Generating Interpretable Play-Style Descriptions Through Deep Unsupervised Clustering of Trajectories

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

***Abstract*—In any game, play style is a concept that describes the technique and strategy employed by a player to achieve a goal. Identifying a player's style is desirable as it can enlighten players on which approaches work better or worse in different scenarios and inform developers of the value of design decisions. In previous work, we demonstrated an unsupervised LSTM-autoencoder clustering approach for play-style identification capable of handling multidimensional variable length player trajectories. The efficacy of our model was demonstrated on both complete and partial trajectories in both a simulated and natural environment. Lastly, through state frequency analysis, the properties of each of the play styles were identified and compared. This work expands on this approach by demonstrating a process by which we utilize temporal information to identify the decision boundaries related to particular clusters. Additionally, we demonstrate further robustness by applying the same techniques to *MiniDungeons*, another popular domain for player modeling research. Finally, we also propose approaches for determining mean play-style examples suitable for describing general play-style behaviors and for determining the correct number of represented play-styles.**

***Index Terms*—Play-style identification, player modeling.**

I. Introduction

A PLAY style is defined as a particular way of playing a game and, typically, reflects the preferences with which a player may engage in progressing through the game. For example, a player may have a preference for exploring the minutiae of a game, or for completing it as quickly as possible. The diversity of possible play styles can be significant, meaning that different players could play a game in very different ways. There are many potential benefits to identifying the play style of an individual player as they engage in the game [2]. From the player's perspective, identifying their style could be used to assist in the tailoring of game mechanics in real-time for the needs and preferences of the player. Additionally, designers gain insight into how their players are interacting with gameplay features and mechanics. There is an added benefit of being able to tailor tutorial mechanics and in-game tips to the type of player identified.

Multiple studies have looked into modeling the style in which an individual engages with a problem and is not a concept unique to games. For example, in learning, studies have been conducted to identify learning styles [3]. Similar studies, specifically in games, have looked to identify player archetypes, as well as personality characteristics and motivations, [4], [5], [6]. Although work has been done in terms of play-style identification, there has been little exploration of applying these techniques to complex trajectory-based data. Without considering the temporal dimension of a user's playthrough we lose the ability to understand how the player arrived at a certain state and instead, the focus is purely on that final state. By doing so we could reveal further insights lost when only utilizing high-level features. This approach has the added value of allowing us to assign labels at different times along the trajectory that more accurately represent the individual. In addition, video games by nature are time series domains with decision-making occurring and changing through the course of gameplay. Therefore, it is fitting to model play-styles using temporal models.

One possible reason for a limited amount of existing work focusing on temporal data could be the lack of this kind of trajectory data labelled with respect to style. This makes applying supervised learning approaches difficult and as a result, we developed and evaluated our unsupervised model on a generated dataset where ground truths were known. The goal of this approach is to improve the credibility of the results obtained from natural datasets where play-styles are unknown.

Limited access to data is not the only issue in trajectory analysis that requires further investigation. Another issue is how to preprocess and analyze large quantities of multidimensional trajectories of variable length. Nonmachine learning applications of trajectory analysis have traditionally only been performed on points moving in 1-D, 2-D, or 3-D spaces [7]. Video game state information, however, is not limited to such low dimensions. In machine learning, the development of recurrent neural networks has allowed for solving complex problems using sequence-based data. Most notable applications are in video tagging [8], generating image descriptions [9], speech modeling [10], language modeling [11] as well as video-games [12]. These architectures allow for the processing of multidimensional trajectory data.

Manuscript received 31 October 2022; revised 8 June 2023; accepted 8 July 2023. Date of publication 26 July 2023; date of current version 15 December 2023. *(Corresponding author: Branden Ingram.)*

The authors are with the School of Computer Science and Applied Mathematics, University of the Witwatersrand Johannesburg, Johannesburg 2000, South Africa (e-mail: branden.ingram@gmail.com; Clint.VanAlten@wits.ac.za; richard.klein@wits.ac.za; Benjamin.Rosman1@wits.ac.za).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TG.2023.3299074.

Digital Object Identifier 10.1109/TG.2023.3299074

Additionally, interpreting generalized behaviors from large datasets remains an issue when trying to identify play-style. This has been done with nontrajectory-based data (summary statistics) within video games to describe player roles [13]. Creating such a technique for trajectory data requires compressing both the spatial and temporal distributions. Such descriptions have recently been obtained by utilizing clustering techniques [14], [15]. These descriptions offer useful insight into characteristics and trends present in the underlying data, however, they are yet to be applied to video game domains as an informative tool for both player and designer.

In this article, we address both the problem of processing complex data and that of finding interpretable descriptions of behaviors. These descriptions should look to describe some general characteristics or patterns exhibited by a group. To that end, we propose a novel system for play-style identification through the clustering of multidimensional variable-length trajectories in video games and demonstrate that these clusters represent varying styles of behaviors. The core of this system is a specialized LSTM-autoencoder that utilizes the benefits of recurrent neural network [16] architectures to handle sequence-based data. We evaluate this model on a generated benchmark dataset as well as a natural domain. We also demonstrate the unique ability of our model to identify the style in both complete and partial trajectories without the need for any additional engineering or training time. We utilize our model to uncover the characteristics of each play-style through state-based cluster analysis. This analysis identifies similar, differing, and unique states across the clusters. Additionally, we present an approach to identify decision boundaries that separate different play-styles. Finally, we demonstrate a technique to determine the appropriate number of play-styles and generate generalized behavioral descriptions for each style.

II. Related Work

Traditionally, play-style identification has been approached through player modeling, which is the study of computational models of players in games. This includes the detection, modeling and prediction of human player traits which are manifested through cognitive, affective and behavioral patterns [17]. The techniques utilized in player modeling usually fall into one of two groups: 1) model-based; or 2) model-free.

A. Model-Based Approaches

In model-based approaches, a player model is built on a theoretical framework whereby the preexisting understanding of the domain is leveraged [17]. Model-based approaches have been inspired by cognitive frameworks [18] as well as general theoretical frameworks of behavioral analysis such as usability theory [19]. Additional examples are models which utilize theories of "fun" [20], [21]. These top-down approaches have also been used to dynamically affect player experience [22], [23], [24]. Although model-based techniques like these are useful, our work seeks to avoid the use of any prior knowledge of the domain and in essence, learn the heuristic classifier in a model-free setting. Doing so has two benefits, with the first being we are not required to impart any domain-based bias, and the second is the ability to learn a more accurate model than a handcrafted one.

B. Model-Free Approaches

Model-free approaches refer to the construction of a mapping (model) between (player) input and a player state representation. In this case, there is no preexisting understanding of the model; rather, it is learned through an iterative process [17]. To achieve this, observations are collected and analyzed to generate a model without strong initial assumptions of its structure. In model-free approaches, we see attempts to model and predict player actions and intentions [25], [26]. Thue et al. [25] implemented a system that learns a label for the player representing their style. However, this required the manual annotation of the different "encounters" with their corresponding style. Our proposed approach looks to learn these play styles in a completely unsupervised setting.

Data mining efforts to identify different behavioral playing patterns within a game have also been implemented using bottom-up approaches [6], [27]. Drachen, Canossa, and Yannakakis [6] used emergent self-organizing maps to classify players into four style categories based on high-level game behavior data. This analysis did not rely on preexisting information or external factors but required identifying cluster characteristics. It is this statistical analysis of the separated data that revealed the semantics behind each cluster and allowed for the identification of player archetypes. We perform a similar step, but unlike previous studies which used metadata summaries, we analyze raw trajectory data without extensive feature engineering.

C. Trajectory Clustering

An approach to the analysis of trajectory data is comparing and grouping based on whole or partial trajectory attributes using a similarity measurement. However, it has been demonstrated that clustering using subsequences lacked meaning as the generated cluster centres are not dependent on the input [28]. Additionally, when clustering trajectories, the choice of similarity measure is important as it should consider both the spatial and temporal features [29]. Common distance measures include Hausdorff distance, dynamic time warping (DTW) [30], Euclidean distance, and longest common subsequence (LCSS) [31]. The choice of metric is dependent on the structure of the data. DTW and LCSS in particular can measure similarities between varying length trajectories.

Techniques such as k-means and hierarchical clustering have been utilized to perform the clustering step [32]. However, these techniques tend to be ineffective when input dimensions are high [33]. Additionally, model-based methods which use statistical approaches (COBWEB [34]), Neural Network approaches (ART [35]), or self-organizing maps (SOM [36]) have been utilized. SOM clustering of time-series features is unsuitable for trajectories of unequal length, as the dimensions of the weight vectors are fixed [37]. Furthermore, data compression techniques utilizing Latent Dirichlet Allocation (LDA) have been applied to play-style clustering [38]. However, by considering

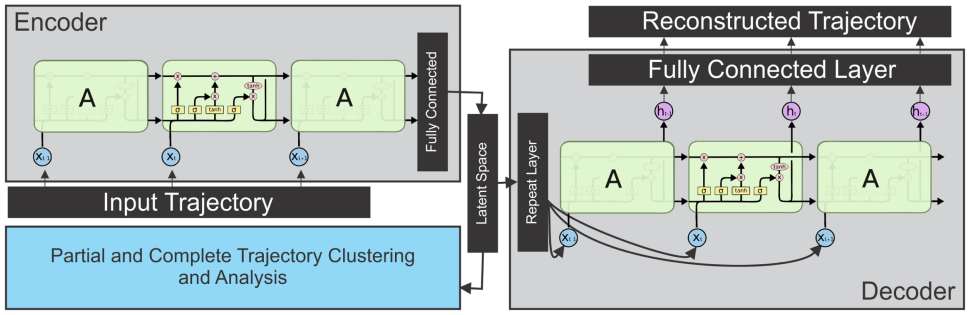

Fig. 1. Play-style identification model architecture.

timesteps as individual data points, LDA approaches ignore the structural significance of temporal data.

In addition to these, deep learning techniques have been applied to unsupervised clustering [14]. Xie, Girshick, and Farhadi [14] used an autoencoder which allowed for the important data compression aspect which was required to make the previously mentioned clustering techniques more feasible. This makes the clustering task easier since clustering can be performed on the encoded data. Xie, Girshick, and Farhadi [14] pretrained a network to minimize the reconstruction loss, and then separately minimize the clustering loss. This separate clustering step used the KL-divergence [39] between a target distribution and an estimated distribution generated from the encoded data. More recently, LSTM autoencoders have been implemented which are capable of handling time-series data effectively [40]. This is a similar approach to Xie, Girshick, and Farhadi [14] with the added LSTM layers to handle time-series data as well as joint minimization of the reconstruction and clustering aspects. Concurrent autoencoder approaches have been utilized for play-style discovery [41] whereby the latent space was used along with ground truth labels in a supervised fashion. It is not common within video game domains for there to exist labelled trajectories with respect to play-style. Therefore, the utilization of supervised approaches limits how easily a model can be applied and as a result, we opt to employ an unsupervised approach. We utilize a similar but unsupervised approach to handling our video game-based trajectories which have the added characteristic of being variable in length as well as being multidimensional.

III. Methodology

We aim to solve the following problem: Given a set of trajectories (X), can we identify a label (y) that categorizes each trajectory $(x_i \in X)$ according to a unique style of play?

A. Play-Style Identification

Our unsupervised approach is based on two key steps. First, we utilize an LSTM-autoencoder network to project a trajectory into a lower-dimensional latent representation. We then perform clustering on this latent space to discover clusters corresponding to related trajectories in this space. The full system is depicted in Fig. 1.

Algorithm 1: Preference-Based Trajectory Generation (PBTG).

Procedure PBTG Environment E
  Define a set of reward functions R
  Initialize our set of trajectories $T = \{\}$
  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 $t(n, r)$ from π_r^n and append to T
    end for
  end for
end Procedure

1) Trajectory Encoding: An autoencoder works to reconstruct each original input trajectory $(x_i \in X)$ after first encoding it as a lower-dimensional state $(z_i \in Z)$. Formally, this is given by (1).

$$z_i = \texttt{Encoder}(x_i) \texttt{ and } x_i' = \texttt{Decoder}(z_i) \tag{1}$$

Our specific model is a temporal autoencoder containing non-stacked LSTM layers similar to Xie, Girshick, and Farhadi [14]. This allows the processing of varied length trajectories by feeding the state at each time step into its own LSTM cell. These cells ("A" in Fig. 1), learn to pass on the important information in sequence until finally outputting a fixed-sized vector representative of our latent space (Z). The latent representation is then decoded using (1) to obtain x_i', the reconstructed trajectory. The network is trained using back-propagation through time [16].

2) Trajectory Clustering: Having projected the trajectories into the latent space, we then cluster them. This clustering step is performed on the set of all generated pairs of (x_i, z_i), where $x_i \in X$ and z_i is the output of the Encoder in (1). Each pair (x_i, z_i) is clustered with respect to z_i to form predicted labels y_i'. Since z_i is a representation of x_i we can use y_i' as the cluster label for the original data. Clustering using Z enables the use of most clustering algorithms as there is no longer an issue of varying length or temporal features.

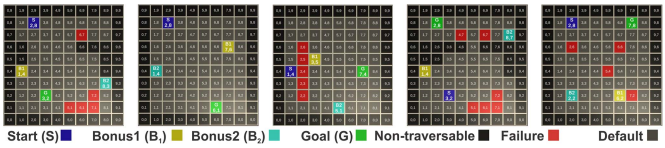

Fig. 2. Randomly generated grid world environments $E_1, \ldots, E_5$.

TABLE I
OBSERVABLE PLAY-STYLES AND REWARD STRUCTURE

R	Behavior	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 and B_2 before G	100	50	50

IV. EXPERIMENTS

A. Datasets

To validate the robustness of our method, we evaluate our model on three different datasets (*GridWorld*, *MiniDungeons*, and *Mario*). Both the *GridWorld* and *MiniDungeons* domains are 2-D grid-like domains where a player seeks a goal with the opportunity of completing other 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 an unlabelled set of trajectories from the game *Super Mario Bros* [42]. This dataset was collected from individuals and serves to showcase our model's performance in natural domains.

1) Grid World: To evaluate an algorithm for play-style identification, it is important to have multiple trajectories from a set of different play-styles. Trajectory-based datasets labelled according to style do not exist and therefore we generate data to account for this. 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 bots with differing styles [43]. We utilize this approach to generate a set of trajectories with differing performance levels for multiple styles.

Our preference-based trajectory generation (PBTG) trajectory generation approach (Algorithm 1) was used to generate 5 individual datasets T_n from five different environments $(E_1, \ldots, E_5)$ with four varying play-styles (reward functions) present. Each environment is a 10×10 grid world, as depicted in Fig. 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. Using this design we generated data with four play-styles, as described in Table I. The set of reward functions R used to emulate these behaviors is also defined in Table I. 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 an agent for each of the combinations of R and E for 20 000 episodes with discount factor $\gamma = 0.99$ and linear ϵ decay 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).

2) 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 [42]. 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.

3) MiniDungeons: MiniDungeons is a 2-D top-down dungeon exploration game, and is a common benchmark research domain for modeling and understanding human play-styles [44]. We use a dataset of player trajectories generated using six player proxies [45]. The state at each timestep is defined as a 15-D vector which encodes event information up until that current timestep and aims to track higher-level player tactics. Table II describes the different play-styles corresponding to the six player proxies as well as their corresponding behavior with example trajectories depicted in Fig. 3. We combine trajectories generated across multiple different levels to form our training and testing sets.

TABLE II
MINIDUNGEONS PLAYER PROXY BEHAVIORS

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

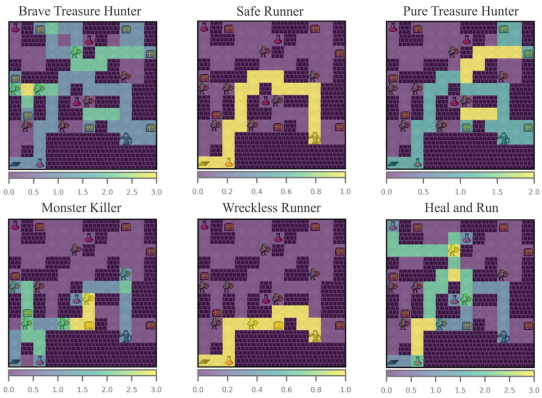

Fig. 3. Example trajectories with state visitation counts (purple → yellow) represented by shaded area for the six player proxies in the *MiniDungeons*.

B. Training

We trained models for seven different environments: $E_1, \ldots, E_5$, *Mario*, *MiniDungeon*. To ensure consistency across domains we used the same parameters for all models. The activation function employed was ReLU. The LSTM cells had output sizes of 20, and the latent vector representation and output layer had a size of 8. The models were trained for 10 000 episodes using the Adam optimizer with a learning rate of 0.001 using mean squared error as the loss function.

C. Clustering

For the clustering step, we evaluated both k-means and Gaussian mixture models (GMMs). For both algorithms, the number of clusters were 4, 6, and 8 for the Grid World, *MiniDungeons*, and *Mario* domains, respectively. For k-means and GMM, we fit our data using 100 restarts and a maximum iteration of 10 000. In addition, k-means used a tolerance of 0.0001 and GMM used a "full" covariance type. Although our model is completely unsupervised, we do compare the ground truth cluster labels (y_i) with the predicted labels (y_i') to validate accuracy on complete trajectories. This validation step uses (2) and (4). First, (2) finds the best match between the cluster assignments from an unsupervised algorithm (y_i') and a ground truth assignment (y_i), where m ranges across all possible one-to-one mappings and n is the number of data points [46].

$$\texttt{Accuracy} = \max_m \left(\frac{\sum_{i=1}^{n} 1\{y_i = m(y_i')\}}{n} \right). \tag{2}$$

Second, (4) defines a confidence measure $(\overline{k})$ giving the probability vector that a given trajectory x belongs to a particular cluster. This is determined using (3) which calculates the Euclidean distance vector $\overline{d}$ between x and C, where C is the set of centroids determined through the clustering step.

$$\overline{d}(x) = ||C - \texttt{Encoder}(x)||_2^2 \tag{3}$$

$$\overline{k}(x) = \frac{exp(\overline{d}^{-1}(x))}{\sum exp(\overline{d}^{-1}(x))}. \tag{4}$$

Additionally, since identifying play-styles on partial trajectories is a key feature, we need to perform a similar evaluation step on these partial trajectories. To this end, we calculate the total accuracy using Algorithm 2 as the average correctly labelled predictions for every trajectory $(t \in T_n)$. In this case, our predicted label (y') is determined by first calculating the weighted moving average confidence $(\texttt{WMAC})$ [47] over all partial trajectories $(p \in t)$ using (5). Here, $\overline{k}_i$ represents the confidence calculated using (3) and (4) as $\overline{k}(p)$ where $p \leftarrow t[0:i]$ and the weighting $w \leftarrow \frac{m(m+1)}{2}$ where m is the length of t. The cluster with the highest confidence is then selected as our predicted label and compared to the corresponding ground truth label y. This process is then repeated for all $(t \in T_n)$ with our final partial trajectory accuracy (PTA) being the percentage of correct predictions.

Algorithm 2: Partial Trajectory Accuracy (PTA).

Procedure PTAT_n
  for all Trajectories $t \in T_n$ **do**
    $w \leftarrow \frac{(m)(m+1)}{2}$ $\triangleright m$ is the length of t
    Calculate $\overline{\texttt{WMAC}}$ with all partial trajectories $s \in t_m$ using (5)
    $y' \leftarrow \text{argmax}_{1 \le j \le N} \overline{\texttt{WMAC}} \triangleright N$: number of clusters
    If y' matches ground truth y **then**
      $PTA \leftarrow PTA + 1$
    end if
  end for
  $PTA \leftarrow \frac{PTA}{n}$
end Procedure

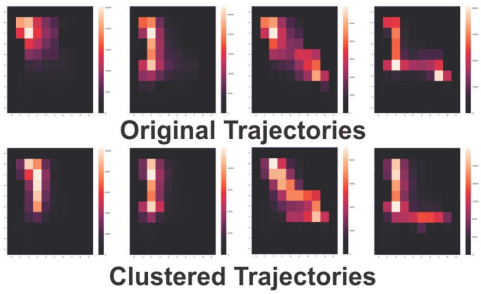

Fig. 4. Heatmap of visited states comparing 8000 clustered trajectories separated by our model (bottom) and original trajectories separated by reward function (top) for E_1.

$$\overline{\texttt{WMAC}} = \sum_{i=0}^{m} \left(\frac{\overline{k}_i \times i}{w} \right). \tag{5}$$

V. Results and Discussion

Through the results of our experimental analysis, we demonstrate the ability of our model to accurately cluster game trajectories into their respective play-styles on both complete and partial trajectories. Additionally, we show how interpretable descriptions for each play-style can be recovered.

A. Complete Trajectory Clustering

To demonstrate the effectiveness of our model in identifying play-styles, we plotted heat maps of the trajectories in each cluster as well as the original trajectories for each $r \in R$ for E_1. In Fig. 4, we observe that there is a correlation between heat maps for both the clustered trajectories and the original trajectories separated by reward function. This shows firstly, that the desired behaviors in Table I are represented in the data through the use of the rewards in Table I. Secondly, we observe that the clustered trajectories depict the same behavior. For example, R_4 sees the agent move to both bonus objectives (top-right in 4) and the same behavior is observed in the corresponding clustered set (bottom-right). For quantitative analysis, we directly compared the set of all predicted labels (y') with

TABLE III
COMPLETE AND PARTIAL TRAJECTORY CLUSTERING ACCURACY

E	Random	Complete GMM	Complete kmeans	Partial GMM	Partial kmeans
E_1	0.25	**0.653**	0.598	0.499	**0.534**
E_2	0.25	0.707	**0.714**	0.602	**0.706**
E_3	0.25	**0.778**	0.705	**0.749**	0.670
E_4	0.25	**0.709**	0.706	0.576	**0.639**
E_5	0.25	**0.778**	0.652	**0.686**	0.678
MiniDungeons	0.17	**0.589**	0.489	**0.446**	0.419

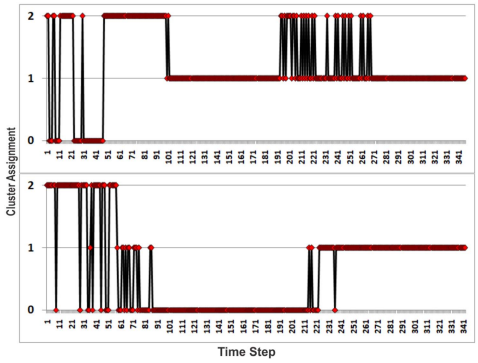

Fig. 5. Change in play-style prediction over time for two particular trajectories from the *Mario* dataset.

the set of all ground truth labels (y) for each Environment $(E_1, \ldots, E_5, MiniDungeons)$ using (2). This resulted in the clustering accuracies shown in Table III for both clustering algorithms. Table III demonstrates our model's ability to accurately cluster play-styles from completed trajectories across multiple varying environments. In particular, we note, weaker performance within the *MiniDungeons* domain resulting from the higher complexity due to the dataset containing trajectories from multiple levels.

B. Partial Trajectory Clustering

To investigate the ability to identify play-styles during gameplay, we clustered partial trajectories and measured the change in clustering confidence as a function of time. This was achieved using Algorithm 2 with the results depicted in Table III. We observe that an environment where trajectories are initially similar such as E_1 has a lower clustering performance. This indicates that the play-styles are initially well aligned while only diverging after some time.

Fig. 5 depicts the change in cluster assignment over time for two particular trajectories from the *Mario* domain. Here we observe that two initially different trajectories converge to the same play-style. Based upon initially differing behaviors it is natural for them to change over time and even converge, based on the changing dynamics of a video game. The occurrence of fluctuations indicates regions of uncertainty in our model indicative of cross-over regions between play-styles. This is the main benefit of identifying style temporally as developers or players have a far denser signal indicating where and when styles become uniquely distinguishable.

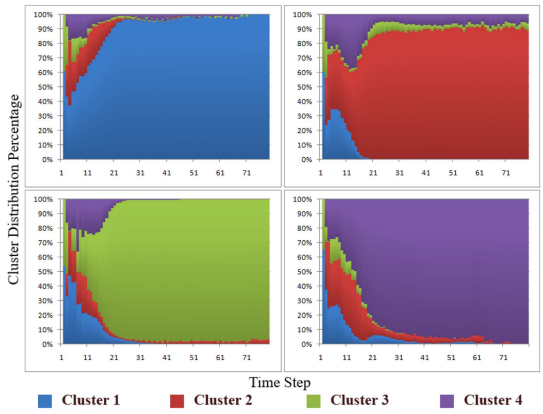

Fig. 6. Clustering assignment over time separated by identified clusters over all five grid environments.

To demonstrate the efficacy of our partial clustering over time we analyzed all trajectories separated by the identified clusters across all five grid environments $(E_1, \ldots, E_5)$. This aggregation is possible since the play-styles we are trying to recover are the same across these different environments, as they would be across multiple levels in a video game. Fig. 6 depicts the results of this, where we observe that for all environments the clustering assignment converges to a unique cluster. This demonstrates that across multiple trajectories our approach to partial trajectory clustering produces consistent desired results.

The PTA (Algorithm 2) is shown in Table III and considered every partial trajectory (p) starting with just the first state in the trajectory $(p = t[0])$, then gradually expanding it to include more states $(p = t[0:1], p = t[0:2], \ldots, p = t[0:n])$, where n was the length of trajectory t. The accuracy does decrease using partial trajectories, but not by a significant amount. This decrease, however, is to be expected with play-styles sharing similarities, most notably the start location. Taken together, these results demonstrate the robustness of the clustering model to unseen data within the domain as well as demonstrating our ability to quickly identify the correct play-style.

VI. IDENTIFYING PLAY-STYLE CHARACTERISTICS

In the case of our grid world domain, we have shown that we can group trajectories based on an expected semantic meaning encoded into the data. However, it is not commonly the case that the semantic meaning behind clusters is known beforehand. The characteristics of each cluster need to be identified for the grouping to be useful for developers or players. We analyze the frequency (f_k) of state (s) occurrences across the identified clusters (k) to identify state-based play-style characteristics defined by (6). A state is an individual time step within a trajectory. In (6), $\sigma, s_i \in S$, where S is the set of all possible states, and n denotes the number of different states. To identify play-style characteristics we identify the shared, differing and unique states across all play-styles.

$$f_k(\sigma) = \sum_{i=1}^{n} 1[s_i = \sigma]. \tag{6}$$

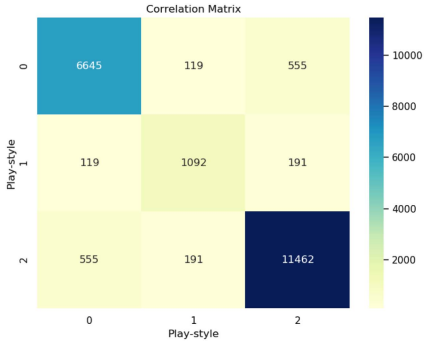

Fig. 7. Frequency of shared states observed relative to all play-styles for *MARIO* (excluding nonvisited states).

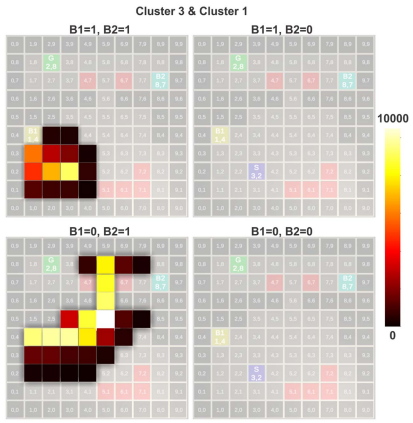

Fig. 8. Shared states observed between Cluster 3 and 1 for E_4 (excluding nonvisited states).

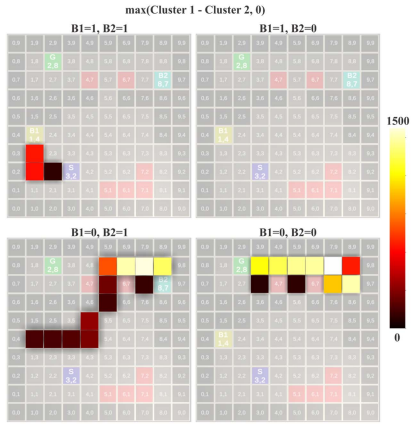

Fig. 9. States found in Cluster 1 without any of shared states with Cluster 3 ($\max(f_1 - f_3, 0)$).

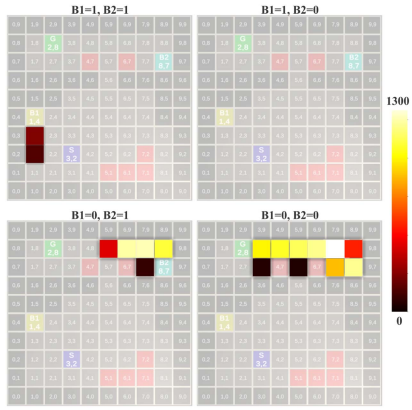

Fig. 10. Unique states for Cluster 1 for E_4 (excluding nonvisited states)

1) Shared States: The shared states between play-styles are calculated as the combined frequency for each common state. A state is said to be common if it exists within trajectories in both clusters. Frequencies are combined by selecting the minimum frequency between the two clusters for each particular state (σ). For example, the similar states between clusters 1 and 3 as seen in Fig. 8, are computed as $\min(f_1(\sigma), f_3(\sigma))$. Here, it is observed that both play-styles congregate around B_1 and after obtaining B_1 which results in the change in state to $B_1 = 0$, both play-styles head towards the goal. It is also outside of these regions, where $\min(f_1(\sigma), f_3(\sigma)) = 0$, which we can conclude that these play-styles diverge.

We conducted an analysis of the number of shared states for all possible play-style pairs of f_k in *MARIO*. Our findings are presented in Fig. 7, where there is a strong correlation along the diagonal. This significant correlation provides evidence that our model effectively separates trajectories into distinct diverse clusters.

2) Differing States: The difference between two play-styles at a state-based level is calculated as $\max(f_k - f_h, 0)$. Conceptually, this can be considered the frequency of state occurrences for a given play-styles k given this state does not occur for play-style h. The conclusion of divergence mentioned above is redemonstrated here in Fig. 9 which depicts the differing states between clusters 1 and 3. Here it is observed that the states surrounding B_2 are found in cluster 1 but not in cluster 3 which also corresponds with the expected behavior. This behavior is to go to the goal after obtaining B_1 rather than also going to obtain B_2.

3) Unique States: To determine the states which are unique to a particular play-style we solve for $\max(f_k(\sigma) - (\sum_{h=0}^{3}[f_h(\sigma)]; h \neq k), 0)$. Here, uniqueness emerges when a particular state occurs more than the total frequency for that state in all the other clusters. In particular, we observe in Fig. 10 that the unique states for cluster 1 reside in the top right of the map for E_4 when $B_1 = 0, B_2 = 1$ as well as when $B_1 = 0, B_2 = 0$ which expectantly corresponds to behavior 4 in Table I. These unique states for cluster 1 ($k = 1$) were calculated as $\max(f_1(\sigma) - (\sum_{h=0}^{3}[f_h(\sigma)]; h \neq 1), 0)$.

By observing the unique states as well as where the clusters are similar and different we can formulate a deeper understanding of the behaviors associated with each cluster. Applying the same form of analysis to the *Mario* dataset, shown in Table IV, allowed us to discover the meaning behind the identified clusters. For example, we note that cluster 0 corresponds to players who die the least while cluster 2 contains players who are more likely to collect coins while also killing enemies. By engaging in this

TABLE IV
THREE MOST FREQUENT UNIQUE STATES FOR EACH IDENTIFIED CLUSTER FOR ALL THE TRAJECTORIES IN THE MARIO DATASET

	State Description					
	Jumps	Kills	Runs	Coins	Deaths	Action
Cluster 0	17	1	2	3	0	9
	17	1	2	3	0	2
	12	1	0	2	0	2
Cluster 1	11	7	0	0	1	9
	11	7	0	0	1	2
	8	7	0	0	1	2
Cluster 2	38	8	0	8	2	9
	38	8	0	8	2	9
	49	8	0	8	1	9

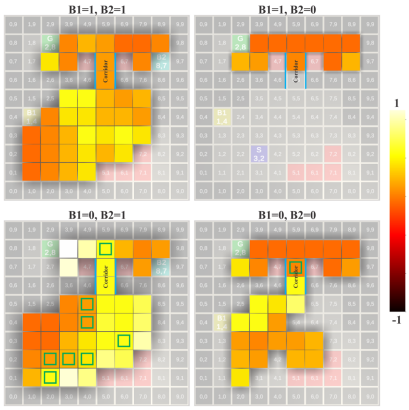

Fig. 11. Decision boundary points indicated by green squares for Cluster 1.

behavior we see that they are more likely to die. Lastly, cluster 1 players tend to jump the least while also ignoring picking up any coins.

VII. IDENTIFYING STYLE DECISION BOUNDARIES

Using state-based frequency analysis for unique state discovery allows us to understand the regions in which players of a particular play-style should occupy. However, this approach does not reveal what decisions said player will make during gameplay to reside in those locations. The limitation of utilizing state-based frequencies for comparison is that this process ignores the temporal nature of trajectories. We incorporate this temporal knowledge into our play-style characteristic identification approach by locating the states which constitute the boundaries between play-styles. It is at these points where players decide to perform some action which is highly indicative of a particular play-style. This is achieved by determining the confidence in each play-style over time for the trajectories separated by our model. This confidence is calculated using (4) with the value stored in the final state σ of the current trajectory x. The neighbours of each state are also determined as any state which is a single time-step away. Therefore, a state is considered to be a boundary point if the variance between the confidence values of its neighbors exceeds a threshold parameter. In Fig. 11, we observe that when $B_1 = 0$, $B_2 = 1$ there is a transition boundary after moving up through the middle corridor where a player could then either go left or right. Here, the play-style prediction accuracy greatly increases if one chooses to

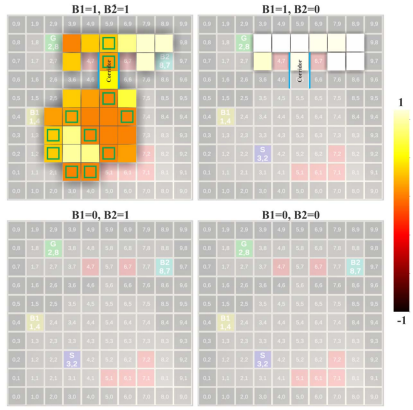

Fig. 12. Decision boundary points indicated by green squares for Cluster 2.

travel left while the accuracy for other actions is far less. This is per the behavior of this particular cluster which is, as previously indicated, to obtain B_1 and head to the goal. The converse can be observed in Fig. 12 which represents the boundary points for cluster 2. For this case, the behavior was to obtain B_2 followed by going to the goal. Once again after a player gets to the end of the corridor they need to decide whether to go left, right or back down, however, once the player decides to go right, the prediction accuracy greatly increases as opposed to following any of the other actions. As a result of this designers would only need to identify how a particular player acts around a small finite set of decision points to classify a player's behavior.

VIII. DETERMINING THE NUMBER OF PLAY-STYLES

A limitation of clustering lies in the choosing of k (expected number of clusters). By choosing k our resultant number of play-styles is not guaranteed to represent the true number of unique play-styles within a certain domain. This is a well-known issue related to both k-means and GMM approaches. A common solution, which we have also used previously [45], has been to employ the ELBOW technique [48]. However, we can utilize the identified shared, differing, and unique characteristics of each cluster to validate the correct number of clusters. By considering whether the frequency of the most unique state was too large or small we could raise and lower the number of clusters we identified. For example, if the frequency of unique states was very low, the particular cluster would be considered too similar to another. In particular, this is achieved by identifying for which value of k the average uniqueness across clusters first stops decreasing. Fig. 13 depicts the average uniqueness across all the identified play-style clusters for differing values of k calculated using (7). Here, it is observed that for E_5 it is best to use 4 clusters to represent all the different play-styles, such that all clusters have a substantial number or unique states. This approach was also employed to determine the optimal number of clusters for the *MARIO* domain, ultimately determining that the appropriate number of clusters is three.

$$\sum_{k=0}^{n}\left(\max\left[f_k(\sigma)-\left(\sum_{h=0}^{n}[f_h(\sigma)];h\neq k\right),0\right]\right). \quad (7)$$

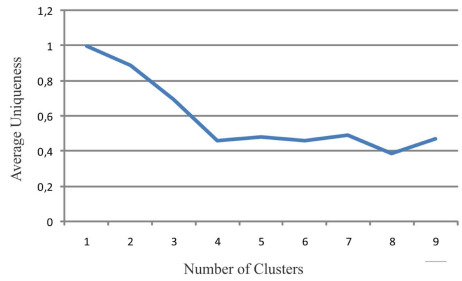

Fig. 13. Average uniqueness across all clusters versus the number of clusters.

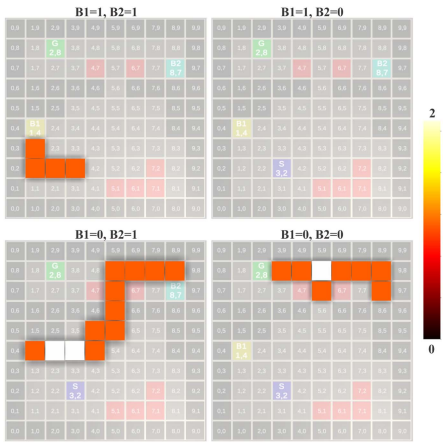

Fig. 14. Mean trajectory for cluster 1 represented as a visitation map.

IX. Mean Play-Style Trajectory

An added benefit of utilizing our autoencoder clustering approach is the ability to utilize the precomputed centroids for each cluster to determine examples of general play-style behaviors. We accomplish this in two ways, the first being to directly feed the centroids through the decoder to generate a mean trajectory. This is possible since the centroid itself exists within the generated latent space of our network. Therefore it can be reconstructed similarly to any other encoded vector. However, this does not guarantee that the trajectory is valid. Our second approach which can be seen in Fig. 14 is to take the closest real encoded trajectory to the centroid and use that as the mean trajectory. This approach to generating mean play-style examples can be used to better describe the expected behavior for a particular style. In this case, we can see the generalized behavior for this play-style is to head to B_1, then B_2, and finally to the goal. Additionally, we can cluster new unseen trajectories using the mean trajectories for each style by applying the longest common sequence (LCS) algorithm [49]. Here the trajectory would belong to the play-style whose mean trajectory shared the LCS of states. By applying this method we obtained results seen in Table V comparable to that of our unsupervised approach, however in some cases, particularly for E_1 and E_5 we observed notably greater clustering accuracy. We note that the high play-style prediction accuracy is due to the mean trajectories for both E_1 and E_5 being highly correlated to the expected general behaviors for each play-style in those environments. Visualizing higher dimensions in this fashion is difficult, however, with this approach, we can obtain state-transition dynamics representative of generalized behavior for both *Mario* and *MiniDungeons*.

TABLE V
Comparison Between LCS Clustering With Mean Trajectories Versus Our LSTM Autoencoder Model

Domain	LCS accuracy	LSTM model accuracy
E_1	0.598	**0.653**
E_2	0.644	**0.707**
E_3	**0.931**	0.778
E_4	0.643	**0.709**
E_5	**0.893**	0.778

X. Conclusion

This article presented an approach to generating interpretable play-style descriptions using an unsupervised LSTM-autoencoder clustering model on variable-length trajectory data. Although our model does not generate the interpretations itself, the descriptions are readily interpretable by designers as per our discussions. We expand upon prior work [1] which demonstrated the ability of our model to accurately recover underlying play-styles in an unsupervised as well as the ability to characterize the identified styles. First, we expanded upon this work by further demonstrating the model's ability to work in the *MiniDungeons* domain. Second, we proposed our technique of identifying the decision boundaries between play-styles which can be further utilized to describe a player's behavior. Third, we dealt with the issue of determining the appropriate number of play-styles within a domain by utilizing our unique state analysis. Last, we showed through the utilization of a play-styles centroid trajectory that generalized mean behaviors can be recovered. All of these components combined go further to describe both an individual's behavior as well as the set of all behaviors possible. Importantly, these descriptions can be created from partial trajectories, crucial for the usage of such methods during gameplay. This component makes it easier for developers and players not only to understand how an individual plays, but also how that individual compares to others of differing styles, thereby allowing players or developers to make more informed decisions on which aspects they should change. In future work, we could look to apply our model to nonstationary players, whereby their behavior changes temporally. Since human players may learn and evolve it would be beneficial to have a model robust to these changes. Future research on identifying play-styles from trajectories could explore two promising avenues: Utilizing image-based autoencoders and modern attention-based models. By leveraging image-based autoencoders, our model can analyze screen recordings instead of state-based trajectory representations. Capturing these visual patterns could improve our understanding of play-styles and enhance the model's integration capabilities. Additionally, incorporating attention mechanisms in our model could also allow us to focus on relevant aspects of gameplay, and capturing intricate relationships between play-styles and trajectories. Lastly, creating an annotated dataset of human play traces encompassing diverse play-styles enriches the training process, enabling our model to learn nuanced patterns and characteristics associated with different play-styles for improved accuracy and generalization.

REFERENCES

[1] B. Ingram, B. Rosman, R. Klein, and C. van Alten, "Play-style identification through deep unsupervised clustering of trajectories," in *Proc. IEEE Conf. Games*, 2022, pp. 393–400.

[2] D. Charles et al., "Player-centred game design: Player modelling and adaptive digital games," in *Proc. DiGRA Conf.*, 2005.

[3] R. Dunn, "Learning style: State of the science," *Theory Into Pract.*, vol. 23, no. 1, pp. 10–19, 1984.

[4] R. Bartle, "Hearts, clubs, diamonds, spades: Players who suit muds," *J. MUD Res.*, vol. 1, no. 1, pp. 6–24, 1996.

[5] N. Yee, "Motivations of play in mmorpgs," in *Proc. DiGRA Conf.*, 2005.

[6] A. Drachen, A. Canossa, and G. N. Yannakakis, "Player modeling using self-organization in tomb raider: Underworld," in *Proc. IEEE Symp. Comput. Intell. Games*, 2009, pp. 1–8.

[7] W. Helland-Hansen and G. Hampson, "Trajectory analysis: Concepts and applications," *Basin Res.*, vol. 21, no. 5, pp. 454–483, 2009.

[8] S. Ilyas and H. U. Rehman, "A deep learning based approach for precise video tagging," in *Proc. IEEE 15th Int. Conf. Emerg. Technol.*, 2019, pp. 1–6.

[9] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3128–3137.

[10] Z. Yu et al., "Using bidirectional LSTM recurrent neural networks to learn high-level abstractions of sequential features for automated scoring of non-native spontaneous speech," in *Proc. IEEE Autom. Speech Recognit. Understanding*, 2015, pp. 338–345.

[11] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Proc. 13th Annu. Conf. Int. Speech Commun. Assoc.*, 2012, pp. 194–197.

[12] P. Bertens, A. Guitart, P. P. Chen, and A. Perianez, "A Machine-learning item recommendation system for video games," in *Proc. IEEE Conf. Comput. Intell. Games*, 2018, pp. 1–4.

[13] C. Eggert, M. Herrlich, J. Smeddinck, and R. Malaka, "Classification of player roles in the team-based multi-player game dota 2," in *Proc. 14th Int. Conf. Entertainment Comput.*. Springer, 2015, pp. 112–125.

[14] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 478–487.

[15] A. Drachen et al., "Skill-based differences in spatio-temporal team behaviour in defence of the ancients 2 (dota 2)," in *Proc. IEEE Games Media Entertainment*, 2014, pp. 1–8.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[17] G. N. Yannakakis, P. Spronck, D. Loiacono, and E. André, "Player modeling," 2013.

[18] J. R. Anderson, C. F. Boyle, and B. J. Reiser, "Intelligent tutoring systems," *Science*, vol. 228, no. 4698, pp. 456–462, 1985.

[19] K. Isbister and N. Schaffer, *Game Usability: Advancing the Player Experience*. Boca Raton, FL, USA: CRC Press, 2008.

[20] T. W. Malone, "What makes things fun to learn? Heuristics for designing instructional computer games," in *Proc. 3rd ACM SIGSMALL Symp. 1st SIGPC Symp. Small Syst.*, 1980, pp. 162–169.

[21] R. Koster, *Theory of Fun for Game Design*. "O'Reilly Media, Inc.", 2013, pp. 11–17.

[22] G. Andrade, G. Ramalho, H. Santana, and V. Corruble, "Extending reinforcement learning to provide dynamic game balancing," in *Proc. Workshop Reasoning, Representation, Learn. Comput. Games, 19th*, 2005, pp. 7–12.

[23] J. K. Olesen, G. N. Yannakakis, and J. Hallam, "Real-time challenge balance in an RTS game using rtneat," in *Proc. IEEE Symp. On Comput. Intell. Games*, 2008, pp. 87–94.

[24] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma, "Difficulty scaling of game AI," in *Proc. 5th Int. Conf. Intell. Games Simul.*, 2004, pp. 33–37.

[25] D. Thue, V. Bulitko, M. Spetch, and E. Wasylishen, "Interactive storytelling: A player modelling approach," in *Proc. AAAI Conf. Artif. Intell. Interactive Digit. Entertainment*, 2007, pp. 43–48.

[26] C. Thurau, C. Bauckhage, and G. Sagerer, "Learning human-like movement behavior for computer games," in *Proc. Int. Conf. Simul. Adaptive Behav.*, 2004, pp. 315–323.

[27] B. G. Weber and M. Mateas, "A data mining approach to strategy prediction," in *Proc. IEEE Symp. Comput. Intell. Games*, 2009, pp. 140–147.

[28] E. Keogh and J. Lin, "Clustering of time-series subsequences is meaningless: Implications for previous and future research," *Knowl. Inf. Syst.*, vol. 8, no. 2, pp. 154–177, 2005.

[29] S. Kisilevich, F. Mansmann, M. Nanni, and S. Rinzivillo, "Spatio-temporal clustering," in *Data Mining Knowl. Discov. Handbook*. Springer, 2009, pp. 855–874.

[30] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proc. 3rd Int. Conf. knowl. Discov. Data Mining,*, vol. 10. Seattle, WA, USA:, 1994, pp. 359–370.

[31] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proc. 18th Int. Conf. Data Eng.*, 2002, pp. 673–684.

[32] M. Nanni, Clustering methods for spatio-temporal data: Ph.D. dissertation, 2002.

[33] M. Steinbach, L. Ertöz, and V. Kumar, "The challenges of clustering high dimensional data," in *New Directions in Statistical Physics*. Springer, 2004, pp. 273–309.

[34] D. H. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Mach. Learn.*, vol. 2, no. 2, pp. 139–172, 1987.

[35] G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Comput. Visi., Graph., Image Process.*, vol. 37, no. 1, pp. 54–115, 1987.

[36] T. Kohonen, "The self-organizing map," *Proc. IEEE Proc. IRE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990.

[37] T. W. Liao, "Clustering of time series data–a survey," *Pattern Recognit.*, vol. 38, no. 11, pp. 1857–1874, 2005.

[38] J. Gow, R. Baumgarten, P. Cairns, S. Colton, and P. Miller, "Unsupervised modeling of player style with LDA," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 3, pp. 152–166, Sep. 2012.

[39] C. Frogner, C. Zhang, H. Mobahi, M. Araya, and T. A. Poggio, "Learning with a wasserstein loss," *Adv. Neural Inf. Process. Syst.*, vol. 28, 2015.

[40] N. S. Madiraju, "Deep temporal clustering: Fully unsupervised learning of time-domain features," Ph.D. dissertation, Arizona State Univ., Tempe, AZ, USA, 2018.

[41] R. Talwadker, S. Chakrabarty, A. Pareek, T. Mukherjee, and D. Saini, "Cognitionnet: A collaborative neural network for play style discovery in online skill gaming platform," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 3961–3969.

[42] M. Guzdial and M. Riedl, "Game level generation from gameplay videos," in *Proc. AAAI Conf. Artif. Intell. Interactive Digit. Entertainment*, 2016, pp. 44–50.

[43] C. Arzate Cruz and J. A. Ramirez Uresti, "HRLB: A reinforcement learning based framework for believable bots," *Appl. Sci.*, vol. 8, no. 12, 2018, Art. no. 2453.

[44] C. Holmgard, A. Liapis, J. Togelius, and G. N. Yannakakis, "Evolving personas for player decision modeling," in *Proc. IEEE Conf. Comput. Intell. Games*, 2014, pp. 1–8.

[45] L. Arendse, B. Ingram, and B. Rosman, "Real time in-game playstyle classification using a hybrid probabilistic supervised learning approach," in *Proc. 3rd Southern Afr. Conf. AI Res. Proc.*Springer, 2022, pp. 60–77.

[46] H. W. Kuhn, "The hungarian method for the assignment problem," *Nav. Res. Logistics Quart.*, vol. 2, no. 1-2, pp. 83–97, 1955.

[47] Y. Zhuang, L. Chen, X. S. Wang, and J. Lian, "A weighted moving average-based approach for cleaning sensor data," in *Proc. IEEE 27th Int. Conf. Distrib. Comput. Syst.*, 2007, pp. 38–38.

[48] R. L. Thorndike, "Who belongs in the family," *Psychometrika*, vol. 18, no. 4, pp. 267–276, 1953.

[49] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. Mol. Biol.*, vol. 48, no. 3, pp. 443–453, 1970.