From Demonstrations and Knowledge Engineering to a DNN Agent in a Modern Open-World Video Game

Igor Borovikov
EA Digital Platform – Data & AI
209 Redwood Shores Pkwy
Redwood City, CA 94065
iborovikov@ea.com

Ahmad Beirami
EA Digital Platform – Data & AI
209 Redwood Shores Pkwy
Redwood City, CA 94065
abeirami@ea.com

Abstract

In video games, there is a high demand for non-player characters (NPCs) whose behavior is believable and human-like. The traditional hand-crafted AI driving NPCs is hard to scale up in modern open-world multiplayer games, and often leads to the uncanny valley of robotic behavior. We discuss a novel approach to solving this problem based on imitation learning. We combine demonstrations, programmed rules, and bootstrapping in the game environment to train a Deep Neural Network (DNN) defining the NPC behavior. Unlike Reinforcement Learning (RL), where the objective is optimal performance, we aim to reproduce a human player style from few demonstrations. We embed the implicit knowledge of the basic gameplay rules which are hard to learn via self-play or infer from a few demonstrations but are straightforward to capture with simple programmed logic. We build a composite model that interacts with the game to bootstrap the human demonstrations to provide sufficient training data for a more complex DNN model capturing stylized gameplay from demonstrations and enhanced with rules. We show that the method is computationally fast and delivers promising results in a game production cycle.

Introduction and Problem Statement

The leading advances in the field of RL in application to playing computer games, e.g., (OpenAI Five 2018; Mnih et al. 2015; Vinyals et al. 2017; Harmer et al. 2018), strive to train an optimal artificial agent ("agent" for short) maximizing clearly defined rewards, and the game itself remains fixed for a foreseen future. In contrast to that, during the game development, the objectives and the settings are entirely different. The agents can play a variety of roles with the rewards that are hard to define formally, e.g., a goal of an agent exploring a game level is different from foraging, defeating all adversaries, or solving a puzzle. Also, the game environment is frequently changing between the game builds. In such settings, it is desirable to quickly train agents that can work as an NPC or as bots for the automated test and game balancing. Throwing computational resources and substantial engineering efforts at training agents in such conditions is not practical and calls for different approaches.

The preferred solution would utilize only a few relatively short episodes played by the developers. The time allowed for augmenting these demonstrations by autoplay would be limited, especially if the game engine doesn't support a dramatic speedup. Thus, the solution has to be sample-efficient and train the agents offline. Using frame buffer for training is problematic due to frequent changes of the game look during development. In contrary, the conceptual types of the game environment rarely change during its production. That allows for the engineering of a substantial part of the prior knowledge about the game states into a low dimensional feature vector replacing the frame buffer. The core gameplay also rarely changes which allows to describe desired behavior as a compact model depending on core game features.

In this position paper, we train a model of an NPC with stylistics traits provided by demonstrations. We include the aspects of behavior that are hard to infer from demonstrations as engineered rules. We use low-dimensional engineered features to provide similar information that is available in organic gameplay. Finally, we train a composite DNN model, which we show to be effective while inexpensive.

Proposed Approach

The methodology we explore assumes modest domain knowledge and an intuitive understanding of the game mechanics of a First Person Shooter (FPS). Such level of understanding is usually widely available to the game designers and software engineers working on the game. We aim at a high level of abstraction to avoid too elaborate or game-specific formalization of that knowledge. With that in mind, the three components we engineer are:

- state space (features),
- action space,
- rules capturing implicit human knowledge.

We complement the rules with explicit human demonstrations to build a Markov ensemble, which provides basic generalization. We use this aggregate model to drive an agent in the game to automate the generation of additional bootstrap data. The bootstrapped data set allows to train a composite DNN model combining the demonstrations and the engineered rules. As a case study, we explore an FPS game,

which is conceptually similar to the one investigated by (Harmer et al. 2018). Its core mechanics is generic enough to make our approach applicable to other games in the same or similar genre.

State Space (Features)

We instrument the game and expose its current state s to the agent as a low-dimensional normalized vector $s = (c_1, \ldots, c_n, d_1, \ldots, d_m)$. Its components, c and d, describe n continuous and m discrete features correspondingly. In an FPS game, the continuous features could include the distance to the adversary, the angle between target Line of Sight (LoS) and the player orientation, velocity components, ammo, and health. We map all continuous components to the range $|c_i| \in [0, 1]$. A mapping function for unbound components like distance could be $\arctan$ or any similarly behaving smooth function. For variables with naturally defined range, like ammo or health, we use linear normalization. The discrete components d may include binary values like the presence of an LoS, sprinting on-off, and one hot coded values of non-binary values, such as current animation type. By construction, all $d_i \in \{0, 1\}$.

The total number of features for a single adversary in our proof-of-concept implementation is quite small: $n \sim 20$ and $m \sim 20$, depending on the experiment. In particular, we explore only on-foot action and remain in the same game modality, e.g., we exclude getting into and driving vehicles. The locations and objects of interest like cover spots, medical packs and ammo clips, can appear in the state vector in different forms but we leave their detailed description to future more detailed publications, where we intend to explore more complex gameplay. Finally, we expose only the features that are observable in organic gameplay to achieve human-like gameplay.

Action Space

The agent's actions naturally map to the game controller input encoded with a few variables. Actions $a_1, \ldots, a_k$ comprise of k continuous and discrete values. There are six analog inputs from the controller: two sticks normalized to $[-1, 1]$ and two triggers normalized to $[0, 1]$ by design. Also there are a few binary inputs from buttons. While all possible combinations of the inputs may look intractable for learning, as stated in (Harmer et al. 2018), our goal of imitating a human player doesn't require to cover all possible combinations of inputs. Instead, we extract only those combinations that occur in organic gameplay and encode them as one hot, which drastically reduces action space complexity. The resulting dimensionality of the action space in our experiments turned out to be also in the low two digits $k \sim 15$, depending on the experiment.

The proposed approach to the game state-action space serves the two purposes: keep the action space dimensionality under control and eliminate actions that humans never take. Thus, the model we train is "fair" (i.e. is not using more inputs at the same time than allowed by human anatomy).

Capturing Gameplay Style with Markov Ensembles

Learning stylistic elements of a policy is challenging. One of the main difficulties comes from style evaluation, which is highly subjective and not easily quantifiable. While inverse reinforcement learning aims at finding reward functions that promote a certain style and behavior, we don't have sufficient data at hand to solve this ill-posed problem. Instead, we incorporate style directly from the demonstrations using imitation learning. Our demonstration data consists of per-frame records of the already described engineered states and the player actions encoded as the game controller inputs.

Notice that we inherently deal with a partially observable Markov decision process (POMDP) as not all the attributes in the state space are observable (Borovikov and Beirami). To support Markov model capturing player style, we include the history of N recent actions as part of the state space as well. The depth N of the history affects the expressiveness of style capture. In our experiments, we limit the depth to less than a second of gameplay ($N < 60$), which still preserves the visual style of a simple reactive policy.

To build the Markov ensemble, we use an approach to style reproduction inspired by natural language processing literature (see review (Zhai 2008)). We encode the demonstrations as symbolic sequences utilizing a hierarchy of multi-resolution quantization schemes ranging from detailed to a complete information loss for the continuous and the discrete channels. The most detailed quantization and higher order Markov models can reproduce sequences of human actions in similar situations with high accuracy, thus capturing gameplaying style. The coarsest level corresponds to a Markov agent blindly sampling actions from the demonstrations. The hierarchical ensemble of Markov models provides minimal generalization from the demonstration data. The ensemble of such models is straightforward to build, and the inference is a lookup process.

We intend to provide a complete description of Markov ensemble by publishing a preprint of our internal technical report (Borovikov and Harder 2018).

Capturing Implicit Human Knowledge with Embedded Rules

The outlined Markov ensemble can formally generalize to previously unobserved states. However, at the coarsest level of quantization, blind sampling from the observed actions easily breaks an illusion of intelligent goal-driven behavior. To address that issue, we trim the coarser levels of quantization from the ensemble, leaving some states unhandled by the resulting incomplete Markov policy. To handle such states, we augment the Markov ensemble with a small number of heuristics captured as rules. The rules provide a reasonable response to the states never observed in the demonstrations and not covered by simple generalization with Markov models. To illustrate the possible types of rules, we briefly examine two of them.

One type of rules illustrates implicit short-term goal setting for the model, eliminating Inverse Reinforcement Learning from the problem formulation. An obvious top-

level goal in an FPS genre is to find, attack and defeat the adversary. A human player would not stand still or wander while receiving damage from the adversary. Such state would rarely if ever appear in demonstrations. Instead, more often than not, the player would face the adversary and engage in combat. The corresponding rule we propose boils to simple "Turn to and approach the target whenever there is nothing else to do", captured with only a couple of lines of code. The target here can be the adversary, a cover spot or other objects of interest. The rule eventually transitions the agent into a state that it can handle from demonstrations.

For the second type of rules, an example could be as simple as "Do not run indefinitely in the same direction if moving in that direction is not possible". Humans proactively avoid blocked states, and they may never occur in organic gameplay. Hence, learning such a rule directly from the demonstrations is not possible since the data for such blocked states is not present.

In both cases, discovering the desired behavior via exploration would require a substantial amount of time, computational resources and a hand-crafted reward function. The costs of such exploration are disproportional to the simplicity of the decisions the agent needs to learn.

To summarize, the engineered rules capture simple human knowledge and complement the ensemble model in the states unobserved in the demonstrations. When the trimmed ensemble model fails to produce an action, the script checks for the conditions like the blocked one to generate a fallback action using the rules. The proposed combination of Markov ensemble and the programmed heuristics provides a segway to the next step which addresses the linear growth of the ensemble with the number of demonstrations.

DNN Model Trained with Bootstrapped Demonstrations and Rules

The traditional RL requires thousands of episodes to learn useful policies. Further, it is hard to engineer reward to achieve desired style. Instead, we resort to imitation learning where we treat the demonstrations as a training set for a supervised learning problem. The model predicts the next action from a sequence of observed state-action pairs. This approach has proved to be useful in pre-training of self-driving cars (Montemerlo et al. 2006) and is the subject of analysis in more recent literature, e.g., (Ross and Bagnell 2010). The main argument against casting IL as a supervised learning framework is the inability to learn from new situations and to recover from mistakes. The rules and feature engineering we present above intend to address these issues by incorporating prior human knowledge and bootstrapping demonstrations to make it part of the supervised learning data. We achieve it by augmenting our small set of demonstrations with bootstrap. We construct an agent controlled by a Markov ensemble enhanced with the rules and let it interact with the game to generate new episodes. The generated augmented data set feeds into training a DNN described next.

The trained DNN model predicts action from the already observed state-action pairs including those previously handled by the scripts. The low dimensionality of the feature space results in fast training in a wide range of model architectures, allowing a quick experimentation loop. We converged on a simple model with a single "wide" hidden layer for motion control channels and a DNN model for discrete channels toggling actions like sprinting, firing, climbing.

Table 1: Comparison between OpenAI 1V1 Dota 2 Bot (OpenAI Five 2018) training metrics and training an agent from human demonstrations, programmed rules, and bootstrap in a proprietary open-world first-person shooter. While the objectives of training are different, the environments are somewhat comparable. The metrics illustrate the practical advantages of the proposed technique.

	OpenAI 1V1 Bot	Bootstrapped Agent
Experience	$\sim$300 years (per day)	$\sim$5 min human demonstrations
Bootstrap using game client	N/A	$\times$5-20
CPU	60,000 CPU cores on Azure	1 local CPU
GPU	256 K80 GPUs on Azure	N/A
Size of observation	$\sim$3.3kB	$\sim$0.5kB
Observations per second of gameplay	10	33

While winning is not everything (Borovikov et al. 2019), we would like the agent to perform at a reasonable level of FPS genre metrics, e.g., demonstrate good kill-death ratio or defeat the adversary within the allowed limit of health and ammo. As we observe in our experiments, the kill-death ratio for the model we train can vary in wide range and is at best around 10-40% of the teacher's performance. It remains an open problem how we can improve performance metrics of a trained model using a limited amount of additional training while preserving the style.

Conclusion and Future Work

We tested our approach on a proprietary open-world first-person shooter game, which resulted in an agent behaving similarly to a human player with minimal training costs. Table 1 illustrates significant computational advantages gained from adding engineered knowledge to the training process of practically useful agents. However, when comparing our approach to the mainstream RL, we need to emphasize the difference between the training objectives, which makes such a comparison only *illustrative*.

The focus of our research is the practical cost-efficient development of human-like behavior in games. Keeping the performance of the model within certain limits is our future secondary objective. Obtaining theoretical guarantees for the style and performance of the trained agents would require substantial additional work. Applying our approach to multi-agent policies and covering multi-modal gameplay is a logical next step. We plan to extend the encouraging results shown here to other games in development.

References

Borovikov, I., and Beirami, A. Imitation learning via bootstrapped demonstrations in an open-world video game. *NeurIPS 2018 Workshop on Reinforcement Learning under Partial Observability*.

Borovikov, I., and Harder, J. 2018. Learning models to imitate personal behavior style with applications in video gaming. Technical report, Electronic Arts, Digital Platforms Data and AI. To be published as a preprint.

Borovikov, I.; Zhao, Y.; Beirami, A.; Harder, J.; Kolen, J.; Pestrak, J.; Pinto, J.; Pourabolghasem, R.; Chaput, H.; Sardari, M.; Lin, L.; Aghdaie, N.; and Zaman, K. 2019. Winning Isn't Everything: Training Agents to Playtest Modern Games. In *AAAI Workshop on Reinforcement Learning in Games*.

Harmer, J.; Gisslen, L.; del Val, J.; Holst, H.; Bergdahl, J.; Olsson, T.; Sjoo, K.; and Nordin, M. 2018. Imitation learning with concurrent actions in 3d games.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Montemerlo, M.; Thrun, S.; Dahlkamp, H.; and Stavens, D. 2006. Winning the darpa grand challenge with an ai robot. In *In Proceedings of the AAAI National Conference on Artificial Intelligence*, 17–20.

OpenAI Five. 2018. [Online, June 2018] `https://openai.com/five`.

Ross, S., and Bagnell, D. 2010. Efficient reductions for imitation learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Sardinia, Italy, May 2010*, 661–668.

Vinyals, O.; Ewalds, T.; Bartunov, S.; Georgiev, P.; Vezhnevets, A. S.; Yeo, M.; Makhzani, A.; Küttler, H.; Agapiou, J.; Schrittwieser, J.; et al. 2017. StarCraft II: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*.

Zhai, C. 2008. Statistical language models for information retrieval a critical review. *Foundations and Trends in Information Retrieval* 2(3):137–213.