Proceedings of the Nineteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2023)

Creating Diverse Play-Style-Centric Agents through Behavioural Cloning

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

School of Computer Science and Applied Mathematics
University of the Witwatersrand
Johannesburg, South Africa
{branden.ingram, clint.vanalten, richard.klein, benjamin.rosman1}@wits.ac.za

Abstract

Developing diverse and realistic agents in terms of behaviour and skill is crucial for game developers to enhance player satisfaction and immersion. Traditional game design approaches involve hand-crafted solutions, while learning game-playing agents often focuses on optimizing for a single objective, or play-style. These processes typically lack intuitiveness, fail to resemble realistic behaviour, and do not encompass a diverse spectrum of play-styles at varying levels of skill. To this end, our goal is to learn a set of policies that exhibit diverse behaviours or styles while also demonstrating diversity in skill level. In this paper, we propose a novel pipeline, called PCPG (Play-style-Centric Policy Generation), which combines unsupervised play-style identification and policy learning techniques to generate a diverse set of play-style-centric agents. The agents generated by the pipeline can effectively capture the richness and diversity of gameplay experiences in multiple video game domains, showcasing identifiable and diverse play-styles at varying levels of proficiency.

Introduction

The development of diverse and realistic agents has become an important goal for game developers required to ensure the game world feels more immersive and believable, which can enhance player satisfaction and enjoyment. Traditional techniques in game design tend to involve hand-crafted solutions while in recent years approaches for learning game-playing agents often focus on optimising for a single objective, such as maximising the score or winning the game (Berner et al. 2019). While these agents may perform well in specific scenarios, they often lack the ability to represent the full range of play-styles that human players exhibit, such as "speed-runner", "completionist", or unpredictable play-styles (Drachen, Canossa, and Yannakakis 2009). In particular, these types of play-styles could be advantageous to designers by serving as a form of play-testing, either independently or in conjunction with human players. Such an approach would enable a more informative development cycle by providing a deeper understanding of how the player base could react to various design decisions. In terms of players, utilising human-like agents maintains a level of realism and fairness not awarded by an optimal agent. There is some empirical evidence indicating that human players prefer playing with and against human-like agents (Arrabales et al. 2012). Therefore, the question becomes not how can we learn an optimal policy but rather how can we learn a set of policies which exhibits diverse behaviours or styles.

Despite the significant success achieved in developing singular optimal policies for multiple video game domains (Berner et al. 2019), there has been considerably less research focused on creating sets of agents exhibiting diverse behaviours. Alternative approaches that aim to replicate observed behaviours have demonstrated their effectiveness in generating human-like behaviours (Pearce and Zhu 2022). In the context of either of these approaches, learning varied policies relies on either designing or automatically identifying sets of characteristics from which similar behavioural policies can be learned. To design a policy, we could define relevant features that describe the desired play-style, and then learn a policy that maps these features to actions (Arzate Cruz and Ramirez Uresti 2018). Alternatively, we can automatically extract relevant play-style features from data using techniques like clustering or dimensionality reduction and then learn a policy from those (Ingram et al. 2022). In both cases, the goal is to learn a set of play-style-centric agents whose behaviour matches that of either a designed or identified play-style.

Creating such a set of play-style-centric agents that play according to a certain style can have several potential benefits. Firstly by generating a group of agents who play according to different styles we can provide players with a more diverse and varied gaming experience, as they will be playing against opponents with different strengths and weaknesses. Moreover, such a technique that generates a set of agents that adhere to a particular gameplay style can offer a cost-efficient alternative for supplying a greater quantity of adversaries for players to confront, in contrast to the conventional design methodology. Many studies have demonstrated that defining appropriate reward functions for complex tasks can be highly challenging and often requires expert knowledge or extensive trial-and-error iteration (Amodei et al. 2016). In contrast, our proposed data-based approach offers a potentially more straightforward and efficient alternative. Lastly, play-style-centric agents can be useful for players who want to learn more about a particular strategy or

play-style, as they can observe the agents in action and analyse their decision-making processes. For example, a player might find enjoyment in being able to play against an opponent of a particular style.

We propose a novel pipeline which aims to generate a diverse set of play-style-centric agents dubbed PCPG (Play-style-Centric Policy Generation). This pipeline consists of a combination of unsupervised play-style identification and policy learning techniques. Through the unsupervised identification process, the demonstrations are divided into subsets based on identifiable behaviours. The policy learning method is then applied to train agents that can mimic a particular play-style. The ultimate goal is to create a diverse set of agents that can accurately represent various play-styles in the game at varying degrees of proficiency. We evaluate the efficacy of our proposed approach in two synthetic domains (GridWorlds, MiniDungeons) and one natural domain (Mario). Demonstrating that the agents generated by our pipeline can effectively capture the richness and diversity of gameplay experiences. Our results show that our agents learn policy with a high degree of accuracy while showcasing identifiable and diverse play-styles.

Related Work

The research on learning to generate policies which exhibit human-like behaviour can be roughly categorized into two groups based on whether they are reward-centric or data-centric. The reward-centric approaches rely on reward shaping which an RL algorithm such as Q-learning can use for optimisation. The most straightforward data-based approach is Imitation Learning (IL), which uses supervised learning to predict and perform actions with the highest predicted probability.

Reward Shaping

With reward shaping, the agent's standard reward function is modified with additional "shaping rewards" that come from a deterministic function $F : S \times A \times S \rightarrow R$ to aid in convergence towards a desired behaviour where S is the finite set of states, and A is is the finite set of all actions (Marom and Rosman 2018). This idea of reward shaping has been used to train a set of human-like agents with differing styles (Arzate Cruz and Ramirez Uresti 2018). Here the standard reward function R was augmented ("shaped") to resemble $R' = R + F$ where F encouraged the play-style-specific behaviours they desired. This resulted in agents with 3 varying styles in the game of Street Fighter 5. Halina and Guzdial (2022) focused on using the same approach of designed rewards to generate a set of diverse agents with a focus on multi-dimensional difficulty. Similarly, Khalifa et al. (2016) demonstrated an approach to encouraging agents to act in a certain style by augmenting the standard Monte Carlo Tree Search algorithm. Here an additional term was added to the Upper Confidence Bound equation which governs which nodes are explored, to select more human-like actions. The difficulty in accurately representing behaviours through reward functions makes it uncertain whether these approaches will produce desired results. Sephton (2016) explored an alternate design-based approach using several techniques of modifying Monte Carlo Tree Search (MCTS) to create different styles of play. These could then be used to change the experience of human opponents playing against them. They also investigate methods of improving the artificial agent's strength, including parallelizing MCTS and using Association Rule Mining to predict opponent choices, thus improving their ability to play well against them. Nonetheless, it was still necessary to utilise substantial domain knowledge to design the heuristics required for MCTS.

Imitation Learning

Imitation learning is a powerful yet simple technique for training a network to imitate a desired behaviour, be it from another algorithm or a human expert (Hussein et al. 2017). During the imitation learning process, the algorithm observes the expert's actions and tries to learn a mapping between the inputs (such as the state of the environment) and the corresponding outputs (such as the expert's actions). This mapping can then be used to make decisions in similar situations. Imitation learning is often used in robotics (Lopes, Melo, and Montesano 2007) and autonomous driving (Abbeel and Ng 2004), where it is difficult to program explicit rules and behaviours for the agents. By learning from expert demonstrations, the agents can acquire complex behaviours and skills that would be difficult to engineer manually. Silver et al. (2016) demonstrate the effectiveness of this approach by training an agent to play the game of Go at superhuman performance levels. This is achieved by utilising Imitation learning as a pre-training step before utilising traditional RL. Q-Learning from demonstrations is another bootstrapping approach which used demonstrations in Atari to bootstrap reinforcement learning against a known reward function rather than learning the reward function from the demonstrations (Hester et al. 2018). A drawback to using IL is that the agent's performance is limited to that of the demonstrator and in practice due to it being an approximation may even perform worse (Ross, Gordon, and Bagnell 2011). Additionally, depending on the scope and variety of the observations the entirety of the state space may be unexplored. If an agent finds itself in a region of uncertainty due to either error or the dynamic nature of the environment, it is unlikely for it to recover. This approach, therefore, requires a large number of expert examples to be effective. Obtaining these expert samples, however, can be difficult in terms of both time and cost, especially when working with human data. An online learning approach called DAGGER looked to mitigate the problem of overfitting by iteratively updating policies by requesting additional feedback from an expert (Ross, Gordon, and Bagnell 2011). This may be impractical for complex tasks with long training runs as the expert needs to be available throughout. Harmer et al. (2018) also acknowledged the difficulties of obtaining large amounts of expert data and instead applied imitation learning as a form of regulariser for a temporal difference RL agent. This allowed for improved performance and generalisability demonstrated in an in-house 3D game. Recently, impressive results in creating human-like agents have been demonstrated in the popular video game Counter-

Strike GO[1] (Pearce and Zhu 2022). Here, Pearce and Zhu (2022) utilised a supervised learning variant of IL called Behavioural Cloning (BC). Vinyals et al. (2019) also used supervised learning along with population-based techniques to learn a set of StarCraft 2 behaviours from demonstrations.

It has been shown that the combination of IL and RL alleviates some of the issues outlined. Such examples of these are Approximate Policy Iteration with Demonstration (APID) and Inverse Reinforcement Learning (IRL). For APID, linear constraints are defined using the expert trajectories utilised by the optimisation process of a Policy Iteration algorithm (Kim et al. 2013). The benefit of IRL approaches is the ability to learn a policy which can outperform the expert that it learnt from. Early IRL approaches assumed optimal demonstrations and a reward function that is linear in a known set of features. Here the key issue was that many differing reward functions could legitimately have led to a given optimal policy for some MDP. This led to Abbeel and Ng (2004) developing a method where both the recovered policy and expert, obtain the same reward on the original MDP. This, however, did not allow for the policy to surpass the expert in performance. Since we look to learn policies which mimic observed behaviours, we do not require our policies to exceed the experts. Bayesian IRL embraced the idea of the existence of multiple valid reward functions by inferring a posterior distribution over rewards rather than committing to a single one (Ramachandran and Amir 2007). By contrast, Maximum Entropy IRL returns a reward function that matches the expected feature counts, favouring rewards that lead to a higher-entropy stochastic policy (Ziebart et al. 2008). These two techniques both allow for the learnt policy to outperform the expert-demonstrated policy. Ranchod, Rosman, and Konidaris (2015) utilised both Bayesian and Maximum Entropy IRL to segment a set of unstructured demonstrations into a set of reusable "skills". Specifically, in video games, there has been limited success in employing IRL. Uchibe (2018) looked to firstly train a classifier which could identify expert versus non-expert state transitions. This classifier was then used as a reward function to train a deep RL algorithm however, their model rarely outperformed a BC baseline method. More recently an approach which looked to jointly learn the policy and reward function in an adversarial manner showed some promise in the Atari domain. However, they concluded that they still performed substantially worse than expert-level games warranting more work (Tucker, Gleave, and Russell 2018).

Unsupervised Clustering

Historically play-style clustering has been conducted on metadata statistics of gameplay experiences using Self Organising Maps (Drachen, Canossa, and Yannakakis 2009) or Kmeans (Bauckhage, Drachen, and Sifa 2014). More recently Justesen et al. (2020) also performed an initial clustering step prior to training behavioural models on StarCraft 2 build order data. However, all these approaches fail to incorporate the temporal aspect present in gameplay experiences or how play-style changes over time or across a level. Alternatively, play-style identification has been attempted temporally on play-through segments (Valls-Vargas, Ontanón, and Zhu 2015). However, Ingram et al. (2022) were able to separate multi-dimensional gameplay trajectories of varying lengths with respect to play-style on both complete and partial trajectories without the need for segmentation. This approach was built off Xie, Girshick, and Farhadi (2016) who implemented an LSTM-autoencoder which is capable of handling time-series data effectively. Although the specific method for clustering trajectories is independent of our methodology we applied this model which jointly optimises for both reconstruction loss as well as a clustering loss in an unsupervised fashion.

[1]https://blog.counter-strike.net/

Behavioural Cloning

Behaviour Cloning (BC) has been utilised for different video game environments using datasets of demonstrations (Gorman and Humphrys 2007; Harmer et al. 2018; Jacob et al. 2022) and is a form of imitation learning. Here an agent learns to exhibit a demonstrated behaviour by mimicking actions $a \in \mathcal{A}$, the set of all actions. The idea is to use an expert to demonstrate desired behaviour instead of using a domain expert to design fined-grained rewards which can be difficult and time-consuming. By utilising an expert to demonstrate the desired behaviour, the process of designing the reward signal can be made simpler since the feedback is provided in the form of a demonstration. This approach is useful when the reward function is difficult to specify or the environment is complex and dynamic. The network is then left to learn how to change its policy to match the actions represented in the expert's behaviour.

These are the actions an expert demonstrator would take given an observed state $s \in T$ where T is the entire trajectory. For our models, a trajectory is defined as a sequence of states $\{s_0, s_1, \ldots, s_N\}$ where N is the length of the trajectory. Learning, therefore, requires a dataset ($(D) = \{X_0, X_1, \ldots, X_h\}$ where h is the number of trajectories) of demonstrated behaviour ($\texttt{policy}{-}\pi$). A loss function is utilised for behavioural cloning, these include cross-entropy or mean squared error to measure the distance between the predicted and demonstrated actions. The model can then be trained by optimising the following equation:

$$\pi = \underset{\pi}{\texttt{argmin}} \sum_{i}^{N} l(a_i, a_i'; \pi) \tag{1}$$

Here l denotes the loss function, a_i represents the expected action as seen in the expert demonstration and a_i' represents the predicted action of the model. According to Bakker, Kuniyoshi et al. (1996), learning sequential action sequences becomes a highly efficient supervised learning task by using labelled examples to train the model and receive clear feedback on prediction accuracy. This enables the model to learn from mistakes and improve performance quickly.

Methodology

We aim to solve the following problem: given a set of trajectories ($\mathcal{D}$) can we separate them into k distinct subsets (H_k)

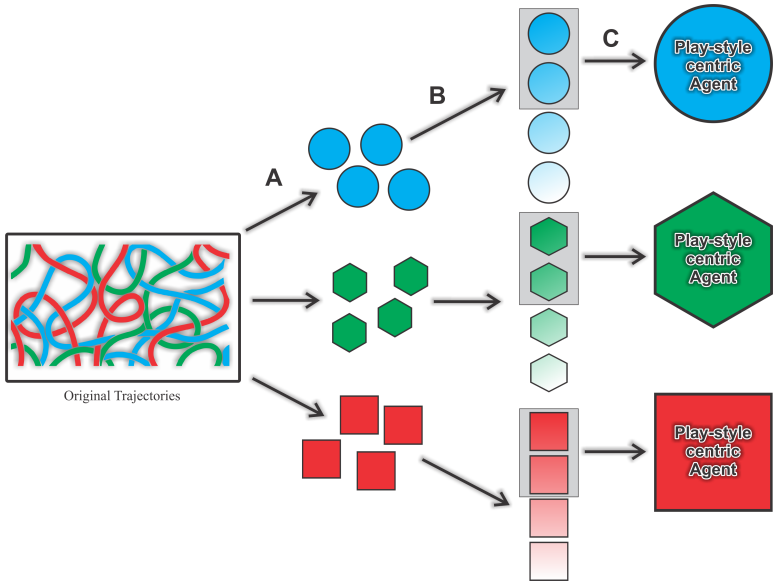


Figure 1: Overall PCPG system; initially the set of trajectories is clustered in step A into subsets H_k, each subset is then rank-ordered in step B indicated by the colour gradient, and finally the top performing trajectories are thresholded ($H_{k,p}$) and used to train the play-style-centric agent in step C.

from which we can generate a policy (π_k) which exhibits the behaviour identified for each subset k? Furthermore, can we subdivide H_k based on a skill level p such that we can generate a policy ($\pi_{k,p}$) which exhibits both the identified behaviour k at the appropriate skill level p?

Play-Style-Centric Policy Generation (PCPG)

Our PCPG system as depicted in Figure 1 is based on three key steps. In step A, we use a clustering technique to separate our observations, or in this case our multi-dimensional trajectory data, according to different identifiable play-styles (Ingram et al. 2022). Step B, we order the trajectories based on a ranking function such that we can filter our separated observations based upon a particular skill range. Step C is then to train individual BC models on this filtered selection of observations for each of the generated subsets. Since each of these selections contains data associated with a specific behaviour, it is expected that the learned policies would display similarities to that behaviour and skill level. Therefore, these models trained using behavioural cloning are referred to as play-style-centric models.

Trajectory Clustering (Step A) To separate trajectories with respect to the style we first project a trajectory ($X_i \in \mathcal{D}$) into a lower-dimensional latent representation ($Z_i \in \mathcal{Z}$) using an LSTM autoencoder network. This network is optimised by minimising the distance between the original trajectory ($X_i \in \mathcal{D}$) and the reconstructed trajectory (X_i') given by Equation 2.

$$Z_i = \texttt{Encoder}(X_i) \;\; \texttt{and} \;\; X_i' = \texttt{Decoder}(Z_i) \quad (2)$$

We then perform clustering on this latent space ($\mathcal{Z}$) to discover k clusters corresponding to related trajectories.

Policy Learning (Steps B and C) Having clustered all trajectories ($X \in \mathcal{D}$) into separate subsets ($H_k \subset \mathcal{D}$) where k is the cluster identifier. In step B each trajectory in H_k is reordered based on a ranking metric. This ranking metric works by assigning a performance or "skill" value to a particular trajectory $X \in H_k$. By applying this approach we obtain an ordered set of H_k such that the first element (trajectory) is the weakest performing and the last is the highest performing trajectory in H_k. This process is necessary such that we can filter trajectories based on a specific performance or "skill" range. To do this, a thresholding parameter (p) is used to choose a subset of trajectories ($H_{k,p} \subseteq H_k$) that perform in the top "p" percent. This subset is utilised as inputs for the corresponding play-style-centric model ($\pi_{k,p}$). This is implemented to guarantee that the trajectories received by our play-style-centric models correspond to a particular skill range based on p and for a particular play-style k. Lastly in step C, the play-style-centric model is trained through a supervised learning approach that minimises the loss between the predicted action by the model and the expected action.

Experiments

Datasets

To validate the robustness of our method, we evaluate our model on two synthetic datasets and one natural dataset. The first is derived from a GridWorld game where a player seeks out a goal with the opportunity of completing two additional optional objectives. By generating this set of trajectories we have access to the ground truth play-styles and as a result, we use this domain to obtain a quantifiable measure of performance. The second is a standard video game research domain called MiniDungeons (Holmgard et al. 2014) containing data generated from different designed human-like

behaviours across multiple 2D levels. The third is an unlabelled set of trajectories from the game Super Mario Bros (Guzdial and Riedl 2016). This data set was collected from individuals and we use it to demonstrate our model's performance in natural domains.

GridWorld To assess the capacity of our model to learn distinct play-style-centric policies, it is crucial to obtain numerous trajectories from a variety of play-styles. Trajectory-based datasets labelled according to style do not exist and therefore we generate data to account for this. Certain datasets aim to capture metrics that are loosely connected to play styles, such as emotions and enjoyment. However, these datasets often depend on limited surveys with self-reported features, which can introduce noise[2]. In contrast, behavioural cloning necessitates substantial amounts of data to be effective. We distinguish two play-styles as being different goals that could be reached by an agent. These we model as different reward functions in the reinforcement learning paradigm. This idea of reward shaping has been used to train a set of human-like agents with differing styles (Arzate Cruz and Ramirez Uresti 2018). We utilise this approach to generate a set of trajectories with differing performance levels for multiple styles.

Our Preference-Based Trajectory Generation (PBTG) approach (Algorithm 1) was used to generate 5 individual datasets $\mathcal{D}_n$ from 5 different environments $(E_1, \ldots, E_5)$ with 4 varying play-styles (reward functions) present. Each environment is a 10×10 grid world, as depicted in Figure 2. The environments each have a start state (S, in blue) and a goal state (G, in green). Walls (black tiles) cannot be traversed and trap states (red tiles) result in failure. The variety in play-styles is introduced through the addition of two bonus states (B_1, in gold and B_2, in cyan). These are the optional objectives that a player with certain preferences might wish to complete. The set of actions is the movement in any of the four primary cardinal directions. These environments were designed and implemented using Unity[3], a free-to-use 3D video game creation engine. These environments incorporated "ml-agents"[4] an open-source project that enables games and simulations to serve as environments for training intelligent agents. Lastly, these environments with the "ml-agents" toolkit were wrapped as Open-AI gym[5] environment using a pre-existing gym-wrapper[6].

Using this approach we generated data with 4 play-styles, as described in Table 1. These are the observable behaviours our model aims to recover through its play-style identification process. The set of reward functions R used to emulate these behaviours is also defined in Table 1. Here we defined large positive rewards for the objectives we wished the agent to accomplish. The respective bonus rewards were only given the first time an agent reached either B_1 or B_2. Following the procedure outlined in Algorithm 1 we trained a Q-learning agent for each of the combinations of R and E for 20000 episodes with discount factor $\gamma = 0.99$ and linear ϵ-decay over the number of training episodes to ensure our agent converges to the global optimal. The state is given by the tuple (x, y, b_1, b_2) where x and y are the Cartesian grid coordinates and b_1 and b_2 indicate whether an agent has visited B_1 or B_2, respectively. Our dataset consists of 8000 randomly selected trajectories per R and E. Therefore our trajectory is a sequence of time steps in the form of (x, y, b_1, b_2).

R	Behaviour	G reward	B_1 reward	B_2 reward
1	Moves directly to G	100	0	0
2	Visits B_1 before G	100	50	0
3	Visits B_2 before G	100	0	50
4	Visits B_1 & B_2 before G	100	50	50

Table 1: Observable play-styles and reward structure

Algorithm 1: Preference-Based Trajectory Generation (PBTG)

procedure PBTG(Environment E)
  Define a set of reward functions R
  Initialise our set of trajectories $\mathcal{D} = \{\}$
  for all reward functions $r \in R$ **do**
    Use Q-learning to learn optimal policy π_r^*
    for n number of required Trajectories **do**
      $\pi_r^n \leftarrow perturb(\pi_r^*)$
      Generate $X(n, r)$ from π_r^n and append to $\mathcal{D}$
    end for
  end for
end procedure

MiniDungeons MiniDungeons is a two-dimensional top-down dungeon exploration game, and is a common benchmark research domain for modelling and understanding human play-styles (Holmgard et al. 2014). We use a dataset of game trajectories generated using six player proxies which are handcrafted policies designed to exhibit particular behaviours (play-styles) (Arendse, Ingram, and Rosman 2022). Table 2 describes the different play-styles corresponding to the 6 player proxies as well as their corresponding behaviour with example trajectories depicted in Figure 3. The paths followed by the system are produced by executing the actions chosen by the designed policies at each level while keeping track of the state-action transitions. The state at each timestep is defined as a 15 dimensional vector which encodes event information up until that current timestep and aims to track higher-level player tactics. We combine a total of 780 trajectories generated across multiple different levels to form our training and testing sets. The specific levels used as named by Holmgard et al. (2014) are as follows:

- $train$: (md-test-v0, md-hard-v0, md-random_1-v0, md-random_2-v0, md-gene_1-v0, md-gene_2-v0, md-strand_1-v0)
- $test$: (md-holmgard_1-v0, md-holmgard_2-v0, md-holmgard_4-v0, md-holmgard_6-v0, md-holmgard_9-v0, md-holmgard_10-v0)

[2]source: https://plt.institutedigitalgames.com/datasets.php

[3]https://unity.com/

[4]https://github.com/Unity-Technologies/ml-agents

[5]https://github.com/openai/gym

[6]https://github.com/gzrjzcx/ML-agents/blob/master/gym-unity/README.md

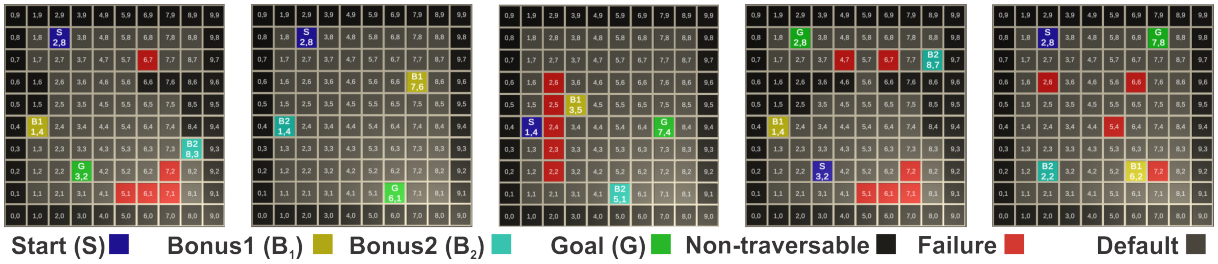


Figure 2: Randomly generated grid world environments $E_1, ..., E_5$

Play-style	Behaviour
Safe Runner	Complete the dungeon as quickly as possible while avoiding monsters.
Wreckless Runner	Complete the dungeon as quickly as possible without avoiding monsters.
Brave Treasure Hunter	Collect treasure even if guarded by enemies.
Pure Treasure Hunter	Collect only unguarded treasure.
Safety-First	Only collect treasure and potions.
Monster Killer	Fight as many monsters as possible without dying.

Table 2: MiniDungeons player proxy behaviours

Figure 3: Example visualisations of trajectories for each of the 6 player proxies in the MiniDungeons domain (Holmgard et al. 2014)

Mario This dataset consists of 74 playthroughs across 11 different levels of Super Mario Bros. These playthroughs were each captured by logging the actions of a unique human participant (Guzdial and Riedl 2016). We then refactored this data into a trajectory, where each time step represents the current state of the playthrough at that point. We defined a state as a tuple given by (j, k, r, c, d, e) where j is the number of jumps, k is the number of enemies killed, r number of times the player has started running, c the number of coins collected, d the number of times the player died and e the unique action encoding.

Training

For the trajectory clustering, we used a pre-trained unsupervised LSTM clustering model as described by Ingram et al. (2022). Our play-style-centric models were made up of 3 fully connected layers of 5 nodes each. The dimensions of the input and output for this model correspond to the length of a single timestep, which is equal to 4, 6 and 15 for the GridWorld, Mario and MiniDungeons domains respectively.

All experiments were conducted using 4 seeds across the 5 GridWorld environments as well as the Mario and MiniDungeons domains with the average accuracy for each being recorded. We trained each play-style-centric model for 2000 episodes. The initial dataset ($\mathcal{D}$) contains 4000 trajectories for each GridWorld environment and 780 for MiniDungeons and 74 for Mario. For our ranking metric, we utilised Equation 3 where x_T denotes the final state in the trajectory X, and β is a constant penalty factor that reduces the score when x_T is not a goal state. This performance metric incentivises shorter trajectories by utilising the length of a trajectory calculated as $|X|$. Here higher values of $|X|$ correspond to lower performance, while smaller values of $|X|$ indicate better performance. Additionally, we adjust this metric based on whether a trajectory successfully reaches the goal state or not. For example, if $\beta = 0.5$, then the score is halved when the trajectory does not end in a goal state. In particular, we used $\beta = 0.1$ heavily penalising all trajectories which did not reach the goal state. While in our domains we employ trajectory length as a performance measure, it is equally feasible to substitute it with a predefined score metric or a different combination of features that can serve as indicators of overall skill. To showcase the advantage of "step C" in our approach, we trained an individual model with the same architecture as each play-style-centric model without clustering, as a baseline approach. Lastly, we tested over a range of thresholding percentages, in particular, where $p = [1, 0.75, 0.5, 0.25, 0.1]$. Here a value of 1 indicates we train our play-style-centric models on all trajectories while 0.1 indicates we train each model on the top 1% of trajectories.

$$\text{Performance}(X) = \frac{1}{|X|} \cdot \begin{cases} 1 & \text{if } x_T \text{ is a goal state} \\ \beta & \text{otherwise} \end{cases} \quad (3)$$

Evaluation

In this study, model accuracy hinged on the model's capacity to predict the correct action for transitioning the agent

between states s_i and s_{i+1}. Employing an autoregressive approach to predict subsequent states at each step proved problematic due to error accumulation. To remedy this and ensure that each prediction can be made independently of historical mistakes, we input, at each step, the ground truth state to the model instead of its previous predictions. This is a widely-used technique in language modelling and model-based reinforcement learning (Moerland et al. 2023). This process is completed for all states in a trajectory and across all trajectories in the testing set. Moreover, alongside all relevant plots, we incorporate error bars representing a single standard deviation of the model. Furthermore, we conduct a comparison with a random baseline. However, it should be noted that although the action space for both domains consists of the four cardinal directions, we apply masking to prohibit agents from selecting actions that are invalid in specific states. Consequently, the average performance of the random baseline is not 0.25, as one might expect, but rather an average calculated across all possible actions for every state.

Results

Through the results of our experimental analysis, we demonstrate the ability of our model to accurately represent desired behaviours that correspond to identified play-styles.

PCPG Model Performance

Figure 4a demonstrates that our PCPG model not only yields high performance but also surpasses the equivalent "No Clustering" baseline. Comparable outcomes can be observed for the MiniDungeons depicted in Figure 4b and for Mario as seen in Figure 4c. This benefit can be observed across all tested values for the performance threshold (p). As our objective is to generate models that emulate the behaviour exhibited by the data, the accuracy of model predictions serves as a suitable performance measure.

Given our knowledge of the GridWorld environment's intrinsic reward function, we can assess each model's (π_k) performance relative to the corresponding underlying reward function (R_k). This is measured as the difference between the model's mean reward (across trajectories and environments) and the optimal reward distribution (Table 1). Trajectories in this case refer to those generated by the model itself. The model's performance is depicted in Table 3, demonstrating notable enhancement in average rewards compared to random performance and a likeness to the optimal scenario. This finding underscores that play-style-centric models, trained on segmented data, can effectively internalize the encoded underlying play-styles exhibited within the trajectories.

Finally, Table 4 reports the prediction accuracy of each play-style-centric model $\pi_{k,p}$ having been trained on its corresponding subset $H_{k,p}$ while then being tested on all subsets $H_{k,p}$. The results are averaged over all values of p. The strong diagonal suggests that each model has specialised in the data within its partition. This indicates that the policies learned by each model are inherently different.

	Play-style-centric Policies			
Model Performance	π_1	π_2	π_3	π_4
Perfect	0.0	0.0	0.0	0.0
Ours (66%)	**8.66**	**44.85**	**46.28**	**50.79**
50%	16.80	89.89	90.52	95.81
40%	27.02	133.59	134.02	141.53
Random (33%)	33.05	180.84	180.12	192.42

Table 3: Distance between average obtained reward and optimal underlying reward function for each play-style centric policy (π_k)

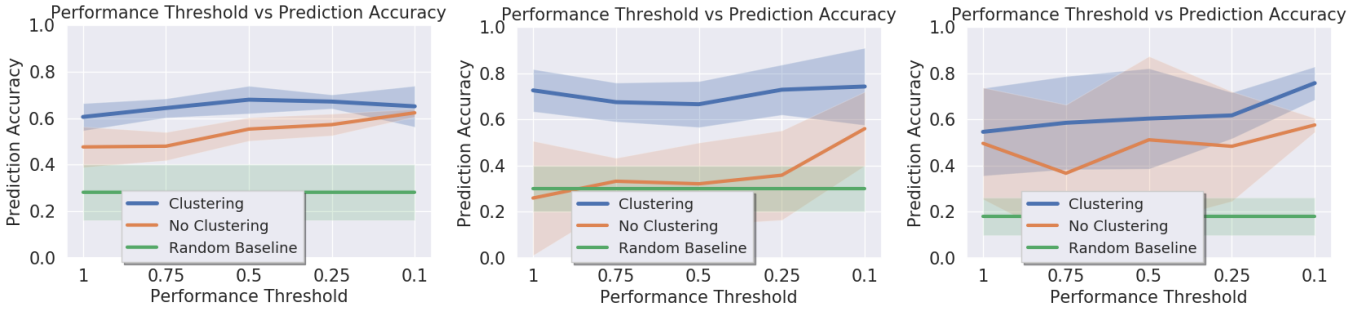
Average over all values of p	$\pi_{1,p}$	$\pi_{2,p}$	$\pi_{3,p}$	$\pi_{4,p}$
$H_{1,p}$	**0.57**	0.45	0.44	0.36
$H_{2,p}$	0.37	**0.64**	0.36	0.49
$H_{3,p}$	0.38	0.39	**0.63**	0.46
$H_{4,p}$	0.33	0.57	0.40	**0.69**

Table 4: Prediction accuracy of each model versus the different partitions averaged across all GridWorlds and all p

Behavioural and Skill-Based Diversity

By varying the value of the performance threshold p we are also able to generate policies of varying skill levels which is another form of diversity. The reason for this outcome can be attributed to the fact that our PCPG model was trained on a particular skill range of trajectories. This is evidenced by Figure 5, which shows a consistent pattern where increasing values of p (wider range of skill) correspond to longer average trajectory lengths. This is a valid analogue for skill as the trajectory length was used as the primary feature of our performance metric (Equation 3). Furthermore, we can observe that each play-style is converging towards its own optimal length, which serves as further evidence of the behavioural diversity that can be achieved through our PCPG model. In contrast, the "No Clustering" approach converges towards a single behaviour. A similar result can be observed in the Mario domain as seen in Figure 6 where the individual play-style-centric models each have a different average trajectory length. Demonstrating diversity among models, Figure 7 displays cross-correlation between play-style-centric models and different data subsets (H_k). Values closer to 1 indicate stronger correlation between play-style and trajectory subset. Notably, the model achieves the highest correlation with its specific training subset. Correlation values beyond the diagonal are generally lower, except for a single exception. Specifically, play-styles one and three show discernible but not significant correlation, suggesting shared traits, while the remaining play-styles significantly differ.

Qualitatively we can view the diversity in behaviours in Figure 8 for a particular environment E_2. Here we observe that the exhibited behaviours match those behaviours encoded in the data, as shown in Table 1. To generate a trajectory of states, each component of our play-style-centric models ($\pi_{k,p}$) takes an initial state as input and produces an action. This action is applied to the current state to transition to the next state. This process is repeated iteratively to



(a) GridWorld (b) MiniDungeons (c) Mario

Figure 4: Comparison of effect of differing the performance threshold percentage p on the models accuracy in GridWorld, MiniDungeons and Mario

	Perfect clustering				**Clustering using our model**				**Clustering with 50% accuracy**				**Random clustering**			
Reward	H_1	H_2	H_3	H_4	H_1	H_2	H_3	H_4	H_1	H_2	H_3	H_4	H_1	H_2	H_3	H_4
R_1	**99.89**	0	0	0	**74.07**	12.03	13.02	18.24	**46.87**	27.90	24.97	35.56	21.97	**39.23**	39.85	50.42
R_2	0	**149.85**	0	0	7.42	**112.32**	11.99	19.22	15.32	**72.73**	29.18	33.27	22.34	36.47	38.09	**55.71**
R_3	0	0	**149.86**	0	7.42	10.25	**115.43**	17.38	15.65	24.62	**76.57**	33.59	**23.50**	36.90	**40.80**	49.20
R_4	0	0	0	**199.81**	7.30	11.09	16.42	**148.52**	13.27	26.17	25.87	**103.88**	22.49	38.27	40.60	49.66

Table 5: The effect of clustering accuracy on the average generated rewards for trajectories contained within each subset H_k across all GridWorlds

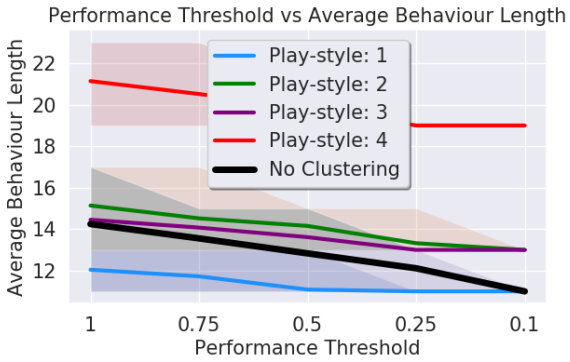


Figure 5: Comparison of effect of differing the performance threshold percentage p on the trajectory diversity across all GridWorld environments $(E_1, \ldots, E_5)$

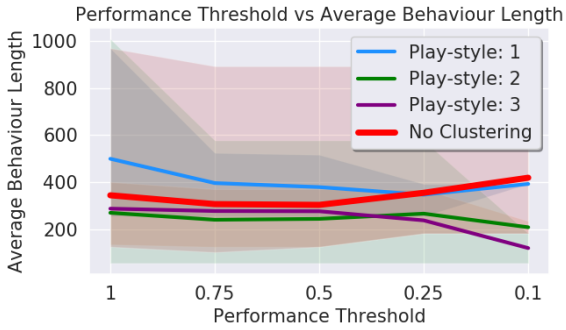


Figure 6: Comparison of effect of performance threshold percentage p on the trajectory diversity in Mario domain

construct a sequence of states and actions that represents the predicted trajectory. Notably, the "No Clustering" behaviour corresponds to a direct path to the goal, reflecting the expected behaviour for the shortest trajectories.

Effects of Varying the Clustering Performance

Since our approach is dependent on first separating trajectories we look to analyse the effect of the clustering performance of our PCPG model's performance. In Figure 9 we observe the results of manually varying the levels of clustering accuracy. Since our GridWorld data was generated we have access to the ground truth clusters. We then manually mislabel a varying portion of these ground truth labels. Therefore, Figure 9 is the result of training our play-style-centric models on datasets of varying corruption levels, here "1" indicates random clustering and "0" is perfect clustering. As a result, we identify that increasing the levels of corruption results in decreasing the effectiveness of our model. However, it does not affect the case where we do not employ any separation. This observation is to be expected as the case whereby we train a single model on the entire dataset does not utilise the play-style information. It is also noted that the variance in performance increases substantially in our models when the level of corruption increases, indicating that our models are no longer able to specialise on a common behaviour found in clustered subsets.

Table 5 displays the impact of clustering accuracy on the average rewards generated by each trajectory in each data subset (H_k) across all GridWorlds. This type of analysis is only feasible when the underlying reward function is known. Here we observe that when clustering is perfect, the expected rewards from each trajectory in H_k are exactly

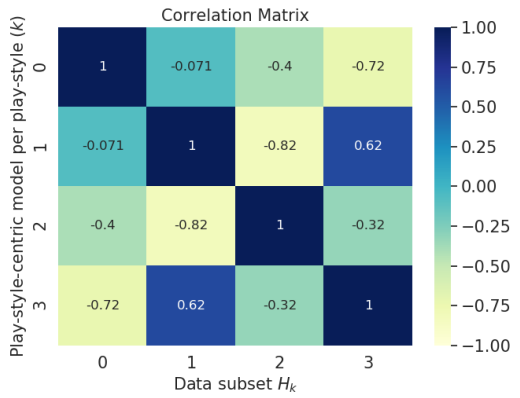


Figure 7: Correlation Matrix depicting relationship between play-style-centric models and data subsets across all GridWorld environments ($E_1, \ldots, E_5$)

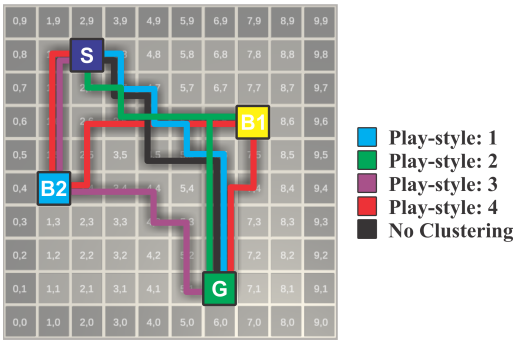


Figure 8: Single trajectory visualisation of behaviours for all Play-style-centric models and the "No Clustering" model for $p = 0.1$ in E_1

the same as those obtained from the original reward functions defined in Table 1, which were utilised to generate the GridWorld trajectories. However, declining clustering performance leads to dispersed average rewards across the reward functions. Off-diagonal values denote noise within H_k, stemming from incorrect trajectory clustering. This noise adversely affects prediction accuracy, as shown in 9.

Discussion and Future Work

Defining or learning a set of play-style-centric behaviours can be challenging due to the subjective nature of play-styles and the diversity of play-style preferences among players. Additionally, it can be difficult to identify and quantify the defining characteristics of each play-style and create a suitable framework for learning these behaviours. Although our focus was solely on grid-based environments, it is essential to highlight that their state space encompassed multiple features beyond just "x" and "y" coordinates. Furthermore, we took into account the temporal nature of these environments and did not overlook it. We also believe that the features we utilised for clustering had the potential to capture variations in a player's style. As a result, there is a possibility to apply this approach in more complex environments and will

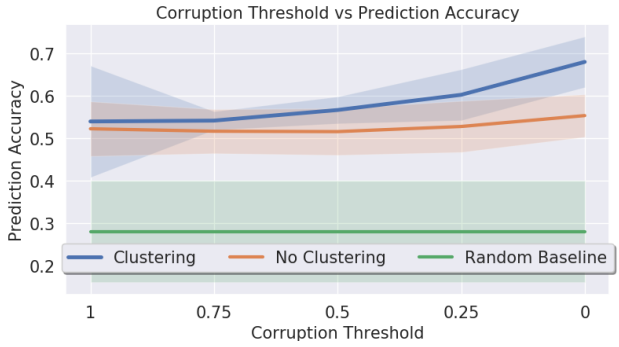


Figure 9: Comparison of the effect of differing the corruption threshold percentage on the models' accuracy. Here "1" indicates random clustering and "0" perfect clustering

be the focus of future work. Moreover, it is worth noting that in many games, access to raw underlying state information is often limited. However, with the advancement of transformer-based models (Wolf et al. 2020), it would be intriguing to explore the application of such an approach to image-based trajectories. This would facilitate easier integration of the system into games. One prospective avenue for future research, which entails greater complexity and access to natural data, centers on the game of Chess. This research could explore the representation of distinct playing styles through the utilisation of well-known Chess openings.

Conclusion

This paper proposes an approach to generating a diverse set of play-style-centric agents that exhibit expected behaviours corresponding to their play-style. Our approach involves utilising unsupervised clustering to separate multi-dimensional varying-length gameplay trajectories by play-style. The resulting trajectory subsets are used to train models using behavioural cloning, enabling the learning of representative policies. This study demonstrates that the separation approach results in models that specialise in the subsets of trajectories on which they are trained and outperform the base case where no separation was used. This paper also demonstrates that model diversity can be achieved through the scaling of a performance threshold, which allows for the learning of agents with varying skill levels. This scalability is useful for developers in creating realistic behaviours that can be more easily scaled in terms of difficulty. Furthermore, this is accomplished without the requirement for designing or learning reward functions, which can be a challenging and non-intuitive process. Notably, this study shows that clustering accuracy has an impact on performance, but decreased clustering accuracy only results in the model prediction accuracy tending towards the baseline. These results were obtained through testing in both synthetic and natural domains, highlighting the method's robustness and generalisability. In conclusion, our proposed PCPG model, which involves both unsupervised clustering and behavioural cloning, is an effective approach for the creation of a diverse set of play-style-centric policies.

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.

Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; and Mané, D. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*.

Arendse, L.; Ingram, B.; and Rosman, B. 2022. Real Time In-Game Playstyle Classification Using A Hybrid Probabilistic Supervised Learning Approach. In *The Third Southern African Conference for AI Research Proceedings. Part of the book series: Communications in Computer and Information Science, Springer, December 2022*. Springer.

Arrabales, R.; Muñoz, J.; Ledezma, A.; Gutierrez, G.; and Sanchis, A. 2012. A machine consciousness approach to the design of human-like bots. *Believable Bots: Can Computers Play Like People?*, 171–191.

Arzate Cruz, C.; and Ramirez Uresti, J. A. 2018. HRLB: A Reinforcement Learning Based Framework for Believable Bots. *Applied Sciences*, 8(12): 2453.

Bakker, P.; Kuniyoshi, Y.; et al. 1996. Robot see, robot do: An overview of robot imitation. In *AISB96 Workshop on Learning in Robots and Animals*, volume 5. Citeseer.

Bauckhage, C.; Drachen, A.; and Sifa, R. 2014. Clustering game behavior data. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(3): 266–278.

Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Dkebiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. 2019. Dota 2 with large scale deep RL. *arXiv preprint arXiv:1912.06680*.

Drachen, A.; Canossa, A.; and Yannakakis, G. N. 2009. Player modeling using self-organization in Tomb Raider: Underworld. In *2009 IEEE symposium on computational intelligence and games*, 1–8. IEEE.

Gorman, B.; and Humphrys, M. 2007. Imitative learning of combat behaviours in first-person computer games. *Proceedings of CGAMES*.

Guzdial, M.; and Riedl, M. 2016. Game level generation from gameplay videos. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Halina, E.; and Guzdial, M. 2022. Diversity-based Deep Reinforcement Learning Towards Multidimensional Difficulty for Fighting Game AI. *arXiv preprint arXiv:2211.02759*.

Harmer, J.; Gisslén, L.; del Val, J.; Holst, H.; Bergdahl, J.; Olsson, T.; Sjöö, K.; and Nordin, M. 2018. Imitation learning with concurrent actions in 3d games. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8. IEEE.

Hester, T.; Vecerik, M.; Pietquin, O.; Lanctot, M.; Schaul, T.; Piot, B.; Horgan, D.; Quan, J.; Sendonaris, A.; Osband, I.; et al. 2018. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

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.

Hussein, A.; Gaber, M. M.; Elyan, E.; and Jayne, C. 2017. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2): 1–35.

Ingram, B.; Rosman, B.; van Alten, C.; and Klein, R. 2022. Play-style identification through deep unsupervised clustering of trajectories. In *2022 IEEE Conference on Games (CoG)*, 393–400. IEEE.

Jacob, A. P.; Wu, D. J.; Farina, G.; Lerer, A.; Hu, H.; Bakhtin, A.; Andreas, J.; and Brown, N. 2022. Modeling strong and human-like gameplay with KL-regularized search. In *International Conference on Machine Learning*, 9695–9728. PMLR.

Justesen, N.; González-Duque, M.; Cabarcas, D.; Mouret, J.-B.; and Risi, S. 2020. Learning a behavioral repertoire from demonstrations. In *2020 IEEE Conference on Games (CoG)*, 383–390. IEEE.

Khalifa, A.; Isaksen, A.; Togelius, J.; and Nealen, A. 2016. Modifying mcts for human-like general video game playing.

Kim, B.; Farahmand, A.-m.; Pineau, J.; and Precup, D. 2013. Approximate Policy Iteration with Demonstration Data. *RLDM 2013*, 168.

Lopes, M.; Melo, F. S.; and Montesano, L. 2007. Affordance-based imitation learning in robots. In *2007 IEEE/RSJ international conference on intelligent robots and systems*, 1015–1021. IEEE.

Marom, O.; and Rosman, B. 2018. Belief reward shaping in reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Moerland, T. M.; Broekens, J.; Plaat, A.; Jonker, C. M.; et al. 2023. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1): 1–118.

Pearce, T.; and Zhu, J. 2022. Counter-strike deathmatch with large-scale behavioural cloning. In *2022 IEEE Conference on Games (CoG)*, 104–111. IEEE.

Ramachandran, D.; and Amir, E. 2007. Bayesian Inverse Reinforcement Learning. In *IJCAI*, volume 7, 2586–2591.

Ranchod, P.; Rosman, B.; and Konidaris, G. 2015. Nonparametric bayesian reward segmentation for skill discovery using inverse reinforcement learning. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 471–477. IEEE.

Ross, S.; Gordon, G.; and Bagnell, D. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 627–635. JMLR Workshop and Conference Proceedings.

Sephton, N. 2016. *Applying Artificial Intelligence and Machine Learning Techniques to Create Varying Play Style in Artificial Game Opponents*. Ph.D. thesis, University of York.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the

game of Go with deep neural networks and tree search. *nature*, 529(7587): 484–489.

Tucker, A.; Gleave, A.; and Russell, S. 2018. Inverse reinforcement learning for video games. *arXiv preprint arXiv:1810.10593*.

Uchibe, E. 2018. Model-free deep inverse reinforcement learning by logistic regression. *Neural Processing Letters*, 47(3): 891–905.

Valls-Vargas, J.; Ontanón, S.; and Zhu, J. 2015. Exploring player trace segmentation for dynamic play style prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 11, 93–99.

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.

Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, 38–45.

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

Ziebart, B. D.; Maas, A. L.; Bagnell, J. A.; Dey, A. K.; et al. 2008. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, 1433–1438. Chicago, IL, USA.