Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2025)

Play-style Identification and Player Modelling for Generating Tailored Advice in Video Games

Branden Ingram, Clint van Alten, Benjamin Rosman, Richard Klein

University of the Witwatersrand, Johannesburg
Branden.Ingram@wits.ac.za, Clint.VanAlten@wits.ac.za, Benjamin.Rosman1@wits.ac.za, Richard.Klein@wits.ac.za

Abstract

Learning new skills often leads to disengagement when individuals face obstacles arising from limited experience, poor instruction, or mismatched preferences. This is common in video games, where players frequently abandon challenging sections. While expert guidance can mitigate these issues, providing personalised advice at scale remains difficult. Automated systems typically offer generic feedback, lacking adaptability to individual playstyles.

In this paper, we present an end-to-end system that generates personalised gameplay advice by learning from both pre-existing datasets and individual player behaviour. The system is evaluated in two domains: a simple GridWorld environment and the more complex MiniDungeons benchmark. Experimental results with simulated agents show that adherence to the generated advice leads to measurable improvements in performance. Our findings advance scalable, personalised guidance in games, with broader implications for learning and skill development.

Introduction

Learning new skills is often challenging and can lead to disengagement, particularly when individuals encounter obstacles due to lack of experience, poor instruction, or misaligned personal preferences. In the context of video games, players frequently abandon games or segments they find too difficult. One effective remedy is expert guidance, which helps learners overcome difficulties—a principle seen in education, sports, and gaming. However, providing such advice at scale is impractical. The key challenge lies in personalisation: effective guidance must align with an individual's unique playstyle, which encompasses preferences, motivations, behaviours, and skill levels. A defensive player, for instance, would benefit most from strategies tailored to a defensive approach as these are more aligned with their natural preferences.

Traditional automated advice systems, such as rule-based or decision tree-based models, have limited adaptability and generalise poorly in complex environments like video games (Ye and Johnson 1995). Alternatively, machine learning (Char, Shah, and Magnus 2018), natural language processing (Papamichail and French 2003) have been utilised for the generation of advice for recommendation systems. These models analyse a user's data and provide personalised feedback and advice based on the specific context and goals of the user. Specifically in terms of games, many also incorporate tutorial systems or walk-throughs that provide advice and guidance to players as they progress through the game (Andersen et al. 2012). However, these systems tend to exhibit a limited scope and lack of personalisation due to their one-size-fits-all nature. Thus, generating tailored advice that is helpful, relevant, and actionable remains a challenging problem that requires overcoming several obstacles, including the need for personalisation, domain expertise, and human-like reasoning.

Video games serve as ideal testbeds for this research, as they mirror real-world learning scenarios and support experiential learning. Ultimately, building a system capable of providing individualised, automated guidance has the potential to enhance not only gaming experiences but also broader educational and skill development contexts. Therefore, our primary contribution is an end-to-end, multi-component system that learns from both a pre-existing dataset and an individual's gameplay to generate personalised advice, as illustrated in Figure 1. The effectiveness of this pipeline is evaluated across two domains: a simple GridWorld environment and a more complex environment, MiniDungeons. Through experiments with simulated agents, we demonstrate that adherence to the generated advice leads to improved agent performance.

Related Work

The problem of advice generation is a computational problem that involves generating personalised recommendations or advice for a user based on their preferences, past behaviour, and other relevant factors. The goal of advice generation is to provide helpful and relevant guidance to users in various contexts, such as recommending products, suggesting courses of action, or providing support for decision-making. Researchers from different fields such as psychology (Harvey, Harries, and Fischer 2000), counselling (Vehvilainen 2001), philosophy (Herrmann 2022), and communication (Garvin and Margolis 2015) have studied this issue. In particular, Garvin and Margolis (2015) investigated the dynamics of giving and receiving advice, providing practical tips on how to communicate effectively and construc-

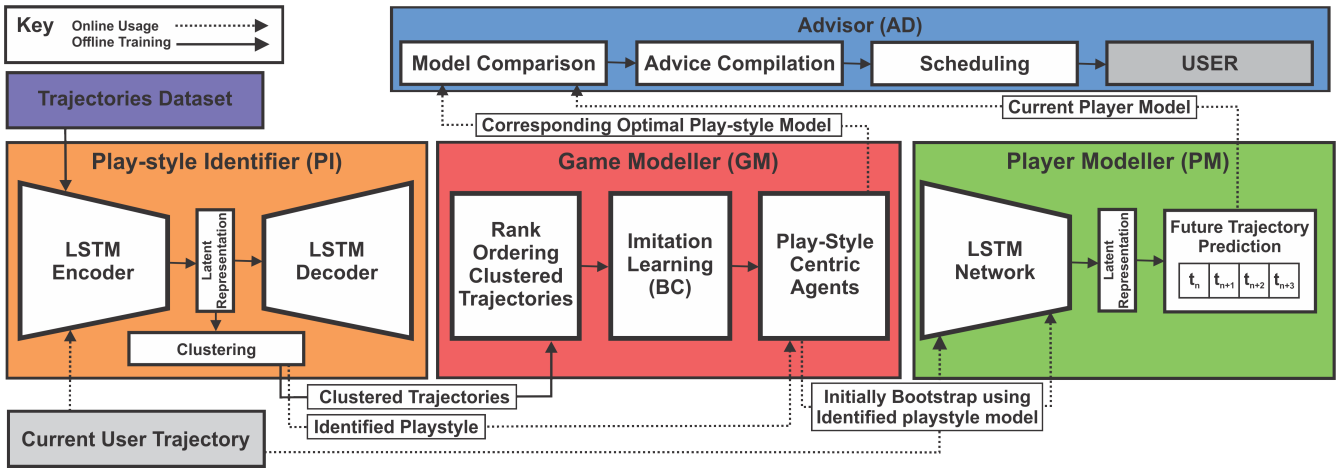

Figure 1: Complete Pipeline of Play-style-centric Advice Generation

tively. Our investigation aims to employ an automated mode of advice generation to differentiate between a user and their corresponding expert, as opposed to prioritising the method of communication.

Automatic advice generation refers to the use of artificial intelligence (AI) and machine learning (ML) techniques to provide personalised advice or recommendations to individuals (Zangerle and Bauer 2022). It involves the use of algorithms that analyse data about the person seeking advice, such as their preferences, goals, and past behaviour, to generate tailored advice. We look to utilise a suite of different ML approaches to generate advice. The general issue of generating effective advice is analogous to the challenge of interpretability associated with deep learning. Although significant advancements have been made in deploying deep learning algorithms that achieve expert-level performance, in the domains of classification such as He et al. (2016), Mensink et al. (2021) and Krizhevsky, Sutskever, and Hinton (2012) as well as in learning such as Mnih et al. (2015) and Silver et al. (2017), concerns have been raised regarding their lack of interpretability. However, if we are to use such techniques to generate useful applicable advice then we require a better understanding of how and why these algorithms produce the results they do.

Gunning (2017) also featured Hu et al. (2017) and Andreas et al. (2016) represent another approach to the problem of explainability making use of the idea of decision trees. Hu et al. (2017) utilised the concept of Neural Module Networks (NMN) which was proposed by Andreas et al. (2016). This NMN architecture makes it possible to answer natural language questions about images using collections of jointly-trained neural "modules", dynamically composed into deep networks based on linguistic structure. This is a similar problem to that of Malinowski, Rohrbach, and Fritz (2015). However, with NMN they showed that it achieves state-of-the-art performance on existing datasets for visual question answering. An alternative approach to explainability is rationale generation, as introduced by Ehsan et al. (2018a). This method involves producing natural language explanations for autonomous system behaviour as if the actions had been performed by a human. Ehsan et al. (2018a) employed a neural machine translation framework to convert internal state-action representations into human-like rationales. The approach required a training corpus of state-action pairs annotated with natural language explanations. Using an LSTM encoder-decoder architecture, the model learned to translate this structured input into coherent rationales. Ehsan et al. (2018a) demonstrated that this technique not only achieved higher accuracy than baseline methods but also produced explanations that were rated as more satisfying by human evaluators. This work was extended by Ehsan et al. (2018b), who tested the robustness of the rationale generation process by withholding portions of the input state information. Additionally, they conducted a more detailed qualitative study using a five-factor evaluation metric with human participants. Ehsan et al. (2018b) showed that the generated rationales consistently outperformed random baselines and closely approximated the quality of human-authored explanations.

All these approaches have focused on generating some understanding or rationale behind the actions produced by the learned model. However, in terms of our goal of creating an advice giver, having the rationale is only half the problem. The other half is to be able to convert that rationale into a form which individual users can utilise to better their performance. In order to accomplish this additional task, we draw on knowledge from the work done in intelligent tutoring systems (ITSs). ITSs are computer-based instructional systems with models of instructional content that specify what to teach, and teaching strategies that specify how to teach (Wenger 2014). ITSs have moved out of the lab and into classrooms and workplaces where some have been shown to be highly effective (Shute 1991; Graesser, Conley, and Olney 2012). While intelligent tutors are becoming more common and proving to be increasingly effective they are difficult and expensive to build. However, with the progress in cognitive science and the development in the theory of cognition, (Anderson 2013) researchers have been

able to design better ITSs such as the geometry tutor developed by Anderson, Boyle, and Reiser (1985). The system worked by generating rules by reducing geometry questions into sub-problems. These rules were used to simulate the sequence of the inferences (correct and incorrect) that students report making in trying to solve a geometry problem. More recently, the adoption of reusable components and learning objects (Ritter, Blessing, and Wheeler 2003), as well as the utilisation of community authoring (Aleahmad, Aleven, and Kraut 2008), has become prevalent in facilitating the development of intelligent tutoring systems. Both ITSs and other rationale generation approaches can benefit from a deeper understanding of the proficiency level of individual students or agents. Similarly, our solution for user advising is enhanced by developing a deeper understanding of the users and environment through play-style identification.

Alternatively, we can also draw from the work done in the fields of learning where researchers have focused on improving the rate at which agents learn. There is a growing number of techniques that are currently being incorporated to improve RL by leveraging knowledge from outside sources. These techniques include; Transfer Learning (Taylor and Stone 2009), Experience Replay, Apprentice Learning (Clouse 1997; Ho and Ermon 2016), Learning from Demonstration (Argall et al. 2009) and Inverse Reinforcement Learning (Abbeel and Ng 2004). In particular, teacher-student learning (also known as knowledge distillation) is a widely used machine learning paradigm, where a large, typically over-parameterized teacher model transfers its knowledge to a smaller, more efficient student model (Gou et al. 2021). The goal is to preserve the teacher's high performance while enabling faster, lighter, or more deployable student models. In a vision setting Xie et al. (2020) demonstrates "Noisy Student" training, where a large model is trained with data augmentation and pseudo-labels from a teacher. Additionally, knowledge distillation has been utilised by Schmitt et al. (2018) for bootstrapping student RL agents using expert teacher guidance. We argue that an understanding of the stylistic traits of a user will aid this distillation process. In particular, we employ a variant of this called Behaviour Cloning (Torabi, Warnell, and Stone 2018) to learn behavioural-centric policies.

Background

The following sections introduce the three core learning problems underpinning our automated tailored advice generation system: Play-style Identification, Play-style-centric Model Learning, and Player Modelling.

Play-style Identification

Play-style identification is an important concept in various fields, including sports, video games, music, and even business. Identifying a player's play-style can help coaches, teammates, or managers to understand their strengths and weaknesses, and tailor their training or strategies accordingly (Charles et al. 2005). This can lead to improved performance and better results. In particular, for video game developers tailoring user experience has become an important goal required to create more engaging and personalised experiences for players. By understanding how different players like to play, game designers can create content and features that cater to their preferences, making the game more enjoyable. One approach to play-style identification is to solve the following problem: Given a set of trajectories ($\mathcal{D}$) can we learn to separate them into k distinct partitions ($\mathcal{P}_k \subseteq \mathcal{D}$) with respect to unknown play-style characteristics. Can these partitions then be used to identify interpretable characteristics describing each play-style as well as their relationships?

Play-style-centric Model Learning

In recent years, there has been growing interest in the development of play-style-centric policy generation approaches, where the aim is not merely to optimise for a single, global objective, but rather to generate policies that reflect diverse play styles or behavioural preferences exhibited by different players (Mouret and Clune 2015; Wang et al. 2024). Broadly, there are two primary methods used to achieve play-style-centric policy generation: reward shaping and data-driven partitioning.

Reward shaping approaches typically define multiple reward functions ($R(s, a)$) that encode different desirable behaviours or play styles (Laud 2004). By adjusting the reward signal as in Equation 1, one can bias the learning process toward policies that embody certain characteristics for example, aggressive, defensive, exploratory, or risk-averse styles can be incentivised through rewarding actions a in state s given function c (Holmgard et al. 2014). However, defining suitable reward functions for complex behaviours is often challenging and may require extensive domain knowledge or iterative tuning.

$$R(s, a) = r(s, a) + c(s, a) \tag{1}$$

Alternatively, data-driven methods seek to identify and learn from distinct behavioural patterns present in existing gameplay data. One such approach frames the problem as follows: given a set of recorded trajectories ($\mathcal{D}$), can we partition this data into k distinct subsets ($\mathcal{P}_k$), each representing a coherent play style? From each partition, a corresponding policy (π_k) can be trained to exhibit the behaviour associated with that style. Furthermore, this process can be extended recursively, where each partition $\mathcal{P}_k$ may be further refined to capture more granular behavioural nuances, enabling the learning of increasingly optimal and style-specific policies (π_k^*). This data-centric perspective removes the need for manually crafted reward functions and allows the system to discover emergent play styles directly from observed behaviour.

Player Modelling

A common implementation of player modelling is that of modelling a player's choices (actions) given different scenarios (states) (Bakkes, Spronck, and van Lankveld 2012). In its basic form, this consists of a model which indicates the likelihood of any available player action given any state. Traditionally action models were implemented for board

game research to improve game tree searches by predicting opponent moves. In this case, the player model was expressed as an evaluation function (Cannel and Markovitch 1993). Similar approaches have been applied to more complex games, however, the increasing sizes of state and action spaces make training accurate models difficult. This has resulted in solutions involving large-scale computing (Vinyals et al. 2019) or utilising abstracted action spaces called "options" (Sutton, Precup, and Singh 1999). Our approach differs from these by using supervised learning instead of reinforcement learning to predict low-level accurate action similar to Muñoz, Gutierrez, and Sanchis (2013). When generating advice in a real-time scenario, it is important to have minimal delay in prediction accuracy, meaning that sample efficiency is critical. Sample efficiency refers to the ability of a sampling method to produce a representative sample using the minimum number of observations possible (Yarats et al. 2021). One potential method to enhance sample efficiency is to make use of pre-training, which involves training a model on a large dataset to learn general features and patterns that can later be fine-tuned on a smaller dataset for a specific task (Zoph et al. 2020). In a manner analogous to this, our model utilizes an initial offline learning strategy, supplemented by a subsequent online learning phase, to enhance its understanding of individual behaviour and improve its predictive accuracy.

Methodology

We aim to generate beneficial tailored advice for players through the identification of high-level play-styles and player modelling in video game domains. This approach involves acquiring a model of both the present state and the optimal expert version of an individual, specifically tailored to their play-style. It is through this method that we can draw comparisons and motivate the user to align their behaviour with their expert self. We suggest that by utilising this process, users can leverage the provided advice to enhance their performance and achieve higher levels of proficiency, specific to their play-style. To this end we propose the pipeline depicted in Figure 1 which is made up of four interconnected systems as outlined below:

The Play-style Identifier (PI)

This component identifies all the different play-styles through unsupervised clustering (Orange Component in Figure 1). To this end, we utilise a fully unsupervised clustering framework for the identification and analysis of play-styles as presented by Ingram et al. (2023b). This approach consists of three principal components. First, a temporal autoencoder with non-stacked LSTM layers, similar to the architecture of Xie, Girshick, and Farhadi (2016), is employed to project trajectories of variable lengths into a fixed-size latent space $\mathcal{Z}$. Each state in a trajectory is processed sequentially by LSTM cells, which capture important temporal dependencies and ultimately produce a latent representation Z_i for each trajectory $X_i \in \mathcal{D}$ via the encoder function. The decoder reconstructs the input trajectory $\hat{X}_i$ from Z_i, which is trained through back-propagation through time (Hochreiter and Schmidhuber 1997). Second, clustering is performed on the set of pairs (X_i, Z_i), where Z_i serves as the latent embedding of X_i. Clustering is conducted on the latent space $\mathcal{Z}$, producing cluster assignments y'_i, which serve as predicted play-style labels for both Z_i and the corresponding original trajectories X_i. By transforming trajectories into a latent representation, the model overcomes challenges related to varying trajectory lengths and temporal structure, enabling the application of conventional clustering algorithms such as k-means and Gaussian Mixture Models. Finally, the resulting clustering defines a set of trajectory partitions $\mathcal{P} = \mathcal{P}_1, \ldots, \mathcal{P}_k$, with associated centroids $\mathcal{C} = C_1, \ldots, C_k$. Each partition $\mathcal{P}_k$ groups trajectories sharing similar latent representations, where $k = y'_i$. Cluster analysis is then performed to extract interpretable behavioural characteristics, decision boundaries, and representative behaviours corresponding to each discovered play-style.

The Game Modeller (GM)

This component works to learn a policy representative of the best way to play for each of the identified play-styles (Red Component in Figure 1). Building on these identified partitions $\mathcal{P}_k$, we train representative policies for each style (Ingram et al. 2023a). Here, each partition $\mathcal{P}_k$ is first ordered according to a predefined ranking metric. In this case trajectory length was used a proxy to performance. A threshold parameter p is then applied to select a subset $\mathcal{P}k, p \subseteq \mathcal{P}_k$ containing only the top-performing trajectories, defined as those within the top p-percentile of the ranking. This selection ensures that the training data used for policy learning reflects demonstrations of a particular skill-range and targeted style. Finally, for each style k, a play-style-centric policy π^k is trained via supervised learning to minimise the loss between its predicted actions and the expert actions demonstrated within $\mathcal{P}_{k,p}$. This process yields optimal, behaviourally aligned policies for each discovered play-style.

The Player Modeller (PM)

This component models the play-style of an individual user to be able to offer tailored advice (Green Component in Figure 1). Here our model operates in two distinct modes: offline and online as described by Ingram et al. (2022b). In the offline mode, the model is pre-trained on a dataset of trajectories $\mathcal{D}$. Each trajectory $X \in \mathcal{D}$ is passed through a gating mechanism that routes it to a specific Predictor Node M_{k_i}, selected based on a cluster index k_i. The cluster index k_i is obtained via the pre-trained Play-Style Identification (PI) model, which partitions $\mathcal{D}$ into play-style-specific subsets $\mathcal{P}k_i \subset \mathcal{D}$, with $k_i \in 1, \ldots, k$, where k is the number of identified play-style clusters. The number of Predictor Nodes in the PM model matches the number of clusters k discovered by the PI model. Each Predictor Node M_{k_i} is responsible for learning the mapping from current state $X[t]$ to the action a which transitioned the world to the next state $X[t+1]$, where t denotes the current timestep.

In the online mode, the PM model is adapted for real-time personalisation to a specific user. This is achieved by continuously fine-tuning the model on partial trajectories

$P = X[1, t]$ generated by the user during interaction with the virtual environment. The online training occurs within the same environment used for offline pre-training. The partial trajectory P is passed through the PI model (pi) to obtain the current cluster index k_i, which is then used by the gating mechanism to direct P to the corresponding Predictor Node M_{k_i}. The Predictor Node then predicts the user's next action based on P. This fine-tuning occurs continuously and in real time, allowing the model to refine its predictions dynamically as additional data is collected during user interaction.

The Advisor (AD)

This final component compares a player's model with their corresponding optimal play-style model to formulate useful advice (Blue Component in Figure 1). Our methodology is designed such that it is capable of being used during real-time while a user is playing, wherein it offers guidance to the user during their interaction with a video game. The interaction between a player and a game environment can be described as a cycle, consisting of four stages:

- Input: The player inputs commands or actions into the game, such as moving a character or selecting an item.
- Processing: The game processes the player's input and updates the game state accordingly, such as moving the character to a new location or updating the inventory.
- Output: The game outputs the updated game state to the player, such as displaying the new location of the character or the updated inventory.
- Feedback: The player receives feedback from the game, such as a reward for completing a task or a penalty for failing to complete a task. This feedback can influence the player's future input and behaviour in the game.

During the feedback phase, our model extends conventional feedback by incorporating personalised advice. Within this context, we address the following question: *Can leveraging identified play-style information (p_k) to train accurate models of the user (p_m) and their expert counterpart (gm_{p_k}) effectively reduce the performance gap between the two?*

Advice Generation This overall process of advice generation in a online setting is outlined in Algorithm 1. We associate each user ($\mathcal{U}$) with a parameter ϖ which serves as a measurement of the likelihood of the user accepting advice (Line 1). By using this thresholding parameter, the model is able to simulate the practical scenario of users rejecting advice, thereby enhancing the realism of the system.

At the outset of a user's interaction with the system, we initialise the three models that were previously introduced, namely the Play-style Identifier (pi), Game Modeller (gm), and Player Model (pm), respectively (Line 2). In addition, a memory buffer is created to keep track of instances when the user accepts advice. The buffer stores the current partial trajectory P, representing the current sequence of states the user has visited, and the corresponding "correct action" provided by the expert. In continuous domains function approximation can be used to generalise experiences, allowing

Algorithm 1: Advice-giving Process

```
1:  procedure ADVICE-GIVING(𝒰, ϖ)
2:      Initialise pi, pm and gm models
3:      Initialise Memory
4:      P ← X[1 : t]   ▷ This is the trajectory observed we observe from 𝒰
5:      pk ← pi(P)      ▷ using Play-style Identifier obtain play-style pk from P
6:      π*_pk ← gm(pk)      ▷ using Game Modeller obtain optimal play-style-centric π*_pk from pk
7:      pm ← pm.train(P)      ▷ Fine-tune player model with new user data
8:      a_e ← π*_pk(P)      ▷ Predict optimal action
9:      if then P ∈ Memory      ▷ If previously advised on current partial trajectory
10:         a_p ← Memory[P]      ▷ Use advised action
11:     else
12:         a_p ← pm(P)      ▷ Predict player action
13:     end if
14:     if a_e ≠ a_p then      ▷ Equation 2
15:         Inform the user of the disparity
16:         r ∈ [0, 1)      ▷ Samples uniform distribution
17:         if r ≤ ϖ then      ▷ Check to see if user takes advice
18:             Memory[P] ← a_e      ▷ Add expert action into memory for current P
19:         end if
20:     end if
21: end procedure
```

the model to learn from a smaller set of experiences (Lillicrap et al. 2015). The subsequent step involves determining whether advice is required, which entails predicting the next action of the player and their corresponding expert, taking into account their play-style. This prediction is made based on the user's current partial trajectory P.

The process of determining the action of the player involves several steps. Initially, the memory buffer is queried (Line 9) to verify whether the user has received advice for the current partial trajectory (P) in the past. If the memory buffer contains P, the advised action that was previously stored in the memory is selected as the user's upcoming action a_p (Line 10). In this way, we assume that if a user had previously decided to remember the given advice then the user will act on it if possible. On the other hand, if P is not present in the memory buffer, the player model pm predicts the user's action a_p (Line 12). Furthermore, we fine-tune the pm by training the model on the observed partial trajectories of the user (Line 7). The key goal of the pm is to achieve a precise representation of the user's actions by focusing on the latest incoming user-specific information during system usage. This objective is accomplished by continuously learning from the observed partial trajectories P as the user continues to interact with the system. As we receive additional data from the user, the model can specialise further on that individual, leading to more reliable predictions.

In addition to acquiring the user's action (a_p), we also de-

termine the optimal action a_e based on the partial trajectory P. This involves the identification of the user's expert play-style policy (gm_{pk}), which is continuously obtained from the set of policies learned by the GM (Game Modeller). This identification process consists of several steps, as shown in Figure 2. During the user's interaction with the game, the system records information on their gameplay in the form of a partial trajectory (P). The trajectory is then encoded (Z_P) using the trained PI model (Play-style Identifier), using the $Encoder(P)$. Next, the play-style identifier (pk) refers to the index of the play-style cluster centroid that is closest to the latent encoding (Z_P). Ultimately, the expert play-style-centric model (gm_{pk}) corresponding to the cluster identifier pk is selected (Line 5). Finally, the optimal action (a_e) for the specific user is determined using the policy (π^*_{pk}) of the expert play-style, given the partial trajectory P (Line 6).

These user-specific models (pm and gm_{pk}) are then compared (Line 14) on a per action basis to identify differences using Equation 2. This function computes the actions outputted by the two models at every step $P[1:i]$ given the partial trajectory of length t and then computes the overall action disagreement between them. Larger values indicating greater disparity between the models being compared.

$$\text{disagreement}(pm, gm_{pk}, P[1:t], t) = \sum_{i=1}^{t} \begin{cases} 0, & pm(P[1:i]) = gm_{pk}(P[1:i]), \\ 1, & pm(P[1:i]) \neq gm_{pk}(P[1:i]), \end{cases} \quad (2)$$

Figure 3 illustrates a sequence of comparisons between the user model (depicted on the left) and the corresponding expert play-style-centric model (on the right). In the first cases actions, the user's actions match those of the expert, and hence, no advice is given. However, in the third case, a discrepancy is observed between the user's actions and the expert's, indicating that advice should be sent to the user. Upon receiving advice we consider the scenario where the user can decide whether to remember the advice or not. If the user chooses to accept the advice, based on the parameter ϖ, the current partial trajectory is stored in the memory for future reference.

Domain

To quantitatively evaluate the proposed advice generation model (Figure 1), a dataset of trajectories with known play-styles was required. Although the model operates in an unsupervised manner, such labelled data were necessary to validate its effectiveness. Firstly we evaluated on a simple Gridworld domain which served as a controlled test-bed for generating quantifiable, style-labelled trajectories using reward shaping. In GridWorld, five distinct grid-based environments were designed in Unity and integrated with the "ml-agents" toolkit and OpenAI gym interfaces. Four distinct play-styles were modelled via specific reward functions as developed by Ingram et al. (2022a). The result was a comprehensive collection of noisy yet behaviourally distinct trajectories, providing a foundation for play-style identification evaluation. Additionally, a domain called MiniDungeons was employed to demonstrate the model's effectiveness in more complex scenarios. MiniDungeons is a standard 2D dungeon exploration benchmark, which provides six handcrafted player proxies (Holmgard et al. 2014). These proxies were used to generate trajectories reflecting distinct behavioural styles, including safe or reckless running, various treasure-collection strategies, and monster-focused play. Data from multiple levels were combined to create robust training and testing sets.

Hyperparameters

To ensure reproducibility, we report the key architectural choices, training parameters, and evaluation metrics used across all experiments. These settings were consistent across the GridWorld and MiniDungeons domains unless stated otherwise. We use a single-layer, unidirectional LSTM architecture for both the encoder and decoder with hidden layer size of 20, and the latent vector of size 8. ReLU is used as the activation function, and the mean squared error (MSE) is employed as the loss function. The models are trained for 10,000 episodes using the Adam optimizer with a learning rate of 0.001. The input dimensionality varies by domain; 4 for GridWorld and 15 for MiniDungeons. We selected the number of clusters k to align with the underlying proxy policies for each domain: 4 in GridWorld and 6 in MiniDungeons. Both k-means and Gaussian Mixture Models (GMM) are initialized with 100 restarts and a maximum of 10,000 iterations. Each play-style-centric model consists of three fully connected layers with five nodes per layer. The input and output sizes correspond to the dimensionality of a single timestep: 4 for GridWorld and 15 for MiniDungeons. These models are trained for 2,000 episodes using 4 different random seeds per domain to ensure robustness. The dataset includes 4,000 trajectories for each of the five GridWorld environments, 780 for MiniDungeons. These are the same protocols used by Ingram et al. (2023b) and Ingram et al. (2023a).

Experiments

This section describes the experimental protocol developed to assess the effectiveness of the entire system in improving user performance using Algorithm 1. The experimental procedure involved constructing a dataset of individual users by randomly selecting five trajectories from overall dataset ($\mathcal{D}$) from each of the investigate domains, this is referred to as a user-set ($\overline{\mathcal{U}}$). For diversity, four selection schemes (outlined in Table 1) were used to select the trajectories, each representing a different user archetype. The term "skill" in this context denotes the level of proficiency of a user, independent of their playing style. This is assessed by the time taken to reach the goal which can be determined by the length of a trajectory. The term "style" represents the user's play-style irrespective of their performance. Therefore, our four user archetypes are represented by the different combinations of these two features.

The aforementioned process (Algorithm 1) was repeated for a total of 20 users, for each selection scheme and for each value of ϖ within the range of $[0, 0.25, 0.5, 0.75, 1]$. A

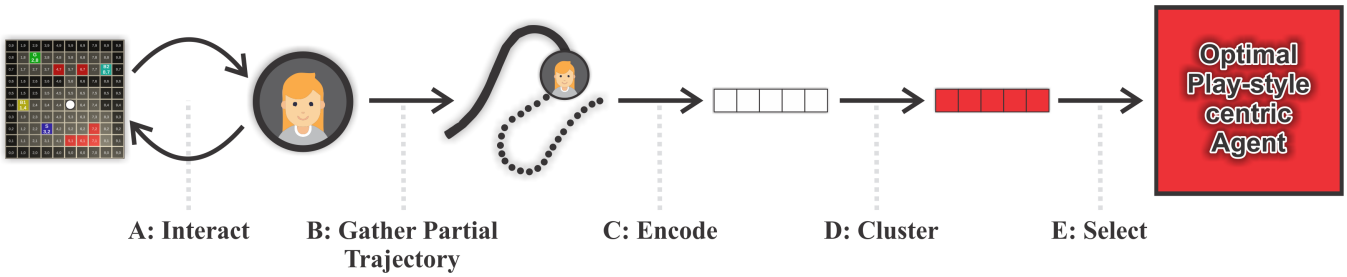

Figure 2: Five-step process of identifying an expert play-style-centric version of the current user. Firstly, the user interacts with the video game (A). Secondly, the system collects the user's current trajectory (B). Thirdly, this partial trajectory is encoded online (C). Fourthly, the encoding is clustered (D). Finally, the corresponding play-style-centric policy is selected using the cluster identifier (E).

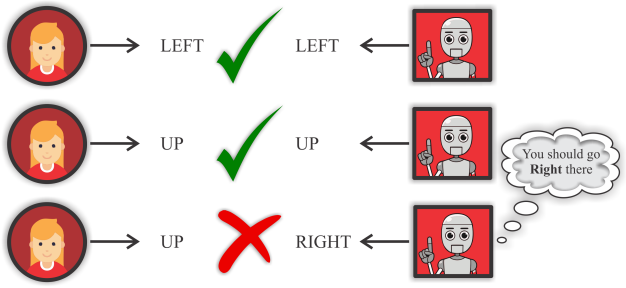

Figure 3: Example comparison between player model and corresponding expert play-style-centric model

Selection Scheme Name (Archetype)	Description (Trajectories are randomly selected from entire dataset $\mathcal{D}$)
full-random	Trajectories do not share play-styles or skill level
random-skill	Trajectories have different play-styles and relatively similar skill levels
random-style	Trajectories share the same play-style but with varying skill levels
set-skill-and-style	Trajectories share the same play-style and relatively similar skill levels

Table 1: Different user-set trajectory selection schemes with their associated description

value of "0" indicates a user who never accepts advice, while a value of "1" denotes a user who always accepts advice. By testing over multiple users we ensured a wide variety of play-styles and skills being represented in each archetype. Additionally, an infinite memory buffer was used, which means that once a user accepted advice, it would never be forgotten. The action correlation score between each user and their corresponding expert was calculated and recorded as the performance metric for our model. This metric is chosen as the objective of our model is to minimise the disparity between a user and their identified expert.

Results

The experimental results regarding the impact of the tailored advice generation pipeline on various advice acceptance thresholds and user archetypes are presented in Figure 4. As expected, it is observed that when advice is never utilised, the advice generation model fails to assist the user in improving their performance. However, in every other case where even marginal levels of advice utilisation occur, the model performance across all archetypes improves. Furthermore, as the degree of utilisation rises, there is a corresponding increase in the proportion of performance improvement.

Notably, it is observed that the "set-skill-and-style" archetype outperforms all others except in the "never take advice" scenario. Nevertheless, improvements were seen across all archetypes, indicating that the system provides tailored advice. The "set-skill-and-style" archetype performs best because the model is best able to exploit the consistent play-style characteristics found in this user-set. Similarly, the model performs the second best in the "random-skill" archetype as this archetype also has a consistent play-style. Both the Game Modeller and Player Modeller are trained in play-style-centric fashions rather than skill-centric, which explains these results. Lastly, it is worth noting that having a consistent skill level is more conducive to improvement compared to randomness.

We employed the identical experimental configuration within the MiniDungeons domain, and the outcomes are depicted in Figure 5. In this context, it is evident that, similar to our previous observations, when the advice is consistently disregarded, our model does not contribute significantly to improving the performance of the simulated user. Furthermore, as we observed in the GridWorlds domain, allowing the model's advice to be followed tends to lead to performance enhancements in our user model. However, this phenomenon is less pronounced in the MiniDungeons domain, particularly for the "full-random" and "random-style" user archetypes. Additionally, we observe slower performance improvements, which can be attributed to the increased complexity of the MiniDungeon trajectories, requiring users to accumulate more experiences before entering states they have encountered previously. Notably, the most effective user archetypes remain "set-skill-and-style" and "random-skill", both of which correspond to users with consistent play styles. This indicates that our play-style identification model can better comprehend our users, enabling the generation of more tailored advice, ultimately leading to a more substantial impact on their ability to emulate their respective experts.

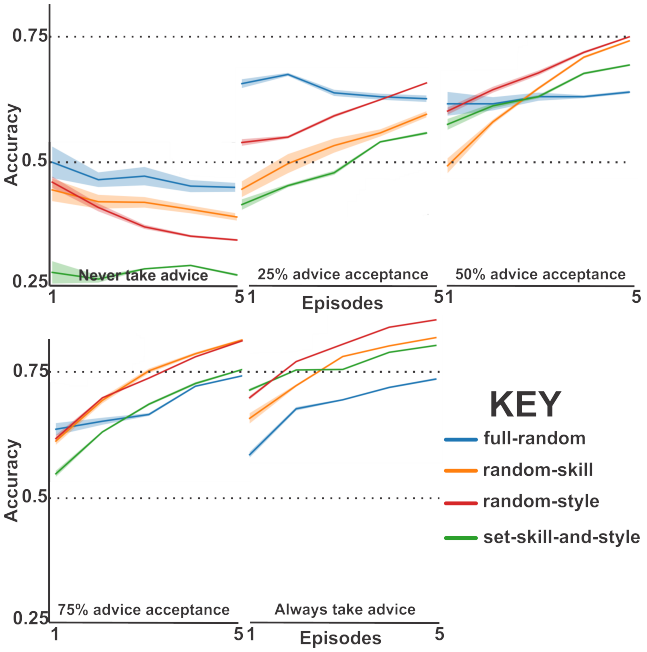

Figure 4: Comparison of the impact of a tailored advice generation pipeline on various advice acceptance thresholds and user archetypes for GridWorlds domain.

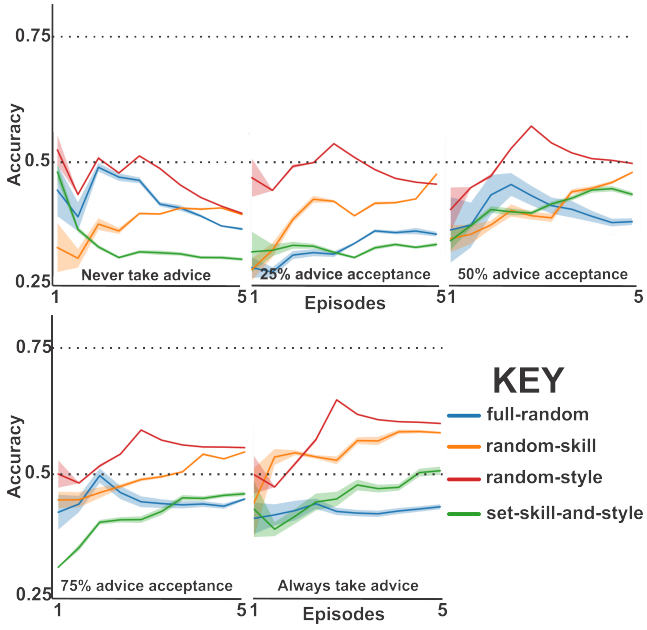

Figure 5: Comparison of the impact of a tailored advice generation pipeline on various advice acceptance thresholds and user archetypes for MiniDungeons domain.

Performance Results for PI and GM Models

Since the overall performance of the pipeline depends on the effectiveness of its individual components, this section presents the results for two key components of the full system: the Play-style Identifier and the Game Modeller.

Play-style Identifier

To evaluate the model's performance in play-style identification, we compared it against two baselines: (1) a random baseline representing the probability of correctly assigning a play-style label through uniform random selection, and (2) a non-learning baseline employing extensive data augmentation and standard clustering. The latter involved padding all trajectories to a uniform length using a zero state, followed by pairwise dissimilarity computation using a custom Root Mean Squared Error (RMSE) metric.

For quantitative evaluation, the predicted labels (y') were compared against ground truth labels (y) across multiple environments (E_1 to E_5 and MiniDungeons). Clustering accuracies for both k-means and Gaussian Mixture Model (GMM) clustering, obtained through offline clustering, are reported in Table 2. The results demonstrate the model's ability to accurately cluster complete trajectories ($X \in \mathcal{D}$) across varying environments, with some reduction in performance observed in MiniDungeons due to its higher complexity and multi-level nature. In all domains, the model significantly outperformed the random baseline and achieved superior accuracy relative to the non-learning baseline.

Environment	Random Baseline	RSME Baseline	Offline GMM	Offline k-means	Online GMM	Online k-means
E_1	0.25	0.298	**0.653**	0.598	0.499	**0.534**
E_2	0.25	0.310	0.707	**0.714**	0.602	**0.706**
E_3	0.25	0.321	**0.778**	0.705	**0.749**	0.670
E_4	0.25	0.267	**0.709**	0.706	0.576	**0.639**
E_5	0.25	0.278	**0.778**	0.652	**0.686**	0.678
Average across GridWorlds	0.25	0.295	**0.725**	0.675	0.622	**0.645**
MiniDungeons	0.17	0.191	**0.589**	0.489	**0.446**	0.419

Table 2: Complete and partial trajectory clustering accuracy

Game Modeller

Given that our approach relies on separating trajectories by play-style, we analysed the impact of clustering accuracy on overall model performance. As shown in Figure 6, we systematically varied clustering accuracy by introducing controlled label noise into the ground truth clusters available in the GridWorld domain. Specifically, we manually corrupted an increasing proportion of the cluster labels and trained play-style-centric models on these progressively noisier datasets. In this context, a corruption level of "0" indicates perfect clustering, while "1" corresponds to random clustering. The results indicate that as clustering accuracy decreases, the performance of the play-style-centric models also degrades. This effect is not observed in the baseline case where no clustering is used, as this configuration does not leverage play-style separation. Furthermore, performance variance increases as corruption levels rise, suggesting that the models lose their ability to specialise in consistent behaviours when trained on poorly clustered data.

Table 3 presents the relationship between clustering accuracy and the average rewards achieved by the play-style-centric models across all GridWorld environments. Because the ground truth reward functions are known, this analysis

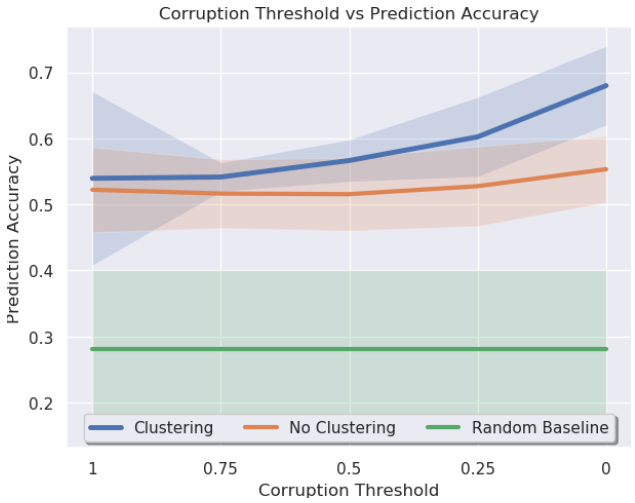

Figure 6: Comparison of the effect of differing the corruption threshold percentage on the models' accuracy. Here "1" indicates random clustering and "0" perfect clustering

Perfect clustering

Reward Function	GM_1	GM_2	GM_3	GM_4
R_1	**99.89**	0	0	0
R_2	0	**149.85**	0	0
R_3	0	0	**149.86**	0
R_4	0	0	0	**199.81**

Clustering using our model

GM_1	GM_2	GM_3	GM_4
74.07	12.03	13.02	18.24
7.42	**112.32**	11.99	19.22
7.42	10.25	**115.43**	17.38
7.30	11.09	16.42	**148.52**

Clustering with 50% accuracy

Reward Function	GM_1	GM_2	GM_3	GM_4
R_1	**46.87**	27.90	24.97	35.56
R_2	15.32	**72.73**	29.18	33.27
R_3	15.65	24.62	**76.57**	33.59
R_4	13.27	26.17	25.87	**103.88**

Random clustering

GM_1	GM_2	GM_3	GM_4
21.97	39.23	39.85	**50.42**
22.34	36.47	38.09	**55.71**
23.50	36.90	40.80	**49.20**
22.49	38.27	40.60	**49.66**

Table 3: The effect of clustering accuracy on the average generated rewards for each play-style-centric model across all GridWorlds

demonstrates that perfect clustering enables the models to accurately reproduce the target behaviours associated with each play-style. While our clustering approach achieves performance close to this optimal case, increasing clustering errors result in a progressive decline in alignment between the underlying reward functions and observed rewards.

Future Work

This work opens several avenues for future research aimed at improving automated, tailored advice generation in video games. The modular architecture of the system allows for targeted enhancements across its components. For the Play-style Identifier, future work could investigate image-based trajectory modelling using convolutional or transformer-based architectures, as well as multi-modal data integration. For the Game Modeller, inverse reinforcement learning presents a promising alternative to behavioural cloning, offering improved generalisation, albeit with higher data and computational demands. Enhancements to the Player Modeller could include richer representations of play-style characteristics to improve behavioural predictions. In the Advisor component, key research questions remain regarding optimal timing, frequency, and personalisation of advice delivery. Incorporating reinforcement learning to decide when and how to provide advice, leveraging large language models to generate natural and motivating feedback, and integrating adaptive game design elements could significantly enhance system effectiveness. Furthermore, extending advice generation to consider player attributes such as experience level or age, and enabling collaborative refinement of advice, offers additional potential. These directions not only promise to improve gaming experiences but also have broader implications for personalised learning and skill development across domains.

Conclusion

This work presents a system capable of automatically generating beneficial, tailored advice by leveraging identified play-style information. Using the motivating example of a defensive player, the approach first identifies the player's style and subsequently generates recommendations designed to enhance performance within that style. The primary contribution is an end-to-end framework for the automatic generation of personalised advice for video game players. To this end, we introduced a comprehensive advice generation model (Figure 1), composed of four interconnected components—Play-style Identifier (PI), Game Modeller (GM), Player Modeller (PM), and Advisor (AD)—trained in both online and offline contexts. While integrated within a unified system, each component addresses a distinct subproblem within the broader task of personalised advice generation. Our results demonstrate that clustering can be effectively used to identify style-based characteristics, which in turn can be leveraged to train more accurate user models as well as expert, play-style-centric models. Furthermore, we show that the Advisor component improves rate in which users become more proficient, we liken this to learning rate, when its advice is followed. These findings were validated across two domains: GridWorld and MiniDungeons. Collectively, the results confirm both the efficacy of the individual components and the overall effectiveness of the proposed pipeline.

In conclusion, automated tailored advice generation has the potential to improve the way people learn and improve their skills in a variety of domains. By identifying and utilising a person's preferred learning or play-style, this methodology can improve the efficiency and effectiveness of learning. Additionally, learning in one's own style can lead to a more positive and engaging learning experience, which is likely to lead to more sustained improvements.

References

Abbeel, P.; and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, 1. ACM.

Aleahmad, T.; Aleven, V.; and Kraut, R. 2008. Open community authoring of targeted worked example problems. In *Intelligent Tutoring Systems: 9th International Conference, ITS 2008, Montreal, Canada, June 23-27, 2008 Proceedings 9*, 216–227. Springer.

Andersen, E.; O'rourke, E.; Liu, Y.-E.; Snider, R.; Lowdermilk, J.; Truong, D.; Cooper, S.; and Popovic, Z. 2012. The impact of tutorials on games of varying complexity. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 59–68.

Anderson, J. R. 2013. *The architecture of cognition*. Psychology Press.

Anderson, J. R.; Boyle, C. F.; and Reiser, B. J. 1985. Intelligent tutoring systems. *Science*, 228(4698): 456–462.

Andreas, J.; Rohrbach, M.; Darrell, T.; and Klein, D. 2016. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 39–48.

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5): 469–483.

Bakkes, S. C.; Spronck, P. H.; and van Lankveld, G. 2012. Player behavioural modelling for video games. *Entertainment Computing*, 3(3): 71–79.

Cannel, D.; and Markovitch, S. 1993. Learning models of opponent's strategy game playing. In *Proceedings of the 1993 AAAI Fall Symposium on Games: Learning and Planning*, 140–147.

Char, D. S.; Shah, N. H.; and Magnus, D. 2018. Implementing machine learning in health care—addressing ethical challenges. *The New England journal of medicine*, 378(11): 981.

Charles, D.; Mcneill, M.; McAlister, M.; Black, M.; Moore, A.; Stringer, K.; Kücklich, J.; and Kerr, A. 2005. Player-centred game design: Player modelling and adaptive digital games.

Clouse, J. A. 1997. On integrating apprentice learning and reinforcement learning.

Ehsan, U.; Harrison, B.; Chan, L.; and Riedl, M. O. 2018a. Rationalization: A neural machine translation approach to generating natural language explanations. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 81–87. ACM.

Ehsan, U.; Tambwekar, P.; Chan, L.; Harrison, B.; and Riedl, M. O. 2018b. Learning to Generate Natural Language Rationales for Game Playing Agents. In *Joint Proceedings of the AIIDE 2018 Workshops*, volume 2282, 1.

Garvin, D. A.; and Margolis, J. D. 2015. The art of giving and receiving advice. *Harvard business review*, 93(1): 14.

Gou, J.; Yu, B.; Maybank, S. J.; and Tao, D. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6): 1789–1819.

Graesser, A. C.; Conley, M. W.; and Olney, A. 2012. Intelligent tutoring systems. *APA educational psychology handbook, Vol 3: Application to learning and teaching.*, 451–473.

Gunning, D. 2017. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2.

Harvey, N.; Harries, C.; and Fischer, I. 2000. Using advice and assessing its quality. *Organizational behavior and human decision processes*, 81(2): 252–273.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Herrmann, D. A. 2022. Prediction with expert advice applied to the problem of prediction with expert advice. *Synthese*, 200: 315.

Ho, J.; and Ermon, S. 2016. Generative adversarial imitation learning. In *Advances in neural information processing systems*, 4565–4573.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Holmgard, C.; Liapis, A.; Togelius, J.; and Yannakakis, G. N. 2014. Evolving personas for player decision modeling. In *2014 IEEE Conference on Computational Intelligence and Games*, 1–8.

Hu, R.; Andreas, J.; Rohrbach, M.; Darrell, T.; and Saenko, K. 2017. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, 804–813.

Ingram, B.; Rosman, B.; Klein, R.; and van Alten, C. 2022a. Play-style Identification through Deep Unsupervised Clustering of Trajectories. In *2022 IEEE Conference on Games (CoG)*. IEEE.

Ingram, B.; Rosman, B.; van Alten, C.; and Klein, R. 2022b. Improved Action Prediction through Multiple Model Processing of Player Trajectories. In *2022 IEEE Conference on Games (CoG)*, 548–551. IEEE.

Ingram, B.; Rosman, B.; van Alten, C.; and Klein, R. 2023a. Creating Diverse Play-Style-Centric Agents through Behavioural Cloning. In *Proceedings of the Eighteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2023, Salt Lake City, UT, USA*. AAAI Press.

Ingram, B.; van Alten, C.; Klein, R.; and Rosman, B. 2023b. Generating Interpretable Play-style Descriptions through Deep Unsupervised Clustering of Trajectories. *IEEE Transactions on Games*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Laud, A. D. 2004. *Theory and application of reward shaping in reinforcement learning*. University of Illinois at Urbana-Champaign.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Malinowski, M.; Rohrbach, M.; and Fritz, M. 2015. Ask your neurons: A neural-based approach to answering questions about images. In *Proceedings of the IEEE international conference on computer vision*, 1–9.

Mensink, T.; Uijlings, J.; Kuznetsova, A.; Gygli, M.; and Ferrari, V. 2021. Factors of influence for transfer learning across diverse appearance domains and task types. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, 44(12): 9298–9314.

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.

Mouret, J.-B.; and Clune, J. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*.

Muñoz, J.; Gutierrez, G.; and Sanchis, A. 2013. Towards imitation of human driving style in car racing games. In *Believable bots*, 289–313. Springer.

Papamichail, K. N.; and French, S. 2003. Explaining and justifying the advice of a decision support system: a natural language generation approach. *Expert Systems with Applications*, 24(1): 35–48.

Ritter, S.; Blessing, S. B.; and Wheeler, L. 2003. Authoring tools for component-based learning environments. *Authoring Tools for Advanced Technology Learning Environments: Toward Cost-Effective Adaptive, Interactive and Intelligent Educational Software*, 467–489.

Schmitt, S.; Hudson, J. J.; Zidek, A.; Osindero, S.; Doersch, C.; Czarnecki, W. M.; Leibo, J. Z.; Kuttler, H.; Zisserman, A.; Simonyan, K.; et al. 2018. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*.

Shute, V. J. 1991. Rose garden promises of intelligent tutoring systems: Blossom or thorn.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *Nature*, 550(7676): 354.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211.

Taylor, M. E.; and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul): 1633–1685.

Torabi, F.; Warnell, G.; and Stone, P. 2018. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*.

Vehvilainen, S. 2001. Evaluative advice in educational counseling: The use of disagreement in the" stepwise entry" to advice. *Research on Language and Social Interaction*, 34(3): 371–398.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Wang, X.; Zhang, S.; Zhang, W.; Dong, W.; Chen, J.; Wen, Y.; and Zhang, W. 2024. Zsc-eval: An evaluation toolkit and benchmark for multi-agent zero-shot coordination. *Advances in Neural Information Processing Systems*, 37: 47344–47377.

Wenger, E. 2014. *Artificial intelligence and tutoring systems: computational and cognitive approaches to the communication of knowledge*. Morgan Kaufmann.

Xie, J.; Girshick, R.; and Farhadi, A. 2016. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, 478–487. PMLR.

Xie, Q.; Luong, M.-T.; Hovy, E.; and Le, Q. V. 2020. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10687–10698.

Yarats, D.; Zhang, A.; Kostrikov, I.; Amos, B.; Pineau, J.; and Fergus, R. 2021. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 10674–10681.

Ye, L. R.; and Johnson, P. E. 1995. The impact of explanation facilities on user acceptance of expert systems advice. *Mis Quarterly*, 157–172.

Zangerle, E.; and Bauer, C. 2022. Evaluating recommender systems: survey and framework. *ACM computing surveys*, 55(8): 1–38.

Zoph, B.; Ghiasi, G.; Lin, T.-Y.; Cui, Y.; Liu, H.; Cubuk, E. D.; and Le, Q. 2020. Rethinking pre-training and self-training. *Advances in neural information processing systems*, 33: 3833–3845.