developer. Each edge in the provenance graph denotes a specific type of relationship—typically causal—between the actions and/or states of game objects. The key advantages of provenance graphs include their ability to model causal dependencies, their structured representation of provenance elements as interconnected nodes,

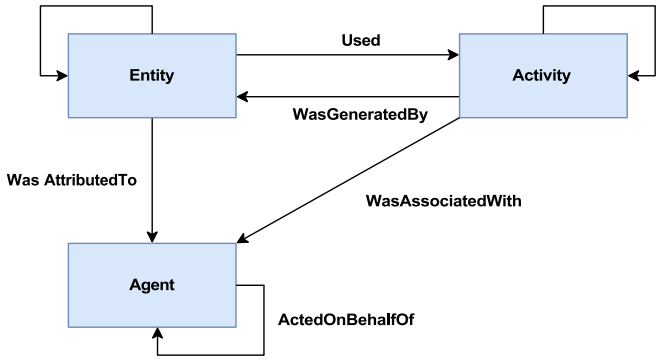


Fig. 2. PROV Core Structures (Informative).

[4] https://github.com/gems-uff/ping.

and their high level of detail, which provides valuable insights into game dynamics and player interactions.

Imitation Learning has emerged as a powerful technique for creating convincing NPCs in virtual environments. The ability of an NPC to act realistically and responsively enhances the player's experience and expands the possibilities for game development and simulations. In video games, NPCs are autonomous agents that interact with the player and the environment, playing a crucial role in shaping the game's narrative. They often drive the storyline by providing quests, sharing vital information, or adding depth and context to the plot. Additionally, NPCs contribute to the gameplay by introducing elements of challenge and fun, serving as adversaries or allies to the player.

Traditionally, creating NPCs involves extensive domain expertise, knowledge engineering, scripting, intuition, and iterative testing. However, the scale and complexity of NPC requirements are continuously increasing. Scale refers to the need for diverse and numerous characters to create the illusion of a densely populated virtual world. Depth, on the other hand, pertains to the intricacy and detail of the game world, mechanics, and narrative, which are further enriched through dynamic interactions between NPCs and players. For players, the game must feature many NPCs, each designed to be as believable, engaging, and human-like as possible [11].

In Imitation Learning (IL), an agent learns manipulation by observing expert demonstrations, enabling the generalization of skills to previously unseen scenarios. This process extracts information about the expert's behavior and the surrounding environment and maps observations to corresponding actions. The robot manipulation task can be framed as a Markov Decision Process (MDP), where the expert's action sequences are encoded into stateaction pairs that align with the demonstrated behavior [15]. In IL tasks, the agent aims to utilize a training set, composed of input–output pairs provided by an expert, to learn a policy replicating the expert's actions as closely as possible [16].

Fig. 3 illustrates the classification process of IL. Currently, the methods of IL can be divided into Behavior Cloning (BC), Generative Adversarial Imitation Learning (GAIL) [17] and Inverse Reinforcement Learning (IRL).

BC uses supervised learning to map states directly to actions based on demonstrations, relying heavily on the quality and coverage of the demonstration data. However, GAIL learns a policy that generalizes better to states outside the distribution of the demonstrations. It achieves this by training the agent to fool a discriminator that distinguishes between the agent's behavior and the expert's, leading to a more robust alignment with the target behavior.

BC requires extensive, high-quality demonstrations that cover a wide

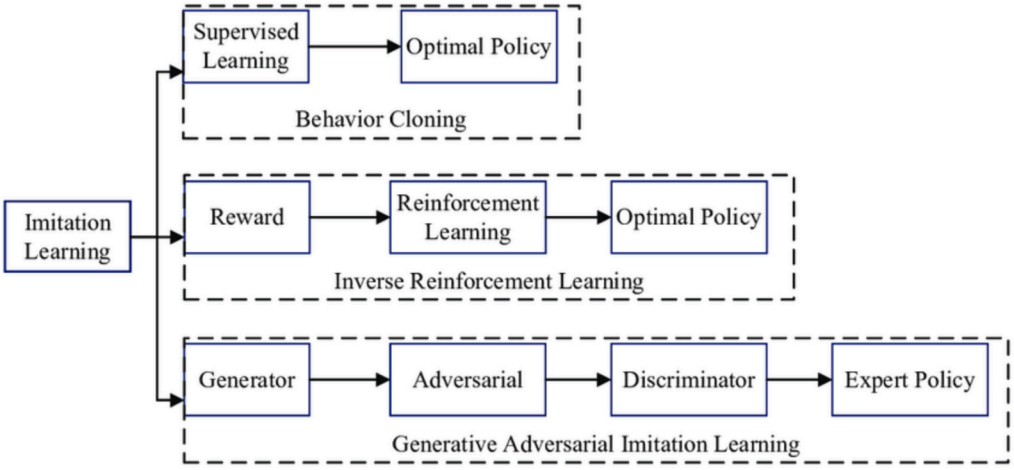


Fig. 3. Classification of Imitation Learning.

range of states. When data is limited or fails to represent the full state space, BC struggles to generalize. In contrast, GAIL is more efficient in scenarios with limited data, as the discriminator guides the policy to explore and adjust its behavior, even with fewer demonstrations. Additionally, BC replicates only the behaviors explicitly present in the demonstrations, which means the agent may fail to act appropriately in unseen states or novel conditions. By incorporating reinforcement learning, GAIL encourages the agent to explore beyond the observed distribution, enabling a more adaptable and comprehensive behavior.

In IRL, the same policy can often be explained by multiple reward functions, creating the reward ambiguity problem, which makes inferring the correct reward function inherently ill-defined. GAIL avoids this issue by focusing directly on aligning the agent's behavior with the observed demonstrations, bypassing the need for explicit reward inference.

Furthermore, IRL requires solving an optimization problem in each iteration to infer the reward function, followed by training a policy to optimize it, making it computationally expensive. In contrast, GAIL, as an end-to- end framework, simultaneously trains the discriminator and the policy in an adversarial setup, offering greater computational efficiency.

Given our objective of efficiently training a believable NPC, we selected GAIL over both BC and IRL due to its strong generalization capabilities, effectiveness with limited data, and computational efficiency.

Another important aspect of our approach is the use of Augmentation Rewards (AR). AR introduces additional or modified rewards to complement the original reward signal, enhancing the learning process. This technique is particularly effective in environments where rewards are sparse, as it provides the agent with more frequent feedback, helping to guide its actions more effectively [18]. By offering intermediate rewards, AR encourages the agent to explore paths that lead to the final goal, reducing the likelihood of becoming stuck in suboptimal states.

Moreover, AR proves beneficial in complex scenarios where the problem requires incremental behavior acquisition. By breaking down the learning process into smaller, manageable steps, AR enables the agent to develop its abilities gradually, improving both performance and adaptability. This incremental approach not only facilitates the agent's learning but also ensures that it can handle increasingly challenging environments with greater efficiency.

The specific implementation of AR within our model and its impact on the training process will be discussed in detail later in this work.

3. Related work

This chapter provides a comparative analysis of existing approaches in IL and the methodology proposed in this work. Traditional IL methods, such as BC and IRL, have demonstrated success in replicating expert behavior. However, these techniques present critical limitations. BC is particularly susceptible to compounding errors when encountering unfamiliar states, while IRL often involves solving complex reinforcement learning problems iteratively, resulting in high computational costs and instability.

To address these issues, Ho and Ermon [17] introduced the GAIL framework, which combines the advantages of BC and IRL. GAIL avoids the need for explicit reward modeling by employing a generator-discriminator architecture to learn expert trajectories. Although GAIL offers improved generalization and reduces reliance on hand-crafted reward functions, it still suffers from sample inefficiency, limited interpretability, and difficulty capturing contextual nuances of human behavior.

The present work builds upon GAIL by introducing two main contributions: the integration of provenance data and the use of augmentation rewards. Provenance offers structured and detailed information about player behavior obtained from gameplay sessions, while augmentation rewards help guide the learning process by encouraging goal-oriented actions. This combination enhances both the realism and efficiency of the learning process.

Previous research has explored the use of provenance in gaming primarily for post-hoc analysis and replay, rather than as a training mechanism. Thuler et al. [19] and Melo et al. [20] proposed systems that utilize provenance to record and replay gameplay for qualitative evaluation and game design improvement. Although valuable, these approaches do not incorporate provenance into the actual training of autonomous agents.

Karpov et al. [21] presented a controller that maps recorded human gameplay to bot actions. However, the approach depends entirely on finding exact matches in the database, leading to failure when encountering unfamiliar situations. In contrast, the approach presented in this work employs GAIL to generalize from the provenance data, allowing the agent to adapt and act in novel scenarios.

Pelling and Gardner [22] utilized supervised learning techniques, including support vector machines (SVMs) and probabilistic models, to develop believable non-player characters (NPCs). While their models performed well in controlled settings, they required manually labeled data and lacked scalability. The method proposed here leverages GAIL, which learns from unlabelled gameplay data and is more suitable for complex environments.

Cruz and Uresti [23] combined safe reinforcement learning with behavior modeling to develop adaptive bots. Their approach relied on real-time data and manually defined reward functions. By contrast, this work uses pre-recorded provenance logs and imitation learning,

eliminating the need for online exploration and manual reward engineering.

Yu et al. [24] introduced the Generative Intrinsic Reward-driven Imitation Learning (GIRIL) framework, which uses intrinsic rewards to encourage agents to outperform experts. Although effective, their objective differs from the present work, which focuses on replicating diverse and realistic human behavior to enhance believability rather than exceeding expert performance.

Iwasaki and Hasebe [25] proposed a method for generating varied playstyles by training agents with distinct reward functions and clustering their behaviors. Although the goal was not human imitation, their use of gameplay logs to influence agent behavior is conceptually aligned. Unlike their method, the current approach uses real human gameplay, captured via provenance, and leverages GAIL to produce emergent behavior without the need for manually designed reward functions.

Hsieh and Sun [26] applied Case-Based Reasoning (CBR) to predict individual player strategies from replayed action sequences. Their system required multiple sessions from the same player to extract consistent behavior. In contrast, the present work aggregates data from multiple players, allowing for generalization across diverse strategies and skill levels through a unified GAIL framework.

Kale et al. [27] conducted a study emphasizing the importance of provenance documentation in the context of earth sciences. Their findings indicate that provenance can improve the efficiency of AI systems when analyzing data, particularly when compared to traditional log-based approaches. However, the study focuses on provenance as a tool for enhancing data analysis workflows rather than as a component of training autonomous agents. Thus, it does not address the application of provenance in imitation learning scenarios, as proposed in the present work.

Ioffe and Szegedy [28] introduced batch normalization as a technique to enhance the training efficiency of deep learning models. By normalizing intermediate activations, their method allows for higher learning rates and reduces sensitivity to initialization, thereby accelerating convergence. Additionally, batch normalization acts as a regularizer, often removing the need for dropout. When applied to image classification tasks, it enabled faster training and achieved state-of-the-art performance on ImageNet, even surpassing human-level accuracy. Although effective, this method targets optimization in supervised learning contexts and does not address learning from demonstration or imitation.

The most directly related research is a previously proposed framework that used provenance data to recreate gameplay sessions and train an NPC using GAIL [11]. This model consists of four key phases, illustrated in Fig. 4:

1. Selection of game parameters required to recreate the player actions;
2. Collection of provenance data from gameplay sessions;
3. Recreation of the players' game sessions based on collected provenance data;
4. Training of the NPC using the GAIL framework.

In the initial stage, the framework identifies and selects gameplay data and input variables necessary for accurately reproducing player actions. During the second phase, data is collected from real users playing against the game's built-in AI. This data is structured into a provenance graph, with nodes representing actions and edges denoting relationships between actions, events, agents, and game state variables. The PinGU framework [13] is used to extract and organize this information.

In the third phase, the provenance file is read and interpreted sequentially. The data, originally in XML format, is converted into numerical variables that control the agent's behavior in the Unity Engine. This allows the system to accurately replay the player's actions within the game environment.

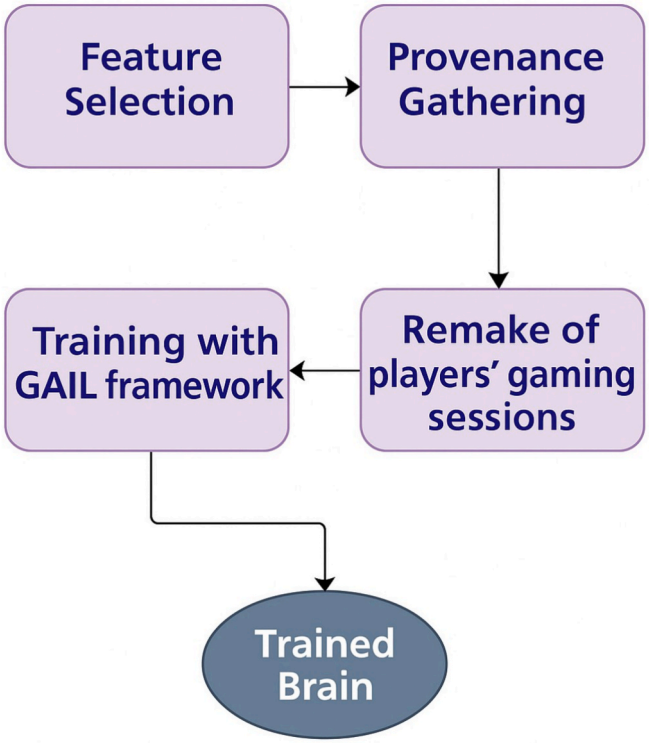


Fig. 4. Phases of the previous model.

Finally, in the fourth phase, the environment is prepared for training. The agent is initialized, and the system uses GAIL to train the policy based on the recreated gameplay. This approach enables the agent to generalize from the data, avoiding rigid behavior replication and improving its ability to respond to dynamic scenarios.

An important aspect of this framework is the reward system. In the previous version [11], the agent receives a reward for each action reproduced from the provenance data. While this approach fosters fidelity to player behavior, it does not account for the underlying objectives of the game, often resulting in inefficient strategies.

To overcome this limitation, a refinement was proposed [11] by incorporating augmentation rewards (AR). These additional incentives guide the agent toward completing meaningful in-game tasks, enhancing its strategic behavior without compromising the realism derived from provenance-based imitation.

This adjustment was essential to ensure the NPC not only mimicked the player's behavior but also demonstrated competitiveness by aligning its actions with the game's overarching goals. Simultaneously, the use of Augmentation Rewards enabled the NPC to maintain its human-like characteristics, striking an effective balance between realism and gameplay efficiency.

Building on the foundations discussed in this chapter, the following section presents the proposed framework developed in this research. By lever-aging the strengths of GAIL, the structured richness of provenance data, and the strategic refinement introduced by augmentation rewards, this framework aims to train NPCs that exhibit both human-like behavior and goal-oriented performance. The next section details the architecture, components, and implementation stages of the proposed solution.

4. Framework for utilizing provenance data to enhance training

We propose an imitation learning approach based on provenance data from previous gameplay sessions to create NPCs with realistic behaviors. Our reward strategy deviates from traditional methods, which typically focus on enhancing correct NPC actions or penalizing failures.

Instead, we reward the NPC when it executes a series of actions that, according to the provenance data, constitute a successful sequence within a winning round. Additionally, we provide small rewards when the NPC successfully completes tasks essential to achieving victory, thereby enhancing its competitiveness.

This reward system ensures a balance between replicating realistic player behavior and fostering strategic decision-making, making the NPC both believable and effective. The details of this reward system will be elaborated later in this section.

We train the NPC using provenance data gathered through the PinGU framework during multiple gameplay sessions. Provenance is critical to our approach, as it captures cause-and-effect relationships between activities within the game. This structured representation enables the NPC to learn in a manner that surpasses the capabilities of regular logs, providing a richer and more insightful training process.

We selected the GAIL framework to train our models, as it directly extracts a policy from data. GAIL addresses a subset of imitation learning challenges: learning to perform a task solely from expert demonstrations. In this setting, the learner can only access the expert's demonstrations and is neither allowed to query the expert for additional data during training nor provided with any reinforcement signal [17].

In our framework, we chose GAIL over BC due to the potentially vast state space in complex environments. When an agent trained with BC encounters a state it has not "seen" during training, it may fail to behave appropriately. In contrast, GAIL focuses on learning a policy that generalizes well across states, closely approximating the behavior observed by experts in similar contexts. This capability allows the agent to act effectively even when faced with unfamiliar states, ensuring robustness and reliability during gameplay.

Another important point is that, although we reward the groups of actions that the NPC performs, which are within a set of actions that led to victory within the round, not all actions in a victorious round can be considered correct, such as a ball throw that did not hit the enemy. This way, the agent will not be perfect, but it will mimic the mistakes that a real player could make.

To train the NPC using provenance data, thereby enhancing the learning process and enabling it to more effectively replicate human behavior within the same number of training steps as outlined in [11], we propose a final model structured into four distinct phases:

1. Find the winning rounds in the provenance file.
2. Identify the actions taken on the nodes of winning rounds.
3. Form groups of actions that resulted in victory.
4. Reward the execution of a group of winning actions.

Our model begins just before recreating the gameplay session described in the model presented in [11]. During the training phase, the last component of the final model is introduced, which involves rewarding the agent when it executes a sequence of actions that led a player to victory during the provenance data collection. This model was designed to enable more efficient training of the NPC by leveraging data stored in a provenance file, previously gathered from gameplay sessions involving real players. Fig. 5 illustrates the implementation of the final model starting from the previous model.

This strategy enables more efficient training of the NPC by leveraging data collected from previous gameplay sessions with real players and stored in a provenance file. Therefore, as a first step, the data from some players who played the game against an NPC that had already been trained using MultiAgent POsthumous Credit Assignment (MA-POCA) [29] were obtained and saved in a provenance file.

We saved the actions of the players and opponents during each executed round, and these actions were divided into: Move, Throw a ball, Dash, Pick a ball, Being Hit, Win, and Lose. The actions were recorded in the provenance file using the PinGU framework [13]. During the game execution, each input is checked, and according to the action execution, when possible, data is performed and saved in the provenance at that moment. An example of an action saved in the provenance is when the player has a ball in his hand and presses the input to throw the ball. Since he has the ball, the movement is executed in the game, and we save the action of throwing the ball in the provenance file. If he

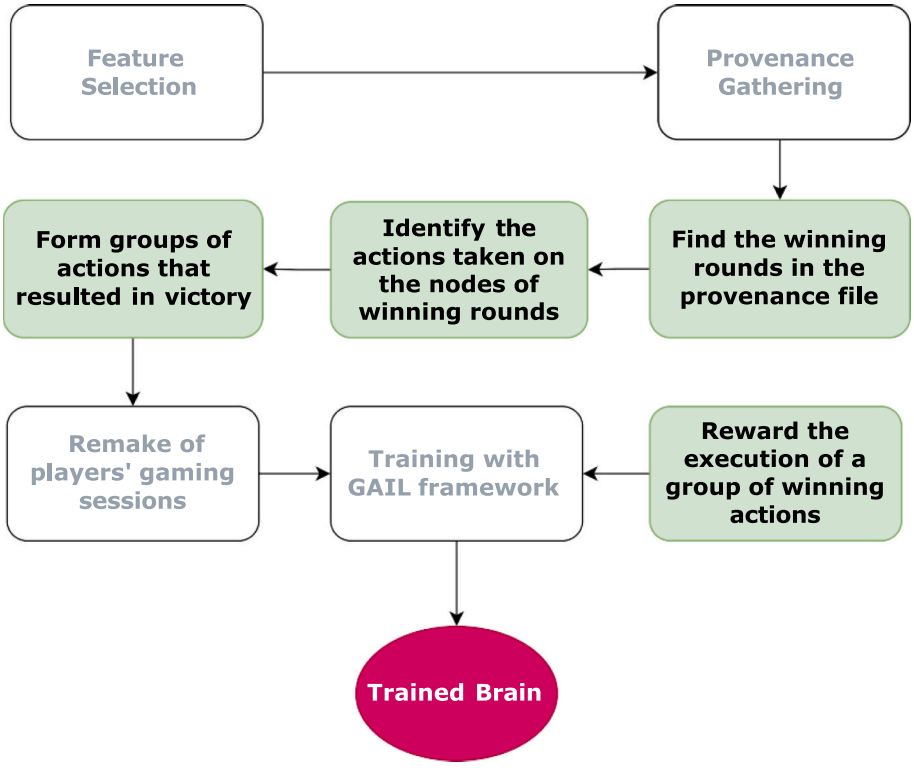


Fig. 5. The complete model.

does not have a ball in his hand when pressing the button, the action is not executed and we do not save this input in the provenance.

4.1. Find the winning rounds in the provenance file

The first stage of our model is a preparatory phase conducted before the training process. During this stage, we methodically process each node from the provenance file, stored in XML format, using time-based navigation. We analyze the label associated with the event for every node, focusing specifically on nodes labeled as "Win." When such a node is identified, we verify its relevance to the player. If the node is confirmed to be associated with the player, we save its identifier for later use in subsequent model stages.

It is important to note that a single provenance file can contain data from multiple matches. If we identify a node from the third match where the player won, we cannot assume that all preceding actions contributed to that victory, as earlier matches may include losses. Therefore, it is essential to establish a method for identifying the starting point of each new game match within the provenance file.

To achieve this, we create a node labeled "Respawn" whenever a new match begins. This allows us to determine the start time of each round. Additionally, we save the identifier of the node immediately following the Respawn event, as it represents the first action of that particular round. This approach ensures precise segmentation of matches within the provenance data, enabling accurate analysis and tracking of actions associated with each round.

4.2. Identify the actions taken on the nodes of winning rounds

The second stage is also conducted before the training process. In the provenance file, all events during the round are recorded, including player actions, opponent actions, and general round events. After identifying the node corresponding to the player's victory in the previous stage, we search through the edge list for the node with the victory identifier, which is stored in the "targetID" attribute. Fig. 6 illustrates the attributes of an edge, specifying the starting vertex (sourceID) and the ending vertex (targetID) of the edge. This step establishes the connections necessary to trace the sequence of actions leading to the victory.

With the identifier of the winning node and the start-of-round identifier, we traverse the edges to determine all events related to the player that occurred from the beginning of the round up to their victory. Starting from the edge containing the player's victory identifier, we recursively trace backward, searching for edges with a sourceID equal to the targetID of the previous edge, continuing this process until reaching the edge associated with the round's initial identifier.

The recursive function used for this process takes the following parameters: the index of the initial node, the index of the final node, and the sourceID of the current edge, which corresponds to the targetID being searched for. After finding the preceding event, the function saves the action's identifier to a text file and then calls itself again, using the identifier of the newly discovered event as the parameter.

Algorithm 1 presents the pseudocode for tracing and recording past actions, demonstrating how the function systematically searches for sequences of events that led to victory.

Algorithm 1 Pseudocode for tracing and recording past actions.

```
procedure FindPreviousActions(start, end, currentID)
  while end > start do
    if connectionType[end] ="WasInformedBy" then
      if sourceID[end] = currentID then
        Save sourceID[end] to the file
        currentID ← targetID[end]
      end if
    end if
    end ← end − 1
    FindPreviousActions(start, end, currentID)
  end while
end procedure
```

In the recursive function, we navigate from node to node by verifying whether the sourceID matches the targetID. If they are equal, we confirm that the current node represents an event within the player's sequence of activities.

The underlying idea is that when an activity is performed, it generates an entry in the provenance file, subsequently creating a vertex in the provenance graph. When another activity is performed later, it creates a new vertex and an edge connecting the two activities. This structure allows us to establish a temporal relationship between events.

For instance, as illustrated in Fig. 7, both activities are connected to the entity Player, with the activity"PickedBall" having its sourceID linked to the activity"Walking." This means that, in the game context, the player first walked and then picked up a ball. Regarding edge navigation, the edge connecting these two activities has a sourceID corresponding to the vertex labeled"PickedBall" and a targetID corresponding to the vertex labeled"Walking." This structure clearly represents the sequence and causality of events within the player's activity timeline.

It is important to highlight that, in a regular log, temporal associations are limited to the chronological order of events within the nodes, without explicitly capturing the connections between nodes timelessly or causally. In contrast, a provenance file saves the nodes in chronological order and establishes links between them, enabling a deeper understanding of their relationships.

For example, in a provenance file, we may observe an event related to the player, followed by several other events associated with the

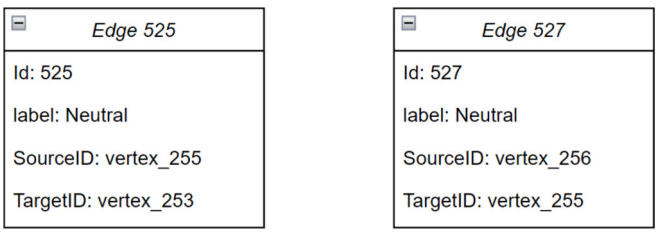


Fig. 6. Example of two edges in the provenance file.

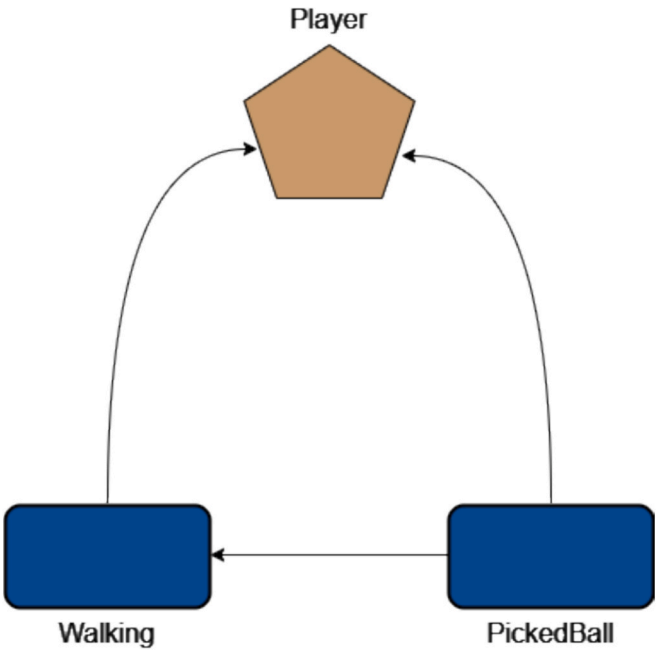


Fig. 7. Snippet from provenance graph.

opponent's character, and then again another event linked to the player. This interconnected structure allows to trace the dependencies and relationships between events, regardless of whether they are temporally sequential or interspersed with activities from other entities. This feature provides a more comprehensive and structured view of the interactions within the system.

However, since the provenance file is represented as a graph where edges connect nodes, we can establish that the first node related to the player has a direct relationship with the next node associated with the player through the edge properties "sourceID" and "targetID." These properties define the linkage and sequence of events within the graph.

Once the scan of a match is complete, we navigate the provenance file to locate the next winning node identified in the previous step. Upon finding it, we repeat the process of traversing the nodes back to the start of the current round, systematically saving the identifiers of the actions in the text file. This iterative approach ensures that all relevant sequences of player actions leading to victories are recorded and preserved for further analysis or training.

4.3. Forming groups of actions that resulted in victory

In the final stage before starting the training step, we scan the entire list of vertices in the provenance file and verify whether each vertex identifier matches an identifier stored in the text file from the previous step. If a vertex identifier is found in the text file containing the winning nodes, we replace the identifier in the file with the label associated with that vertex. This label specifies the action or event recorded at that node.

After completing the replacement, to identify the sequence of activities that led the player to victory, we divided these activities into groups of three items. This grouping is essential to prevent the agent from receiving rewards for every single action, which would result in constant rewards. For instance, consider a match where the player lost. The player might have walked, dashed, or thrown a ball during that match. If we rewarded every action indiscriminately, the reward system would lack specificity and could reinforce ineffective behaviors.

To make the reward system more targeted, we only reward the agent when it performs a precise group of actions that matches a sequence found in the text file of victorious actions. To facilitate this, we separate every three sequential items in the text file by inserting a new line with the label"LineBreak" after each group. This formatting aids in the navigation and processing of action sequences in the subsequent stages of the model, ensuring more accurate reinforcement of behaviors associated with success.

The decision to use three items per group was made after conducting various tests, as this number struck the optimal balance between challenge and feasibility. With a smaller number of items, players often repeated the sequence of actions, leading to overly generic behaviors and reducing the specificity of the reward system. On the other hand, increasing the number of items in each group made it significantly harder for players to replicate the exact sequence of actions, resulting in rewards becoming too infrequent.

By choosing groups of three items, the system ensures a level of complexity that discourages simplistic repetition while still allowing the sequence of actions to be achievable, thus maintaining an effective and meaningful reward mechanism. Fig. 8 shows a final snippet of the text file, with the winning action groups separated by lines containing the entry "LineBreak".

4.4. Reward the execution of a group of winning actions

The previous steps resulted in creating a text document containing sets of three actions that led to victory in each recorded game session, which is stored in the provenance file. When initiating the training process, we load this text file into the code and read its contents line by line.

For each line, we add the action it contains to the same array until a line with the entry "LineBreak" is encountered. At this point, this array of actions is added to a list of arrays, ultimately storing all sets of actions. After processing the "LineBreak", we continue reading the subsequent lines, adding them to a new array of actions and repeating the process.

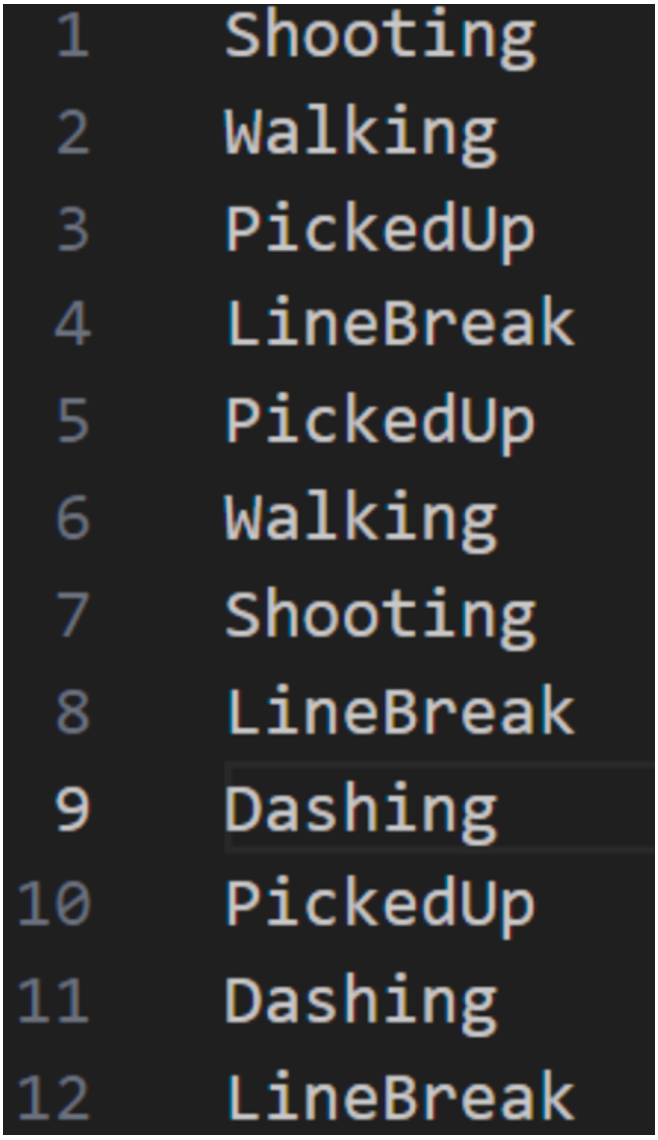


Fig. 8. A snippet of the text file showing the sequence of actions in a winning round.

As previously described, it is important to emphasize that each array will contain exactly three actions. By the end of this procedure, the list will include multiple sets of actions, capturing sequences from all game sessions recorded in the original provenance file. This structured format is essential for efficiently utilizing the data during training.

In the final stages of our previous model, the agent starts executing actions based on what was previously recorded in the provenance file generated during the gameplay sessions of real players. At this point, the step of our previous model is executed, which involves recreating the actions of real players read from the provenance file. This is done at the beginning of the final stage, the training phase using the GAIL framework.

Also at the very beginning of the training process, we create an initially empty vector, and for each action performed by the agent, we add the action to this vector. The action is named in the same way as in the text document containing the winning actions. We then check if the vector contains three actions to compare it with the groups of winning

actions in the list. When the vector reaches three actions, we verify if all the actions within this vector match any winning action group in the list. If an identical sequence of actions is found, the agent is rewarded, and the vector is cleared to begin checking the next group of actions the agent performs.

In our final model, in addition to this reward, a small reward is also given when the agent successfully achieves the game's objective. This approach ensures that the agent is not only incentivized to replicate winning sequences but also motivated to accomplish the game's main goal.

Algorithm 2 presents the pseudocode that details the process of validating and rewarding action sequences performed by the NPC. Initially, the algorithm verifies whether the number of recorded actions in the vector reaches the required threshold of three consecutive actions. Once this condition is satisfied, the collected sequence is compared against stored sequences of successful actions from previous gameplay sessions. If a match is found within the predefined list, a reward is assigned to reinforce the correct behavior, encouraging the NPC to learn and replicate effective strategies while maintaining human-like characteristics.

5. Results

This section presents and analyzes the results obtained through the evaluation of the proposed model. The effectiveness of the approach was assessed from two complementary perspectives: the agent's training efficiency and its believability when interacting with human players. Each aspect was examined through distinct experimental setups, as detailed in the subsections below.

Algorithm 2 Pseudocode for validating and rewarding action sequences.

```
procedure ProcessAction
  Add the Action done to the actions list
  if size of actions list = 3 then
    Set exist ← False
    for all sequence in actionsList do
      for each index from 0 to length of sequence − 2 do
        if 3 consecutive actions in sequence match actions list then
          Set exist ← True
          break
        end if
      end for
    end for
    if exist is True then
      Add a reward of 0.2
    end if
    Clear the actions list
end if
end procedure
```

5.1. Methodology and outcomes regarding training efficiency

For the evaluation of training efficiency, we performed a competitive comparison between the model presented in this work and our previous model, which employs a reward method based on all of the player's recreated actions. We set both trained agents to compete in 30 matches. Both models were trained with the same number of steps to ensure similar conditions. The number of trained steps was 4,500,000.

The number of matches won by the model developed in this work was 23, while our model presented in our previous work had 7 matches won. This number indicates that our model achieved a winning percentage of 76.7 % compared to the 23.3 % win rate of our previously trained model. This result demonstrates that our model exhibited more efficient learning with the same number of training steps, as it significantly outperformed the previous model in terms of victories.

To further validate these findings, a 95 % confidence interval was calculated for the winning percentages of both models. The confidence interval for our current model ranges from 61.5 % to 91.8 %, while for the previous model, it ranges from 8.2 % to 38.5 %. These intervals provide strong statistical support for the superiority of our model, as the intervals do not overlap, indicating a statistically significant difference in performance. The lower bound of our model's interval remains well above the upper bound of the previous model's interval, reinforcing the reliability of our results and confirming that the proposed model provides a substantial improvement in performance.

Table 1 displays the results after the confrontation of the two models, highlighting the winning percentages and their respective confidence intervals.

5.2. Methodology for assessing believability

Despite the efficiency in training regarding performance in the number of wins against another trained agent, we also validate its believability. For the evaluation of believability of this model we compare this model to our previous model developed in [11]. The findings from the previous work had already demonstrated that the previous model had already surpassed the traditional reward model in believability. The traditional reward model gives positive feedback to the agent when it performs a correct action, such as winning a match, and penalizes the agent when it performs an incorrect activity, such as colliding with a wall.

We divided the evaluation of believability into two aspects: movement and ball throwing. We invited 9 players, aged 17–27, including 3 females and 6 males, to play the DodgeBall game. Each participant played against both models for five rounds and was then asked to complete a questionnaire in which they rated the NPC's movement and ball throwing on a scale from 1 to 10. A rating of 1 indicated behavior completely opposite to that of a player, while a rating of 10 indicated behavior close to real player behavior. Both models were evaluated using the same number of steps.

At no point did we indicate which model was the Previous or Final Model. Five participants began the test by playing against the NPC trained with the Previous Model, while four participants started by playing against the NPC trained with the Final Model.

To evaluate the NPC's movements, we instructed participants to observe how the actors moved through the scenario, dodging obstacles, collecting balls from the ground, and approaching the player naturally and realistically. They also considered the movements' fluidity and naturalness, considering whether the NPC seemed to be acting according to the game and environment logic. To evaluate the NPC's ball throwing, participants were instructed to observe how the NPC collected the ball and executed the throws, including both hits and misses. Participants also had a section in the questionnaire where they were asked to write their perceptions of each model and were encouraged to compare them as well.

In this game the player must defeat the opponent's avatar by throwing a ball at it, and the round finishes when one of them hits the other twice. The Dodgeball game offers players the flexibility to employ various strategies, such as playing aggressively or carefully launching balls from a distance.

Our goal is to create an agent that behaves like a player and can be perceived as a real human player, including making correct and incorrect actions. We considered all actions in a winning match during training to prevent our agent from being invincible and looking like a

Table 1
Competitive comparison with confidence intervals.

Model	Matches Played	Matches Won	Winning Percentage (%)	95 % Confidence Interval
Our model	30	23	76.7	[61.5 %, 91.8 %]
Previous model		7	23.3	[8.2 %, 38.5 %]

machine. As a basis, we used a provenance file created from game sessions as real players.

This means that some actions may have been incorrect during the winning matches, but the player won in the end.

5.3. Evaluation of NPC's movement

To evaluate the movements of the NPC, players observed how the actors moved through the scenario, dodging obstacles, collecting balls from the ground, and approaching the player naturally and realistically. They also considered the movements' fluidity and naturalness, considering whether the NPC seemed to be acting according to the game and environment logic.

According to the players on the questionnaire, the NPC trained using the proposed model moved well, colliding rarely, usually during backward dashes or when passing between an obstacle and a wall. In the Dodgeball game, the character can carry one ball in its hand and up to 4 balls at the same time. The NPC always sought to collect balls from the ground so that it always had the possibility of throwing the ball, which is why the NPC often had more than one ball available for throwing. The NPC also always used the dash as soon as it was available, remembering that in the Dodgeball game, the dash has a small cooldown, preventing the player from using it sequentially.

The players also noted that our agent's movement was not "robotic style" being fluid and simulating a real player's movement. Table 2 shows the ratings given by the players regarding the movement of our NPC and the NPC trained by the previous model.

The results showed that both models received good ratings for their movement, indicating that the players found the movement to be close to what a real player would do. The average rating for the movement of our trained agent was 9.11, while the average rating for the agent trained by the previous model was 8.77. This shows that our agent has a very realistic movement compared to a real player.

The statistical analysis presented in Table 3 provides valuable insights into the performance and believability of the NPC models evaluated. The standard deviation for both models is relatively low, with the final model showing a slightly smaller value (0.782) compared to the previous model (0.833). This indicates that the player ratings for the final model are slightly more consistent, suggesting a narrower variation in player perceptions of the NPC's believability. The coefficient of variation (CV) further supports this observation, as the final model exhibits a lower CV (8.58 %) compared to the previous model (9.49 %), demonstrating higher stability and reliability in player evaluations.

Although the Wilcoxon signed-rank test did not reveal a statistically significant difference between the two models ($p = 0.313$), the consistency of the ratings provides important insights into player perception. The final model achieved slightly lower standard deviation and coefficient of variation values compared to the previous model, indicating a more stable and consistent evaluation among participants. These results suggest that, while both models were perceived as similarly believable in terms of movement, the refinements implemented in the final model contributed to a more uniform and reliable user experience. This consistency, even in the absence of strong statistical significance, supports the robustness of the proposed approach.

Table 2
Ratings given by players on the believability of NPC movements.

Model	Players									Average Rating
	A	B	C	D	E	F	G	H	I	
Previous Model	9	8	9	8	8	10	9	10	8	8.77
Final Model	9	8	10	9	8	10	9	10	9	9.11

Table 3
Statistical Analysis Results for NPC movements.

Statistic	Previous Model	Final Model
Standard Deviation	0.833	0.782
Coefficient of Variation (%)	9.49	8.58
Wilcoxon Signed-Rank Test (P-Value)	0.313	
Effect Size (r)	0.00	

5.4. Evaluation of NPC's ball throw

Regarding ball throwing, our NPC demonstrated good accuracy, especially by making throws when the opponent was in a position to be hit. This was an area where the agent developed by our previous model struggled the most, often picking up the ball but throwing it without aiming at the opponent.

Our NPC would collect balls and wait for a clear, obstacle-free line of sight to make the throw, making it resemble a real player. The players supported this analysis, as the average score for throwing was 8.55, while the average score for the agent trained by the previous model was 7.66. This difference is shown in Table 4, which displays the ratings given by the players related to the ball throw in both trained agents.

Some players noted that the throwing behavior has improved significantly, with the agent holding the ball and waiting for a good opportunity to hit the player. However, there were instances where the agent could have thrown the ball because it was in front of the player with a clear path but did not do so at that moment. According to a group of players, this behavior was observed but rarely occurred again. After achieving the scores, we asked other players if they had noticed this behavior, but they did not indicate any issues during their matches.

The results presented in Table 5 provide important insights into the statistical evaluation of the NPC ball-throwing performance for the previous and final models. The standard deviation of the final model (0.726) is notably lower than that of the previous model (1.000), indicating that the ratings for the final model were more consistent among players. This suggests that the improvements made in the final model led to a more uniform perception of its believability.

The CV further supports this conclusion, with the final model exhibiting a CV of 8.49 %, compared to 13.04 % for the previous model. A lower CV for the final model highlights a reduced relative dispersion in player ratings, emphasizing its stability and reliability regarding player perception.

Given the small sample size and paired nature of the data, we applied the Wilcoxon signed-rank test. Although the calculated effect size was small, the statistically significant difference observed ($p = 0.023$) indicates that participants consistently rated the final model's ball-throwing behavior as more believable. This suggests that the improvements introduced in the final model—such as more deliberate throwing decisions and enhanced spatial awareness—resulted in a perceptible enhancement of realism. The consistency of player evaluations, as shown by the reduced standard deviation and coefficient of variation, further supports the reliability of this perception. Therefore, despite the modest magnitude of change, the outcome highlights the success of the final model in refining the NPC's behavior in a way that is meaningful to players.

Overall, the statistical analysis underscores the success of the final model in enhancing both the consistency and overall believability of the NPC's ball-throwing behavior. These findings validate the effectiveness

Table 4
Ratings given by players on the believability of NPC ball throws.

Model	Players									Average Rating
	A	B	C	D	E	F	G	H	I	
Previous Model	7	6	9	8	8	8	7	9	7	7.66
Final Model	8	8	9	8	8	9	9	10	8	8.55

Table 5
Statistical Analysis of NPC Ball Throws.

Statistic	Previous Model	Final Model
Standard Deviation	1.000	0.726
Coefficient of Variation (%)	13.04	8.49
Wilcoxon Signed-Rank Test (P-Value)	0.023	
Effect Size (r)	~0.00	

of the modifications implemented in the final model.

5.5. Which NPC appeared more human?

Players were asked in the questionnaire which of the two NPCs (trained with the previous or final model) they considered to be more human-like in behavior. This subjective question aimed to gather a direct impression of believability from the player's perspective. The majority of participants (8 out of 9) identified the NPC trained with the final model as more human-like. This reinforces the quantitative findings and highlights the effectiveness of our proposed approach in preserving realistic behavior. Fig. 9 presents the detailed results of this survey.

6. Conclusion and future work

Immersion is one of the key experiences a player should have when playing a game and NPC behavior holds an important role in the field. Our proposal suggests using information from provenance obtained from game sessions with real players to increase learning efficiency through the Imitation Learning method. Within Imitation Learning, we use the GAIL framework, allowing the NPC to learn a policy and create actions that roughly approximate what a player would do.

One of the persistent challenges in training NPCs is the need for vast amounts of high-quality training data, which are often expensive and time-consuming to collect. In this context, data provenance presents itself as an innovative solution. Provenance offers an additional layer of information that can be used to enrich the training process. By integrating provenance data collected from previous sessions, it is possible to accelerate training and provide NPCs with a deeper capacity for adaptation and learning.

We previously proposed using provenance with Imitation Learning, involving cause-and-effect links and providing more specific knowledge about the events that occur while a user play was used only as a logging record strategy. Using the cause-and-effect relationship provided by provenance to improve training by achieving better results with the same number of steps has not been studied in the literature until the present work.

In this work, the environment used for validating was the DodgeBall game from the Unity Engine ML-Agents package. Nine students were selected to participate in the game and played three different rounds. The NPC was rated from 1 to 10 by the players based on its movement and ball-throwing behavior in the game. A rating of 1 means the agent did not resemble a real player at all, while a rating of 10 means a real player can easily mistake it. The comparison was made between our novel model and our previous solution. Regarding movement, our model achieved an average rating of 9.11 with a standard deviation of 0.737 compared to an average of 8.77 with a standard deviation of 0.746 from the previous model. For ball throwing, players gave our model an average rating of 8.55 with a standard deviation of 0.684, while the previous model received an average of 7.66 with a standard deviation of 0.942.

The results demonstrated that our NPC improved in believability, which is significant given our objective to maintain or enhance believability even when employing more efficient training methods. A second validation was also conducted, in which both models were directly compared. A total of 30 matches were played, with our model winning

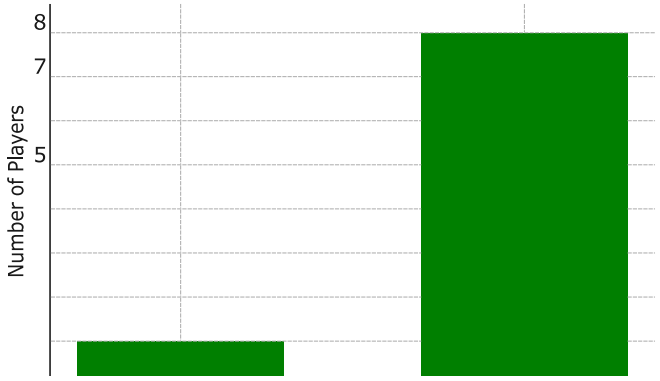


Fig. 9. Players' responses on the model perceived as most similar to human behavior.

23 of them (76.7 %), demonstrating clear superiority and supporting the hypothesis that incorporating provenance information during training can enhance efficiency within the same number of steps.

In the 95 % confidence interval tests, our model also demonstrated superiority over the previous model. All the results were quite satisfactory and lead us to believe that this solution is very promising and has significance for future advances in using provenance in conjunction with AI, especially with Imitation Learning.

For future work, we want to test the framework in other games to verify how generic it is and if it works in different game genres. We also want to increase the tests by using more steps and a larger number of matches and players involved in the tests. Another future improvement could be the expansion of the comparison of the efficiency of our model against other training models.

CRediT authorship contribution statement

Lauro V.R. Cavadas: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Esteban W.G. Clua:** Writing – review & editing, Supervision, Project administration, Conceptualization. **Troy C. Kohwalter:** Writing – review & editing, Supervision, Project administration, Conceptualization. **Sidney A. Melo:** Software, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] T. Osa, J. Pajarinen, G. Neumann, J.A. Bagnell, P. Abbeel, J. Peters, et al., An algorithmic perspective on imitation learning, foundations and trends®, Robotics 7 (1–2) (2018) 1–179.
[2] S.K.S. Ghasemipour, R. Zemel, S. Gu, A divergence minimization perspective on imitation learning methods, in: Conference on Robot Learning, PMLR, 2020, pp. 1259–1277.
[3] P. De Haan, D. Jayaraman, S. Levine, Causal confusion in imitation learning, Adv. Neural Inf. Proces. Syst. 32 (2019).
[4] A. Billard, D. Grollman, Robot learning by demonstration, Scholarpedia 8 (12) (2013) 3824.
[5] A.G. Billard, S. Calinon, R. Dillmann, Learning from humans, in: Springer Handbook of Robotics, 2016, pp. 1995–2014.
[6] J.A. Bagnell, An invitation to imitation, Tech. rep., Carnegie-Mellon Univ Pittsburgh Pa Robotics Inst, 2015.

[7] L.V.R. Cavadas, E. Clua, T.C. Kohwalter, S.A. Melo, Training human-like bots with imitation learning based on provenance data, in: 2022 21st Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), IEEE, 2022, pp. 1–6.
[8] J. Ortega, N. Shaker, J. Togelius, G.N. Yannakakis, Imitating human playing styles in super Mario bros, Entertain. Comput. 4 (2) (2013) 93–104.
[9] A. Khalifa, A. Isaksen, J. Togelius, A. Nealen, Modifying mcts for human-like general video game playing, 2016.
[10] J. Togelius, R. De Nardi, S.M. Lucas, Towards automatic personalised content creation for racing games, in: 2007 IEEE Symposium on Computational Intelligence and Games, IEEE, 2007, pp. 252–259.
[11] L.V.R. Cavadas, E. Clua, T.C. Kohwalter, S. Melo, Using provenance data and imitation learning to train human-like bots, Entertain. Comput. 48 (2024) 100603.
[12] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, et al., The open provenance model core specification (v1. 1), Fut. Gener. Comput. Syst. 27 (6) (2011) 743–756.
[13] T. Kohwalter, E. Clua, L. Murta, Provenance in games, in: Braz. Symp. Games Digit. Entertain. SBGAMES, 2012, pp. 162–171.
[14] K. Belhajjame, R. B'Far, J. Cheney, S. Coppens, S. Cresswell, Y. Gil, P. Groth, G. Klyne, T. Lebo, J. McCusker, et al., Prov-dm: the prov data model, W3C Recommendation 14 (2013) 15–16.
[15] J. Hua, L. Zeng, G. Li, Z. Ju, Learning for a robot: deep reinforcement learning, imitation learning, transfer learning, Sensors 21 (4) (2021) 1278.
[16] A. Attia, S. Dayan, Global overview of imitation learning, arXiv preprint arXiv: 1801.06503 (2018).
[17] J. Ho, S. Ermon, Generative adversarial imitation learning, Adv. Neural Inf. Proces. Syst. 29 (2016).
[18] r. t. p. Author names (if known, Goal exploration augmentation via pre-trained skills for sparse-reward reinforcement learning, Mach. Learn. (2023), URL https://link.springer.com/article/10.1007/s10994-023-06503-w.
[19] L. Thurler, S. Melo, L. Murta, T. Kohwalter, E. Clua, Using provenance and replay for qualitative analysis of gameplay sessions, Entertain. Comput. 52 (2025) 100778.
[20] S. Melo, L. Thurler, A. Paes, E. Clua, Game provenance graph-based representation learning vs metrics-based machine learning: an empirical comparison on predictive game analytics tasks, Entertain. Comput. 52 (2025) 100755.
[21] I.V. Karpov, J. Schrum, R. Miikkulainen, Believable bot navigation via playback of human traces, in: Believable Bots, Springer, 2013, pp. 151–170.
[22] C. Pelling, H. Gardner, Two human-like imitation-learning bots with probabilistic behaviors, in: 2019 IEEE Conference on Games (CoG), 2019, pp. 1–7.
[23] C. Arzate Cruz, J.A. Ramirez Uresti, Hrlb 2: a reinforcement learning based framework for believable bots, Appl. Sci. 8 (12) (2018) 2453.
[24] X. Yu, Y. Lyu, I. Tsang, Intrinsic reward driven imitation learning via generative model, in: International Conference on Machine Learning, 2020, pp. 10925–10935.
[25] Y. Iwasaki, K. Hasebe, A framework for generating playstyles of game ai with clustering of play logs, in: ICAART, vol. 3, 2022, pp. 605–612.
[26] J.-L. Hsieh, C.-T. Sun, Building a player strategy model by analyzing replays of real-time strategy games, in: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 3106–3111.
[27] A. Kale, X. Ma, Provenance in earth ai, artificial intelligence in Earth, Science (2023) 357–378.
[28] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, 2015, pp. 448–456.
[29] A. Cohen, E. Teng, V.-P. Berges, R.-P. Dong, H. Henry, M. Mattar, A. Zook, S. Ganguly, On the use and misuse of absorbing states in multi-agent reinforcement learning, arXiv preprint arXiv:2111.05992 (2021).