Real Time In-Game Playstyle Classification Using A Hybrid Probabilistic Supervised Learning Approach

Lindsay John Arendse, Branden Ingram, and Benjamin Rosman

School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg, South Africa.
https://www.wits.ac.za/csam/
Lindsay.Arendse@students.wits.ac.za, ORCID 0000-0002-5134-5637
Branden.Ingram@wits.ac.za, ORCID 0000-0001-7376-1327
Benjamin.Rosman1@wits.ac.za, ORCID 0000-0002-0284-4114

Abstract. In interactive digital media, such as video games, bringing about an adaptive or personalised experience requires a mechanism for correctly classifying or identifying the player style, before attempting to modify the experience in some way that improves player interest and immersion. This work presents a framework for solving this problem of in-game real time playstyle classification. We propose a hybrid probabilistic supervised learning approach, using Bayesian Inference informed by a K-Nearest Neighbors based likelihood, that is able to classify players in real time at every step within a given game level using only the latest player action or state observation. This improves on current approaches dependent on previous episodic player action trajectories in order to classify the player. Furthermore, we highlight the effect that this representation of the player state-action observation has on the in-game playstyle classification's accuracy, prediction stability, and generalisability. We apply and test our framework using MiniDungeons, a rogue-like dungeon exploration game, and further evaluate our framework using a natural dataset containing human player action data from the platforming game Super Mario Bros. The experimental results obtained from our approach outperforms existing work in both domains. Furthermore, the evaluation results highlights the ability of our framework to generalise to unseen levels, without the need for additional retraining. Additionally, the Super Mario evaluation results illustrates the scalability of our framework to a more complex game environment with human player data.

Keywords: Game AI · Playstyle Identification · Playstyles · Player Modeling · Supervised Learning · Bayesian Inference · K-Nearest Neighbour · Rogue-like · Platforming · MiniDungeons · Super Mario Bros.

1 Introduction

In video games there are sometimes several *ways* or *styles* in which players can play a game. Different players find different parts of a game challenging and rewarding. Catering for this diversity in player playstyle, preference, and skill is quite challenging for game creators [3]. Adapting to player preference and skill is important in achieving higher levels of player engagement and is especially vital in games used for education [2]. Furthermore, from a game design perspective, in order for game developers to maximise the number of target players it is essential to create games which cater for different gameplay experiences or player playstyles [22].

A player playstyle can be seen as a representation of the player's strategy and player profile [1]. In order to understand or react accordingly to a given playstyle a model approximation of the player is needed. Approaches within the research area of Player Behaviour Modeling are centered around the creation of this model approximation of players [24]. In other words, a player model can be defined as an abstracted representation of a player's behaviour in a game environment [1]. Player modelling provides a mechanism which game designers and researchers can use to gain insight and understanding into how players are feeling and how players might act [25]. Player modelling in itself is an interesting challenge and problem domain. Player models are additionally useful when combined with game personalisation or game adaptability (also known as *dynamic difficulty adjustment* [4], *adaptive player experience* or *game balancing* [21]). This application of player models to game personalisation or adaptability is of increasing importance in video games and is especially necessary when the purpose of game AI is to improve the experience or enjoyment of the human player [1].

Before attempting to modify the gameplay experience in some way which improves player interest and immersion, a mechanism for correctly classifying or identifying the player preference or player style is required. We present a framework for solving this problem of in-game *real time* playstyle classification. We propose a hybrid probabilistic supervised learning approach, using Bayesian Inference informed by a K-Nearest Neighbors based likelihood, that is able to classify players in real time at every step within a given game level using only the latest player action or state observation. As part of our experiment we compare our hybrid classifier approach to a comparative approach based on unsupervised clustering of the player action trajectories, using an LSTM-autoencoder [12]. Furthermore, we highlight the effect that this representation of the player state-action observation has on the in-game playstyle classification's accuracy, prediction stability, and generalisability.

We apply and test our framework using the MiniDungeons game domain. MiniDungeons is a turn-based top-down tile based dungeon exploration game, created as a benchmark research domain for modeling and understanding human playstyles [8][9]. From an implementation perspective we make use of a python, OpenAI Gym compatible, re-implementation[1] [14]. Our framework is further evaluated using a natural dataset[2] containing human player action data from the platforming game Super Mario Bros [6]. We respectively obtain accuracies

[1] https://github.com/ganyariya/gym-md
[2] http://guzdial.com/datasets.html

and prediction stability results which highlight the success of our framework when compared to existing work. Prediction stability is a necessary feature, not present in related studies, for real time playstyle classification since frequent fluctuations in the prediction can adversely affect the game experience. The rest of the work is organised as follows, Section 2 and 3 describes related work and background, Section 4 outlines our research methodology, and our experiment discussion and analysis is presented in Section 5.

2 Related Work

There are a number of closely related areas of work centered around the problem of creating adaptive or personalised games. Several studies focus on the player model creation process, in other words, modelling or mimicking human players [8][9][10][14]. Other works aim to investigate methods for identifying, classifying, or clustering the player, with the aim of trying to answer the question 'does the observed player belong to a known player type or playstyle?' [5][11][12][16]. There are several other researchers focused on how the game experience can be changed or personalised to the identified player [4][13][20][21]. Our work looks to recover the underlying playstyles present in play logs which could be used to aid all of these studies.

Normoyle and Jensen [16] investigates a method which uses Bayesian semi-parametric clustering for creating player clusters. Normoyle and Jensen take post-match data and cluster based on how the player's choices affect the end game result, in contrast to clustering on the outcomes directly. Iwasaki and Hasebe [14] use a clustering approach, called x-means, to cluster play log data with the aim of evolving different agent playstyles created using a genetic algorithm, also called C-NEAT [11]. We apply this idea of player personas in order to generate our set of rule based player proxies. Additionally, we look to further explore the concept of play logs by analysing the impact different representations have on performance.

Past work in this field has been largely concerned with identifying playstyles at the end of the game episode, in order to evolve or create diverse player persona agents, or to analyse player trace data for analytical or game testing purposes.

The user-study work by Valls-Vargaswork et al. [23] present a player modeling framework to capture non-stationarity within player strategy by using sequential machine learning techniques which incorporate predictions from previous temporal player observations. However, the effect of prediction stability is not featured as a component of their approach. Our work looks to highlight the impact the stability score has when predicting playstyles during gameplay. Additionally, Valls-Vargaswork et al. [23] relied on a manual annotation process for providing playstyle labels to the collected player trace data at fixed intervals. This manual annotation is not only time consuming but may also be infeasible from a cost perspective. Our framework's Trajectory Processing step which utilises player personas and clustering to provide a feasible solution to the lack of labelled player datasets.

Hernandez-Leal et al. [7] also considered the concept of non-stationary strategies whereby they utilised a Bayesian framework to train an agent which learns an optimal policy against an opponent whose strategy switches. In our work we look to apply Bayesian inference to track the belief over a set of playstyles which was inspired by their belief tracking of observed policies.

The work by Ingram et al. [12] looked to remedy the issue of requiring labelled data by utilising an unsupervised lstm autoencoder clustering approach. Their autoencoder model works by projecting trajectory information into a lower dimensional latent representation which could then be clustered using vanilla clustering approaches like Gaussian Mixture Models. This approach, however, does not guarantee that the clusters identified correspond semantically to the playstyles we wish to recover. Additionally the clustering step relies on previous episodic player action trajectories. In contrast we propose a hybrid probabilistic supervised learning approach, using Bayesian Inference informed by a K-Nearest Neighbors based likelihood, that is able to classify players in real time at every step within a given game level using only the latest player action or state observation.

Gow et al. [5] also utilised an clustering approach which incorporates multi-class Linear Discriminant Analysis (LDA) to model players in Snakeotron and Rogue Trooper. Similar to Valls-Vargaswork et al. [23] they relied on play log segmentation and summaries. The ability to classify players at every game step may allow for the game's adaptability to become more responsive and granular than these segmentation based approaches. Our model can accommodate this important capability which is required for assistive companion agents, dynamic difficulty adjustment made during game play, or for tailoring tutoring based games.

The classification of players needs to happen in near real time and concurrently with other computations, such as graphics rendering, non-player character behaviour logic, and game physics [1][24]. Therefore techniques used in real time playstyle identification should strive to be as computationally efficient and inexpensive for applications within game AI [1]. For this reason we have chosen to make use of more computationally efficient algorithms, such as Bayesian Inference and K-Nearest Neighbors search, in building our framework.

Computationally complex classifiers pose a risk at runtime by interrupting or halting the game experience. The user study work by Scott and Khosmood [19] highlights this adverse effect which computationally heavy approaches have on a game's playability and player experience. Scott and Khosmood conclude that approaches which reduces the playability of the game and thus the overall player experience are unacceptable in a commercial video-game setting. Furthermore, from a model explainability perspective understanding the playstyle prediction outcome, identifying which game mechanics or features certain playstyles use, and gaining insight into how the different player type clusters relate are useful to game designers when deciding on which game mechanics to change, and which new game features to implement or remove [12][22]. For this reason, we have chosen explainable machine learning approaches such as Bayesian Inference and K-Nearest Neighbors. As part of our experiment we analyse the player action trajectory data and identify the key generalised behaviours observed. Our methodology section outlines how we collect, pre-process, analyse, train, and evaluate our approach, as well as our definition of the playstyle in-game classification problem.

3 Background

This section describes the terminology and definitions necessary for defining the in-game playstyle classification problem.

3.1 Play Log Definition

Game metrics are the recorded numeric data generated, during gameplay, as a result of the player interacting with the game [22]. Playstyles in games are identified based on the observed in-game event information [14]. This is a reason why the use of a *play log* [11], which captures such event information, is valuable for identify play styles. Thus our classification approaches uses this concept of a play log, which is a vector containing the essential game metrics required to identify or discriminate between playstyles [11] [14].

We define the play log as pl, an n-dimensional vector containing k^n game metrics m: $pl = [m_1, m_2, \cdots, m_k]^n$ where $n, k \in \mathbb{N}$ and $m \in \mathbb{R}$. Let $\varUpsilon$ be an instance of pl, with a specific n, k, and m values. Let $\varPsi$ be an instance of pl containing unseen play log game metric data with the same n, k, and m values as $\varUpsilon$. It is important to note that the game metrics $m_{1...k}$ are recorded and updated at every step within a given level of a game.

3.2 Playstyle Set Definition

Our aim is to build playstyle classifiers which make use of play log metric data to distinguish between different playstyles. During the early stages of game development, especially when mechanics of a game are being tested, changed, and developed it is challenging for game designers to collect game metric data from human players [14]. Furthermore, playtesting with human players can be time consuming and a costly exercise. This is why quite a number of studies create and make use of agents as a proxy for human players [8][10][14]. Hence, we will also make use of proxy agents, also called personas [9][10], to mimic human playstyles within the given game and thus generate the play log metric data. We define a set $\varGamma$ of *proxy agent implemented playstyles*, called $\varGamma \subseteq P_g$, where P_g is the global set of all possible playstyles for a given game g.

3.3 Game Levels

It is common for games to have one or more levels. Thus for a given game, we divide all available levels into three sets: the first called *train* which is used for the training of the playstyle classifiers, the second is called *seentest* which is used for the seen levels prediction evaluation, and the third called *unseentest* which is used for the unseen levels prediction evaluation.

3.4 Playstyle In-game Classification Problem Definition

For a given game g, a player agent α, and a level l from g. We aim to find a model classifier $\varOmega_{\varUpsilon}$ which is trained on labeled play log data, $\varUpsilon$, obtained using the set, $\Gamma \subseteq P_g$, of proxy agents on the set of training levels, *train*, for the game g which can classify unseen play log entries, Ψ, to a given playstyle in Γ.

$$\Omega_\Upsilon : \Psi \rightarrow \Gamma \,,\, (\forall\, step \in l \,, \exists\, \Omega_\Upsilon) \tag{1}$$

In other words, we would like to find a model Ω_Υ which can at every *step*, within a level l, in the game g classify the unseen play log entries in Ψ, of agent α, to a playstyle in Γ. The model classifier Ω_Υ is evaluated using the respective level sets *seentest* and *unseentest*.

4 Playstyle Classification Method

Our proposed framework provides a method for solving the problem of in-game real time playstyle classification. As illustrated in Figure 1 our framework involves multiple steps where low-level player action trajectories are processed to generate corresponding playstyle labels. These steps are broken down into: Trajectory Processing and Playstyle Classification.

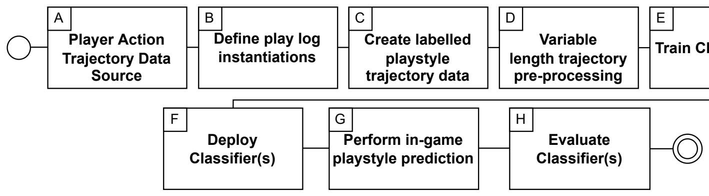

A
Player Action Trajectory Data Source
B
Define play log instantiations
C
Create labelled playstyle trajectory data
D
Variable length trajectory pre-processing
E
Train Classifier(s)
F
Deploy Classifier(s)
G
Perform in-game playstyle prediction
H
Evaluate Classifier(s)

Fig. 1: Proposed Framework Overview

4.1 Trajectory Processing

Our playstyle identification framework is able to account for two situations related to the availability of existing player play log data. Our framework begins with the *Player Action Trajectory Data Source* step, as illustrated in Figure 1(A). Regarding the source data, two situations could occur, which we respectively refer to as Case I and Case II. Case I arises when there is no existing player play log data, thus for Case I we make use of rule-based proxy agents to mimic human player playstyles and generate the source player action trajectory data. Case II corresponds to when there already exists player action trajectory data. The next step in our framework is to define the play log instantiations, as shown in Figure 1(B). As discussed in Subsection 3.1 the play log contains the the essential game metrics required to identify or discriminate between playstyles. For this reason the construction of the play log is important and several play log representations should be considered. As part of this work we highlight some of the key considerations and effects that the play log representation has on the in-game playstyle prediction.

Once the play log representations has been defined, the next step in our framework is to respectively in Case I and II obtain labelled playstyle trajectory data, as shown in Figure 1(C). In Case I, the proxy agents are used to generate the play log data along with the agent playstyle label. For Case II, traditional unsupervised clustering is used to obtain the playstyle label for the respective play log data. In order to handle the variable length of player action trajectories across different episodes, we pivot the episode play log such that each entry within the play log is assigned the play log playstyle label (Figure 1(D)). This pre-processing step prepares the data for the training and development of our classifier.

4.2 Playstyle Classification

In this section we present our hybrid probabilistic supervised learning approach, which uses Bayesian Inference informed by a K-Nearest Neighbors (KNN) based likelihood, that is able to classify players in real time at every step. At any given point in a game, there exists a belief over which playstyle the player is currently using. In other words, there is a probability distribution across the playstyles which exist in the game. At the start of the game episode the probability belief distribution across the playstyles will be equally likely. As the player takes actions within the game the playstyle belief probabilities will shift in the direction of which ever playstyle is likely to have taken the observed actions. For this reason, Bayesian Inference is a suitable solution for modelling and tracking these changes in player playstyle. A core component of Bayesian Inference is the likelihood probability function. We train a KNN classifier using the labelled play log data, from steps A-E in Figure 1, which when given an unseen play log entry will return a playstyle classification as well as the probability belief of this classification across the playstyle label classes. This probability belief distribution of the KNN classification is what we use as the likelihood probability during the posterior belief update, as shown in Equation 2 which illustrates our Bayesian Belief Update process:

$$P(playstyle|observation) = \frac{P(observation|playstyle)P(playstyle)}{P(observation)} \tag{2}$$

Where $playstyle \in \Gamma$, $observation \in \Psi$, and where $P(observation|playstyle)$ is the likelihood probability distribution, provided by the KNN classification's probability distribution trained using the labeled play log data in Υ. The prior distribution is initialised to equally likely at the start of the Bayesian belief update process, and at every step the highest posterior playstyle belief probability is returned as the playstyle prediction.

5 Evaluation by Experiments

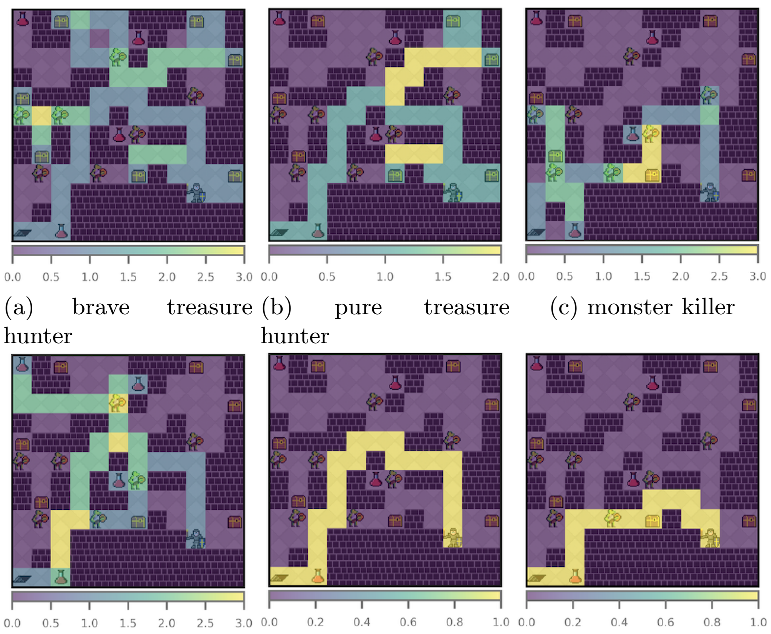
As part of our experiment we develop two variants of our hybrid supervised Bayesian classifier based on two different play log instantiations which we have defined. We make use of MiniDungeons as our Case I domain, and the human player Super Mario Bros dataset as our Case II domain. Furthermore from an evaluation perspective we compare our approach to the the comparative unsupervised approach by Ingram et al. [12]. We begin our Case I evaluation of our framework using the MiniDungeons game domain [8][9]. Subsection 5.1 outlines the experiment setup and the implementation details related to the application of our framework method. The Case II human player experiment setup, implementation, and evaluation is presented in Subsection 5.3.

5.1 Case I: MiniDungeons Experiment

MiniDungeons is a two-dimensional top-down dungeon exploration game, and is a common benchmark research domain for modeling and understanding human playstyles [8][9]. From a play log instantiation perspective we define two types of play logs, each of which measure the player interaction at different granularities:

– Low level visitation grid (LLVG): is a two-dimensional vector which tracks the number of visits made to each position in a level (i.e. models at the player action level [1]). For a given MiniDungeons level the LLVG play log dimensions will be equal to the level's row and column count.
– Tactic level information (TLI): is a one-dimensional vector which stores the number of times a specific action type is taken, as well as the number of times the player visits each level cell or square type [11]. The TLI play log aims to track the higher level player tactics [1].

To generate these two respective play log datasets we define a set of six player proxy agents as described in Figure 2. The playstyles are created based on the



(a) brave treasure hunter (b) pure treasure hunter (c) monster killer

(d) heal and run (e) safe runner (f) wreckless runner

Fig. 2: Playstyles Heatmap for MiniDungeons (Level 9) [9].

main objectives available in a given level of MiniDungeons, for example: collect treasure, restore hit points (HP) by collecting potions, battling monsters for experience points, and reaching the dungeon exit. Furthermore, the choice of our playstyle proxy agents is informed by the clustering analysis work done by Iwasaki and Hasebe [14] who evolve multiple human proxy agents without predefining the desired playstyle or reward function. The core playstyles generated by their approach are: a *runner* based playstyle, which prioritises exiting the dungeon as quick as possible, a *completionist* player, which aims to interact with treasure, monster, and potion elements within the dungeon, a *treasure centric* based playstyle, which focuses on collecting treasure, and a *safty-first* playstyle, which focuses on collecting treasure and potions. Respectively for our work, as illustrated in Figure 2, we have the treasure centric agents called *brave treasure hunter* and *pure treasure hunter*. The *pure treasure hunter* only collects unguarded treasure, whilst the *brave treasure hunter* will collect treasure guarded by monsters and replenish HP if needed. The *monster killer* playstyle agent is concerned with battling as many monsters as possible without dying. If the monster killer agent's HP is low, potions will be collected to restore health. The *heal and run* agent aims to collect all the potions in a given level. The *safe runner* and *wreckless runner* playstyles respectively aim to exit the dungeon as fast as possible. The *safe runner* agent takes the shortest available safe path to the dungeon exit (i.e. a path which does not encounter any monsters), whilst the *wreckless runner* agent will take the shortest path to the dungeon exit.

From the training and evaluation level perspective the respective MiniDungeons levels[3] used are:

– *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)
– *seentest* (md-strand_2-v0, md-holmgard_0-v0, md-holmgard_3-v0, md-holmgard_5-v0, md-holmgard_7-v0, md-holmgard_8-v0)
– *unseentest*: (md-holmgard_1-v0, md-holmgard_2-v0, md-holmgard_4-v0, md-holmgard_6-v0, md-holmgard_9-v0, md-holmgard_10-v0)

The levels assigned to the *train* set were selected in order to reserve the Holmgard levels for the seen and unseen evaluations. Each Holmgard level was randomly assigned to the respective *seentest* and *unseentest* sets. Each level run was repeated ten times in each respective evaluation case.

Initially two types of classifiers were developed, namely KNN and Bayes, each of which has a play log variant based on the LLVG and TLI play log instantiations. The classifiers trained on the *train* levels and evaluated on the *seentest* and *unseentest* levels are respectively called: KNN (LLVG), KNN (TLI), Bayes (LLVG), and Bayes (TLI). Each level run was repeated ten times for each playstyle within each respective evaluation case.

For both the KNN (LLVG) and KNN (TLI) classifiers the number of nearest neighbours used is three (i.e. *k=3*, determined empirically during experiment). The KNN (LLVG) model classifies the observed play log entry using the nearest or most similar log entries in the form of the LLVG play log [18]. Similarly the

[3] https://github.com/ganyariya/gym-md/blob/main/README/resources/md_stages_screenshots/README.md

KNN (TLI) model classifies using the nearest entries in the form of the TLI play log.

For the Bayesian Inference based classifiers, Bayes (LLVG) and Bayes (TLI), the initial prior probability belief distribution is equally likely across the playstyles. The likelihood probability that the in-game player observation belongs to a specific playstyle is informed by the historic LLVG and TLI play logs respectively. The posterior probability is then updated using the likelihood probability and the prior probability distribution. This process of updating the playstyle probability belief distribution is repeated as the new evidence (i.e. the player observation) is observed [18].

In our initial evaluations the KNN classifiers outperformed the Bayes classifiers, and upon careful inspection we attribute the poorer Bayes performance to the likelihood probability function not having enough historic data to discern and provide probabilities which can adequately influence the playstyle belief distribution. The problem with the Bayes classifier was that in unseen situations the default equally likely probability was being returned by the likelihood function. Which had little effect on the belief update because the observation was equally likely. We then improved the likelihood probability function by including a cosine similarity lookup for nearest similar observations. Which results in a more informed probability likelihood being used when compared to the default equally likely probability. The likelihood probability function used to inform the playstyle belief update is core to the success of the Bayes classifier. This insight as well as the KNN and Bayes comparative accuracy and prediction results, presented in subsection 5.2, suggested that a well balanced approach would be to combine the two classifier types into a hybrid classifier - which we refer to as the Hybrid Bayesian Supervised Learning classifier. In our hybrid classifier the KNN output prediction probability distribution is used as the Bayesian likelihood function. Two variants of our Hybrid classifier, called Hybrid (LLVG) and Hybrid (TLI), was created based on the LLVG and TLI play log instantiations and are respectively informed by the KNN (LLVG) and KNN (TLI) likelihood probability classifiers.

In order to baseline our hybrid approach, two baselines were used. The first baseline is a simple 'random' classifier which will at each step make a random prediction on the player's playstyle. The second comparative baseline is an unsupervised approach which makes use of an LSTM-autoencoder to cluster the player action trajectories [12]. The Autoencoder is used to project variable length trajectories in a uniform latent space which is then used for the clustering step. Although, this approach is unsupervised we used the ground truth labels generated for both the Mario and the MiniDungeons domain in order to compute the performance of this model.

5.2 Case I: MiniDungeons Experimental Results

As outlined in the MiniDungeons experiment setup, we have run two kinds of evaluations on the respective classifiers. The first evaluation is run on levels which have been seen before by the respective classifiers. The seen case is an appealing evaluation in situations where a game's levels are known and defined. For example, a game developer would like to adapt non-player character attack patterns, during the encounter with the player, within the game's existing levels. In this situation the game developer can train and deploy our classifiers and adapt the game accordingly. Next let us say that the game has been released successfully and our classifiers have been deployed and shipped with the game's initial release. In this situation our second evaluation on unseen levels may assist the game developer. As new levels are added to the game via updates, the classifiers which can generalise to unseen levels do not need to be re-trained or re-deployed. The representation of the play log is key to enabling this generalisability across unseen levels.

The LLVG play log representation is at a lower granularity, than the TLI representation. The LLVG play log is level specific since the play log's dimensions and the positional relevance of the player is coupled to the specific level under consideration. This means that, in this case, the LLVG classifier cannot be applied to unseen levels. This is why the respective LLVG classifier results are absent in the unseen evaluations (as shown in Figure 4 and Table 1). Since the TLI play log is at a higher 'tactic level' abstraction the same TLI vector can generalise across levels. The results shown in Figure 3 confirm that in the

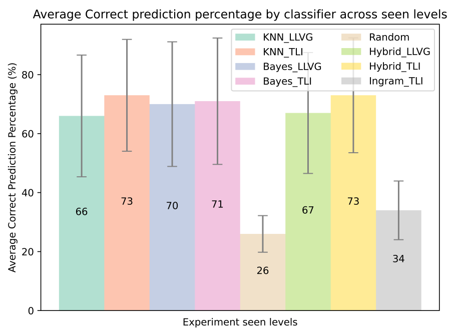


Fig. 3: Seen Evaluation

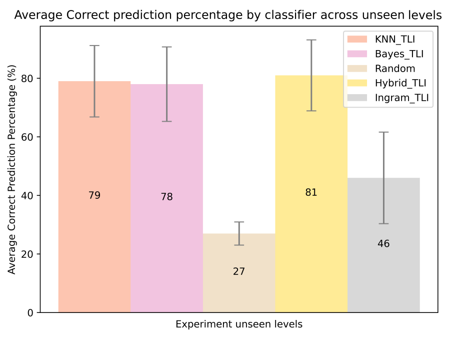


Fig. 4: Unseen Evaluation

seen case that each classifier is able to on average accurately predict the correct playstyle, well above the random classifier and our comparative case. The KNN (TLI) and Hybrid (TLI) classifiers marginally performs better than the other classifiers. The interesting observation is that in all three classifier types the TLI based representation resulted in better average accuracies when compared to the LLVG variants. This suggests that the TLI play log representation better captured the player interaction in comparison to the LLVG representation. Initially, we expected the LLVG representations to perform better since the play logs are better fit to each level and are at a lower granularity. However, this is not the case since the KNN LLVG performed marginally weaker.

Figure 4 summarises the unseen evaluation of the respective TLI based classifiers. On average our classifiers accurately predict the correct playstyle well above the random classifier and to our comparative case. The results in Figure

4 confirm that the higher level TLI play log representation is able to generalise to unseen levels. The Hybrid (TLI) classifier marginally outperformed the KNN (TLI) and Bayes (TLI), which we attribute to the difference in stability scores, which is summarised in Table 1. However, before discussing the stability score metric, it is important to note that, in both the seen and unseen evaluations, our classifiers at each step received only the latest play log entry, whilst the Ingram (TLI) classifier received the latest play log entry as well as the previous temporal play log entries. Thus highlighting the success of our framework's variable length pre-processing step in removing the dependency on previous episodic player action trajectories in order to classify the player. As highlighted in Figure 4 we outperform the comparative case. Although some of the poor performance associated with the approach by Ingram et al. [12] can be attributed to the unsupervised nature of the model, our performance is substantially greater and can be done using only the current play log state rather than requiring the past temporal play log entries.

As mentioned, Table 1 summarises an important result relating to the average playstyle prediction stability for each classifier. The stability of the classifier is a count of how many times, during an episode, the classifier changes its playstyle type prediction (e.g. transitioning from a Monster Killer prediction to a Brave Treasure Hunter prediction). The lower the stability score, the more stable the

Table 1: Overall Average Stability Score by Classifier

Classifier Type	**Average Stability Score**		
	Seen Levels		**Unseen Levels**
	LLVG	**TLI**	**TLI**
KNN	8.317	4.5	2.778
BAYES	0.222	0.111	0.444
Random	51.017		54.105
Hybrid	2.583	1.611	1.556
Ingram et al. (TLI)	1.333		1.167

classifier. Since the random classifier makes a random prediction at every step the stability score is high. Table 1 highlights the big difference in stability between the KNN and Bayesian approaches. We respectively attribute these to the nearest neighbour distance logic in KNN and the gradual belief update in the Bayesian update. If you need a more stable classfier, then the Bayesian approach may be better. If the rate of prediction type change is not a concern then the KNN approach can be considered. The idea behind the Hybrid classifier is that the classifier should ideally bring about good balance between the rate of change in playstyle prediction type, i.e. the stability, and the accuracy of the believed playstyle prediction. When comparing the unseen evaluation TLI stability scores of the KNN (2.778), Bayes (0.444), and Hybrid (1.556), we see that the Hybrid classifier's stability sits between the KNN and Bayes classifiers, whilst obtaining a higher average accuracy in the unseen case, as seen in Figure 4.

The choice of classifier, in terms of stability, is dependent on how the playstyle prediction classifier is used to adapt a game accordingly. For example, when adapting a game's difficult at the end of a level or after an enemy encounter, then the prediction stability of the classifier is less of a consideration because the game adaptability occurs less frequently. However, if you are adapting the non-player character (NPC) behaviour in real time in response to player actions then the stability of the classifier is quite important. In this case of playstyle-adaptive NPC agents, players may be confused or the gameplay experience may be adversely affected if the playstyle prediction changes more often, which may result in conflicting or inconsistent NPC agent behaviour. We continue the evaluation of our framework using the Super Mario Bros human player data, in Subsections 5.3 and 5.4.

5.3 Case II: Super Mario Bros Experiment

The Super Mario Bros evaluation dataset was created as part of a user study, conducted by Guzdial and Riedl, consisting of seventy-four human players which played through twelve levels of Super Mario Bros [6]. Super Mario Bros is a platforming game which involves moving the main player character called Mario through a two-dimensional level traversing it from left to right whilst navigating platforms, jumping gaps, and overcoming enemies, until the end of the level is reached. Mario is able to move left and right, jump, run, and shoot fireballs, if the enabling *fire flower* item is collected [17]. We make use of the Super Mario Bros natural human player action dataset in order to ascertain whether our framework is able to scale to a more complex domain. Furthermore, Super Mario Bros is a suitable domain for studying playstyles since there are a number of different ways to play the game. The global goal of Super Mario Bros is to reach the end of the level, however, there are coins which can be collected and various enemies which can be defeated in different ways based on collecting different items which all affect the final score obtained. There is also an exploratory element where players can try find shortcuts, hidden areas, or bonus rooms accessed by traversing green warp pipes or by breaking platform blocks to enable new pathways.

For this evaluation we define two types of play logs, based on the TLI representation, each of which summarise the player interaction at different granularities:

- Summarised Tactic level information (S-TLI): is a one-dimensional vector which stores the number of times the player jumps, kills, runs, breaks bricks, and dies. The S-TLI is a summarised play log view of the player's in game interactions.
- Extended Tactic level information (E-TLI): is a one-dimensional vector which is an expanded more granular form of the S-TLI tracking the occurrence of forty action and in-game interaction events.

To generate these two respective play log datasets we processed the source Mario data and performed K-means clustering on the respective play log data to obtain the prospective playstyles within the dataset. Figure 5 and 6 respectively show the S-TLI and E-TLI PCA-Reduced Mario data after K-means clustering. This processing step was necessary as the mario dataset is unlabeled. The efficacy of this approach to surfacing prospective playstyles, is demonstrated by applying the same unsupervised K-means clustering to our MiniDungeons dataset. Here we were able to successfully extract the six desired clusters which correspond to

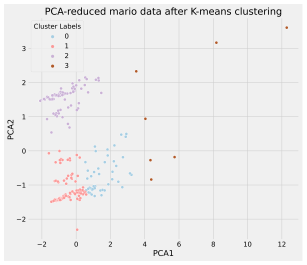


Fig. 5: PCA-Reduced Mario (S-TLI) data after clustering.

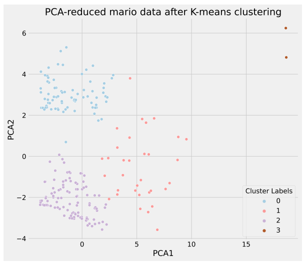


Fig. 6: PCA-Reduced Mario (E-TLI) data after clustering.

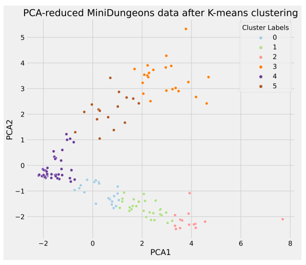


Fig. 7: PCA-Reduced MiniDungeons (TLI) data after clustering.

the six playstyle proxy agents, as shown in Figure 7. The respective values of k was determined using the silhouette coefficient and elbow method [15].

It is interesting to see that the K-means clustering on the S-TLI and E-TLI play log representations both resulted in the same number of clusters, despite being at different levels of granularity. Which highlights a possible commonality or relationship in the underlying play log data. The cluster labels obtained will be used as the playstyle labels during training. The idea as shown in the MiniDungeons case, Figure 7, is that similar play log entries belong to the same playstyle cluster. Thus unseen play log entries should fall within a cluster containing the most similar play log data. From a training and evaluation perspective we took the player action mario dataset and partitioned eighty percent of the player action trajectories for training and the remaining twenty percent of trajectories for validation and testing. In total two classifier types were created, where each classifier type has a play log variant based on the S-TLI and E-TLI play log instantiations. The first classifier type being a weighted KNN classifier and the second being our Hybrid Bayesian classifier type which is informed by a weighted KNN classifier as the likelihood probability function. From a baseline perspective we make use of a simple 'random' classifier which will at each step make a random prediction on the player's playstyle. In terms of our comparative baseline we again make use of the Ingram et al. approach [12]. For the unsupervised Ingram et al. classifier we make use of the ground truth cluster labels generated in order to compute the performance of this model.

5.4 Case II: Super Mario Bros Experimental Results

For the Super Mario Bros evaluation the intuition behind our choice of play log representation is that a lower level play log granularity in terms of the player interaction should better surface or distinguish the underlying prospective playstyles. Which in turn should bring about a stronger prediction accuracy performance. The average correct prediction percentage results obtained for the S-TLI and E-TLI evaluations are respectively shown in Figures 8 and 9. These shown results confirm the ability of our framework to operate and scale within a more complex domain using human player action trajectory data. Secondly, the difference in the average correct prediction percentage results between the S-TLI and E-TLI Hybrid classifiers confirm our intuition related to the player interaction play log granularity. The only difference between the S-TLI and E-TLI evaluations is this play log granularity, which results in a substantial difference in the playstyle classifier's prediction accuracy. When comparing the weighted

Table 2: Overall Average Stability Score by Classifier

Classifier Evaluation	**Average Stability Score**	
	S-TLI	***E-TLI***
Weighted KNN	0.196	0.080
Hybrid	0.014	0.004
Random	0.745	0.754
Ingram et al.	0.040	0.161

KNN classifier results to our Hybrid classifier we see a greater difference in average prediction performance, which we attribute to the respective stability scores obtained. As shown in Table 2, our Hybrid classifier's Bayesian belief update results in a better stability score than the Weighted KNN which more rapidly changes the playstyle prediction type due to the nature of the nearest neighbour search. The lower the stability score, the more stable the classifier. In comparison to the MiniDungeons evaluation, the Mario evaluation better highlights the effect that the stability of the classifier has on the end prediction result. The Hybrid classifier's more gradual Bayesian belief update allows for the classifier output to be more certain about the playstyle under observation. It is important to note that the trade-off between the prediction and the model stability still exists. The MiniDungeons evaluation showed that a better stability does not necessary guarantee a better average prediction result. In the MiniDungeons case both the stability scores of the Ingram et al. (TLI) and Bayes (TLI) were more stable in compared to the Hybrid (TLI), but the average prediction percentage results were not better than the Hybrid (TLI). The strength of our Hybrid classifier and framework is again highlighted as our classifiers at each step received only the latest play log entry, whilst the Ingram classifiers received the latest play log entry as well as the previous temporal play log entries. Thus again confirming the success of the variable length pre-processing step.

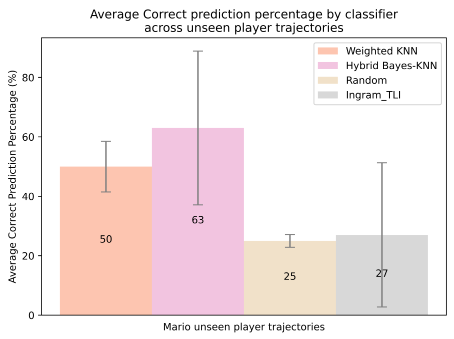


Fig. 8: Mario S-TLI average correct prediction percentage by classifier across unseen player trajectories.

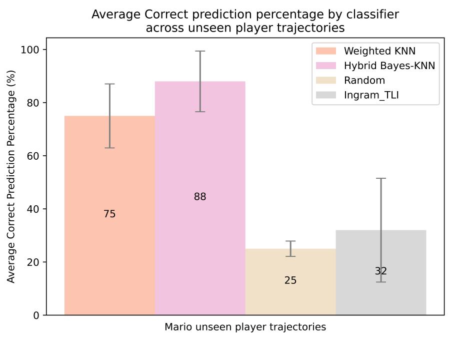


Fig. 9: Mario E-TLI average correct prediction percentage by classifier across unseen player trajectories.

6 Conclusion and Future Work

Our work contributes a hybrid probabilistic supervised learning framework, using Bayesian Inference informed by a K-Nearest Neighbors based likelihood, that is able to classify players in real time at every step within a given game level using only the latest player action or state observation. Furthermore we outperform our comparative baselines whilst using only the latest player observation to make our prediction. Our experiment highlights the success of our framework in a complex human player setting and the effect the play log representation has on the prediction's accuracy, stability, and generalisability. As part of our future work we plan to use our playstyle classification framework to adapt the gameplay experience in a meaningful way. We would also like to apply our work in a competitive online game setting in order to illustrate the potential use and benefit to the e-sports community from a player strategy analysis and game analytics perspective.

References

1. Bakkes, S., Spronck, P., Lankveld, G.V.: Player behavioural modelling for video games. Entertainment Computing **3**, 71–79 (2012), date accessed: 2021-05-15
2. Bontchev, B.: Holistic player modeling for controling adaptation in video games. Proceedings of the 14th International Conference e-Society 2016 (04 2016)
3. Charles, D., Black, M.: Dynamic player modelling: A framework for player-centred digital games. Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education (01 2004)
4. Gonzalez-Duque, M., Palm, R.B., Risi, S.: Fast game content adaptation through bayesian-based player modelling. In: 2021 IEEE Conference on Games (CoG). pp. 01–08 (2021). https://doi.org/10.1109/CoG52621.2021.9619018
5. Gow, J., Baumgarten, R., Cairns, P., Colton, S., Miller, P.: Unsupervised modeling of player style with lda. IEEE Transactions on Computational Intelligence and AI in Games **4**(3), 152–166 (2012). https://doi.org/10.1109/TCIAIG.2012.2213600
6. Guzdial, M., Riedl, M.O.: Game level generation from gameplay videos. In: Proceedings of the Twelfth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (2016)
7. Hernandez-Leal, P., Taylor, M.E., Rosman, B., Sucar, L.E., de Cote, E.M.: Identifying and tracking switching, non-stationary opponents: A bayesian approach. In: AAAI Workshop: Multiagent Interaction without Prior Coordination (2016)
8. Holmgard, C., Liapis, A., Togelius, J., Yannakakis, G.: Monte-carlo tree search for persona based player modeling. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment **11**(5), 8–14 (Jun 2021)
9. Holmgard, C., Liapis, A., Togelius, J., Yannakakis, G.N.: Evolving personas for player decision modeling. In: 2014 IEEE Conference on Computational Intelligence and Games. pp. 1–8 (2014). https://doi.org/10.1109/CIG.2014.6932911
10. Holmgard, C., Liapis, A., Togelius, J., Yannakakis, G.N.: Personas versus clones for player decision modeling. In: Pisan, Y., Sgouros, N.M., Marsh, T. (eds.) Entertainment Computing – ICEC 2014. pp. 159–166. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
11. Iawasaki, Y., Hasebe, K.: Identifying playstyles in games with neat and clustering. In: 2021 IEEE Conference on Games (CoG). pp. 1–4 (2021). https://doi.org/10.1109/CoG52621.2021.9619024

12. Ingram, B., Rosman, B., van Alten, C., Klein, R.: Play-style identification through deep unsupervised clustering of trajectories. In: Proceedings of the IEEE Conference on Games (2022)
13. Ingram, B.: Generating tailored advice in video games through play-style identification and player modelling. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment **17**(1), 228–231 (Oct 2021)
14. Iwasaki., Y., Hasebe., K.: A framework for generating playstyles of game ai with clustering of play logs. In: Proceedings of the 14th International Conference on Agents and Artificial Intelligence - Volume 3: ICAART,. pp. 605–612. INSTICC, SciTePress (2022). https://doi.org/10.5220/0010869500003116
15. Kodinariya, T., Makwana, P.: Review on determining of cluster in k-means clustering. International Journal of Advance Research in Computer Science and Management Studies **1**, 90–95 (01 2013)
16. Normoyle, A., Jensen, S.: Bayesian clustering of player styles for multiplayer games. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment **11**(1), 163–169 (Jun 2021)
17. Pedersen, C., Togelius, J., Yannakakis, G.: Modeling player experience in super mario bros. pp. 132 – 139 (10 2009). https://doi.org/10.1109/CIG.2009.5286482
18. Russell, S.J., Norvig, P.: Artificial Intelligence: a modern approach. Pearson, 3 edn. (2009)
19. Scott, G., Khosmood, F.: Complementary Companion Behavior in Video Games. Master's thesis, School of Computer Science, California Polytechnic State University, San Luis Obispo, https://digitalcommons.calpoly.edu/theses/1744 (2017), https://doi.org/10.15368/theses.2017.55, Arxiv Short Paper Version: https://arxiv.org/abs/1808.09079
20. Togelius, J., Nardi, R.D., Lucas, S.M.: Making racing fun through player modeling and track evolution (2006), date accessed: 2020-07-30
21. Tremblay, J., Verbrugge, C.: Adaptive Companions in FPS Games. In: FDG (2013), date accessed: 2020-04-27
22. Tychsen, A., Canossa, A.: Defining personas in games using metrics. In: Future Play (2008)
23. Valls-Vargas, J., Ontañón, S., Zhu, J.: Exploring player trace segmentation for dynamic play style prediction. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment **11**(1), 93–99 (Jun 2021), https://ojs.aaai.org/index.php/AIIDE/article/view/12782
24. Yannakakis, G., Spronck, P., Loiacono, D., Andre, E.: Player modeling, pp. 45–59. No. 6 in Dagstuhl Follow-Ups, Dagstuhl Publishing (2013), date accessed: 2021-05-04
25. Zhang, M., Verbrugge, C.: Modelling player understanding of non-player character paths. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment **14**(1) (September 2018), date accessed: 2021-05-04