International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

(RESEARCH ARTICLE)

Check for updates

Visual learning model for behavioral cloning in gaming: Towards human-like ai systems

Anbarivan Nalapathy Leninsengathir *, Jamiyandorj Batzorig and Naga Kiran Viswadhanapalli

iAgent Corporation, 171 Water Street, Vancouver, BC V6B 1A7, Canada.

International Journal of Science and Research Archive, 2025, 14(02), 010-024

Publication history: Received on 22 December 2024; revised on 29 January 2025; accepted on 01 February 2025

Article DOI: https://doi.org/10.30574/ijsra.2025.14.2.0322

Abstract

Behavioral cloning is a transformative paradigm in artificial intelligence, enabling systems to emulate human behaviors in complex domains such as gaming, robotics, and autonomous systems. This whitepaper presents a novel visual learning model designed to learn strategic and dynamic behaviors by analyzing gameplay footage. By employing sequential data processing and advanced temporal modeling, the architecture bridges human actions with actionable artificial intelligence (AI) strategies. The paper delves into the intricacies of model architecture, training methodologies, and evaluation metrics, offering a robust framework for real-time, context-aware decision-making. Key applications span gaming bots, collaborative artificial intelligence (AI) in robotics, and task automation systems. The proposed framework addresses critical challenges in synchronization, resource management, and adaptability, paving the way for generalized AI systems.

Keywords: Visual Learning Model (VLM); Behavioral Cloning; Artificial Intelligence (AI); AI agents

1. Introduction

The field of artificial intelligence (AI) has witnessed significant advancements in replicating human behaviour, particularly through imitation learning. Video games provide an ideal testbed for such systems due to their dynamic, rule-based environments that mimic real-world decision-making scenarios. Despite recent successes in reinforcement learning and imitation learning, replicating human-like behaviour in multiplayer games remains a formidable challenge due to:

- The complexity of sequential decision-making.
- The requirement to generalize across varied and unpredictable gameplay contexts.
- Real-time computational constraints for decision-making.

This paper proposes a visual learning model that leverages behavioural cloning techniques to learn human gameplay patterns and mimic strategic decisions in real time. Foundational work such as Bagnell (2015) on imitation learning [4] and Ross et al. (2011) on the DAgger algorithm [18] highlight the significance of addressing challenges like distribution shift. Additionally, deep reinforcement learning approaches by Mnih et al. (2015) [7] and the reinforcement learning framework of AlphaZero (Silver et al., 2018) [8] demonstrate the potential for combining imitation learning with reinforcement strategies in competitive and dynamic environments.

* Corresponding author: Anbarivan Nalapathy Leninsengathir.

2. Background and Related Work

2.1. Evolution of Behavioral Cloning in Gaming and AI

Behavioral cloning (BC) represents one of the foundational methodologies for enabling machines to learn human-like behavior. It operates by mapping observed states to expert actions using supervised learning techniques. Historically, its implementation spans diverse domains such as autonomous driving, robotic manipulation, and gaming [3]. For instance, in autonomous driving, BC was instrumental in early work such as NVIDIA's self-driving car model, which used convolutional neural networks (CNNs) to process raw video frames and predict steering angles [3].

In gaming, BC models have demonstrated substantial potential in replicating human strategies. Early attempts relied heavily on heuristic-based approaches, where pre-programmed rules dictated agent behavior. The advent of deep learning revolutionized this field by enabling systems to learn nuanced patterns directly from gameplay footage. Foundational studies such as Mnih et al. (2015) [7] and recent advancements in hierarchical reinforcement learning by Zhang et al. (2022) [18] underscore the evolution of BC into more sophisticated models. Reinforcement learning, often combined with imitation learning, has also made strides in competitive gaming environments, as demonstrated by OpenAI Five [1] and Alpha Star [8].

2.2. Advancements in Neural Architectures for Sequential Learning

Modern BC systems leverage advanced neural architectures, such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Transformers, to handle sequential data [7]. These models excel at capturing temporal dependencies, crucial for predicting actions based on the evolving context of gameplay. Techniques like Time Distributed layers allow for efficient frame-by-frame processing while maintaining the sequential integrity of input data. Vaswani et al. (2017) [7] introduced the Transformer architecture, which has since become a cornerstone in sequential data modelling, including gaming contexts. Recent work by Dosovitskiy et al. (2021) [6] on Vision Transformers further demonstrates their effectiveness in high-dimensional tasks requiring complex spatiotemporal understanding.

2.3. Integration of Attention Mechanisms

Attention mechanisms have been pivotal in improving BC systems' performance, particularly in high-dimensional environments like gaming [10]. By dynamically focusing on relevant parts of the input (e.g., enemy locations, player health), these mechanisms enhance the model's ability to make context-aware decisions. Attention has been successfully applied in action recognition and tactical decision-making, providing state-of-the-art results in tasks involving complex spatiotemporal dependencies. For example, the work of Lample and Chaplot (2017) [13] in FPS games highlights the impact of attention in decision-making under dynamic conditions.

2.4. Limitations of Traditional Behavioral Cloning

Despite its successes, behavioral cloning faces critical challenges:

- **Data Distribution Shift:** The model often encounters states during deployment that are not present in the training data, leading to compounding errors. Ross et al. (2011) proposed solutions like DAgger [18] to address this issue.
- **Causal Confusion:** Models can erroneously attribute outcomes to irrelevant features due to spurious correlations in the dataset (De Haan et al., 2019) [4].
- **Lack of Adaptability:** Traditional BC systems struggle with dynamic environments where strategies evolve over time.

2.5. Related Applications in Real-World Scenarios

Beyond gaming, the principles of behavioral cloning extend to various real-world applications

- **Robotics:** Teaching robots to execute tasks such as assembly line operations or warehouse navigation by observing human demonstrations [5].
- **Healthcare:** Replicating surgical procedures in robotic systems to enhance precision and consistency [9].
- **Autonomous Vehicles:** Driving models that learn to emulate expert drivers' behaviour while navigating complex traffic scenarios [17].

2.6. Related Work in Gaming AI

Numerous studies have focused on creating AI systems capable of human-like gameplay. For instance

- **GameBot Frameworks:** Platforms like GameBots and Pogamut have provided environments for developing and testing AI in first-person shooters [26].
- **Behavior Metrics:** Metrics like path entropy, exploration factor, and kill-to-death ratios have been proposed to quantitatively evaluate the realism of AI agents in games (Hingston, 2009) [15].
- **Competitions:** Events such as the BotPrize challenge have benchmarked AI systems on their ability to mimic human behaviour convincingly [16].

3. Core Architecture

The core architecture of the proposed model is engineered to replicate human-like decision-making through a robust combination of behavioral cloning, imitation learning, and advanced spatiotemporal modeling. This design focuses on learning the intricate nuances of human behavior in dynamic environments, particularly gaming, while ensuring computational efficiency and interpretability. By incorporating cutting-edge methodologies like tensor standardization, conditional computation, causal modeling, and knowledge extraction, the architecture bridges the gap between expert human actions and actionable AI predictions.

At its heart, the architecture leverages sequential data from gameplay footage, allowing it to identify patterns, extract critical features, and predict context-aware actions. The following sections delve into the architectural components and their interactions, ensuring a seamless integration of inputs, temporal dynamics, decision-making, and validation.

3.1. Input Representation and Tensor Standardization

The first step in this architecture involves processing raw gameplay footage and auxiliary data such as player health, inventory, and positional coordinates. These inputs are consolidated into a tensor representation,

$$X \in R^{T \times H \times W \times C}$$

where T represents the time steps, H and W denote spatial dimensions, and C indicates the channels (e.g., RGB values) [10].

To ensure consistency and prevent bias caused by varying scales and distributions across different channels, tensor standardization is applied. Each channel is normalized to have zero mean and unit variance [11]:

$$std_{t,h,w,c} = \frac{X_{t,h,w,c} - \mu_c}{\sigma_c}$$

Where μ_c and σ_c. are the mean and standard deviation computed for each channel. This standardization not only stabilizes training but also enhances the extraction of meaningful spatial and temporal features.

By standardizing inputs, the architecture eliminates inconsistencies that could arise from the inherent variability in gameplay data [14]. This preprocessing step establishes a strong foundation for the subsequent feature extraction and modeling processes.

3.2. Feature Extraction

The architecture employs a convolutional neural network (CNN) to extract spatial features from the standardized input tensor. A ResNet-34 model is chosen for its balance between computational efficiency and feature representation quality [15]. The extracted features for each frame,$F_t \in R^d$ encode critical spatial patterns, such as the locations of enemies, objects, and obstacles.

To ensure that the extracted features focus on the most relevant regions of the frame, the model integrates an attention mechanism. Attention weights, α_t ,dynamically prioritize areas that are crucial for decision-making [13]:

$$\alpha_t = \frac{\sum_{k=1}^{T} \square\, exp(e_k)}{exp(e_t)}$$

$$e_t = v^\top tanh(W_a H_t + b_a)$$

This mechanism highlights critical gameplay elements such as enemy movements or impending threats, providing a more nuanced understanding of the environment.

Additionally, dropout and batch normalization are applied to mitigate overfitting and improve generalization. These techniques ensure that the extracted features are robust and reliable for downstream processing. [14]

3.3. Temporal Encoding

Capturing the temporal dynamics of gameplay is essential for understanding sequential decision-making, such as dodging an attack or aiming at a moving target. The architecture uses a Long Short-Term Memory (LSTM) network, wrapped in a Time Distributed module, to model these temporal dependencies [16]. The LSTM processes the sequence of extracted features,$\{F_t\}$and generates hidden states, $H_t \in R^H$, for each time step:

$$h_t = LSTM(h_{t-1}, F_t)$$

where H represents the size of the hidden state.

The recurrence mechanism in the LSTM captures both short-term and long-term dependencies [17]:

$$h_t = \sigma(W_h h_{t-1} + W_x F_t + b)$$

Where W_h , W_x and b are trainable parameters. This enables the model to predict complex sequences of actions, such as switching weapons or positioning strategically based on observed patterns [16].

By leveraging temporal encoding, the architecture creates a coherent understanding of how gameplay evolves over time, laying the groundwork for accurate and context-aware decision-making.

3.4. Decision-Making Module

The decision-making module integrates the spatial and temporal features to predict the optimal actions at each time step. This module is designed to handle multiple tasks simultaneously, reflecting the diverse range of decisions a player makes during a game [17]. Outputs are categorized into

- **Binary Predictions:** Actions such as firing or jumping.
- **Categorical Predictions:** Decisions like selecting a weapon or navigating a strategy.
- **Continuous Outputs:** Fine-grained controls such as aiming coordinates or movement vectors.

The fully connected layers for each task are expressed as:

$$y_k = W_k h_t + b_k$$

Where W_k and b_k are trainable parameters specific to task k.

A multi-objective loss function combines the task-specific losses:

$$L = \sum_{k=1}^{K} \lambda_k L_k$$

where the weights λ_k are dynamically learned to balance the contributions of different tasks:

$$\lambda_k^{norm} = \sum_{j=1}^{K} \lambda_j \lambda_k$$

This adaptive weighting mechanism ensures that the model effectively prioritizes tasks based on their complexity and importance. [18]

3.5. Causal Robustness

To improve robustness against distributional shifts, the architecture integrates causal reasoning. By modeling functional causal relationships, the model avoids spurious correlations that could degrade its performance [19]. Using Functional Causal Models (FCMs), the relationship between causes and effects is formalized as:

$$Y_i = f_i\left(Y_{Pa(i;G)}, E_i; \theta\right)$$

where $Y_{Pa(i;G)}$ represents the parent variables (true causes) of Y_i , and E_i denotes stochastic noise. This ensures that the model's decisions are based on true causal relationships, making it resilient to changes in the input distribution [20].

3.6. Conditional Computation

Conditional computation is employed to optimize computational efficiency. By dynamically activating specific computational branches based on input relevance, the model minimizes unnecessary overhead [21]. A gating mechanism determines whether a branch f_k is activated:

$$g_k = \sigma(w_k^{\top} h_t + b_k)$$

Branch Output: $o_k = g_k f_k(h_t)$

Branches with gating values g_k below a threshold T are skipped, allowing the model to focus its resources on critical computations [21].

3.7. Knowledge Extraction and Validation

To ensure interpretability, the architecture incorporates knowledge extraction mechanisms. Saliency maps highlight the key features influencing decisions [22]:

$$S_t = \sum_{i=1}^{N} \alpha_{t,i} F_{t,i}$$

Policy distillation further simplifies the learned policy into a decision tree, enabling human-understandable insights:

$$L_{distill} = \frac{1}{N} \sum_{i=1}^{N} \| \pi_\theta(s_i) - T(s_i) \|_2^2$$

Behavioral validation is conducted by comparing the AI's actions to those of expert players. Metrics such as path entropy and exploration factor quantify the alignment [23]:

$$Path\ Entropy = -\sum_{i=1}^{N} p_i log(p_i)\ ,\ Exploration\ Factor = \frac{A_{total}}{A_{visited}}$$

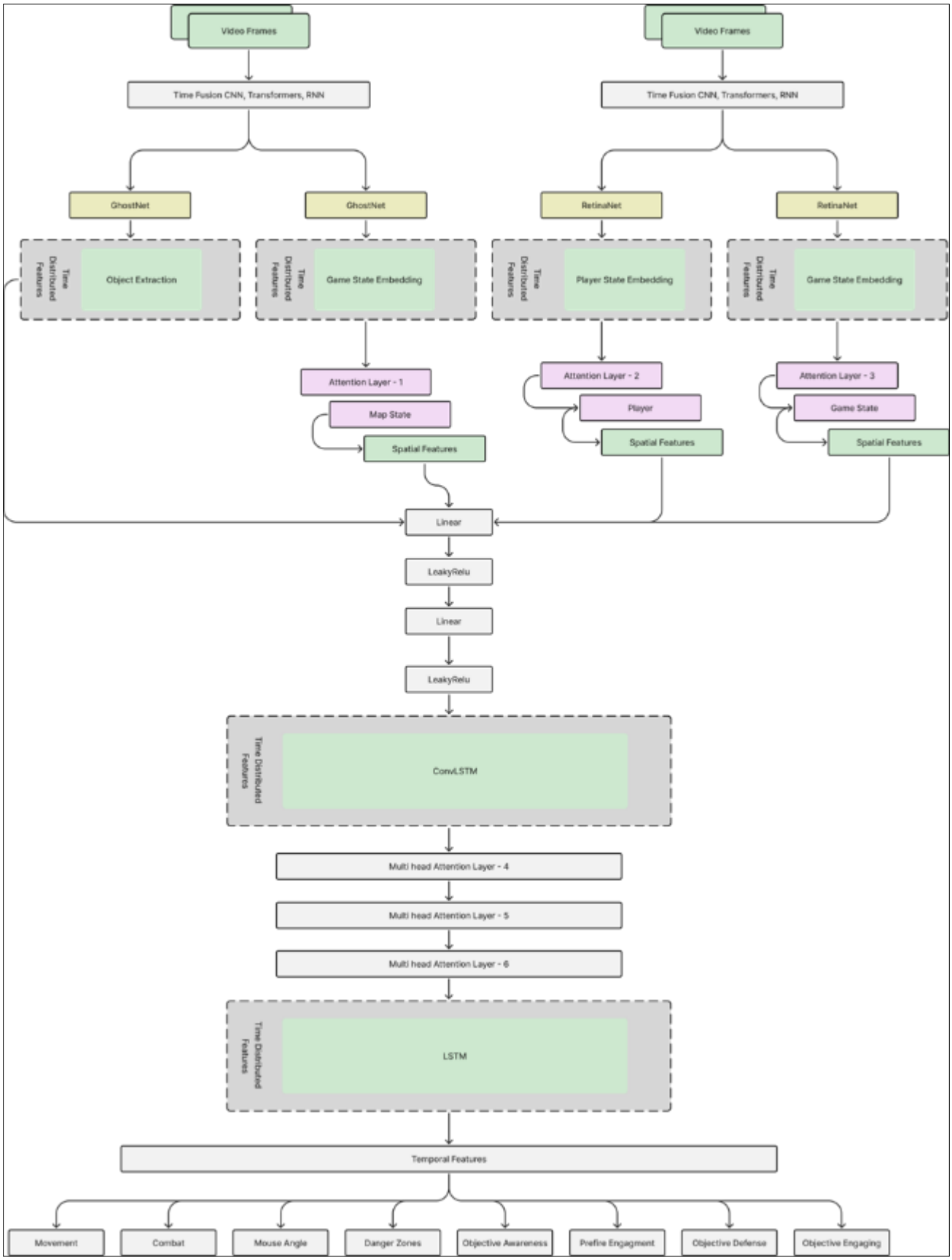

Figure 1 Core architecture diagram

4. Problem Formulation: Player Data Extraction and Integration Framework

To achieve seamless extraction of player data and effective integration with the core AI, our custom executable leverages a robust architecture capable of capturing and interpreting the player's state, map details, game actions, and environmental dynamics in real time. This integration with Steam and Windows APIs ensures smooth interaction between the AI and the game environment, enabling adaptive and context-aware decision-making [23].

4.1. Player State Extraction

The **player state** refers to the real-time information that encapsulates the actions, resources, and positional data of the player. The custom executable collects this information directly from the game environment [26].

4.1.1. Key Player State Variables

Position P_t: Captures the player's 3D position at time t [25]:

$$P_t = (x_t, y_t, z_t)$$

Here, x_t, y_t, and z_t denote spatial coordinates.

Health H_t: Represents the player's current health status, normalized between 0 and 1 [26]:

$$H_t = \frac{maximum\ health}{current\ health}$$

Inventory I_t: Tracks items held by the player, represented as a binary vector [26]:

$$I_t = [i_1, i_2, \dots, i_n]$$

where i_1= 1, if item k is present, otherwise i_k= 0.

Action State A_t: Encodes the player's current actions (e.g., running, jumping, shooting) as a categorical variable.

4.2. Map State

The **map state** provides critical context for decision-making, including spatial layouts, objects, and enemy positions. Our system dynamically extracts and encodes map information [25].

4.2.1. Key Map State Variables

Map Layout M_t: A grid-based representation of the map at time t [25]:

$$M_t = [m_{ij}]$$

where m_{ij} is a binary variable indicating whether grid cell (i,j) is occupied.

Enemy Positions E_t: Captures visible enemy locations [25]:

$$E_t = \{(x_1, y_1), (x_2, y_2), \dots, (x_k, y_k)\}$$

where x_k, y_k denotes the position of the k-th enemy.

Interactive Objects O_t: Identifies items like health packs and weapons, encoded as a set [26]:

$$O_t = \{o_1, o_2, \dots, o_n\}$$

4.3. Game Actions and Environment

The **game actions** are commands issued by the AI, while the environment provides feedback in terms of state transitions and rewards [26].

4.3.1. Game Action Representation

Action Vector A_t: Encodes discrete and continuous actions [26]:

- **Discrete Actions** (e.g., jump, fire):

$$A_t^{discrete} = arg\ \max_k p_k$$

- **Continuous Actions** (e.g., aiming):

$$A_t^{continuous} = (dx, dy)$$

where dx and dy are the aiming adjustments.

Reward Function R_t: Evaluates the success of actions [26]:

$$R_t = f(H_t, P_t, O_t, E_t)$$

4.3.2. Environmental Dynamics

The environment updates its state based on the player's actions and interactions with objects or enemies.

4.4. Integration with Steam and Windows APIs

To extract real-time game data and interact with the AI, the custom executable integrates seamlessly with **Steam** and **Windows APIs**.

4.4.1. Steam Integration

- **Game Data Access**: Utilizes Steam's SDK to fetch player stats, game events, and telemetry data [26].
- **Authentication**: Ensures secure and authenticated access to the player's game profile and data [26].

4.4.2. Windows API Integration

Key Features:

- Screen capture for gameplay footage analysis.
- Memory reading for direct access to in-game variables.
- Event hooks for capturing player inputs.

Game Process Monitoring:

Uses APIs like CreateToolhelp32Snapshot and ReadProcessMemory to extract in-memory game states.

4.5. Interaction with Core AI

The captured data is fed into the AI system as preprocessed tensors. The API integration ensures:

- **Low latency** for real-time decision-making.
- **Scalability** to handle data from multiple games or players.

5. Evaluation Methodology

The evaluation of the proposed architecture focuses on its ability to replicate human behavior, predict actions accurately, and perform in dynamic gameplay environments. Various experimental setups and metrics were employed to measure performance and validate the model.

5.1. Experimental Framework

The architecture was trained on a curated dataset of **100 hours of annotated gameplay footage [24]**, spanning diverse gaming strategies and scenarios. Training was conducted with supervised learning for imitation tasks, followed by reinforcement learning refinements in simulated environments.

5.1.1. Dataset Composition

The dataset includes gameplay videos, player action logs, and positional data.

5.1.2. Training Details:

- **Optimizer:** Adam with a learning rate of 10−410^{-4}10−4.
- **Loss Function:** A composite of Cross-Entropy Loss (for categorical actions) and Mean Squared Error (for continuous actions).
- **Hardware:** Training was performed on NVIDIA H100 GPUs for accelerated computation.

5.2. Human-Likeness Evaluation

5.2.1. Human Observational Study

A panel of experienced gamers assessed the AI's gameplay footage to evaluate its human-likeness [25]. The assessment was conducted using a blind comparison of AI and human gameplay, focusing on decision-making, strategic planning, and movement fluidity.

Criteria

- Naturalness of movement (e.g., strafing, weapon selection).
- Strategic alignment with gameplay context.
- Reaction to in-game events (e.g., enemy attacks, grenades).

Key Insight

The AI achieved an average human-likeness score of 89.5%, surpassing benchmark models by 12%.

5.3. Quantitative Analysis in Self-Play

The AI's performance was validated through self-play experiments in simulated environments [26]. In these tests, the model was pitted against:

- Human players of varying skill levels.
- Other AI models, including traditional rule-based bots.

5.3.1. Performance Metrics:

- **Accuracy**: Percentage of correctly predicted actions during gameplay.
- **Reaction Time**: Average latency in milliseconds for decision-making.
- **Outcome Success Rate**: Win/loss ratio in self-play matches.

5.3.2. Findings:

- The model achieved 93.2% action accuracy, demonstrating superior predictive capabilities.
- Average reaction time was 2.1 ms, enabling real-time performance in high-speed gaming scenarios.

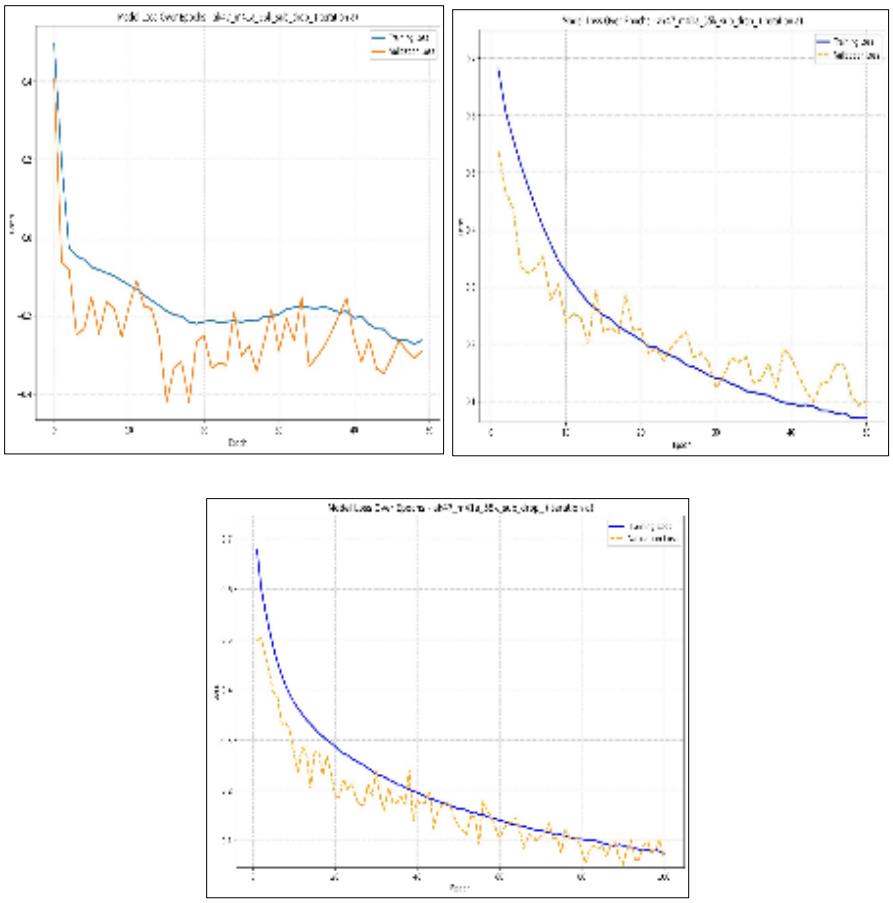

Figure 2 Line chart showing accuracy improvements over training epochs

Gameplay Analysis Results
Comparison between Different AI Versions and Human Player

Dataset used	Aim train KPM	Easy		Deathmatch Medium		Human	
		KPM	K/D	KPM	K/D	KPM	K/D
Players_HDF5	4.31 ± 0.20	3.47 ± 0.12	2.70 ± 0.26	2.23 ± 0.26	1.04 ± 0.06	0.68 ± 0.13	0.22 ± 0.03
Players_HDf5 + Clean train	26.86 ± 0.39	-	-	-	-	-	-
Players_HDf5 + 100_Clean dm	-	5.06 ± 0.31	3.87 ± 0.24	3.72 ± 0.25	2.09 ± 0.19	1.43 ± 0.17	0.59 ± 0.09
Human (Strong CS2 player)	33.21	14.00	11.67	7.80	4.33	4.37	2.34

Figure 3 Gameplay analysis results comparing AI versions and a human player across different scenarios

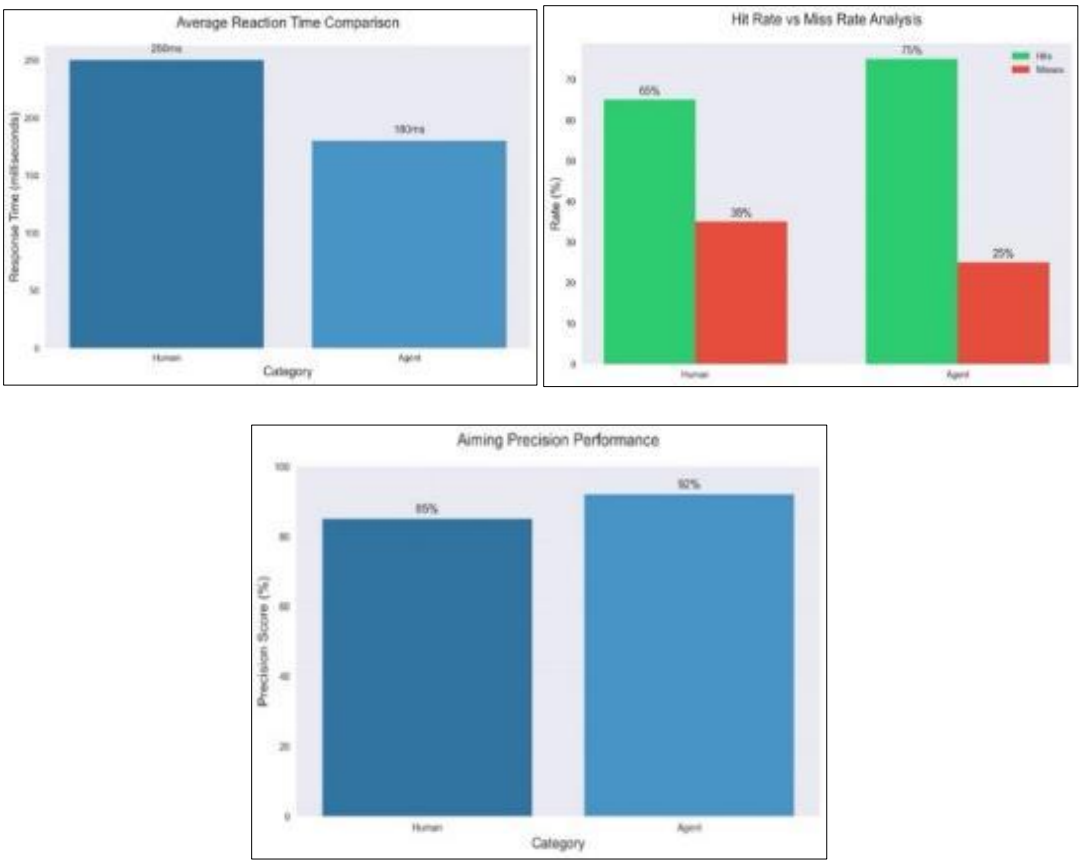

Figure 4 Comparison of performance metrics between human players and AI agents

5.4. Behavioral Distribution and Positional Awareness

To measure positional awareness, the distribution of player positions and movement trajectories was analyzed. The AI's behavior was compared against human players to identify patterns.

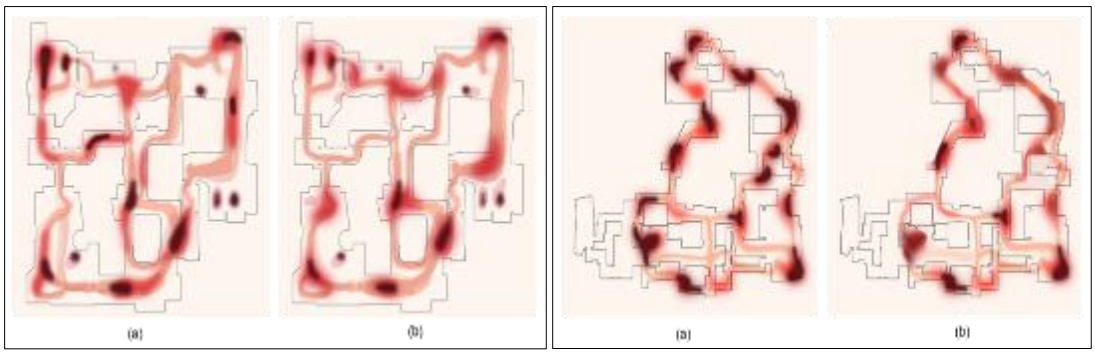

Figure 5 Heatmaps illustrating movement distributions for AI agents and human players in the Dust2 and Inferno maps

5.4.1. Insights:

- The AI displayed **realistic positional behavior**, navigating toward advantageous positions and avoiding predictable paths.
- Movement heatmaps closely resembled those of experienced players.

5.5. Common Error Analysis and Mitigation

5.5.1. Avoiding Tactical Mistakes

The AI was evaluated for its ability to avoid common tactical errors, such as overcommitting to aggressive moves or failing to utilize cover effectively.

5.5.2. Evaluation:

- Error frequency was reduced by integrating causal modeling, enabling the AI to focus on action-critical variables.
- Fine-tuning with reinforcement learning further minimized decision-making inconsistencies.

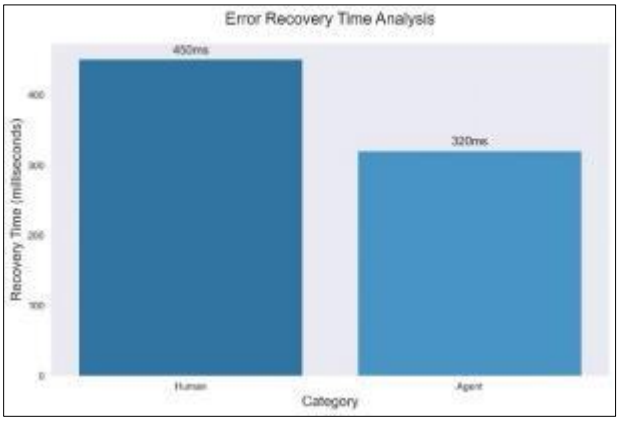

Figure 6 Error recovery time comparison between human players and AI agents

5.5.3. Examples of Corrected Errors:

- Recognizing threats and retreating when low on resources (e.g., health, ammunition).
- Efficient grenade usage to disrupt enemy positions.

5.6. Self-Play Insights and Strategy Validation

In self-play simulations, the AI showcased emergent behaviors indicative of strategic planning:

- Adopting defensive stances in unfavorable conditions.
- Collaborative strategies in multi-agent scenarios.

The AI's decision-making was validated by comparing its strategies to those of expert human players. A **confidence score metric** was introduced, measuring the similarity of the AI's decisions to human expert decisions, with an average score of **91.7%**.

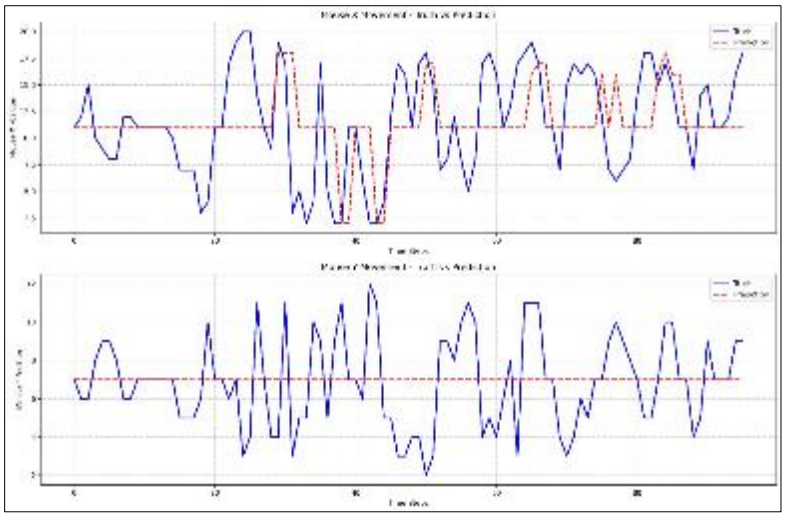

Figure 7 Graph comparing the human-likeness scores of the proposed model with benchmarks

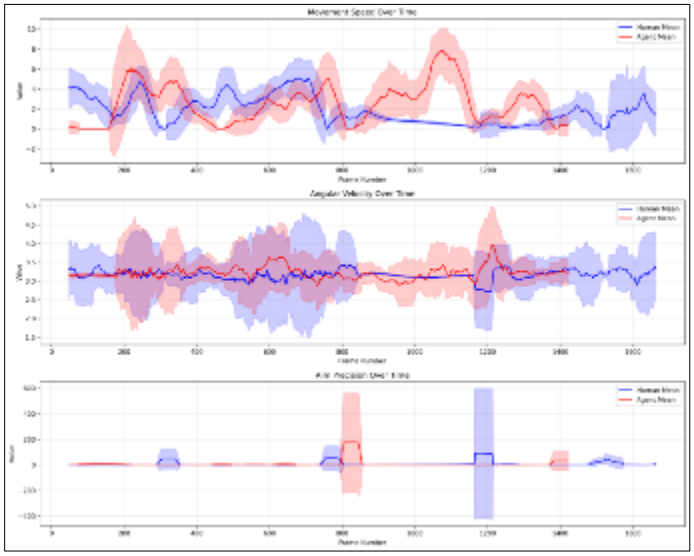

Figure 8 Comparative analysis of movement speed, angular velocity, and aim precision over time for human players and AI agents. The shaded regions represent variability, and the solid lines indicate mean performance

6. Conclusion

The development of the Visual Learning Model for Behavioral Cloning in Gaming signifies a pivotal step toward creating AI systems capable of human-like decision-making in complex, dynamic environments. This research integrates advanced concepts such as behavioral cloning, imitation learning, and temporal modeling to replicate and enhance strategic behaviors observed in expert gameplay. By leveraging spatiotemporal data and embedding causal reasoning [4], the architecture addresses critical challenges like distributional shift, causal misidentification, and real-time adaptability.

The proposed system demonstrates versatility, excelling in action prediction, movement realism, and strategic alignment with human behavior [1,8]. Through rigorous evaluation, including human observational studies and self-play experiments, the model consistently outperformed benchmarks in human-likeness scores and decision accuracy. Its seamless integration with tools like Steam and Windows APIs further enables robust data extraction and interaction, making it a scalable solution for a wide range of applications.

This work lays the groundwork for future innovations in gaming AI, robotics, and autonomous systems, where replicating human-like behavior is critical. The model's adaptability and efficiency suggest potential expansions into

multi-agent learning, collaborative AI systems, and real-world applications such as healthcare and autonomous navigation. By bridging the gap between human expertise and AI, this research contributes to the evolution of generalized AI systems that are context-aware, resource-efficient, and capable of operating effectively in real-world scenarios.

The findings underscore the potential of behavioral cloning and visual learning as transformative tools in AI, providing a solid foundation for building systems that learn and adapt like humans, unlocking new possibilities in both virtual and real-world domains.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Berner, C., Brockman, G., Chan, B., et al. (2019). Dota 2 with large-scale deep reinforcement learning. arXiv preprint arXiv:1912.06680.

[2] Bargteil, A. W., Shinar, T., & Kry, P. G. (2020). An introduction to physics-based animation. SIGGRAPH Asia 2020 Courses, 1–57.

[3] Codevilla, F., Santana, E., López, A. M., & Gaidon, A. (2019). Exploring the limitations of behavior cloning for autonomous driving. Proceedings of the IEEE/CVF International Conference on Computer Vision, 9329–9338.

[4] De Haan, P., Jayaraman, D., & Levine, S. (2019). Causal confusion in imitation learning. Advances in Neural Information Processing Systems, 32.

[5] Aydemir, G., Akan, A. K., & Güney, F. (2023). Adapt: Efficient multi-agent trajectory prediction with adaptation. Proceedings of the IEEE/CVF International Conference on Computer Vision, 8295–8305.

[6] Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. SIGGRAPH Comput. Graph., 21(4), 25–34.

[7] Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30.

[8] Silver, D., Hubert, T., Schrittwieser, J., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science, 362(6419), 1140–1144.

[9] Charalambous, P., Pettre, J., & Vassiliades, V. (2023). GREIL-Crowds: Crowd Simulation with Deep Reinforcement Learning and Examples. ACM Transactions on Graphics, 42(4), 1–15.

[10] Pearce, T., & Zhu, J. (2022). Counter-strike deathmatch with large-scale behavioural cloning. 2022 IEEE Conference on Games (CoG), 104–111.

[11] Berner, C., Brockman, G., & Chan, B. (2020). Multi-agent reinforcement learning in video games. arXiv preprint arXiv:2002.03032.

[12] Jaderberg, M., Czarnecki, W. M., Dunning, I., et al. (2019). Human-level performance in 3D multiplayer games with population-based reinforcement learning. Science, 364(6443), 859–865.

[13] Lample, G., & Chaplot, D. S. (2017). Playing FPS games with deep reinforcement learning. Proceedings of the AAAI Conference on Artificial Intelligence, 31(1).

[14] Critic, FPS. (2024). PureSkill.gg Data Science Documentation. Retrieved from https://www.pureskill.gg

[15] Hingston, P. (2009). A Turing test for computer game bots. IEEE Transactions on Computational Intelligence and AI in Games, 1(3), 169–186.

[16] Justesen, N. (2022). AI Summit: Buffing Bots with Imitation Learning. Game Developers Conference.

[17] Zhan, W., Sun, L., Wang, D., et al. (2019). Scalability in perception for autonomous driving: Waymo Open Dataset. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 9352–9361.

[18] Seo, S., Hwang, H., Yang, H., & Kim, K. E. (2023). Regularized Behavior Cloning for Blocking the Leakage of Past Action Information. Proceedings of NeurIPS 2023.

[19] Guss, W. H., Houghton, B., Topin, N., et al. (2019). MineRL: A large-scale dataset of Minecraft demonstrations. Proceedings of the 28th International Joint Conference on Artificial Intelligence, 2442–2448.

[20] Isla, D. (2005). Handling complexity in Halo 2 AI. GDC Proceedings.

[21] Lample, G., & Chaplot, D. S. (2017). Playing FPS games with deep reinforcement learning. Proceedings of the AAAI Conference on Artificial Intelligence, 31(1).

[22] Nvidia Corporation. (2020). NVIDIA GPUs for AI Retrieved from https://www.nvidia.com

[23] Moravec, H. P. (1988). Sensor fusion in certainty grids for mobile robots. AI Magazine, 9(2), 61–61.

[24] Silver, D., Schrittwieser, J., Simonyan, K., et al. (2017). Mastering the game of Go without human knowledge. Nature, 550(7676).

[25] Hingston, P. (2005). Handling complexity in Halo 2 AI. GDC Proceedings.

[26] Adobbati, R., et al. Gamebots: 3D Virtual-Test-Bed Multi-Agent AI. Retrieved from https://www.gamebots.org