Humanlike Behavior in a Third-Person Shooter with Imitation Learning

Alexander R. Farhang*†, Brendan Mulcahy†, Daniel Holden†, Iain Matthews† and Yisong Yue*
*Caltech, Pasadena, CA, USA, †Epic Games, Cary, NC, USA

***Abstract*—We tackle the problem of generating humanlike bot behavior by learning from human demonstrations. We developed a controlled gym environment to collect data on a subset of human behavior—namely aiming and target acquisition in single opponent settings. We introduce an identity-conditioned causal transformer to produce humanlike behavior of a controllable quality on a per-frame basis that captures the differences in skill and style between conditioned players.**

***Index Terms*—imitation learning, sequence modeling, humanlike behavior**

Video: alexfarhang.github.io/humanlikebehavior

I. Introduction

Non-player characters (NPCs) are a crucial component in many modern games, from single-player role-playing games to large multiplayer combat oriented games. Highly engaging NPCs, or bots, can bring life to a game and increase a player's immersion through competition or collaboration. Common approaches to developing game artificial intelligence (AI) often involve bespoke, finely crafted behavior trees to try to approximate human behavior—which is challenging to both specify and capture. These challenges are exacerbated when developing diverse NPC behaviors, such as those with variable personas or skills.

In this paper, we develop a controlled game environment to collect human gameplay for aiming in one-versus-one combat. We then use this data to train neural network models for humanlike behavior—specifically aiming and movement control. This controlled setup allows the capturing of human behavior for target acquisition and tracking with a known target, which would otherwise have to be heuristically chosen in a multi-target environment. With this approach, a relatively small amount of data (on the order of hours) is needed for models to begin to approximate human behavior.

Our agent model is based on a causal transformer neural network trained on human trajectories. The transformer network is autoregressive, and learns to reconstruct future actions based on previous state-action sequences. Furthermore, we add a novel type of identity-conditioning by prompting the transformer with a prepended learned token signifying a player's identity, modifying the quality of gameplay. This allows us to incorporate multiple skills or styles of gameplay in a single model, which enables batch processing of neural network controlled bots with differing characteristics. Our approach can replicate qualitative and quantitative aspects of humanlike behavior and capture elements of individual player style.

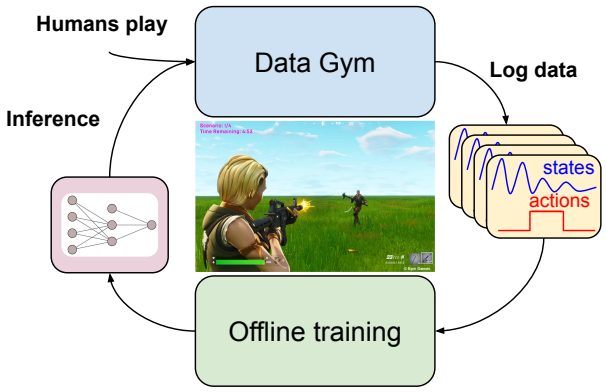

Fig. 1. Data collection and inference framework. Human demonstrations are logged, models are trained offline, then models are imported into the engine for inference.

II. Related Work

Deep learning has enabled considerable advances in the creation of agents to play modern games, including real-time strategy games like StarCraft and multiplayer online battle arenas like Dota 2 [1], [2]. The research community has also applied deep learning to first-person shooter gameplay using platforms built upon Quake 3 and Doom [3], [4]. Large-scale behavior cloning demonstrated success on the more recent first-person shooter, Counter-Strike; Global Offensive, from pixels, using unlabelled datasets with inverse dynamics models alongside smaller datasets of expert demonstrations [5]. Similar approaches, though with learned inverse dynamics models, have worked at even larger scale in Minecraft, achieving game milestones that can take skilled humans over 20 minutes of gameplay [6]. Other behavior cloning approaches synthesize language and gameplay to produce generalist agents with the ability to play multiple 3D games [7]. Adversarial imitation learning approaches have also been used to generate policies with multiple behavioral personas in racing and navigation environments [8]. In this work, we use imitation learning to generate controllable humanlike behavior of differing player styles and skills in an interactive environment involving competition with another dynamic agent.

III. Data

A. Data Gym

Our Data Gym is a custom one-versus-one combat map built using Fortnite, a popular, modern third-person shooter game

in Unreal Engine 5 (UE) [9]. The player engages in weapon-based combat with a bot (traditional behavior tree-based game AI) spawned near the player (Fig. 1). Following the standard control mechanics of third-person shooters, the player can navigate and aim with forward/backward and left/right translational controls and can rotate in yaw and pitch with mouse movements. The player also controls firing the weapon, aiming down sights, crouching, sprinting, and jumping.

The game consists of a series of matches versus the bot with the following structure.

1) The player is reset to a starting location and a 3 second countdown begins, during which the player's controls are locked.
2) After the countdown, a bot is spawned at a random location within a fixed distance of the player. The player's controls are unlocked, and the player attempts to acquire and eliminate the target.
3) Once the target or the player is eliminated (100 shield and 100 health points are lost), the process restarts.

Data was collected in 20 minute sessions from 5 players ranging from 1 to 2.5 hours each, totalling 8 hours or 800k frames. The UE Learning Agents plugin [9] enabled state-action trajectories to be logged directly from the game state. Learning Agents also supports runtime inference and direct game state control through the UE Neural Network Engine (NNE) plugin for agent evaluation [9].

B. Frame Rate

Handling variable frame rates is a common challenge in learning from and generating realistic gameplay. Moreover, the additional overhead of running neural network inference results in a lower frame rate, which can cause distribution shift from training data collected at faster frame rates. We found in initial experiments that encoding the instantaneous frame rate improved model performance. As the viable frame rate for in-engine neural network inference with our models was approximately 30 frames per second (fps), we found further improvements by requiring the fps of human data collection to stay between 20 and 40 fps.

C. State Space

A benefit of interfacing with the game engine is that game state information can be extracted directly, bypassing expensive pixel-based computations on rendered scenes. The Data Gym trajectories log state information of the player's own character as well as information about the opponent and game state. These "self" features include the player's rotation and whether a bullet was just fired. "Opponent" state features include pitch and yaw error from the player's rotation centered at the weapon to the pelvis of the opponent, as well as the opponent's velocity and distance.

D. Action Space

The action space included all ways that the player can control the playable character via a combination of key presses and mouse movements that have been processed by the game engine's input system. Aiming actions are the changes in character rotation (pitch and yaw) and movement actions are the inputs along the forward/backward and left/right axes. Crouching, jumping, weapon firing, and weapon aiming-down-sights are additional controlled actions. Because all data logging is integrated into the game engine, no inverse dynamics model is required to infer actions—the player's ground truth actions are recorded. Actions are treated as continuous outputs and are optimized with mean squared error during model training. In cases where discrete actions must be handled by the game engine, predicted actions are thresholded.

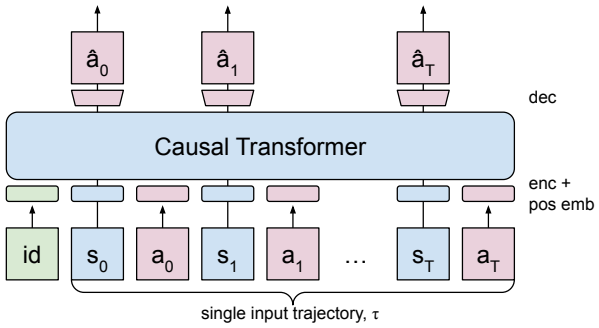

Fig. 2. Identity-conditioned causal transformer. To predict actions, the transformer processes a state-action history sequence (with an optional, prepended identity token).

IV. Agent

A. Neural Network Architecture

Transformers were introduced to efficiently model sequential data and consist of repeated blocks of attention [10]. Sequence tokens are linearly projected into Queries (Q), Keys (K), and Values (V); Values are weighted by the normalized (often scaled) similarities of tokens across the sequence (1).

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T)V \tag{1}$$

Decoder-only transformer models were subsequently proposed for generative pre-training (GPT) which introduced causal masking for batched autoregressive predictions [11]. Two approaches utilizing the GPT architecture for offline reinforcement learning include the Decision Transformer, which focused on reward conditioning, and the Trajectory Transformer which used beam-search-based planning [12], [13]. We build on the Decision Transformer due to its simplicity of use at inference time and reduced computational cost. This paper uses a causal transformer to process a trajectory, τ, of alternating embeddings of states and actions, with temporal length T,

$$\tau = (s_0, a_0, s_1, a_1, \ldots, s_T, a_T).$$

State and action features are separately encoded with linear layers then added to learned positional embeddings for each time step (Fig. 2). As the sequence has been split into states and actions, the context window is $2T$. The embedded sequence is processed by multiple stacked decoder-only transformer blocks using causal attention masking. Finally,

the action is predicted with a linear layer at the appropriate sequence token. This base transformer structure resembles a Decision Transformer without return inputs [12].

B. Identity Conditioning

In identity-conditioned experiments, an additional "id" token is prepended to the state-action sequence, to prompt the transformer with a trajectory identity when generating actions (Fig. 2). A shared learned embedding akin to a positional embedding is added to identity tokens. In this paper, id can refer to group membership or an individual player id. In the skill group-conditioned experiments, the id indicates which group of players the trajectory belongs to (high skill group or low skill group). In individual identity-conditioned experiments, the id is unique to each player. At inference, conditioning a trajectory is simply prompting the model with the desired id embedding.

C. Training and Inference Details

We utilized a 4 layer stacked decoder with 256 dimensional embeddings and 4 attention heads. Training took $1e5$ iterations with the Adam optimizer (learning rate: $1e{-}4$) and $1e4$ steps of linear warmup. Dropout was set to 0.1 and the attention window was over 60 frames (120 tokens due to the state-action split and 121 when including identity conditioning). 60 frame trajectory segments were sampled for training. Models were converted to ONNX and run in UE using the NNE plugin.

Because the weapon in the Data Gym was automatic, players typically held down the trigger once they acquired the target, resulting in relatively few trajectories exhibiting the transition from no-firing to firing. This also resulted in the model controlling firing in a highly autoregressive way, such that firing would be predicted mainly when there was firing in the previous 60 frames. To improve the model during inference, it was augmented with a simple in-engine rule-based controller that actuated additional firing when the character was aiming roughly at the target, effectively seeding the model for autoregressive prediction. As we are primarily concerned with humanlike rotational aiming, this was sufficient for our needs. We found performance on a held out validation set to be only weakly informative of the model's ultimate performance in engine rollouts, consistent with prior observations [7].

V. Results

Our results analyze two main questions: 1) do our trained models behave in a more humanlike way than the existing hand-crafted bots; and 2) whether we can train models to achieve variable behaviors (e.g., skill levels or personas).

A. Humanlike Behavior of the Unconditioned Model

We first analyze our learned base model, without any identity conditioning, trained on two highly skilled players—each with win rate over 97%. The goal of this analysis is to determine whether the basic version of our models can generate generic humanlike behavior by comparing our neural networks to humans and bots. Notably, the neural network

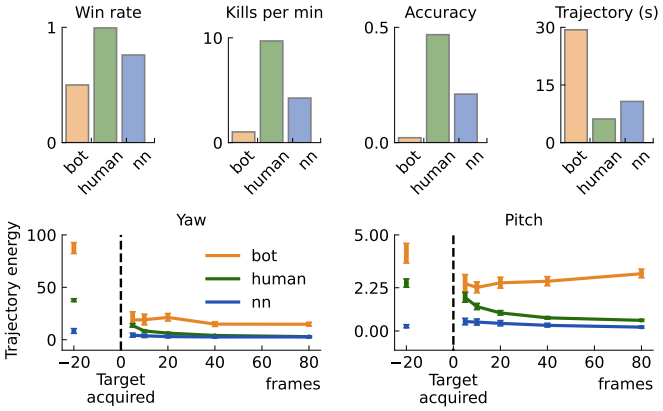

Fig. 3. Top: combat statistics of humans, learned agents (nn), and bots. Bottom: yaw and pitch trajectory energy comparisons for windows before and after target acquisition (with 95% confidence intervals).

model (nn) has a win rate surpassing that of the bot and weapon accuracy and match completion speeds more similar to humans (Fig. 3).

To quantitatively measure the aggregate structure of aiming before and after target acquisition, we use a metric similar to the average rotational energy over a time period t, where $r_{\hat{t}}$ is a rotational action at time $\hat{t}$. We show the energy in cascaded windows before and after initial target acquisition (when yaw error first becomes 0), with different windowing periods: 20 frames before acquisition and 5, 10, 20, 40, and 80 frames after acquisition (Fig. 3).

$$\text{Trajectory Energy}(t, r) = \frac{1}{t}\sum_{\hat{t}=0}^{t} r_{\hat{t}}^2 \tag{2}$$

The energies drop upon target acquisition for bots, humans, and neural networks, because large turns are no longer necessary to acquire a target. The neural networks exhibit energy levels more similar to those of humans, especially after acquisition when characters are actively tracking the target. Neural network energy levels are lower overall, perhaps due to underfitting.

When compared to bots qualitatively, the trained neural network models and humans exhibit smoother target acquisitions as well as fewer large jerky movements during target acquisition and tracking (Fig. 4, example video). The learned models are also robust to rotational and translational perturbations—

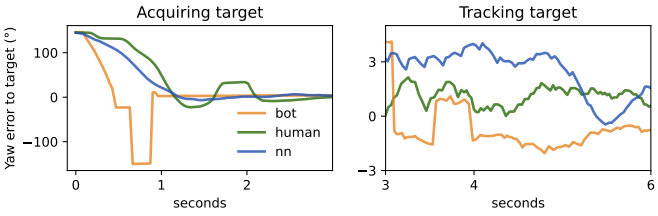

Fig. 4. Aiming behavior. Yaw error to the target for example trajectories of bots, humans, and neural networks playing the Data Gym. The first half of the trajectory shows the acquisition of the target (left), and the second half shows the tracking behavior (right).

they can recover from sudden, large (injected) turns away from the opponent. This ability was enhanced by initiating matches in the Data Gym with random pitch rotations; providing demonstrations on large pitch errors helped prevent the learned agents from becoming stuck looking at the ground or sky. Overall, these results demonstrate that, in the context of one-on-one combat, our model generates more humanlike behavior than our hand-crafted bot.

B. Skill Group Identity Conditioning

We next addressed whether our model can be prompted to produce behavior of a variable caliber. Players were split into two coarse groups separated by win rate: high skill group (win rate $\geq 90\%$) or low skill group (win rate $< 90\%$). An identity-conditioned transformer was trained with group membership identity labels. We find that prompting with skill group enables a single model to exhibit different qualities of behavior, a convenient trait when trying to minimize memory allocation to neural network weights in engine. We find differences in win rate and kills per minute corresponding to group identification—low skill group conditioning resulted in lower performance than high skill grouped conditioning (Fig. 5, left).

C. Individual Identity Conditioning

Finally, we explored whether identity conditioning enables the model to capture unique play styles. We trained an identity-conditioned model on individual player identities with 4 players. We quantify some low-level features of gameplay that vary drastically between players: jumping and crouching. Some players' play styles include frequent crouching or jumping and others never engage in the behaviors. The neural network can reproduce these behaviors per identity, though imperfectly—at lower rates and Player 3's crouching behavior is not replicated (Fig. 5, right). Identity conditioning on individual players picks up elements of their play styles, though can come at a cost of reduced skill replication for some identities, which may be improved with larger datasets.

D. Failure Modes

If the distance to the target becomes too large, the models can have difficulty accurately tracking the target. Players also prefer to engage in up-close combat. Additionally, sometimes low skill group-conditioned models or player identity-conditioned models actuate lower aiming skill by exhibiting offsets in aiming.

VI. Discussion and Conclusion

Our imitation learning approach captures quantitative and qualitative elements of humanlike aiming behavior with a relatively small data requirement. Additionally, our proposed identity-conditioning framework allows controllable generation of differing qualities and styles of play with a simple prompting mechanism. This approach to replicate aspects of a desired behavior, in this case humanlike aiming, can be synthesized with behavior-tree based AI such that the neural network aiming module can be recruited in particular contexts,

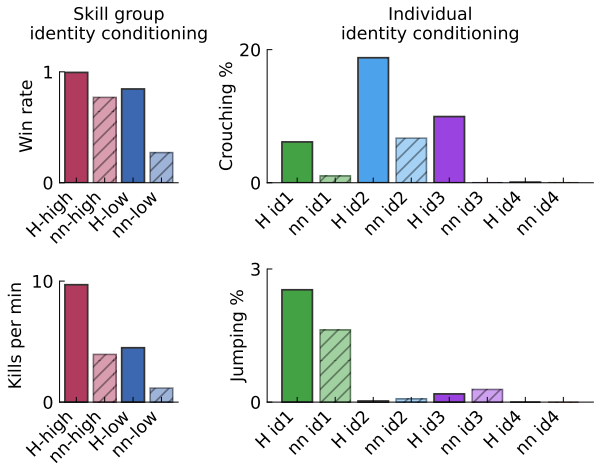

Fig. 5. Identity conditioning results with skill group-conditioned (left) and individual player-conditioned neural networks (right). Combat statistics for humans (solid) and neural networks (hatched).

like one-on-one engagements. This is particularly useful when the desired behavior is difficult to specify and hand craft, but demonstrations are straightforward to obtain.

Future directions could explore scaling up the amount and diversity of the data and complexity of the behavior. Instead of conditioning on simple skill group-based or player-level identities, large scale datasets of diverse play styles could be used to learn a useful latent space for a desired downstream task. While this paper focused on aiming, our approach can be applied to other modalities of humanlike gameplay, for example navigation, structure building, or multiplayer collaboration.

References

[1] O. Vinyals *et al.*, "Grandmaster level in Starcraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[2] OpenAI *et al.*, "Dota 2 with large scale deep reinforcement learning," 2019.

[3] C. Beattie *et al.*, "Deepmind lab," 2016.

[4] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "Vizdoom: A Doom-based AI research platform for visual reinforcement learning," in *IEEE CIG*, 2016.

[5] T. Pearce and J. Zhu, "Counter-strike deathmatch with large-scale behavioural cloning," in *IEEE Conference on Games*, 2022, pp. 104–111.

[6] B. Baker *et al.*, "Video pretraining (VPT): Learning to act by watching unlabeled online videos," 2022.

[7] S. Team *et al.*, "Scaling instructable agents across many simulated worlds," 2024.

[8] W. Ahlberg, A. Sestini, K. Tollmar, and L. Gisslén, "Generating personas for games with multimodal adversarial imitation learning," in *IEEE Conference on Games*, 2023.

[9] Unreal engine. [Online]. Available: https://www.unrealengine.com/

[10] A. Vaswani *et al.*, "Attention is all you need," in *NeurIPS*, vol. 30, 2017.

[11] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[12] L. Chen *et al.*, "Decision transformer: Reinforcement learning via sequence modeling," in *NeurIPS*, vol. 34, 2021, pp. 15 084–15 097.

[13] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," in *NeurIPS*, vol. 34, 2021, pp. 1273–1286.