Capturing, Clustering and Copying Playstyles in Games

Mark Alan Ferguson

Doctor of Philosophy

University of York

Computer Science

June 2023

Abstract

For several years there has not only been a lot of interest in enhancing the user experience within video games but also in the development of algorithms to allow AI agents to better assist humans. It is felt that if it were possible to cluster and imitate playstyles, this would lead to an enhanced human AI interaction. The main aim of this thesis is to lay out a three-step process to allow the encoding, clustering, and imitating of playstyles.

To encode the gameplays into a representation space, the STDIM-VAE architecture has been proposed. This architecture was then compared to both its individual components. Further, a more in-depth analysis of the STDIM-VAE learnt representation space was conducted and it was shown that the architecture was able to encode playstyle relevant features across the three games investigated.

Clustering was then attempted using a novel method, namely, Reference-Based Clustering. The results of which were then compared to clustering within the other encoder type representation spaces as well as clustering on extracted gameplay variables. It has been shown that the combination of the STDIM-VAE and Reference-Based Clustering techniques allows good quality clustering across all considered games.

Finally, imitation of three different playstyles was explored within a stealth game. A new technique called DTWI was used to imitate across a range of altered levels, to investigate the ability to correctly imitate the playstyle. It has been shown that it was possible to correctly imitate playstyles across many level variations.

Although the majority of this work has focused on use cases within the video game domain, it was expected that the methods created would have use cases beyond this application area. To this end, the STDIM-VAE was used in the generation of representations from which model design novelty was evaluated.

I dedicate this thesis
to my family for their
love, support and
encouragement.

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost I would like to to express my deepest gratitude to my supervisors Dr James Walker, Dr Sam Devlin and Dr Daniel Kudenko for all their help, advice and guidance throughout my PhD.

Special thanks to my collaborating partners, Dr Andreas Lieberoth, Dr Marc Andersen and Dr Mihaela Taranu, from the Interacting Minds Centre at the University of Aarhus, Denmark and also to Professor Sebastian Deterding at the University of York for useful discussions.

I would also like to extend my sincere thanks to both Engineering and Physical Sciences Research Council (EPSRC) and Microsoft Research Lab, Cambridge for funding this research.

Many thanks to my family and friends who have supported me throughout the whole process.

Declaration

I declare that this thesis is a presentation of original work, and I am the sole author. This work has not previously been presented for a degree or other qualification at the University of York or elsewhere. All sources are acknowledged as references.

Two of the datasets utilised within this work were collected by collaborating partners at the Interacting Minds Centre at the University of Aarhus, Denmark. This data was provided to me and used as a starting point for my independent research, analysis and conclusions via the methods developed within this thesis. The first dataset contained images of model ducks and was used within Chapter 7. Whereas the second dataset contained images of transport models alongside a range of metrics and was used in the discussion of future work in Section 8.3.4.

Some of the material in this thesis has appeared in the following published papers:

1. Ferguson, M., Devlin, S., Kudenko, D. and Walker, J.A., 2020, September. Player style clustering without game variables. In *Proceedings of the 15th International Conference on the Foundations of Digital Games* (pp. 1-4).

2. Ferguson, M., Devlin, S., Kudenko, D. and Walker, J.A., 2022, September. Imitating Playstyle with Dynamic Time Warping Imitation. In *Proceedings of the 17th International Conference on the Foundations of Digital Games* (pp. 1-11).

3. Ferguson, M., Deterding, C.S., Lieberoth, A., Malmdorf Andersen, M., Devlin, S., Kudenko, D. and Walker, J.A., 2020, June. Automatic similarity detection in lego ducks. In *ICCC'20: Eleventh International Conference on Computational Creativity.* Association for Computational Creativity (ACC).

1 Introduction

This thesis aims to lay out a three-step process in order to allow the imitation of a diverse set of playstyles, when provided only with a set of unlabelled gameplay videos. The first step in this process was to extract a set of gameplay clusters for each of the playstyles within the demonstration set. Initial experiments showed that it was beneficial to first convert the gameplay videos into an intermediate representation space. To achieve this, a preliminary step was required, namely the training of an encoder that can successfully encode the relevant playstyle information into the representation. The final step was to imitate the given playstyles based on the provided demonstrations. Further, the aim of the developed method was to be able to learn the playstyle, even in situations which differed from the provided demonstrations.

1.1 Motivation

The ability to successfully imitate a given playstyle has many possible applications in games. Further, the developed techniques could be utilised in applications beyond games. In order to develop the proposed methods, a testbed domain had to be chosen. It was decided to use the video game domain as the experimental testbed in this work for several reasons. Firstly, video games are essentially simulations and hence can provide a diverse set of tasks for the agent to complete, while not having any real-world safety consequences of any errors the agent might make. However, it is noted that it would be beneficial to utilise these techniques in order to solve real-world problems, and as such it is recognised that provable assurances would be required in these cases. Secondly, experiments conducted during the development of the algorithms can be performed faster and at a lower cost in a video game domain when compared to using real world situations. Finally, in general, video games have a known goal and set of strategies commonly used to complete them. This information can help accelerate any research by assisting in the selection of playstyles for which data could be collected.

In games, possible applications include both the direct use of the playstyle agents, as well as their indirect use in the development of future techniques and it is considered that this work lays the path for both of these types of applications. In regards to the direct use of the playstyle agents, it has been suggested in the literature that agents with a given persona can be used to automate the testing of new game content [54, 53, 46, 47], by allowing them to explore the relevant game space. Traditionally, these personas were achieved by designing a reward function to encourage a given behaviour. The personas selected could be of a general form and hence not require any level of game knowledge. Although, it is recognised that it is unlikely that these general personas would fully cover how human players may perform in any new game content. Alternatively, personas could be designed for the game in question in an attempt to fully cover all possible behaviours, however this would most likely require a level of game knowledge. This hence introduces a level of human bias and would limit the testing to playstyles that the human selector already was aware of. Alternatively, clustering could be applied to extract the playstyles from the playerbase. However, within the literature this is commonly achieved by clustering on game features and hence requires access to these features. Further, it is common that feature selection is often required, and this

potentially reintroduces human bias. The three-step process laid out in this thesis allows this to be achieved in a fully automated manner. This is achieved by encoding the gameplay into representations and hence does not require feature selection. Clustering can then be applied within this representation space to extract the playstyles that exist in the playerbase. Finally the imitation of each of these clusters can generate a policy for each playstyle, which can then be used for playtesting.

Alternatively, the learnt playstyle models could be integrated more directly into the game, by using them to control Non-Player Characters (NPCs) within the game. By utilising these models, each Non-Player Character (NPC) can be given their own playstyle and hence this would result in each NPC having a unique feel. Further, the use of imitation based methods aims to allow more complex behaviours in a believable manner. This results in the NPC acting in a more complex and rational manner, which is especially important when attempting successful collaboration. These ideas attempt to improve the human character interactions, with the likely knock on effect of improving the overall player experience.

This premise is supported by a behavioural study that in part investigated the desired characteristics of AI agents, both teammates and enemies, within a First Person Shooter (FPS) video game [23]. One common overarching desire, that almost all participants indicated, was the need for the AI agent to act human-like in any given situation. This idea was also refined by some participants to suggest potential behaviours, such as seeking 'health' when hurt or searching for cover when attacked. In addition, the majority of participants suggested that the AI agents shouldn't always just fight to the death when attacked. This further supports the idea of using imitation based techniques to develop NPCs, as it is likely to result in a more human-like behaviour. Further, just over half of the participants discussed the idea of having variable playstyles. This would be achieved using the proposed idea of providing different NPCs with different playstyle models learnt from imitating each of the found clusters. Interestingly, common behaviours that would normally be considered bad, such as making mistakes or even freezing, were also suggested by some participants as desirable. This most likely links back to the AI agents acting more like a human and hence in turn being expected to make mistakes. Further, all participants considered it bad if the AI agent's accuracy was too perfect, and almost all participants thought it was bad if the AI agent's reflexes were too fast. Regarding teammate AI agents, most participant comments revolved around being able to communicate with these AI agents, to direct them. This is thought to distil down to a failure of the AI agent and human understanding what each of them was trying to achieve. The move to a more human-like behaviour should result in a more rational behaviour, so it is easier to understand what a teammate AI agent was trying to achieve overall.

The player's immersion within a game is also a key element that affects the player experience [56]. Immersion can broadly be described based on three key features. Firstly, the reduction in the perception of time passing. Secondly, the absence of attention to what is happening in the real world. Thirdly, the engagement and feeling of being in the environment. In other words, 'When you stop thinking about the fact that you're playing a computer game and you're just in a computer' [15].

While it has been discussed in the literature whether immersion is truly desirable [38], this ultimately comes down to the type of experience that looks to be designed. A

German playwright Bertolt Brecht suggested that you should 'alienate' your audience at certain points to break immersion. By breaking immersion, you allow your audience a chance to step back and think more critically about what they are experiencing. This referenced work also suggested how this could be achieved within games, however if the temporary breaking of immersion was not desired then these referenced ideas provided a guide of what should be avoided. When designing AI agents, if their behaviour is either 'buggy' or lacks complexity, the player is more conscious of the behaviour construction and hence can break immersion. Again, the use of imitation methods to design NPCs with more complex and rational behaviours should improve player immersion.

The idea of a 'flow' state has also been discussed in relation to the player's experience and can be thought of as 'a state of concentration so focused that it amounts to absolute absorption in an activity' [24]. Naturally, both immersion and 'flow' have a certain amount of overlap in how the experience affects the player. In some cases, immersion in an experience is a precursor to achieving a 'flow' state. This is because a 'flow' state is seen as a more optimal or extreme experience [56].

Exploration to discover what was required to achieve a 'flow' state suggested it could be achieved with the combination of enjoyment and involvement or immersion [71]. This optimal combination can potentially be found by balancing the player's skill against the challenge they will encounter. If the experience strays outside this sweet spot, it can either lead to player anxiety if the challenge is too hard or boredom if the challenge is too easy. Although, it was suggested that enjoyment is possible even outside of the 'flow' state, and hence 'flow' was thought to be achieved by the combination of a high intensity of player experience (immersion) and high valence of player experience (enjoyment). However, if both of these are low then boredom is experienced. On the other hand, high immersion and low enjoyment leads to player anxiety. Conversely, high enjoyment and low immersion causes relief.

Based on this both immersion and enjoyment are required to achieve a 'flow' state. It has been suggested that the tripartite model of media enjoyment can be applied to the video game domain to explain what impacts a player's enjoyment of a game [33]. In this model three types of reaction have been described, namely 'Affective', 'Cognitive' and 'Behavioural'. For 'Affective' reactions, this mainly focuses on the emotional side of the player's experience, whether that be horror, sadness, *etc.* On the other hand, 'Cognitive' reactions more focus on rational analysis of this experience. This in itself can be split into two. Firstly, story assessment, which looks to evaluate the realism of the experience. This could range from the perceived realism of characters' actions to the story coherence and hence lack of plot holes. Secondly, personal assessment, this kind of assessment is more player dependent and relates to how the player relates to the experience. The last reaction type is called 'Behavioural', which considers how the player acts and in its rawest form how they interact with the medium, if at all. By utilising playstyle imitating agents to control NPC behaviours, the hope is that a diverse set of experiences can be realised while having the NPCs act in a realistic way based on some underlying playstyle.

The elements that make up the player video game interaction were also investigated [20]. From the player side, this highlights the player's engagement and enjoyment. Each of these can be related back to aspects of the 'flow' state, as engagement is similar to the idea of immersion. The video game aspects, which can be thought of

as game design, can be split into three categories. The first of these is 'Input/Output', which relates to the player's controls (*e.g.* motion controls) and how the game provides feedback to the player (*e.g.* visuals on a screen). The second is 'Content', which relates to game content the player interacts with, including both the gameplay and narrative, as well as what challenges the player will face. The final category is called 'Multiplayer' which relates to the additional players the player will meet within the game. This work looks more at the 'Content' side of video game aspects, in regard to the NPCs the player is going to interact with through the story. By improving these interactions, it is thought that this would help improve both player engagement and enjoyment.

In addition to these direct applications, there are many interesting applications for which this work lays the foundation. In the literature it has been shown, particularly in single agent systems, that AI agents can successfully outperform humans for example in popular games such as Chess, Go, Atari, *etc.* [17, 92, 7]. However, there are still certain tasks where humans are much stronger than AI agents [86]. For example, extrapolation to situations outside of training scenarios [111] or the ability to learn new complex tasks quickly, presumably because as noted by Bengio et al 'Humans and animals seem to be able to learn massive amounts of background knowledge about the world, largely by observation, in a task-independent manner' [9]. Given the generally accepted proverb 'two heads are better than one' [2], it is expected that successful collaboration between humans and AI agents would potentially allow the discovery of effective solutions to complex tasks. However, it has been shown that when agents are trained in isolation, while they can perform very well between themselves, issues can arise when collaborating with humans [21]. This is most likely caused by agents falling into given behavioural patterns and having no flexibility in how to overcome an impasse and achieve a given task. It has also been shown that even when training with a simple Behaviour Cloning (BC) agent, this issue can be mitigated [21]. This is key, as inflexibility can cause clashes in approaches and potentially cause a deadlock. When considered within the video game domain, while successful collaboration can provide interesting interactions, these impasses can cause frustration for the player and have a potentially negative impact on the player experience. Consequently, the development of human agent collaboration also has valuable applications within the gaming industry.

Before laying out ideas to improve this human agent collaboration, it is key to provide a clear definition of exactly what is meant by collaboration. In the literature [16] three types of interactions have been discussed, namely 'instruction', 'cooperation' and 'collaboration'. In the 'instruction' type of interaction, one participant is essentially in control and instructs the second participant in what to do in order to complete the overall goal. In contrast, within the 'cooperation' type of interaction, both participants share one overall goal and each of them complete subtasks in order to complete this goal. However, each of these subtasks is independent from one another. This is the key difference between the 'cooperation' interaction and the 'collaboration' interaction. In the latter type, namely 'collaboration', the subtasks are interdependent and hence a greater understanding of what the other participant is doing and is planning to do is required to avoid potential issues.

To achieve successful 'collaboration', due to the interdependent subtasks, it is important to be able to infer the required subgoals. This is something humans can often do fairly easily, especially within the chosen test domain, namely video games, due

to common practices used within the industry. For example, if a human player comes across a locked door, they will infer that a subgoal is to find and collect a key with which to open the door. Additionally, humans can often infer supplementary information from the game environment. For instance, coins should be collected, and spikes should be avoided. However, it is hard for AI agents to effectively learn even such simple inferences.

In addition to inferring the required subgoals, it is also key to be able to link these to the other player's actions to predict which subgoals each player is looking to complete. This is important for two reasons. Firstly, to ensure time is not wasted by unnecessarily targeting the same subgoal and secondly, if a subgoal is to be achieved, the correct level of assistance can be provided if it is indeed required. However, this is not a one-way street, a human player should also be able to infer and understand what an AI agent is also trying to accomplish.

In the literature three categories of human robot interaction have been defined based on three types of Sensorimotor Contingencies (SMCs)) [16]. Based on the requirements for collaboration, one of these types seems most appropriate, namely sync SMCs, due to the involvement of bidirectional predictions within the system. The ability to predict the goals of the collaborating partner can help to avoid impasses, such as when the agent unintentionally blocks the human player as they fail to predict where the human player was intending to go. Additionally, as mentioned, the agent should act in a rational manner and hence allow the human player to predict what the agent was attempting to do and provide assistance as required.

As previously stated, agents that trained with BC agents showed a greater ability to successfully collaborate. It is assumed that being able to train with human imitation agents, allows the training agent to learn how to predict what the other player was attempting to do and hence act accordingly. However, it is expected that the agent should be able to successfully collaborate with a range of playstyles different players may use. This could be achieved by training with a range of different human players, but this type of training is very time consuming.

However, a reasonable alternative approach would be to train with a range of agents, each with its own playstyle. This hence requires methods in order to generate a diverse set of human-like playstyle agents. In this thesis, as mentioned earlier, a three-step process is laid out in order to achieve this. Playstyle gameplays can be encoded, clustered and imitated to generate a diverse set of playstyle agents. An agent can then be trained with this diverse set of playstyles and hence aim to enable successful collaboration with a range of human-like playstyles.

This would result in successful collaboration between human and agents no matter what playstyle the human used. Currently, and often for complex collaboration in games, human players are paired with other human players. This means that the successfulness of the collaboration is reliant on other human players. Also, the playstyles that can be used are dependent on the other human partners willingness to change to complementary playstyles, and this often means that one player has to adapt their playstyle to complement the other. Consequently for a successful two person collaboration, only one player can use their preferred playstyle, unless by chance their playstyles are inherently complimentary.

However, if it were possible to pair human players with AI agents, the human players

would be free to use any playstyle they desire, and the AI agent would attempt to adapt. This would look to improve any collaboration and hence most likely the player experience. Whereas poor collaboration would frustrate the player and hence lead to a poorer player experience.

Overall, the research in this thesis aims to develop a system to encode, cluster and imitate a diverse set of playstyles based on gameplay videos collected from the playerbase. These trained agents can then be used in several ways. Either, directly to test game content or be used to control NPCs to give each a unique feel and lead to more interesting player NPC interactions. Alternatively, in a more indirect capacity they can be used to train or test developed NPCs to test their abilities to successfully collaborate with a range of playstyles they are expected to encounter once deployed within a game. Further, the aim to improve human agent collaboration has many applications beyond games. For example, real world use cases could include intelligent assistants, home care assistants and medical applications *etc.*

1.2 Hypothesis

The main aim of this thesis is to allow the imitation of a diverse set of playstyles from a set of gameplay demonstrations. This led to the investigation of the following hypothesis:

> A set of agents can be trained to act with a diverse set of playstyles shown within a demonstration set that contains only raw gameplay videos.

This hypothesis can be further broken into three main questions. These are:

1. **Encoding**
   Is it possible to learn a representation space that contains the key features that define a playstyle?

2. **Clustering**
   Is it possible to cluster within the aforementioned representation space to create a separate cluster for each playstyle?

3. **Imitation**
   Is it possible to train agents to accurately imitate human playstyles based on a set of demonstrations extracted by clustering?

1.3 Research Steps

As previously mentioned, a feasible method to potentially improve human / agent collaboration is to train with a set of models that imitate human playstyles. A natural approach in order to generate these models is to use Imitation Learning (IL), but options arise for choosing the datasets used to train these models. In an ideal situation, these models should display a diverse set of playstyles, to ensure the agent can perform ad hoc teamwork with a range of humans.

To achieve this, techniques need to be developed to allow the grouping of players based on their playstyles. This then means imitation methods can be applied to each of these groups to generate a diverse set of models. To ensure the generalisation of the clustering method, the technique developed was based on raw observation data (*i.e.* game screen). However, in order to compare the similarity of this raw data effectively, it should first be transformed into an intermediate representation. This means that an unsupervised representation learning method was also required. It was decided to generate a representation at a frame by frame level and then use a matching process to determine the similarity of the representation sequences. This meant that a matching process could then be chosen that was robust to temporal shift, as it was thought that the order of actions was more important than the timing in which they happened. To fully test this encoding method an example in a non-gaming domain was studied.

When it comes to imitation methods, it is key that they correctly imitate the playstyle seen in the demonstration and act in a human-like manner to ensure that the actions seem rational to a human observer. This is to ensure that when working together, the human can understand what the agent is trying to achieve.

The overall workflow for this thesis can be seen in Figure 1. Once the dataset was collected, the first step was to encode the raw gameplay videos into a meaningful representation space. Then clustering was applied within this learnt representation space to extract each of the playstyle clusters. Finally, imitation was applied to each playstyle shown in the found clusters. It should be noted that this workflow was designed in a modular manner and hence as better methods are developed they can be substituted into each step of the workflow.

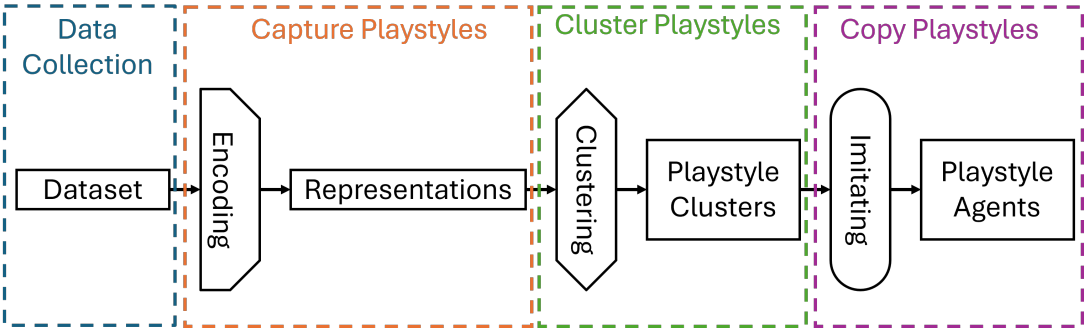


Figure 1: Proposed Workflow

1.4 Thesis Structure

This section presents how the steps within this thesis are organised. Chapters 2 and 3 will perform a deep dive into the existing relevant literature to set the context for this work. Chapter 2 will focus on the analytical side of this work. This includes the literature around being able to cluster human data based on playstyle and the generation of intermediate representations through unsupervised representation learning. Chapter 3 focuses on the learning side of this work. This starts with a general explanation of Reinforcement Learning (RL). The paradigms of Imitation Learning (IL) and Imitation

from Observation (IfO) are discussed and the literature in each of these areas will be explored. Finally, the possible applications for imitation will be considered.

Chapter 4 will introduce a new hybrid encoder architecture, Spatiotemporal DeepInfomax Variational Autoencoder (STDIM-VAE) to generate the intermediate representations from the raw gameplay videos. Analysis will then be conducted on the representation space in order to gain an understanding of what has been encoded within the gameplay and also how this compares to other gameplays.

Chapter 5 will introduce a novel clustering technique, Reference-Based Clustering, to allow clustering of the learnt representation space in order to extract a cluster for each of the playstyles. This clustering will be applied to multiple case studies that span different games, with each looking at different sub-datasets, where appropriate, to explore what kind of playstyles can be successfully extracted.

Chapter 6 will discuss a new imitation technique called Dynamic Time Warping Imitation (DTWI), to allow the imitation of the playstyles shown within each of the clusters found.

Chapter 7 will examine other possible applications for the encoder previously introduced. This will focus on the ability to design a system that rates the novelty of a set of models provided, based only on videos showing each of the models.

Chapter 8 will draw everything together and provide the main conclusions and takeaway points from this work. Following this, possible avenues for future work will be discussed, as well as some initial experiments which were undertaken within these areas including the concurrent learning of playstyles through multi-modal imitation. Finally, a few closing marks will round off this thesis.

2 Literature Review: Game Analytics

Overall, the work in this thesis can be broadly split into two sections. The first of these concerns game analytics, which involves the generation of a representation space. A representation space aims to allow the expression of a given dataset in a lower dimensional space, while ensuring data points that resemble one another are close together within the space. The representation space can be used to cluster datasets, as well as evaluate the meaning behind the representations to gain an understanding of how any two data points are similar or different. This can then be used to gain an understanding of how any two clusters compare. The second section addresses the learning side of the work in this thesis, which involves training agents to imitate given data. This current chapter focuses on the first of these two sections and the subsequent chapter will focus on the second of these *i.e.* the learning side of the work. Both of these chapters will discuss the relevant literature around each of these sections with the aim of providing a firm foundation of understanding for the remainder of this thesis.

2.1 Player Persona

In order to understand how the ideas of player personas relate to video games, it is first key to explore the general definition of persona. Within psychology, a definition of persona is 'the mask or facade presented to satisfy the demands of the situation or environment and not representing the inner personality of the individual' [1]. When considering video games, the 'mask' can be thought of as the 'character' and the 'situation or environment' as the 'video game' itself. This means in video games, player personas define the kind of character the player aims to emulate in order to be successful and hence how they will act in order to complete the game.

The ideas around player personas have also been investigated within the scope of game design [18] allowing players to express themselves as they wish. This has led to the creation of a four-step process, in order to design games around the idea of player personas. Below each of these steps will be explored to generate a video game specific definition of both 'Playstyles' and 'Play Personas'.

The first step in the process is known as 'Play-Values', this relates to the overall goal that the player is attempting to achieve. This in itself can be split into one of two categories. Firstly, 'Narrative Play-Value' (NPV) which focuses on the story elements of these values. One given example is 'a highly moral man cast by misfortune in a highly immoral environment, trying to do right by doing wrong' [18]. The second category is 'Functional Play-Value' (FPV) which focuses purely on the actions taken, rather than the story elements of the game. An example from this type of category is 'action, emphasis on fighting rather than opening doors or operating equipment' [18].

The second step is known as 'Play-Modes', within the gaming industry these are more commonly known as 'mechanics' or 'features', which ultimately define all the actions any player is capable of utilising within the game in order to negotiate any situation. This action set can be split into two types known as 'Navigation' and 'Interaction'. The 'Navigation' type relates to the player's location within the game and how they move throughout the game space. This can include the character's pace of movement through the environment, which is dictated by the player's choice to sneak, sprint, *etc.* as well

as the players' use of additional tactics such as flanking or the use of cover. The 'Interaction' type on the other hand focuses on how the player interacts with both Non-Player Characters and obstacles within the game. This could take the form of the use of a preferred weapon, *etc.*

The third step is called 'Playstyle', this can be distilled down to a player consistently using a certain set of 'Play-Modes' in order to overcome any given challenge within the game. In this four-step process, the 'Playstyle' is defined only to occur over a short time period or segment within the game.

The fourth and final step is called 'Play-Personas' which applies when longer time periods of gameplay are involved, this could even extend to the game as a whole. If a given 'Playstyle' is maintained over a significant amount of time, this then identifies the player with a given 'Play-Persona'.

Having introduced a basic definition for 'Player Persona', we next need to consider how these ideas could possibly be used within the games industry. In modern literature, there are two main ways that ideas around 'Player Personas' can be used. Firstly, they can be used for automatic playtesting of games and secondly, they can be used in the analysis of a game and/or the playerbase to understand the 'Playstyles' that exist within a game. Below each of these options is explored in turn, as well as relevant literature demonstrating these applications.

Initially, we will focus on how 'Player Personas' can be used to allow fast automatic testing of newly developed content. To achieve this, a number of 'Player Personas' need to be defined for example, this has been performed on MiniDungeons/MiniDungeons 2 [54, 53] and the 'Player Personas' are described below:

Runner
: Gets to the exit as quickly as possible.

Killer
: Kills as many enemies as possible and then gets to the exit.

Collector
: Collects as much treasure as possible and then gets to the exit.

Completionist
: Consumes as many potions as possible, collects as much treasure as possible, kills as many enemies as possible and then gets to the exit.

Exit
: Just tries to get to the exit.

Survivalist
: Avoids taking damage but tries to get to the exit.

Based on these descriptions, agents were designed in order to act with different 'Player Personas'. These agents can then be evaluated with new content to emulate how 'real' players would interact with this new content to allow game 'bugs' to be found and solved before any content is provided to customers.

However, the 'Player Personas' described above are heavily tailored to the specific game they are designed for and hence would need to be reworked for any other game.

This led to the question, can ‘Player Personas’ be defined in general terms in order to make them game agnostic [46, 47]. This led to a new set of definitions, some of which are provided below:

Winner
Purely focuses on winning the game.

Map Explorer
Aims to explore as much of the reachable game space as possible.

Curious
Aims to maximise interaction with game elements, with a focus on elements not interacted with before.

Record Breaker
Aims to maximise the score, without a focus on winning.

These definitions can be used in the same way as described above to generate agents and then test new game content. The main benefit of these definitions and the previous set of definitions is that the speed at which testing can be performed compared to the use of human Quality Assurance (QA) game testers is markedly quicker. It should be recognised that while these general ‘Player Personas’ mean that game specific knowledge is not required, it is unlikely that these general personas would actually cover the whole set of ‘Player Personas’ possible within the game environment.

The second use case is to extract ‘Player Personas’ either directly from the game or from how the playerbase actually played the game. Each of these extracted ‘Player Personas’ could then be inserted into the automated testing system to be used as game specific ‘Player Personas’. As an example, in the literature ‘Player Personas’ have been extracted directly from the ‘Hitman: Blood Money’ game [101]. Here the ‘Player Personas’ were constructed based on choices both in the ‘Navigation’ and ‘Interaction’ fields. These choices can then be mapped to a set of relevant ‘Play-modes’ available to the player under a given situation. Within the ‘Navigation’ field there are four options suggested. The first two of these are ‘Brawn’ or ‘Brain’. Here in the first case, namely ‘Brawn’ the player pushes the character to its physical limits by climbing, running, *etc.* whereas in the second case, ‘Brain’, the player aims to use rational thought to accomplish the goals instead of physical means. In this case, for example, this can take the form of using coins to distract enemies, spying through keyholes, *etc.* The last two options focus on the player’s use of disguise. These are ‘Role-Play’ and ‘Stealth’. The first of these aims to use disguises in order to blend into the environment and gain access to restricted areas. Whereas the second aims to proceed without the use of any disguises and continues to wear the trademark black suit. This hence means that the player must move stealthily around the game space in order to gain access to the required areas within the game. When we consider the ‘Interaction’ field, this mainly concerns the type of weapons used by a player. These weapons can be split into a number of categories *e.g.* ‘Non-Lethal’ *vs.* ‘Lethal’, ‘Silent’ *vs.* ‘Loud’ and ‘Clean’ *vs.* ‘Bloody’.

Having now introduced the definition of these terms and how they may manifest themselves within games, discussed next is how these ideas will be used within this

work. The main way that these ideas are employed is in the generation of gameplay data. This was done in two different manners, firstly, AI agents with a given player persona were used in order to automatically generate data with a given playstyle. The second was to pre-define a set of personas and then manually generate data under a chosen player persona. The aim behind the use of these methods was to generate datasets that not only contained realistic playstyles but were also close to human-like gameplay data. This was done to reduce the complexity in the collection phase in order to allow quick development of the newly designed algorithms.

As mentioned above, an alternative to focusing on the game mechanics is to consider the playerbase. Although it is unlikely to fully contain all valid 'Player Personas', it does show the personas being employed and hence can be used to evaluate the current state of the game. A common approach here is to extract this information from the playerbase data, by the use of clustering directly on extracted game variables. Each of the clusters can then be identified as a 'Player Persona'. This information can then be utilised in a number of ways. Firstly, it can be fed back to the game designers to generate agents with the found 'Player Personas' to allow automatic playtesting, as discussed previously. Secondly, the size of each cluster can be used to infer how successful a given 'Player Persona' is, as players will tend to drift towards the more successful personas. Consequently, if the cluster sizes of the playerbase data are unbalanced, this suggests that a rebalancing of the game mechanics may be required to resolve this issue [19]. Alternatively, the 'Player Personas' found through clustering could be used to improve the viewer experience of Electronic Sports (Esports) events (*e.g.* League of Legends World Championship, Fortnite World Cup). This could be achieved by providing additional information on top of the game stream, in the form of overlays, to give the viewer an enhanced understanding of what any player was trying to achieve. This could then help explain the complexities of the strategies the player was using in the game to new viewers, which could lower the barrier of entry and improve the enjoyment of watching these games [26].

Next, the literature around clustering will be explored, as this is a fairly common technique to extract 'Player Personas'. Following this, a set of examples will show how these clustering techniques have been applied within the video game domain.

2.2 Clustering

To date, the most common tool to discover playstyles from within a dataset of player data is clustering. Clustering algorithms are used to group or cluster a dataset into subgroups based on some underlying structure of the data. However, this is far from the only application of clustering, in fact, clustering is used in a wide range of scientific fields (*e.g.* Molecular Biology [77], Medical Diagnostics [43], Crime Analysis [4], *etc.*). The fact that clustering has been broadly used across many fields is particularly beneficial as it has led to the development of a wide range of different algorithms.

Of these developed techniques the traditional algorithms can be separated into groups. A number of these were discussed in the literature [110] including, but not limited to, methods based on: 'partition' (*e.g.* K-means [69]), 'hierarchy' (*e.g.* BIRCH [112]) or 'distribution' (*e.g.* GMM [84]). Due to differences in approach, each of these types is generally suitable in different situations depending on the properties of the data

generated within that domain.

When considering clustering in the video game domain, although time independent features are possible, it is considered that to fully represent all playstyles the features should be time dependent. This is thought to be the case, as the time actions were taken may demonstrate the player's priorities in the game and hence have a bearing on the playstyle being displayed.

2.3 Time Series Clustering

As discussed, it is believed that it is beneficial to use time dependent features in order to fully represent all possible playstyles within the video game domain. Consequently, a subset of clustering algorithms must be used, known as 'Time Series Clustering', and this section focuses on these different approaches.

Below are the findings from a review paper on 'time series clustering' [5], which is explored below. The authors found that generally the techniques can be labelled in one of three ways. Firstly, 'whole time-series clustering' algorithms which take multiple time series and cluster these based on the similarity between the whole sequences. Secondly, 'subsequence clustering' techniques take a single long sequence, then cluster subsequences extracted from this single longer sequence. Finally, 'time point clustering' which clusters time points using a combination of temporal position and similarity where it not only matters what values are seen but also when they occur.

When considering these options, with regard to which of these approaches would be best used in the video game domain, a few properties need to be considered. Firstly, as the aim is to cluster different players' gameplay sequences, this inherently rules out the use of 'subsequence clustering' techniques, as this only considers a single long sequence. Secondly, it is important to note that while the order in which actions are taken is considered to define the style, the speed at which they are undertaken does not. It is suggested that highly skilled players are likely to be able to make decisions quicker and hence act quicker within the game environment when using the same playstyle compared to players of a lower skill. Based on this consideration it is deemed that 'whole time-series clustering' techniques are most appropriate for the video game domain.

Additionally, when these types of techniques were scrutinised, they can also be further separated into three groups. Firstly, 'shape-based' techniques that stretch and contract the series in the time domain to find the best match. This is used as a distance metric and inserted into standard clustering algorithms to make them compatible with time series data. Secondly, 'feature-based' algorithms which take the time series data and transform it into a lower dimensional vector space. From here classic clustering algorithms can then be applied to the feature vectors. Finally, 'model-based' approaches convert the time series into model parameters and then clustering is applied to these model parameters. When considered within the context of this work 'shape-based' techniques are most appropriate. This is because 'feature-based' and 'model-based' require either the definition of features or a model respectively, in order for clustering to be applied. As the aim of this work is to remove human intervention in order to reduce bias, it does not make sense to introduce another source of human bias at a different point in the process.

The simplest method that falls into the 'shape-based', 'whole time-series clustering'

category, is a one-to-one matching distance metric. This involves taking two sequences and comparing the sequence values at the same timestep at each point in time and combining these distances into a single value to describe the distance between the two sequences. This is then repeated for all other possible pair-wise comparisons within the dataset to create a distance matrix, that can be used to cluster the data. However, due to the simplicity of this method, it has two major drawbacks when considering its suitability for this work. Firstly, as it compares the sequences at each timestep, it is required that the sequences have the same length. However, when comparing two gameplays this cannot be guaranteed. In addition, this method is not robust to temporal shift, which as discussed above, is beneficial for this work.

An alternative method that is robust to temporal shift is Dynamic Time Warping (DTW) [10]. The robustness is achieved by stretching and/or compressing one time series in order to find the optimal match with the other time series. Additionally with the windowing parameter, defining the gap in time with which points can be matched, sequences can be compared even if they are of different sequence lengths. Although this windowing parameter can be set to a finite value in order to reduce the compute time, this restricts the maximum difference in length between the sequences compared.

A further approach uses a similarity function based on Longest Common Subsequence (LCSS) [104]. This method is also robust to temporal shift by allowing the stretching and/or compressing of the sequences. As it is based on previous edit distance techniques, it is more robust to noise. This is realised by allowing points in the sequences to be left unmatched.

For this work, either of the last two mentioned approaches would be considered appropriate. However, as noise is not expected to be a major issue within this application, Dynamic Time Warping was chosen as the best approach to use. Due to the modular fashion of this work, *i.e.* the encoding of gameplay videos before DTW and the clustering after DTW, it would be possible to replace the technique of DTW in the future.

2.4 Player Data Usage in Games

Within the gaming industry, the practise of recording player data has become commonplace. However, the way companies utilise the data post collection, has been evolving over time. Initially, player data collection mainly occurred during the development phase of a game [45]. Often interviews were performed, in parallel with this data collection, with the aim of answering key questions about a game in development. An example of such a question for a 'driving game' would be 'Are the driving controls and game level design successfully balanced such that the driving experience is both realistic and approachable for the player'. The feedback gathered via this data/interview collection can then be used to make the required adjustments to the game, prior to the game being released. At this stage of development, the use of interviews is feasible due to the small number of players when compared to the number of potential users following the game being released.

It should be recognised, however, that if data is collected post game release, the richness of the data available opens up other opportunities. One potential option would be the extraction of 'Player Personas' as previously discussed. The literature cited below considers how best this can be achieved by the use of clustering within games.

The initial aspect that needs to be considered is how to apply clustering within the video game domain and subsequently identify any issues that arise from this. A total of seven main issues have been identified in the literature [8] and are listed below, although not all of these are unique to the game domain.

1. **Validation**: Testing procedures are not always available.
2. **Interpretation and Visualisation**: It is key to be able to visualise and interpret the found clusters in order to effectively act on the found insights.
3. **Time Dependency**: How a player interacts with a given game is most likely to evolve over time and hence this must be accounted for in clustering.
4. **Progress Dependency**: Progression systems in games mean that a player may gain access to additional equipment throughout the game and hence their playstyle may adapt to these newly acquired attributes.
5. **High Dimensional Data**: Within games, the feature space is often extremely large and hence systems must be able to deal with this.
6. **Data Type Mixing**: The available feature set is commonly a mixture of data types, for example, continuous data (*e.g.* kill/death ratios) or categorical data (*e.g.* character type) and hence the systems have to be able to account for this.
7. **Feature Selection**: Given the large number of available features within games, it is unlikely to be beneficial to interact with all of them and hence it is key to select a good set of features.

Next cases in the literature will be reviewed where clustering has been attempted in the video game domain. In each case, an overview is given describing what has been attempted. Following this, a discussion is provided into how each of these approaches deals with the previously introduced issues.

The first example attempts to use standard clustering techniques (*e.g.* K-means, Gaussian Mixture Models, Archetype Analysis, *etc.*) to cluster playstyles within a First Person Shooter game called 'Destiny' [27].

Several issues that have been discussed above can be seen in this approach. Firstly, the data itself has a high number of available features (High Dimensional Data), hence a subset of features needs to be selected for the clustering to be applied (Feature Selection). 'Proportion of Kills using Shotgun' and 'Average Kill Distance' are two of such selected features used within the author's analysis. Although human feature selection was required in this particular case, it is possible that these features could be transferred between games of the same genre, however, some very game specific aspects may be missed in this transfer.

After clustering was conducted, interpretation (Interpretation and Visualisation) was attempted by examining the feature values seen in each cluster. This was performed on both the Player vs Player (PvP) and Player vs Environment (PvE) datasets and led to the following suggested playstyles. Firstly, 'Aggressive Close Range' playstyle which use 'melee attacks' and the 'shotgun' to get kills at a short range. On the other hand, the 'Marksmen' cluster uses either the 'hand canon' or 'sniper' from a long distance in

order to achieve kills. Thirdly, 'Sitting Duck Sniper' stays in one location on the map. The first two playstyles were found in the PvP dataset whereas the third was found in the PvE dataset. Interestingly, the defining features for most of the found playstyles in this paper can be split into the two categories 'Navigation' and 'Interaction' which were introduced when defining 'Player Personas', see Section 2.1.

It is clear that playstyles can be linked directly to a set of features chosen at the beginning of the analysis. Hence it is important for successful clustering to select a good set of initial features. Although this selection can often be a difficult challenge for the game analyser, it becomes worse significantly for a new game when the types of playstyles expected are unknown.

Further, it is also noted that as the features are taken as final values, the time dependency of the playstyle (Time Dependency) is not accounted for within this referenced work. This results in the authors not capturing nuances of the playstyle as they change through time.

A second literature source also took a similar approach of applying standard clustering algorithms to a subset of variables from two additional games: The Exiled Realm of Arborea (TERA) and Battlefield 2: Bad Company 2 (BF2BC2) [28]. In these cases, the authors used normalisation strategies in order to deal with the various data types of features (Data Type Mixing).

In this literature source, interpretation of the clusters was also conducted on both games (Interpretation and Visualisation). An example from TERA was known as 'Stragglers' which had low values for all features, which meant that players were not only not achieving kills, but they also were not being killed either. A further example from BF2BC2 was known as 'Veterans' which were characterised by having a long 'playtime' as well as a high level of skill. These can also easily be traced back to specific features and hence further reinforces the importance of choosing suitable features. However, when analysing the playerbase data, an understanding of the game is required, as is demonstrated by the differences in features chosen across the considered games.

Similar clustering algorithms were also applied to the game World of Warcraft (WOW) [29]. However, in this case, the feature vector contained game variables (*e.g.* player level) over time. Although, this does not fully address the time dependency issue (Time Dependency), as this approach considered how the variables changed over time and attributed that to a playstyle, rather than understanding that a playstyle may change over time evolving as the environment changes. Hence actions should be considered within a shorter time window.

Within this literature, interpretation was also attempted (Interpretation and Visualisation). One possible playstyle was characterised as 'Casual Player', where the player slowly transitions between different levels within the game, namely level 10 and level 20. The second playstyle was named 'Hardcore Player' and those who were deemed to have adopted this playstyle quickly progressed to level 70 and then progressed to level 80 at a later point.

Hierarchical Clustering has also been applied within the game 'StarCraft: Brood War' [49]. In this case, a greater focus was given to the time dependency (Time Dependency) of a given playstyle. This was achieved by not only tracking how the chosen feature, 'military composition', (Feature Selection) changed over time but also re-clustered at a given time interval. This allowed the strategy to be tracked over time to see how it

evolved, as many strategies may start in the same way before diverging over time. The strategies that were found often showed the player's preference in what units to build and also in which order buildings were constructed. Further predictions were made based on the player's adopted strategy (*e.g.* 'Player's Faction' or 'Likelihood of Winning'). These could also be implemented within overlays for any viewer to further improve the viewer's experience.

Up to this point, all the introduced clustering techniques have focused on clustering purely based on feature information, by either final feature values or a trace of the features through time. Although, within the literature, a tool was proposed called 'Gamalyzer' [79] which instead focused on the sequence of actions. This however makes the strong assumption about the availability of the player's actions and hence reduces the dataset types on which the tool can be used.

An alternative use for player data is the classification of playstyles, instead of using clustering to extract them. This has been applied in 'Dota 2' [31] in order to identify the player's role, or in other words the strategy the player has adopted. However, this requires methods in order to extract a selected set of features (*e.g.* 'lane', 'damage type', *etc.*), as well as a known set of strategy labels (*e.g.* 'Gankers', 'Pushers', *etc.*) for the dataset. Similarly to the game predictions for 'StarCraft: Brood War' discussed above, these could also be integrated into overlays to improve the viewer's experience.

Within the work in this thesis, it was decided to focus on an unsupervised clustering approach to extract 'Player Personas' from the playerbase data. Below consideration will be given to how each of the seven issues raised in the literature and stated above should be addressed when clustering within the video game domain.

Firstly, although Ground Truth (GT) labels will not be utilised during training, it is assumed that the Ground Truth playstyle labels will be available. This allows the evaluation of the quality of the clusters found within each game (Validation). Although this is only applied to the games explored within this thesis, it does provide a certain level of assurance of the method's ability to correctly cluster playstyles.

Secondly, visual artefacts will be generated to demonstrate how gameplay sequences differ. In addition, projections will be produced to show how playstyles relate to one another (Interpretation and Visualisation).

Thirdly, the data sequences will be taken either over a short period of time or a single level to cluster the current playstyle (Time Dependency). This would allow future work to re-cluster over later gameplay sequences in order to monitor how the playstyle evolves over time.

Further, although the dependency of the player's progress (Progress Dependency) is not directly accounted for, it is assumed that all players are at an equal level of progression. This means that all players are considered to have the same abilities.

Finally, a choice was made to cluster based on the game screen data. This clustering was performed on this data source as it was suggested that whilst the player chooses which playstyle to use, any playstyle can be extracted from the data provided to the player (*i.e.* game screen). By using this data source, the feature selection from a large number of game features (Feature Selection) is not required. Additionally, the screen data is expressed as a single data type and hence the mixing of types is not an issue (Data Type Mixing).

In theory, as all the relevant features are captured within the game screen data, no

biases are introduced by selecting a subset of the features. Further, the game screen data is time dependent and hence the time dependent parts of playstyles can be found.

Once the data source has been selected, the clustering algorithm must be chosen. As mentioned above, DTW will be used in order to generate a distance matrix before clustering is applied. However, most current clustering algorithms require a defined set of parameters, for example, the number of clusters within the data. Given these parameters need to be tuned to a given dataset, the aim is to design a clustering algorithm with a fixed set of parameters that do not need to be manually tuned for different datasets.

Initial experiments attempted to compare the game screens directly, but this was found to be unsuccessful. This failure was deemed to be the case as even minor frame image changes (*e.g.* object translation) caused large frame distance differences, even though they had very little effect on the playstyle being demonstrated. This led to the investigation of the use of unsupervised representation learning in order to transform the frames into representations in an automated fashion. However, the representation should contain both 'navigation' and 'interaction' information to ensure that all playstyles are fully encapsulated within the representation. In the following section, the literature on unsupervised representation learning will be discussed.

2.5 Unsupervised Representation Learning

Unsupervised representation learning is used within Machine Learning (ML) to transform complex data into a simpler representation space. Generally, this space is then used to solve a downstream task (*e.g.* classification, clustering, reinforcement learning, *etc.*), with the success of the solution being directly linked to the representation space chosen. This has hence led to a range of approaches being designed to encourage the generation of different types of representation depending on the downstream task requirements. A total of four types of approaches have been defined [65] based on the goals that have been used to train the encoder:

Observation Reconstruction
: Aims to learn a representation space from which the original observation input can be generated.

Forward Model
: Aims to learn a representation space from which a prediction can be made about the future.

Reverse Model
: Aims to learn a representation space that allows the prediction of the action that caused the transition between two sequential representations.

Prior Knowledge Constraints
: Aims to learn a representation space that minimises/maximises a given set of loss/objective functions that expresses the prior knowledge.

2.5.1 Observation Reconstruction

A number of methods have used the observation reconstruction goal to train encoders to transform high dimensional data into low dimensional data. This is achieved by 'bottlenecking' the network into a low dimensional layer before reconstruction of the observation is attempted. The network is then trained to minimise the reconstruction error between the generated and original observation. This is known as an Autoencoder (AE) [51].

This technique was developed and refined further into the so called Variational Autoencoder (VAE) [62]. This employs a similar setup to AE by 'bottlenecking' the network, but a noise term is introduced to the representation layer prior to its reconstruction. This aims to ensure that a smooth transition is seen as you move through the representation space. Additionally, a regulating term related to the Kullback-Leibler (KL) divergence is implemented within the overall loss.

These methods work on the idea that if you are able to accurately reproduce the provided input from the low dimensional layer, the learnt representation space effectively contains all of the relevant information of the provided input.

2.5.2 Forward Model

Forward models aim to learn to predict the future. The Action-Conditional Video Prediction method [78] aims to learn a representation space, in which the next representation can be predicted either just from the current representation or the current representation combined with the action taken. The predicted future representation is then decoded into a future frame prediction and the error of this prediction is then optimised.

An alternative approach, Contrastive Predictive Coding [102], is optimised to predict the representation directly. Additionally, these predictions are performed over multiple time steps.

These techniques work on the assumption that if you are able to successfully predict the future from the representation, the representation must contain relevant information about how the observed situation operates.

Further applications integrated forward models within the learning process, for example, to generate rewards based on curiosity [81]. This was achieved by providing an additional reward signal based on the accuracy of the future prediction. It was suggested that inaccurate predictions indicate unexplored situations, and hence the reward signal encouraged the exploration of these parts of the environment.

2.5.3 Reverse Model

Reverse models, on the other hand, aim to predict the action that caused the transition between two sequential states. This can be achieved by one of two methods depending on the action type. For discrete actions (*e.g.* move left), a classifier is trained to select the action that caused the transition. Alternatively, for continuous actions (*e.g.* turn the steering wheel by x), regression is used to predict the action values.

Generally, reverse models are used within multi task optimisation [91, 81] in order to improve the learning of the main task. This is based on the idea that having a

representation that is able to predict the action that caused the transition must include relevant information about how the environment operates.

2.5.4 Prior Knowledge Constraints

Prior knowledge constraints are integrated into the loss function of the network to encourage a set of characteristics based on prior knowledge. For example, within the robotic domain, prior constraints can be introduced based on our knowledge of the laws of physics seen in the real world [57], a couple of examples are discussed below.

Firstly, 'Temporal Coherence Prior' could be implemented. This suggests that any task relevant information should change slowly over time, as this is how it is observed within the real world. This can be encouraged by the minimisation of the difference between sequential representations.

Secondly, 'Repeatability Prior' could be implemented. This proposes that the current task's relevant information and the action taken, determine how the task information changes. This means that if the same action is taken in similar states, then similar outcomes are expected to be observed. This can be encouraged by minimising the differences between both state changes when their initial conditions are similar, and the same action is taken.

Alternatively, priors could be defined based on the relationship between features in sequential states. This is what Spatiotemporal DeepInfomax (ST-DIM) [6] was developed to do. Before the definition of the priors, two layers of features were defined at different points within the encoder network. These were the local features that still contained spatial information and global features which did not. Two priors were then defined on the local and global features. Firstly, the mutual information between local feature patches in the same spatial location in sequential states should be maximised. Secondly, the mutual information between the global features and all the patches of the local features of the next frame should also be maximised.

In summary, all of the aforementioned priors aimed to encourage the representation space to have certain characteristics which humans have deemed as useful to generate a good representation space.

3 Literature Review: Autonomous Agents

Within this chapter, focus will be given to the literature related to the learning side of this work. Over recent years, research in AI has led to the development of an array of techniques that have allowed AI agents to learn and complete a diverse set of tasks. This was commonly achieved using Reinforcement Learning. Initially, the general ideas around these techniques will be explored. However, given that the aim of this work is for AI agents to be able to learn human-like behaviours, the Imitation Learning algorithms that learn from human demonstrations seem an appropriate approach and hence the literature on this area will be specifically discussed. Following this, methods for Imitation from Observation (IfO) will be explored. These techniques aim to remove the need for player actions in the demonstrations. Finally, applications for these techniques will be discussed.

3.1 Reinforcement Learning

The overarching idea of Machine Learning is to learn to make predictions on unseen data, based on an understanding gained from the provided data. Generally, ML is split into three learning paradigms, namely supervised learning, unsupervised learning and reinforcement learning [66]. In the supervised learning paradigm, data is provided with a set of labels. The aim is to then be able to predict the label for any new data using either classification or regression, depending on whether the labels are discrete or continuous. On the other hand, the unsupervised learning paradigm takes unlabelled data and aims to gain an understanding about the structure of this data. This is commonly achieved via either clustering or representation learning. In the final paradigm, namely Reinforcement Learning, algorithms are provided with a reward signal where the aim is to learn how to maximise the total received reward via trial and error. This hence requires the system to be able to interact with the environment. The development of deep learning has led to the use of deep neural networks in a range of cases, and this includes all three of the previously introduced ML paradigms [64].

Within this work RL techniques [94] were used in the development of the IL algorithm and hence further details relating to the Reinforcement Learning paradigm will be discussed. The overall aim of Reinforcement Learning is to learn how to maximise the received reward. To achieve this, the agent must learn which action to take in any given situation. However, the agent must consider that rewards may be delayed, meaning that the reward is not received until many steps after a decision is taken. To evaluate these delayed rewards, a trial and error approach is required in order to find them due to the delayed consequences of actions taken.

In order to trial new ideas, interaction with the environment must be possible and this is a key feature of Reinforcement Learning. This interaction takes the form of a continuous loop, as shown in Figure 2. The agent receives an observation based on the current environment state, but it should be noted that this may not contain all the information about the environment's current state. Based on this observation, the agent must then choose an action to take, which is passed back to the environment. This action then has some effect on the environment. After which, a new observation is passed to the agent alongside a reward value which is based on a reward function. The

agent is hence trying to maximise the reward and therefore the reward function rewards the behaviour that is trying to be learnt. For example, within the game domain, the game score is commonly used as the reward to encourage good performance.

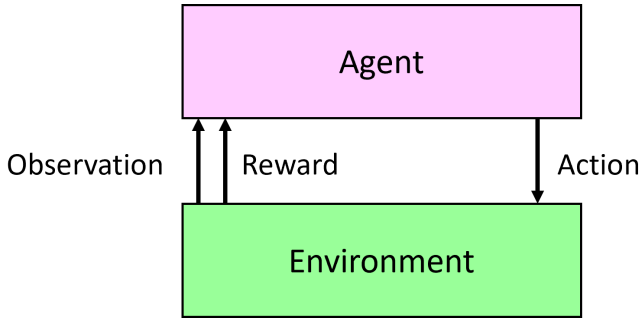


Figure 2: Reinforcement Learning Loop

A significant issue within Reinforcement Learning is the balancing of exploration and exploitation. To maximise the reward, the agent should exploit what has been learnt. However, given that it is possible that better rewards are available, the agent should also explore their possible actions to find better rewards. Further, this exploration must be conducted for many steps, as the effect of an action may not be realised until many steps into the future. This issue becomes even more prominent within a stochastic environment as the action sequence must be tried multiple times in order to gain a true understanding of the reward it manifests. Additionally, issues arise as the agent's actions affect which future state the agent transitions to, and hence long-term planning may be required to arrive at a state in which a large reward can be achieved. Further, as the agent can assess the developing situation based on sequential observations, changes to any plan may be required if progress is not being made as anticipated. It should be noted that in every case the agent aims to improve its overall performance based on the previous experiences it has had.

When considering approaches within the Reinforcement Learning paradigm, generally they can be split into three categories called actor-only, critic-only and actor-critic [44]. The actor-only techniques look to learn the policy directly by outputting the action to be taken based on the current situation. On the other hand, critic-only techniques aim to learn a value function to predict the future reward based on the situation depending on what action will be taken. In this case the policy can be indirectly found by taking the action that predicts the highest future reward. The final technique, namely actor-critic, looks to learn both of these elements in tandem. In this case, the critic is used to evaluate the policy.

One of the earliest developed algorithms in the Reinforcement Learning paradigm was called Q-learning [108, 107]. This algorithm aims to learn the state action value function, referred to as $Q(S_t, A_t)$, where S_t and A_t are the state and action at time t respectively. In its simplest form, this is recorded as a table. The values in the table are then updated based on the difference between the immediately experienced reward, R_{t+1}, plus the discounted maximum reward expected from the next state and the current predicted state action value. The discount factor is represented by γ. In addition, this update is multiplied by a learning rate, α, in order to allow a smooth transition to the true value, the update formula is shown in Equation 1 [94].

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \quad (1)$$

As mentioned, deep learning can be used within the Reinforcement Learning paradigm. This has been applied to Q-learning in the development of Deep Q-Network (DQN) which uses a neural network as a function approximation for the value function. Further, convolutional neural networks can be used to learn directly from image inputs and here great success has been previously found in a range of games [72, 73].

Since the initial conception of DQN, many extensions have been suggested to the original algorithm. These include having two networks, one online for the policy and the second which is used for evaluation. The parameters of the evaluation network are updated from the online network at a given interval. This aims to help address the over estimation problem [103]. Further, network alterations were suggested by estimating both the state value and action advantage. These two values can then be combined to estimate the Q-values [106]. Additionally, prioritised replay was proposed to help in the selection of important experiences, instead of just uniformly sampling the experiences [89]. On top of these, many more changes to the algorithm have been proposed in the literature.

Above, only a brief description of Reinforcement Learning has been given, as no major alterations to the algorithm have been suggested. This work proposes a new method to generate a reward function based on the similarity of the agents' playstyle to a set of playstyle demonstrations. However, this reward definition allowed a minor alteration, in the form of controlling exploration and exploitation. Given the known reward definition, the exploration can be reduced as the predicted reward approaches the maximum theoretical reward.

3.2 Imitation Learning

Imitation Learning is a paradigm used to train a policy to complete a task by providing the learning system with a set of demonstrations which shows how to complete the given task. This allows the teacher to transfer knowledge via a set of demonstrations instead of having to define a reward signal for the given task.

A survey of imitation techniques [113] explored different categories in which the taxonomies could be divided, a couple of options are discussed below. Classically, imitation methods can be split into two category types, namely 'model-based' and 'model-free'.

The first of these categories, namely the 'model-based' methods, learn the dynamics of the environment, commonly by either a 'forward model' [37] or an 'inverse model' [74]. This model is then used to help train the policy. A 'forward model' aims to predict the next state given the current state and action. On the other hand, an 'inverse model' aims to predict the action that caused the transition between the current and next state.

The second category, namely the 'model-free' methods on the other hand do not aim to learn about the dynamics of the environment. This category in itself can be further split into three subcategories. Firstly, Behaviour Cloning which aims to learn to map directly between the observations and actions. Secondly, reward engineering techniques, that aim to extract a reward function, that can then be used to train a policy

[14]. Finally, adversarial style methods that utilise Generative Adversarial Networks (GANs) [40] in order to train the policy [52].

Within the same survey [113] an alternative category split was also offered. Conventionally, the category split would normally only contain Behaviour Cloning (BC) and Inverse Reinforcement Learning (IRL). BC aims to map directly between the observation space and action space, whereas IRL aims to extract a reward function and then use standard RL techniques to learn the policy. However, it was suggested by Zheng et al that a third type should be added to the split namely Adversarial Structured Imitation Learning. Although this new category is similar to IRL, the aim was to identify real/fake transitions and hence learn a policy based on this signal. However, this signal is unlikely to recover the reward function and hence differs from IRL in this respect.

In the above survey it was noted that 'Inspired by the presence of GAIL, many recent papers adopt this adversarial structure, and inevitably, GAIL becomes the baseline for comparison.' Based on this it was decided that any developed techniques would be baselined against both Generative Adversarial Imitation Learning (GAIL) and GAIL variants that operate purely based on observation.

A second survey [55], explored the workflow of the methods used, instead of splitting the types of techniques. The first step of this workflow relates to the source of the demonstrations. This source can be one of three sensors: teacher sensors, external sensors or learner sensors. Although ultimately the sensor type differences distils down to a couple of questions. Firstly, is the point of view the same for the collected demonstrations and the learner? Secondly, do the teacher and learner have the same capabilities? Within this work, demonstration collection is performed on the learner's sensors and hence the point of view and capabilities are the same for the agent collecting the demonstrations as for the learning agent. However, the environment the agent is in may be altered.

The second step in the workflow considers the feature representations used. In its simplest form, the raw features can be used, although, this can often result in a high dimensional feature representation. An alternative is to use designed features, although, these need to be manually created and hence require expert game knowledge to ensure the relevant features for the task are kept. Further, access to the game code may be required to extract these features. On the other hand, feature extraction can be used, as it allows the automatic extraction of a lower dimensional feature representation. However, it should be noted that consideration must be given to the method used to once again ensure that the relevant features are maintained. In this work, feature extraction was used using an unsupervised representation learning technique that was designed to learn the key features that relate to the playstyle being used.

The next step was to learn from the demonstrations. This can be achieved by mapping the observation to the action via either classification or regression depending on whether the action space is discrete or continuous. However, the direct learning approach is generally not enough to learn a robust and human-like policy. In these cases, policy refinement is required in order to generate a policy with improved generalisation. Although this refinement can take many forms, *e.g.* Active Learning (querying an expert for the best response) or Apprenticeship Learning (attempting to recover the reward function), each of these approaches aims to explore situations not seen within the demonstrations and aims to learn how to act appropriately.

Once the imitation has been completed, questions arise about how to evaluate how well the imitation has been performed. This can be achieved using quantitative evaluation. This normally takes one of two forms. Firstly, by error calculation which measures how close to completing the task the agent is, *e.g.* when the task is to place a ball in a cup, the error is the distance the ball is from the cup. However, this kind of measurement is not always possible. Another such method is quantitative scores, for this type of evaluation a set of criteria are defined for success. For example, for driving, the number of collisions could be recorded, and correct imitation of a good driving technique should lead to fewer collisions. These kinds of evaluations are useful when the completion of the task is more important than how it was achieved and if the behaviour was believable. Although, if the believability of the behaviour is more important than the task, the second type of evaluation, namely qualitative evaluation, is appropriate. For this type of evaluation, examples of the behaviours are given to different human evaluators and a combination of their rating scores are used to score the quality of the imitation.

For the imitation part of this work, two evaluation methods were used to evaluate the quality of the imitation of the playstyle. Firstly, a criterion was set to define what playstyle was being used based on the game variables. This allowed the evaluation of which playstyle was used and in addition if the level was completed under that playstyle. Secondly, a metric that measures the similarity of a playstyle within the gameplay was used to evaluate the similarity of the playstyle from the policy, compared to the demonstrations provided. This aimed to provide an indication regarding how believable the gameplay was.

Up to this point, all the imitation methods mentioned have focused on single-modal imitation. This type of imitation aims to learn a single task. The other method type is called multi-modal imitation which attempts to learn multiple tasks at the same time. Techniques have been developed that can achieve this [48, 67], for example by passing a latent code into the network to control the task attempted. Here, if the system was imitating the ability to drive a car, the latent code would control whether the car turned left or right.

Within the imitation section of this work, a distinction has been made between multi-task and multi-style imitation. For multi-task imitation, each of the tasks commonly relies on similar skills to complete the task however the end states are often different. For multi-style imitation, the overall task was the same and hence this results in the same/similar end state, although the strategies used to complete the task vary across the playstyles.

In this work, it was decided to focus on single style imitation to develop relevant methods for this type of imitation. Following this, it is suggested that future work could look to expand this method from single-style imitation to multi-style imitation. The literature discussed in the rest of this section will focus on these single-style imitation methods.

One of the simpler options for imitation, previously mentioned is called Behaviour Cloning [88, 12]. This approach transforms the IL problem into a supervised learning problem. This is achieved by teaching a policy to map the current state to the action taken. Although, this makes the assumption that the learning system has access to both the state trajectory and the action sequence for the demonstrations. However,

the simplicity of this technique generally means that it runs into three main problems. Firstly, the policy does not generalise, and hence problems are encountered if there is a change in the state distributions between the demonstrations and the environment in which the imitation was being performed. Secondly, the predictions are only made based on the next action and this means that the policy is not capable of any long-term planning. Finally, any small errors made by the policy are likely to be compounded over time as the policy is unlikely to be able to recover from any deviation from the path.

Further methods have also been built on top of standard BC, in an attempt to fix some of the aforementioned issues. One such method is called Disagreement-Regularized Imitation Learning [13] and aims to address the state distribution issue. Initially, the algorithm starts by running standard BC, but this is applied to an ensemble of policies. Following this, a two-step process is used to train the final policy. The first step performs an update using the BC loss function. This step aims to ensure that when the agent is acting within the demonstration distribution, the agent acts as shown in the demonstrations. The second step performs a single policy update with the aim of minimising the variance of the policy ensemble. This step aims to ensure that when the agent is outside the demonstration distribution, the agent moves towards the expert demonstration distribution. This is achieved by minimising the variance, as in these cases within the distribution, all the policies should agree and hence have a small variance. On the other hand, when outside the distribution, the policies are more likely to disagree and hence have a larger variance. By minimising this variance, the agent is encouraged to move back towards the demonstration distribution.

Another type of approach, also previously mentioned, is called Inverse Reinforcement Learning [75]. The first step of IRL is to identify a reward function that can be used to explain the demonstrations, assuming that they are acting optimally under the reward function. Once this has been defined, a policy can be trained on the reward function using a standard RL technique. These two steps are then repeated to further refine the reward function and hence result in a policy that acts more like the demonstrations. However, this means that the Reinforcement Learning problem needs to be repeatedly solved in a loop. Solving this type of problem even for a single instance can be time consuming and repeatedly solving it can take a great deal of time. Although, this can be mitigated to a degree by only stepping towards the solution within each loop iteration instead of fully solving the problem.

One technique that falls into this category [3] uses a linear combination of the feature vector to define the reward function. However, in this thesis, the aim was to operate on the raw frames and hence a feature extraction technique would be required to use this method. An unsupervised representation learning technique could be used to generate an encoder to extract a feature vector from the frame. However, issues are likely to arise as many reward functions could be defined to explain the demonstrations. One such reward function that would explain the demonstrations, would be a reward based on the path the agent had taken through the level. While this would explain the demonstrations, it would not allow the transfer of the playstyle to different situations. This is because such a path may need to be altered if an object that was required, in order to maintain the playstyle, was moved.

Finally, the aforementioned category, Adversarial Structured Imitation Learning was considered. Each of the methods in this category builds on Generative Adversarial

Networks (GANs) [40] in order to train a policy based on a reward signal from the discriminator. Generative Adversarial Imitation Learning (GAIL) [52] is a technique that falls into this category. In tandem with training the generator (policy), this technique trains a discriminator to be able to distinguish between real (from the demonstrations) and fake (from the generator) transitions. In this case, the transition takes the form of a state action pair. The two networks are trained in competition and hence this encourages the policy to generate transitions similar to the demonstrations. Within a code implementation [115], an alternative definition of the transition was also considered. In this version, the transition took the form of the current state, action, and next state triple, and will be referred to as Generative Adversarial Imitation Learning - Triple (GAIL-T). These types of methods aim to help improve the generalisation of the learnt policy. By creating a reward based on the 'realness' of the transition, it is possible to provide a good reward signal in situations that are not seen in the demonstrations, if the transition shares characteristics with these demonstrations. However, if there were differences between the demonstration level and the imitation level, then the discriminator could split the transition based purely on this and hence no good rewards would be seen, even in transitions that match the behaviour shown in the demonstrations. Further, GAIL and GAIL-T reintroduce the assumption that the learning system has access to the actions used in the demonstrations, which cannot always be guaranteed.

Other methods have also been developed that define a reward function based on the similarity to the demonstrations. One such method is called Soft Q Imitation Learning [85]. In this technique the replay buffer is loaded with the demonstrations and the reward for each transition is set to one. Further, throughout training, the agent's experiences are then added to the buffer, with the reward signal always being set to zero. Although, it should be noted that when samples are drawn from the replay buffer, an equal split of demonstrations and experiences are chosen. Overall, this technique looks to encourage the agent to follow the demonstrations, by rewarding on the demonstration transitions. However, as the plan in this thesis was to learn in unseen environments, issues are likely to arise as the observation from the new environment will never see a reward.

Another such method is called Primal Wasserstein Imitation Learning [25], which aims to imitate the demonstrations by looking to minimise the wasserstein distance between the state action distributions of the demonstrations and the learnt policy. Once again, this technique assumes the availability of the player's actions in the demonstrations. Not only this, but if the environment in which the imitation was performed was changed compared to where the demonstrations had been collected, the state action distribution would not be the same when acting with the same playstyle. Potentially, this could mean that minimising the difference in the state action distribution does not result in the recovery of the displayed playstyle.

Further, methods have looked to imitate a diverse set of behaviours [105]. To achieve these diverse behaviours, an adversarial based method has been designed where the discriminator was conditioned based on the embedding of the current state. Here the aim was to allow separate rewards for each trajectory and hence avoid mode collapse, where only one type of behaviour was performed. On the face of it, this seemed to be a relevant technique, as the aim was to imitate a set of playstyles, which should be diverse. However, in this work, the aim was not just to create diverse behaviour, but the behaviour should relate to what was found in the data. The plan in this thesis was to

first cluster the data into separate playstyles and hence the imitation should only learn a single style. Additionally, this method also assumed that the actions were provided within the demonstration set.

Other methods have also used demonstrations to help speed up the convergence of Reinforcement Learning, particularly for more complex tasks. This can take many forms, for example, they could be used to pre-train the network [50] or to mix in samples from the demonstrations into the sampled batch at a given ratio [80]. Either way, the aim was to expose the agent to transitions of interest for completing the task, without having to rely on exploration to find them. This idea has been shown in many cases to help the learning of complex tasks. Ultimately however, these kinds of techniques care more about completing the task, than how it was completed. The demonstrations essentially provide guidance towards possible solutions to the task, but on completion, the algorithms aim to find the optimal solution. While in many use cases, this is useful, in this work, more interest is given to replicating the strategies that were used to complete the task, even if they were suboptimal. This means that this type of method is not suitable for this work.

3.3 Imitation from Observation

Until this point most of the imitation methods discussed have assumed the availability of the player's action sequence alongside the state trajectories in the demonstration set. If this assumption could be removed, the collection of the demonstrations becomes easier, as the collection process is now only required to record the screen. Further, this increases the type of data sources that could be used. Recently, uploading gameplay videos to video sharing platforms has become more common and if the action sequence is not required, then this could be used to collect demonstrations.

A survey [99] explored the techniques that attempted imitation based only on observations. Similarly, as discussed for imitation in general, these techniques fall into one of two groups. These are called 'model-based' and 'model-free'. Again the 'model-based' techniques fall into either the 'inverse model' or 'forward model' categories. These models have been defined above. The second group, namely 'model-free', also split into two similar subgroups as before, these being 'adversarial' and 'reward engineering'. Often it can be seen that techniques have been built based on similar ideas used in the methods above, but they have removed the need for the action sequence in the demonstrations.

One method that falls into the 'inverse model' subcategory of the 'model-based' methods is Behaviour Cloning from Observation (BCO) [96]. To remove the need for the demonstration's actions, an 'inverse model' is taught to predict the action based on the current and next state. This is learnt by interacting with the environment and recording the observation trajectory, alongside which action the agent performed. This model is iteratively improved based on new experiences, but the 'inverse model' allows the agent to predict the actions used in the demonstrations. These action predictions combined with the demonstration trajectories, then allow a standard BC technique to be applied. However, as this technique ultimately uses standard BC, issues such as bad generalisation are still present.

The other subcategory for the 'model-based' techniques, attempts to learn a 'forward

model' instead of an 'inverse model'. One such method is called Imitating Latent Policies from Observation (ILPO) [30]. In this type of technique, the aim is to learn a 'forward model' that can predict the next state when provided with the current state and the action taken. However, in this algorithm, the 'forward model' is provided with an encoded version of the state and a latent action in order to predict the next state. Although here the prediction made is the delta in the states, instead of the state directly. This model is trained to ensure that one latent action results in a correct prediction for the next state in the demonstrations. In parallel to this, a latent policy is learnt which aims to select the action that results in the best prediction of the next state. Once these models have been learnt, the next step is to remap the actions, to allow the selection of the correct action, based on the current state and latent action. To achieve this, the agent interacts with the environment and collects current state, action, and next state triplets. Each latent action is looped over to predict the next state, based on the current state using the 'latent forward model'. The best latent action was found by the selection of the latent action that caused the best next state prediction. Remapping then learns how to map from the current state and latent action onto the real environment action. By learning the latent action first, the method requires fewer interactions with the environment. However, as the 'forward model' is trained based on the demonstration, if the environment level is changed, it is likely that the 'forward model' predictions would be less accurate and hence cause issues for the imitation.

Alternatively, an 'adversarial' based method could be used to imitate the demonstrations. The previously discussed 'adversarial' based methods have provided the player action in the transition information which are then passed to the discriminator. In order to convert this method to an Imitation from Observation method, the transition definition must be changed. The transition definition could be defined as the current state and next state, and this is called Generative Adversarial Imitation from Observation (GAIFO) [97]. Additionally, within a code implementation [115], another transition definition was defined. This transition definition meant that the discriminator only received the current state and from now on will be referred to as Generative Adversarial Imitation from Observation - Single (GAIFO-S). On the assumption that frame stacking was being used, this falls in line with another piece of literature that used a single observation but contained a stack of four frames [98].

However, these types of techniques can run into issues if either the perspective of the teacher and learner are different or there are changes in the dynamics of the environment. Each of these provides indicators to the discriminator about whether the transitions are real or fake. The discriminator reward signal to the policy is likely to focus solely on these differences and hence the signal provides no useful information on how to imitate the demonstrations. Several techniques have been designed to overcome these issues, a few of which are discussed below.

Firstly, consideration was given to the issue regarding the change of perspective between the teacher and learner, the referenced method [93] aims to overcome this issue. The ability to imitate from a third person perspective can be beneficial, as under some circumstances, it can be hard to collect first person demonstrations. The base 'adversarial' system is set up to extract features from both the current state (S_t) and the state four timesteps later (S_{t+4}). Both of these feature vectors are concatenated together, both being used to predict whether the transition is real or fake. However, a process

must be introduced to ensure that the features extracted are domain agnostic, in this particular case, if they are first or third perspective views. To achieve this, using just the current frame features, the system attempts to predict which domain the features are from. When training, the sign of this loss is flipped. This means that on backpropagation, all of the information about the domain should be removed from the features. This method has been designed to allow the imitation from demonstrations from a different perspective, however in this thesis, consideration was only given to cases where the environment was changed and not the perspective.

A different method, Indirect Imitation Learning (I2L) [39], aims to address issues around when the dynamics of the environment change. The algorithm was tested on, and shown to be effective in imitation, when controlling different robots, even when the gravity of the environment, the density of the robot or the friction of the system was changed. To achieve this a wasserstein critic was used to score the agent trajectories and this score was used to control which data was added to the buffer. A priority-queue was then used to control the data the discriminator was trained on. This aimed to ensure that throughout training, data added to the buffer has a distribution more similar to the expert distribution than the current training data. Although this technique addressed some changes to the environment transition dynamics, these changes always just altered the effects of the player's action on the robot. Ultimately, the states the agent must transition through to complete the task were still the same. This meant that matching the state distribution was likely to cause the correct behaviour. However, in this work, the aim was to imitate in altered level environments. This meant objects within the environment may be repositioned or the level layout changed, based on this it is suggested that simply minimising the difference in state distribution may not result in the desired behaviour.

Although a set of categories have been defined within the surveyed literature, new methods have been developed that span multiple categories. One such method is called State Alignment-based Imitation Learning [68]. In this case, the method contains the discriminator from the 'adversarial' based methods, as well as learning models relating to the environment from the 'model-based' method category. The technique was designed to deal with changes in the environment dynamics, which took the form of changing the robot the agent controls. To achieve this, the policy was updated on two levels, namely locally and globally. When updating locally, a VAE was used to predict the target next state and then the learnt inverse dynamics model can predict the action that would be required. Then at the global level, the aim was to minimise differences in the state distribution. Minimising the distribution differences alone was not enough to extract the correct policy, as it was possible to maintain the distribution while not fully copying the behaviour. Further, this global element was required due to the changes in the environment dynamics. Similarly, as before, these changes to the environment altered the effects of the agent's actions but not the state trajectory to complete the task.

The final category identified was called 'reward engineering'. These methods aim to generate a reward signal based on the similarity to the demonstrations, and then train an agent using a standard RL technique. One such method Time-Contrastive Networks (TCN) [90] generates a reward signal by attracting towards a 'positive' sample and repelling away from a 'negative' sample. The main version of the algorithm assumed that two viewpoints were available and the 'positive' sample was taken from the same

point in time but from the alternate view. Whereas the 'negative' sample was taken from the same view but later in the frame sequence. Within the game domain, it is unlikely to have access to multiple views. An alternative version of the algorithm was designed with a single view, where the 'positive' sample was within a certain range of the current step and the 'negative' sample was outside this region plus some time margin. Again, this assumed the path to success was the same in both the demonstrations and the environment. As the plan was to alter the version of the level, then the route to success was likely to change. Questions then arise, does the reward signal based on the demonstration path to success translate to success in the changed level layout?

3.4 Imitation Learning Applications

Within the literature, an array of possible applications have been discussed where Imitation Learning can be applied. Below, a number of these applications will be explored and a short discussion about how the proposed methods in this thesis could be used within these areas.

The first and largest commercial use case is within the multi-billion dollar video game industry. Interest has been developing in the use of systems to allow automatic playtesting of new game content. Previous approaches commonly relied on a level of game knowledge to design a set of personas to fully explore the game space [54, 53]. This however limits the variety of personas that can be used in testing based on the selector's knowledge of playstyles. Alternative systems have designed general persona types that can be applied to any game [46, 47], but it is possible that these general personas would not fully explore the game space. The proposed idea is to apply the previously discussed imitation method post-clustering of the encoded gameplay videos from the playerbase. This would allow playtesting to be tailored to the game based on the expected playstyles, instead of testing being driven by metrics such as curiosity [41] *etc.*

A further application is in the development of NPCs. This could be achieved either directly by controlling the NPCs with different playstyle policies learnt through the imitation method. This would aim to give each NPC a unique feel as well as aiming to provide more human-like behaviour, which is getting harder to achieve with the increasing complexity of new games. Previous literature has suggested imitation based methods are able to generate more believable NPCs [95]. Further, a behavioural study [23] found human-like behaviour was a desirable trait for AI agents and hence would be expected to improve the player experience. Alternatively, the learnt playstyle models could be used in a more indirect capacity. In this capacity, the models could be used to either test AI agents to see how they react to playstyles they are likely to encounter when integrated into a game or in the training of AI agents. When training with different playstyle agents, the AI agent should be able to learn how to cooperate with a range of playstyles they might encounter. In the literature, it has been shown that training even with just a BC agent can lead to better cooperation [21]. Each of these ideas looks to improve human character interactions within games and hence aims to lead to a better player experience.

In addition to these applications within games, the proposed methods have several possible applications beyond games [55]. A popular task and test for Imitation

Learning is the control of simulated vehicles. This task is based on the development of autonomous vehicles for the real world, which is currently gaining a lot of traction. The general ideas of imitation are appropriate here, as it should result in rational acting agents. This is particularly important in the cross over region when public roads are full of both autonomous vehicles and human controlled vehicles to enable humans to understand what the autonomous vehicles are trying to do and hence avoid crashes. The idea of training with different playstyles or driving styles could also be useful to allow autonomous vehicles to adapt to human drivers as well. Further, these techniques could be used in the development of assistive robots for either elderly or recovering patients. Each of these individuals is likely to have their own eccentricities and hence being able to train or test with different types of patients should allow the development of assistive robots that are able to adapt to the patient they are with.

4 Encoding Playstyles

This chapter focuses on the encoding of gameplay videos into a meaningful representation space. This will be achieved using a proposed hybrid unsupervised representation learning technique called Spatiotemporal DeepInfomax Variational Autoencoder (STDIM-VAE) designed for the video game domain. Within this chapter, this method will be introduced, alongside a number of methods to interpret the learnt representation spaces. Once these have been described, three case studies will be discussed where a representation space has been learnt using the STDIM-VAE technique and then analysis conducted to gain an understanding of this learnt representation space.

4.1 Spatiotemporal DeepInfomax Variational Autoencoder

When designing a system for unsupervised representation learning, the choice of the loss function is pivotal to the type of representation learnt. The success of the downstream task is dependent on the type of representation being learnt, hence the loss function should be chosen based on this downstream task.

In this case the downstream task is the clustering of playstyles. Based on the reviewed literature, there are two main feature types that need to be considered when describing a playstyle, namely 'Navigation' and 'Interaction'. This means that the encoder system is required to contain aspects designed to encode each of these types of features into the representation to allow the successful clustering of playstyles.

Firstly, consideration was given to the 'Navigation' features. These types of features are heavily based on the player's location and their relative position to objects, other players and Non-Player Characters in the game space. It is suggested that this type of information can be determined based on what the player can see in each frame and hence this should be encoded into the representation. This led to the inclusion of an observation reconstruction goal into the loss function. It was decided to use the VAE, previously discussed, within the proposed hybrid encoder system.

Secondly, attention was given to the 'Interaction' features. These generally contain more high-level concepts such as the type of weapon used. Hence, it is important to encode high-level features into the representation as well. From the literature, it has been shown that ST-DIM is able to generate representations from which high-level game features (*e.g.* Items in Inventory) can be linearly predicted. This is achieved by looking to maximise the mutual information between sequential states. It was therefore also decided to include the ST-DIM architecture, as previously discussed, within the proposed hybrid encoder system.

These decisions culminated in the development of a hybrid encoder that encompasses elements from both VAE [62] and ST-DIM [6] encoder systems. This resulted in the proposed STDIM-VAE system, the architecture for which can be seen in Figure 3. Following is a discussion of the elements taken from each of the previous works and how each of their loss functions have been combined together.

Firstly, focus was given to the VAE [62] section of the hybrid encoder. The overall VAE loss function is made up of two components, these are the reconstruction loss and the KL regularisation loss term. The reconstruction loss component aims to allow the observation to be generated from the representation by minimising the error between

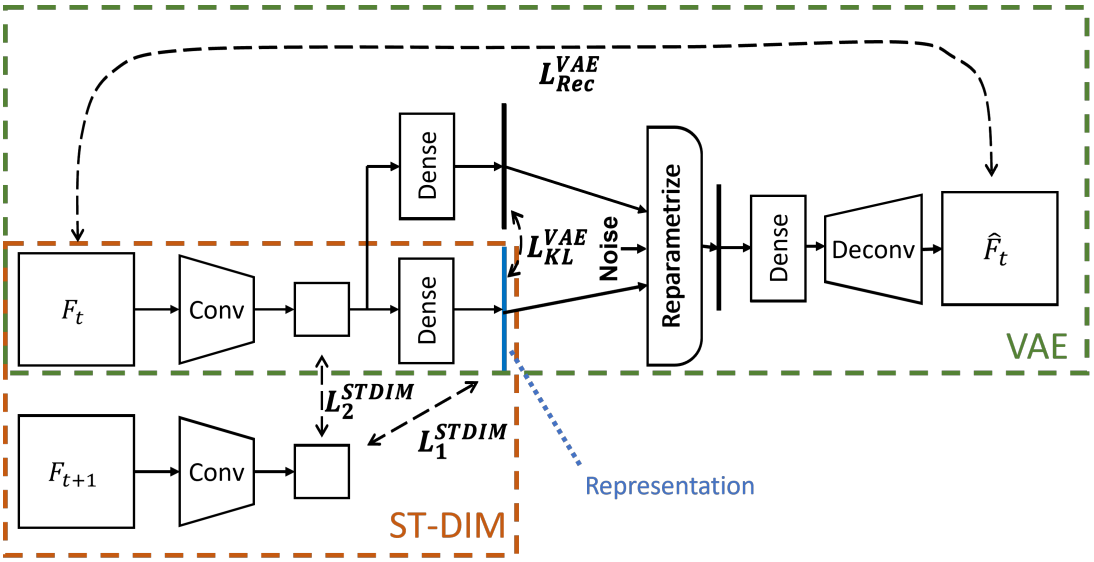


Figure 3: STDIM-VAE Architecture

the original and generated observations. The loss function to achieve this is defined in Equation 2, where $F_{t,x,y}$ is the pixel value at position (x,y) of the original frame at time t and $\hat{F}_{t,x,y}$ is the reconstructed pixel value at position (x,y) of the reconstructed frame at time t.

$$L_{Rec} = \sum_{x,y} \left(F_{t,x,y} - \hat{F}_{t,x,y} \right)^2 \tag{2}$$

The second component of the VAE loss function aims to regulate the representation by the use of KL divergence. This component of the loss is defined in Equation 3 where μ is the mean and σ is the standard deviation.

$$L_{KL} = -0.5 \sum_{i} \left(1 + log(\sigma_i^2) - \mu_i^2 - \sigma_i^2 \right) \tag{3}$$

With each of the loss components stated above, an overall loss function must be defined in order to combine each component into a single loss term that can be optimised. The general version of this is defined in Equation 4 by using a β parameter to balance the regulation term. Within this work the β value was set to 1.0 to get the loss function defined in the literature.

$$L_{VAE} = L_{Rec} + \beta L_{KL} \tag{4}$$

Secondly, focus was given to the ST-DIM [6] section of the hybrid encoder. Similarly, this section also has two loss function components. One aims to maximise the mutual information between global and local features in sequential frames. The second aims to maximise the mutual information between local features in sequential frames. Each of these terms are shown in Equation 5 and 6 respectively. The global features at time step t are represented by GF_t and local features at time step t at patch location (x,y) are represented by $LF_{t,x,y}$. The weights are represented by W_g and W_l.

$$L_{GL} = \sum_{x,y} -log\left(\frac{\exp(GF_t^T W_g LF_{t+1,x,y})}{\sum_{f_i \in F_{next}} \exp(GF_t^T W_g LF_{i,x,y}}\right) \tag{5}$$

$$L_{LL} = \sum_{x,y} -log\left(\frac{\exp(LF_{t,x,y}^T W_l LF_{t+1,x,y})}{\sum_{f_i \in F_{next}} \exp(LF_{t,x,y}^T W_l LF_{i,x,y}}\right) \tag{6}$$

Each of these losses were then divided by the size of the local feature layer in the X and Y dimensions before being summed together to calculate the total loss, this is shown in Equation 7.

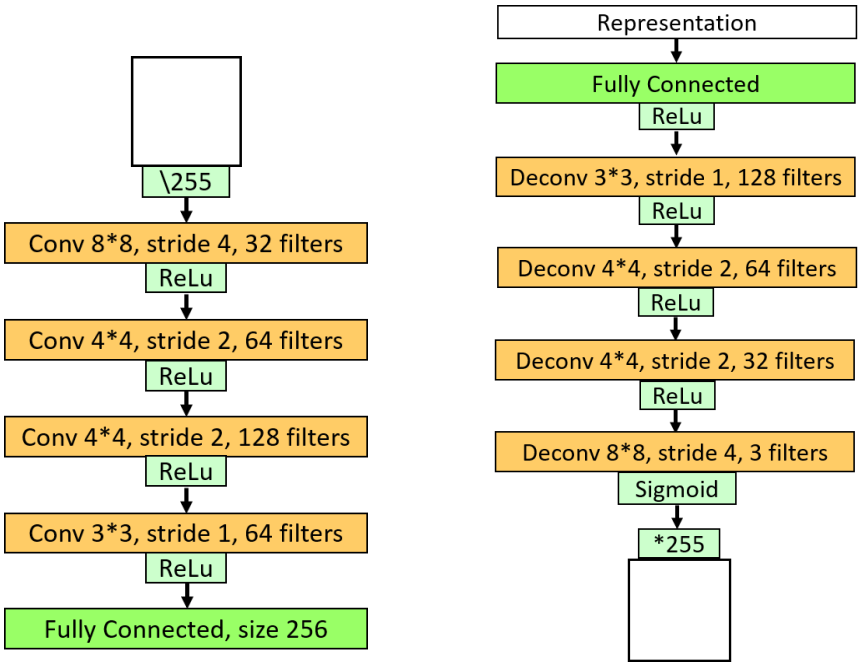
$$L_{STDIM} = \frac{L_{GL}}{XY} + \frac{L_{LL}}{XY} \tag{7}$$

The hybrid STDIM-VAE architecture then needs to combine both of these loss functions. An investigation of these loss values throughout the optimisation process was conducted to find the possible values that each loss component could take. Based on this, it was found that the VAE component was significantly larger, which meant that if the two loss functions were naively added together the VAE component would dominate the total loss. So, it was decided to combine these loss functions in a weighted manner in order to transform them into similar ranges and hence give equal importance to each of the components. It was felt that this was required to ensure that each of the sub architectures contributed equally to the learnt representation. Equation 8 shows how this was achieved, where W is frame width, H is frame height, C is the number of colour channels and B is batch size.

$$Loss = \frac{1}{2}\frac{L_{VAE}}{WHCB} + \frac{1}{2}\frac{L_{STDIM}}{10} \tag{8}$$

After the loss function was designed, the next step was to decide on the architecture for both the encoder and decoder. The structure for each of these blocks can be seen in Figures 4a and 4b respectively.

Ultimately, the aim was to use the learnt representation space to allow the clustering of playstyles based on gameplay videos alone. To allow this, the representations must contain both types of features ('Navigation and 'Interaction'), that are used to describe the playstyle being demonstrated. To accomplish this, a hybrid encoder was designed combining the two previous encoder systems. It is believed that one of these (*i.e.* VAE) will encourage a representation that contains player location or 'Navigation' information, whereas the second (*i.e.* ST-DIM) will encourage higher level information which describes the 'Interactions' that have occurred.



(a) Encoder (b) Decoder

Figure 4: Network Architecture

4.2 Playstyle Interpretation

Once the encoder training had been completed, analysis was conducted in order to explore the meaning behind each learnt representation space. This was conducted on three levels. Firstly, at the single gameplay level which looked to explain what the representation had encoded from each frame. Secondly, at the gameplay *vs.* gameplay level which aimed to explore differences between the gameplays, based on the differences between the representations. Thirdly, at the dataset level which looked to investigate, at a higher level, how each of the gameplays compared.

4.2.1 Explaining Gameplay Representations

The first level of analysis focused on exploring which parts of the gameplay's frames were encoded to the representation. This aimed to help provide an explanation for the representation.

Previous literature has provided methods to allow an explanation from a 'black-box' model for a classification task. One method called Randomized Input Sampling for Explanation (RISE) [82], generates an importance map that highlights parts of the image that are key for a given class classification. This is produced by firstly generating a random set of masks. These masks are then applied to a given input image and the effect of the class label probability is monitored. The set of masks are then combined in a weighted manner using these probabilities as weighting factors. This results in masks that produce a high probability of a given class being weighted highly, and this subsequently leads to parts of the image which led to the model predicting that given

class, being highlighted in the importance map.

However, within this work the aim was to highlight parts of the input that had been encapsulated into the representation. To adjust for this, the similarity between the original representation and the representation generated from the masked input image was calculated. This similarity was then used as the weight, instead of the class probability. This meant that masks that create similar representations are weighted highly, as they contained parts of the input that were contained in the representation. Equation 9 shows how the masks were combined for classification explanation using Monte Carlo sampling. In this case a set of N masks were produced and M_i represents mask i. Further, $\mathbb{E}[M]$ represents the expected value of M. Each mask is then combined with image, I, via element wise matrix multiplication which is represented by $\odot$. This result is then passed through function f which is the classification function being explained. Equation 10 shows how this has been altered for representation explanation, in doing so the classification function f is replaced with the encoding function enc. The overall system, that will be referred to as Randomized Input Sampling for Explanation of Representations (RISER), is shown in Figure 5.

$$S_{I,f}(\lambda) \overset{\text{MC}}{\approx} \frac{1}{\mathbb{E}[M] \times N} \sum_{i=1}^{N} f(I \odot M_i) \times M_i(\lambda) \tag{9}$$

$$S_{I,enc}(\lambda) \overset{\text{MC}}{\approx} \frac{1}{\mathbb{E}[M] \times N} \sum_{i=1}^{N} Cos\_Sim(enc(I \odot M_i), enc(I)) \times M_i(\lambda) \tag{10}$$

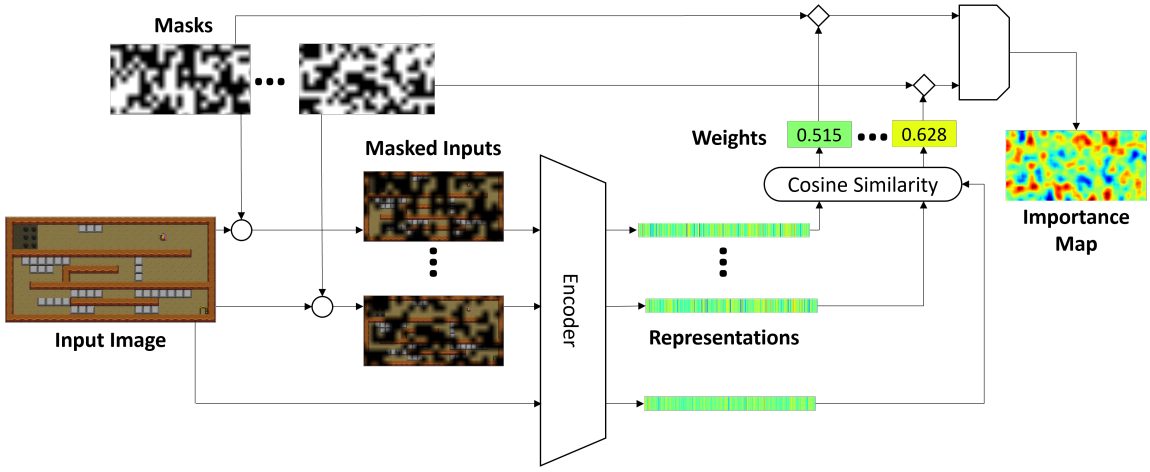
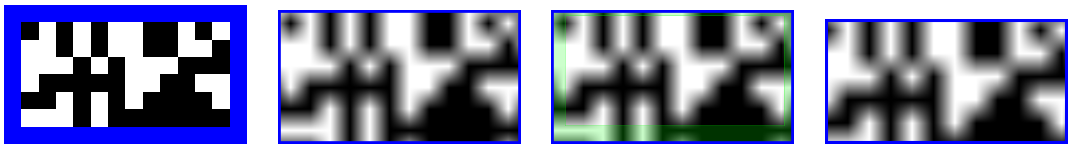


Figure 5: RISER Schematic

Within the referenced work [82], a three-step process was used to generate masks. Firstly, a binary mask was generated with a smaller size than the input image, this is shown in Figure 6a. This was then upscaled using bilinear interpolation to a mask that was bigger than the input image, an example is shown in Figure 6b. The final step was to randomly crop this mask to the size of the image input, as shown in Figure 6c to create the resultant mask shown in Figure 6d. The use of bilinear interpolation creates a set of masks without sharp edges by producing regions with decaying intensity as you reach the edge of the region. Additionally, the use of cropping allows flexibility in

where these regions are placed on the image. By combining both smoother edges and placement flexibility this technique aims to create smoother importance maps. From here on, mask generation using this technique will be referred to as ‘Literature Based’ or ‘LB’ masks.



(a) Binary Mask (b) Upscaled Binary Mask (c) Marked Upscaled Binary Mask (d) Final Mask

Figure 6: Process Steps in the Creation of ‘Literature Based’ Masks

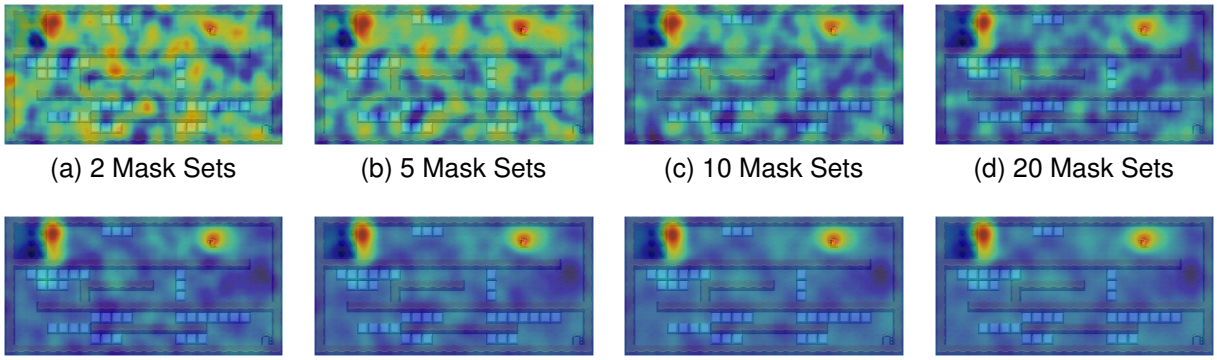
An additional mask generation technique was also experimented with, as a number of games that were investigated had game screens made up of multiple square assets. Hence the effect of simply blocking out one asset per mask was investigated. An example of this kind of mask can be seen in Figure 7 and from here these types of masks will be called ‘Single Block’ or ‘SB’ masks.

Figure 7: Example of a ‘Single Block’ Mask

The aim of the ‘Single Block’ mask type was to isolate the importance of separate assets, the block size was simply set as the size of the assets used in the game. However, the block size still needs to be decided upon for the ‘Literature Based’ mask type. In the literature [82], the block number was set to 7 for an image size of 224, hence it was decided to choose a block size for each game individually in order to maintain a similar ratio between image size and number of blocks.

The final parameter that needs defining was the number of masks to be used. In the system diagram in Figure 5 a total of 2000 masks were used and this appeared to be an insufficient number to obtain high quality importance maps. Due to this it was decided to calculate an importance map for different mask sets and then create a final importance map by averaging them. This was appropriate as the resulting map was then the same as if the system was run with a larger number of masks within a set. To investigate how many mask sets were required, an importance map was calculated for a single frame with an increasing number of mask sets, this can be seen in Figure 8.

It can be seen that for a low number of mask sets, random hotspots are evident. However, as the number of mask sets was increased, these hotspots became less and less prevalent. It should be noted that as the mask number increases, the required compute time also increases and hence it was decided to use 20 sets of 2000 masks to balance the quality of the importance map to the compute time necessary to calculate them.



(a) 2 Mask Sets (b) 5 Mask Sets (c) 10 Mask Sets (d) 20 Mask Sets

(e) 50 Mask Sets (f) 100 Mask Sets (g) 150 Mask Sets (h) 200 Mask Sets

Figure 8: Importance Maps for Different Numbers of Mask Sets

4.2.2 Explaining the Differences between Gameplays

The next level of analysis aimed to explore how any given pair of gameplays differed. This has been achieved using a minor adjustment to the RISER system. Instead of simply blanking out the masked region of the input, this was mixed with a second input image. The masks were generated using the same methods as in the RISER case. The frames were paired based on the matching found by Dynamic Time Warping on the representation sequences. In this altered system Equation 11 shows how the masks were combined to generate the importance map for the difference between the two input images, namely I_1 and I_2. The resulting altered system, called Randomized Input Sampling for Explanation of Representation Differences (RISERD), is shown in Figure 9.

$$S_{I_1,I_2,enc}(\lambda) \overset{\text{MC}}{\approx} \frac{1}{\mathbb{E}[M] \times N} \sum_{i=1}^{N} Cos\_Sim(enc((I_1 \odot M_i) + (I_2 \odot (1 - M_i))), enc(I_1)) \times M_i(\lambda) \quad (11)$$

The output from this system highlighted parts of the image that differed between the two frames but were also important to the representation. It is suggested that if this was performed on gameplays from different playstyles, the output would highlight the differences between these playstyles. On the other hand, if it was performed on gameplays from the same playstyle, it would highlight 'noise' elements within the playstyle. These noise elements were aspects of the gameplays that differ but were not caused by a change in playstyle.

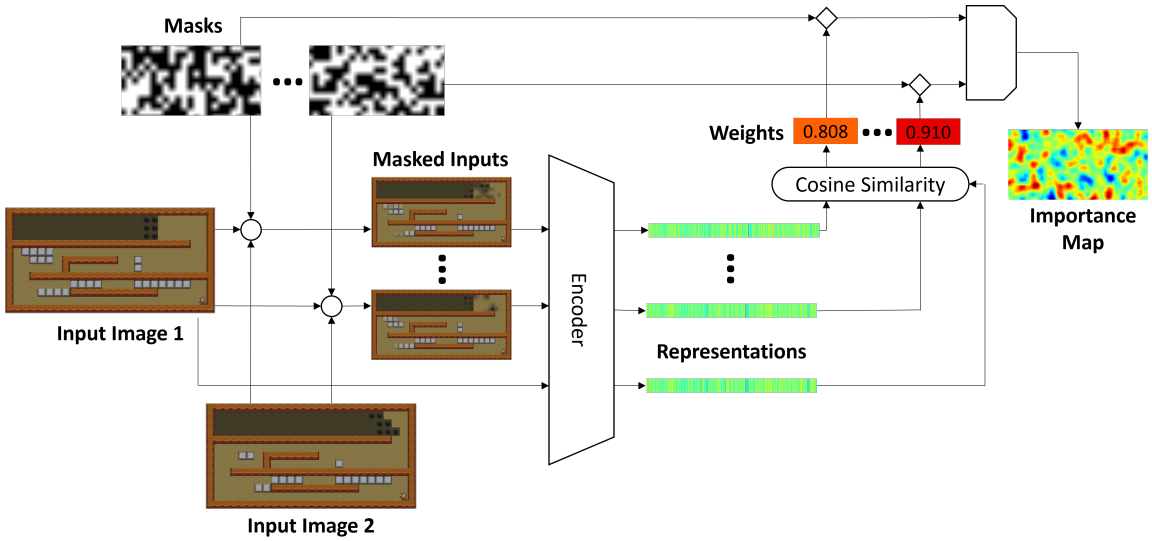


Figure 9: RISERD Schematic

4.2.3 Gameplay Projections

The final level of analysis looked at the dataset as a whole. Firstly, a distance matrix was calculated by using DTW [10] together with the cosine distance metric to calculate the distance between two gameplay sequences. Uniform Manifold Approximation and Projection (UMAP) [70] was then used to project each gameplay onto a 2D plane based on the calculated distance matrix, while attempting to maintain the structure of the dataset. After the gameplays were projected onto a 2D plane the relative position to other gameplays could then be investigated to gain an understanding of how the gameplays related to one another.

4.3 Representation Evaluation

Alongside examining the meaning of the generated representation, the quality of the representation space was also evaluated. This evaluation took two forms, both utilising the fact that ground truth labels were available.

4.3.1 Group Average Linkage Between Playstyles

The first evaluation process examined the group average linkage [76] between playstyles. This allowed the generation of a matrix showing the group average linkage between each playstyle pair. This can be used to ensure that not only are videos of the same playstyle embedded close together, but also check that they are far away from videos of other playstyles. These two properties would be expected of good quality representations. This hence should result in a low group average linkage when comparing the playstyle against itself, particularly when comparing it to the group average linkage to other playstyles. Further, a low group average linkage between different playstyles would suggest possible similarity, and if these can be mapped to known factors in the playstyle definition, this can further show the quality of the representation space.

4.3.2 Silhouette Score

The second evaluation method summarises the quality of the representation by taking the average silhouette score for each sample, where the sample silhouette score [87] combines the intra-cluster distance and the average nearest cluster distance. This in essence evaluates how compact the given cluster is and how close it is to other clusters. This metric is traditionally used to evaluate the quality of the predicted cluster labels. However, in this case the cluster labels were given as the ground truth labels. This hence meant that in order to get a good silhouette score, the encoder must encode videos of the same playstyle closer together whilst ensuring other playstyles are encoded further away. This metric returns a value between 1 and -1, where a higher value indicates better quality clusters or in this case a better learnt representation space.

4.4 Case Studies

In the following case studies, a representation space will be learnt from gameplay videos and analysis conducted with the intention of exploring the meaning of the representations. Each of these case studies will first introduce the game used in the study alongside how the gameplay dataset was generated. A range of games will be explored with varying properties (*e.g.* 2D/3D, grid-based, partially/fully observed). After which the training of the encoders will be discussed, including loss graphs and the time requirements for training. Finally, the representation space will be explored, using the previously introduced methods, *i.e.* RISER, RISERD and UMAP projections, to explain the meaning of the representations. Within this section a number of examples will be provided for each of this analysis techniques, however further examples can be found in Appendix A. This will not only be performed on the introduced hybrid encoder (STDIM-VAE), but also on each of the individual encoder designs that are utilised within this hybrid. The aim of this was to investigate the information each individual encoder type encodes and if the hybrid version had learnt a mixture of these.

4.4.1 Black Smoke

The first case study focused on the game ‘Black Smoke’, which is part of the GVGAI Framework [100]. This game is a 2D, grid-based, fully observed game, in which the player’s overall goal is to reach the exit of the level. In order to reach this exit, additional game mechanics may need to be utilised, for example this can include the breaking of blocks or finding a key to unlock a door. Although, a time constraint is also introduced through the use of harmful ‘smoke’ which is stochastically propagated from source blocks, forcing the player to progress through the game level. A level example is shown in Figure 10.

In the furtherance of providing validation on the success of the clustering, data was generated with known Ground Truth (GT) labels. Within ‘Black Smoke’, this was achieved by designing agents to play with a given playstyle. To achieve this, a set of playstyles had to be defined, this was done based on the literature discussed in Section 2.1. The first playstyle simply aimed to get to the level exit as quickly as possible, this

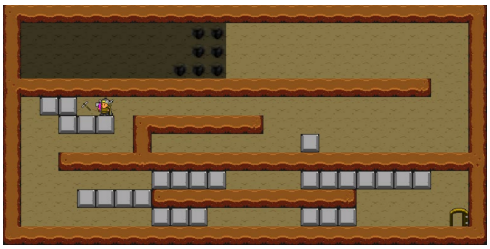


Figure 10: Frame Example from BlackSmoke

was based on the previously defined ‘Runner’ persona. The second playstyle aimed to break blocks while on route to the level exit, this was based on the ‘Collector’ persona.

The next step was to convert these definitions into agent policies that could play this game to generate the necessary gameplay data. It was decided to use a two-step look ahead agent to achieve this. This agent used the value function defined in Equation 12 to decide upon its next action. The variable B counts the numbers of blocks broken and E is the distance to the exit.

$$V = cB - E \tag{12}$$

The c parameter was used to set the priority between breaking blocks against navigating to the level exit. Given that one step towards the exit would increase the value function by one, a c_1 value of less than one focused on navigating to the exit, whereas a value of greater than one would result in prioritising the breaking of blocks before proceeding to the exit. A value of 10 therefore was chosen to give a high priority to block breaking for the ‘Collector’ based playstyle. On the other hand, a value of 0.01 was chosen for the playstyle based on the ‘Runner’ persona. A non-zero value was required to ensure that the agent would break blocks if they prevented progression towards the exit. It was noted that the designed agents would act deterministically and hence would be easier to cluster. To add further difficulty to the clustering process, Random Action Noise (RAN) was introduced at varying levels. This led to the creation of datasets with the following RAN levels: 0%, 1%, 5%, 10% and 20%. The RAN level was used to control the probability that the agent would take a random action instead of choosing the next action based on its policy. In addition, a final dataset was constructed which combined the data collected at all the above mentioned RAN levels. This aimed to construct a dataset of gameplays from players with different levels of error when selecting the correct action under the chosen playstyle.

In order to train the encoder on this game, a number of additional parameters needed to be selected. These are listed in Table 1. Further, the time requirement for each encoder type is provided in Table 2. It should be noted that as the hybrid encoder had to calculate the losses from both the VAE and ST-DIM encoders, this was expected to take longer. This should mean that the combined time for both the individual encoders was equal to that of the hybrid encoder, which based on these values appears to be true. However, it can be seen that this was only the case if the same number of epochs were undertaken. Although this was not always the case, as an early stopping mechanism was used to allow early termination if the validation loss stopped decreasing over a significant number of sequential epochs.

Table 1: Gameplay Recording Settings for BlackSmoke

Setting	Value
Image Width	780
Image Height	390
Down Sampling	None
Frame Limit	None

Table 2: Comparison of Average Encoder Training Times for BlackSmoke

Encoder	Training Time (m) $\pm$ SD
VAE	37.31 $\pm$ 00.10
ST-DIM	172.83 $\pm$ 35.79
STDIM-VAE	216.55 $\pm$ 26.24

Next, an investigation of the loss terms was conducted. This involved looking at both the total hybrid loss as well as the individual components. Firstly, the total hybrid loss was studied. This can be seen in Figure 11 with each line representing a different training run. These results looked promising as there appeared to be reliable convergence to a common value of approximately 0.377.

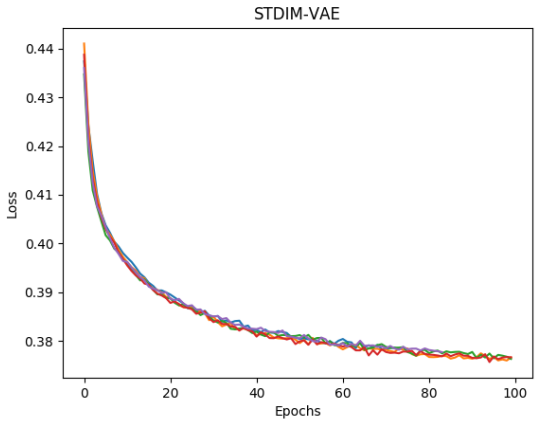
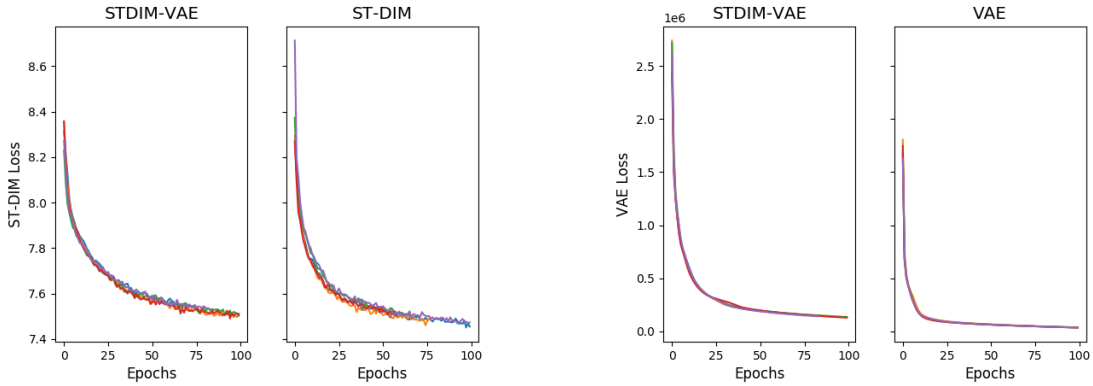


Figure 11: STDIM-VAE Training Loss for BlackSmoke

Following this, consideration was given to each of the individual loss components. Although each of these components were weighted before being combined to form a total hybrid loss value, the unweighted values were investigated to allow a comparison between these values and the loss value of the encoder that purely optimised for that loss term. These can be seen in Figures 12a and 12b.

The hybrid encoder was able to optimise each of the loss components reasonably well as shown by the fact that they converged to similar values. Although this convergence point was slightly lower when the encoder purely focused on each component. However, this was to be expected given that the hybrid encoder had to balance the



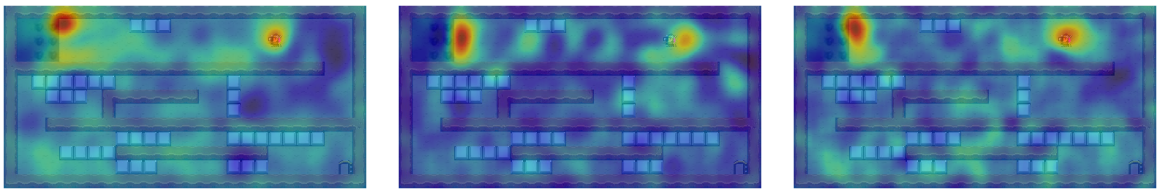
(a) ST-DIM Loss Component

(b) VAE Loss Component

Figure 12: Comparison of Loss Components for the Hybrid and Individual Encoders for BlackSmoke

optimisation of both components.

After all the encoders were trained, the next step was to apply the previously introduced interpretation techniques to the learnt models to gain an understanding about what had been encoded by each of the encoder types. Given that the playstyles used for this game differed based on the priority with which the player breaks blocks, it was hoped that the focus would be on both the player's location and the number of blocks that had been broken. If this were the case, then this would suggest that the encoder had successfully learnt the relevant information about each of the playstyles. Firstly, analysis was applied to each encoder type to investigate the differences in what had been learnt. The importance maps for ST-DIM and VAE can be seen in Figures 13a and 13b respectively using 'Literature Based' masks on the initial frame of a 'Runner' gameplay.



(a) ST-DIM

(b) VAE

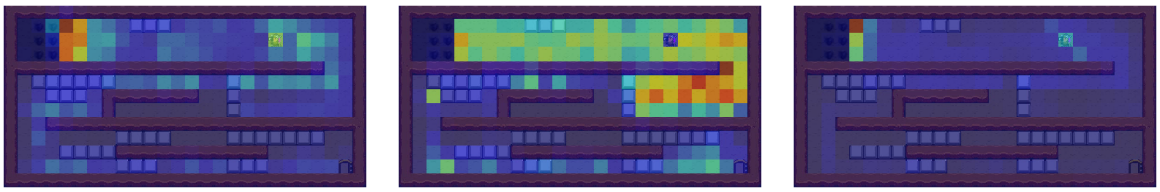
(c) STDIM-VAE

Figure 13: 'Runner' No RAN Run 1, Frame 0 Importance Maps using 'LB' Masks

It can be seen from the above that the ST-DIM encoder appeared to focus on the top of the smoke cloud edge as well as having a secondary focus point on the current player location. On the other hand, the VAE appeared to focus on the whole smoke cloud edge with a secondary focus point being slightly in front of the player's current location. This showed that each of these encoder types had learnt different aspects from this frame. As the developed encoder method, STDIM-VAE, aimed to combine both techniques, the regions of focus should be a mix of the ST-DIM and VAE importance maps. The importance map for the STDIM-VAE for the same frame in question is shown

in Figure 13c. In this case, although there was more focus on the top of the smoke front edge, it does also incorporate the rest of the smoke front. Additionally, the player focus region, contained both the player and the region in front of the player. Both of these observations provide evidence that the hybrid encoder method does indeed learn a combination of the features learnt by each individual encoder method.

A similar investigation was conducted for the 'Single Block' mask type and the resulting importance map for each encoder type can be seen in Figure 14.



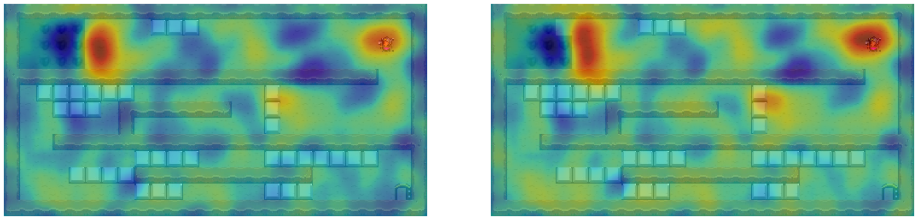
(a) ST-DIM (b) VAE (c) STDIM-VAE

Figure 14: 'Runner' No RAN Run 1, Frame 0 Importance Maps using 'SB' Masks

In this case the ST-DIM importance map again focused on both the smoke front and the player location. However, in this case there appeared to be interest in both the path the player took and the location the smoke could occupy later in the episode. A similar importance map was seen for the VAE but with a stronger focus on the possible position of the smoke cloud and less focus on the player's current location. Given that both of these importance maps were fairly similar, the importance map for the STDIM-VAE was, as expected, similar to both of these maps but appeared to be more similar to the ST-DIM importance map.

After comparing the importance maps for each encoder, the next step was to investigate the importance map for the whole gameplay sequence. At this point it was decided to focus on just the STDIM-VAE on two runs of each playstyle.

Firstly, focus was given to the 'Literature Based' mask type. When investigated, it was found that often at the beginning of the gameplay sequence, the encoder focused on a combination of both the smoke cloud edge location and the player's location. This can be seen in Figure 15, which shows the importance maps for a run of each playstyle at frame 5.



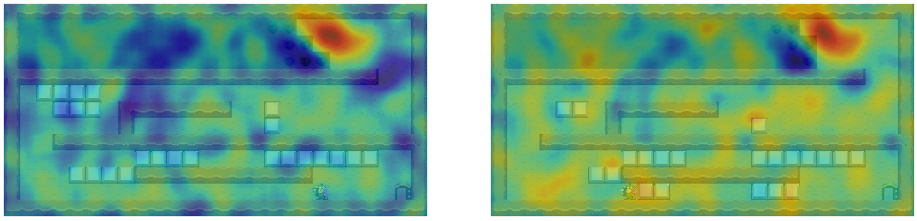
(a) 'Runner' with 5% RAN Run 1 (b) 'Collector' with 5% RAN Run 1

Figure 15: Frame 5 Importance Maps using 'LB' Masks

This suggested that the encoder had successfully encoded the relevant playstyle information, as a major difference in the playstyles was the number of blocks broken.

This linked back to the route the player had taken, and that information appeared to have been successfully encoded.

When analysis was conducted further into the gameplay, the encoder appeared to switch to focusing more heavily on the smoke cloud edge location. This is shown in Figure 16, which shows the importance map for a run of each playstyle at frame 75.

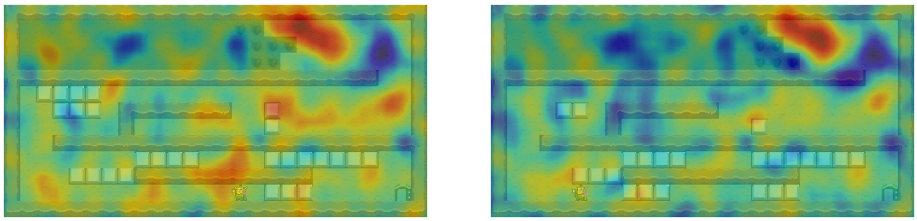


(a) 'Runner' with 5% RAN Run 1 (b) 'Collector' with 5% RAN Run 1

Figure 16: Frame 75 Importance Maps using 'LB' Masks

In a way, this was essentially encoding a timestamp on the gameplay, as the distance the smoke cloud had propagated could be linked back to roughly how long the player had been playing the game. Interestingly the two playstyles considered in this game vary based on the time to complete the level, due to extra time being required to break blocks, this meant that in theory, based purely on where the smoke cloud was the playstyle could be determined.

However, between these two points in the gameplay, namely frames 5 and 75, the importance maps commonly started to show what appeared to be random hotspots, where part of the frames have a fairly high focus region with no clear reason. This is shown in Figure 17, which shows the importance map for a run of each playstyle at frame 65.

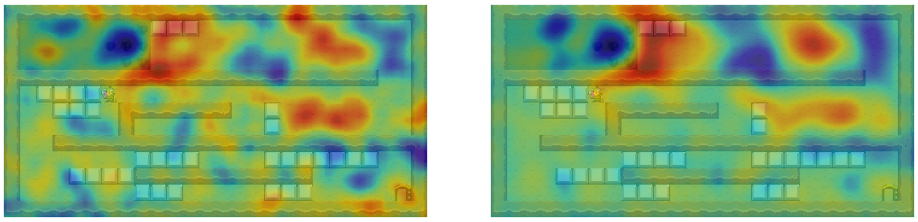


(a) 'Runner' with 5% RAN Run 1 (b) 'Collector' with 5% RAN Run 1

Figure 17: Frame 65 Importance Maps using 'LB' Masks

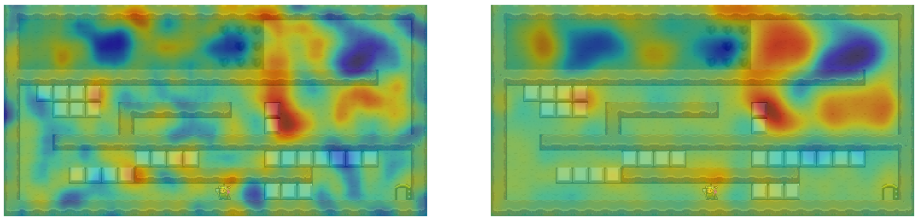
This led to the question, as to whether 20 mask sets were indeed a truly good balance, of compute time against quality for later frames in the gameplay. To investigate this, a higher number of mask sets were experimented with on frames taken from the 5 and 75 frame range. An example of this can be seen for frames 30 and 60 with 20 and 200 mask sets in Figures 18 and 19.

Based on these two examples, it has been shown that overall, the most intensely highlighted areas were maintained even with a higher number of mask sets. Additionally, these apparently random hotspot areas were still present with 200 mask sets, all be it



(a) 20 Mask Sets (b) 200 Mask Sets

Figure 18: ‘Runner’ No RAN Run 1, Frame 30 Importance Maps using ‘LB’ Masks



(a) 20 Mask Sets (b) 200 Mask Sets

Figure 19: ‘Runner’ No RAN Run 1, Frame 60 Importance Maps using ‘LB’ Masks

with slightly less intensity. From these investigations, it was decided that the use of 20 mask sets was still appropriate.

The next step was to average the importance maps calculated across the whole game playstyle sequence. This aimed to generate a ‘fingerprint’ of the focus of the encoder for each gameplay. These can be seen in Figure 20.

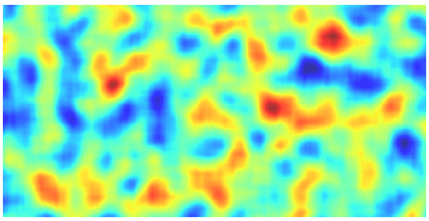
Although, it was hard to accurately interpret the meaning behind the areas of focus across the whole gameplay, there was a clear difference when looking at the ‘fingerprints’ of different playstyles. This suggested that the encoder was potentially learning information, which was specific to each playstyle and hence causing the areas of focus to change depending on the playstyle being considered.

Following this, analysis was conducted using the ‘Single Block’ mask type, which focused on the importance of individual assets to the representation. When considering the first frame of each gameplay, it was shown that the main focus was still on the player’s location and the location of the smoke cloud edge. This is shown in Figure 21.

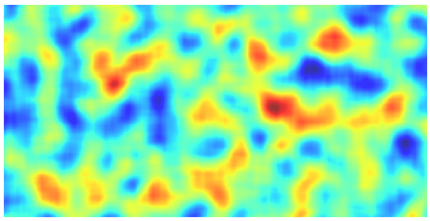
However, it was seen that in addition there was a small focus on the path that the player would take through the level. This becomes even more prominent when you consider the final frame in each gameplay, this is shown in Figure 22.

This further suggests that the representation contained information about how the player moved through the level and in some cases the blocks that had been removed, as these differed between the playstyles. Although, the intensity of this interest varies from block to block, with the blocks that make up the first wall that needed to be removed being the most intense. When averaged over the whole gameplay, a similar pattern was found and is shown in Figure 23.

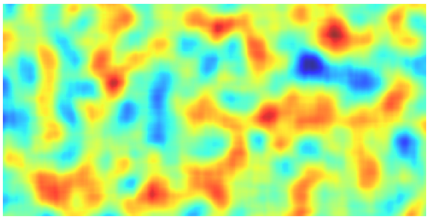
Next, analysis was conducted to extract an understanding of the differences between gameplay pairs, using the introduced RISERD method. The first pair that was explored, compared two demonstrations of the ‘Runner’ playstyle with no RAN. Given



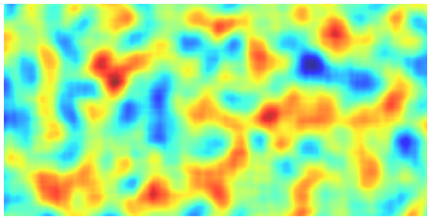
(a) ‘Runner’ with 5% RAN Run 1



(b) ‘Runner’ with 5% RAN Run 2

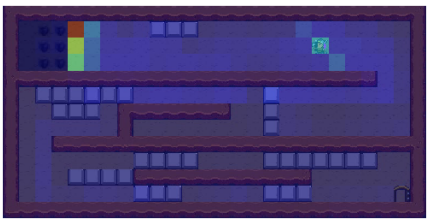


(c) ‘Collector’ with 5% RAN Run 1

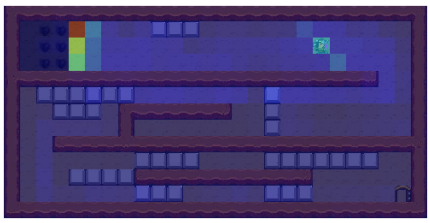


(d) ‘Collector’ with 5% RAN Run 2

Figure 20: Focus ‘Fingerprints’ using ‘LB’ Masks



(a) ‘Runner’ with 5% RAN Run 1



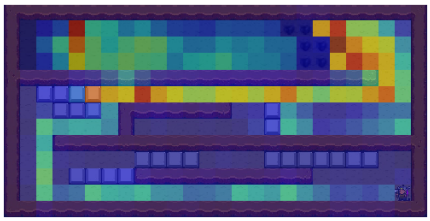
(b) ‘Collector’ with 5% RAN Run 1

Figure 21: Frame 0 Importance Maps using ‘SB’ Masks

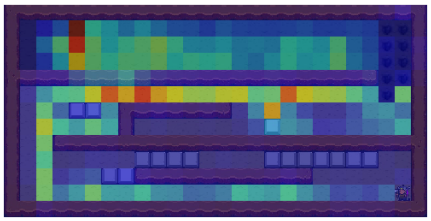
that the comparison was done between gameplays of the same playstyle, any extracted differences identified cannot be down to the playstyle itself. Early in the gameplay, as shown in Figure 24, the main focus region was seen to be on the front of the smoke edge. Further, when examining the matched frame, the two areas of focus related to the two different locations to which smoke had propagated in the paired frames. In these two gameplays there was no RAN, therefore the path the player took was the same and hence the only difference was how the smoke had propagated, due to its stochastic nature.

A later frame in the gameplay was also investigated and the results are shown in Figure 25. In this case, the focus region expanded to where the smoke had been, the reason for this can be determined by inspecting the matched frame. It can be seen, that while the smoke front was located in the same place in the frame pair, there were three squares that differed based on the existence of a smoke ‘swirl’ and these were indeed the three squares that received focus. The combination of both of these results showed that the main difference in the gameplays was how the smoke had moved through the level. This allowed the identification of a difference in the gameplays which was not caused by the playstyle.

Following this investigation, a comparison was conducted on two ‘Collector’ game-

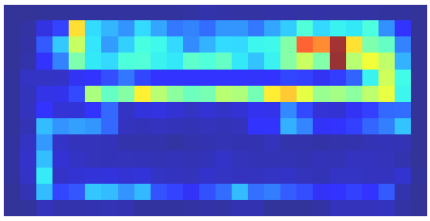


(a) ‘Runner’ with 5% RAN Run 1

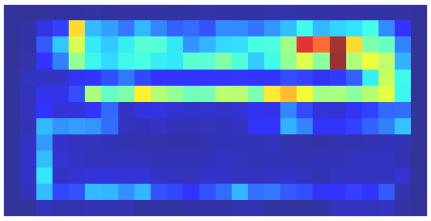


(b) ‘Collector’ with 5% RAN Run 1

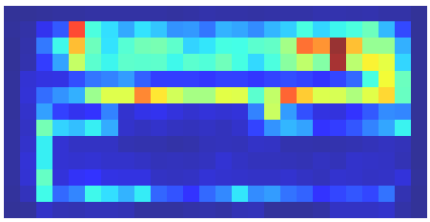
Figure 22: Last Frame Importance Maps using ‘SB’ Masks



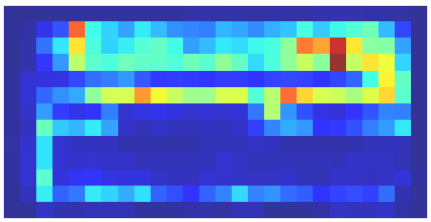
(a) ‘Runner’ with 5% RAN Run 1



(b) ‘Runner’ with 5% RAN Run 2



(c) ‘Collector’ with 5% RAN Run 1



(d) ‘Collector’ with 5% RAN Run 2

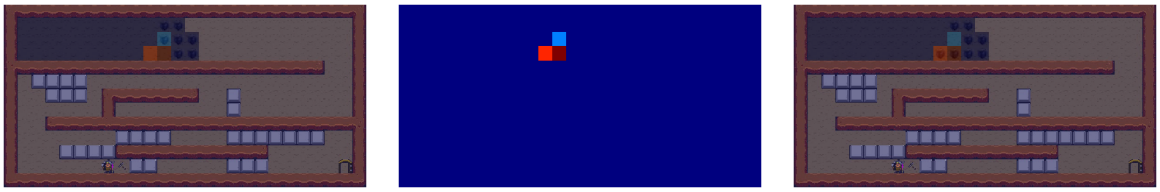
Figure 23: Focus ‘Fingerprints’ using ‘SB’ Masks

plays, although this time each of these had 20% RAN. A frame early in the gameplays was initially considered and the importance map is shown in Figure 26. At this point the main focus appeared to be in front of the smoke, however a small focus difference can be seen directly in front of the player compared to the rest of the frame. To explore this further the matched frames were examined, in this case the frame being considered had been matched to two different frames (*i.e.* frame 3 and frame 4). Based on the matched frames, it can be seen that in run 2 an unnecessary axe hit had been performed due to RAN. This action delayed the player’s progress and hence, presumably based on a combination of the same player position and similar smoke location, resulted in an additional frame (*i.e.* frame 4) to be matched with the frame being considered from run 1. The additional matched frame can be used to explain each of the focus regions. As this was one timestep later the smoke had started to propagate forward, which led to the focus in front of the smoke. Secondly, the axe can be seen in front of the player showing the axe hit and hence is different to what was normally seen. Although, this is only a small visual difference, it was likely that this was why only a small focus difference could be seen in front of the player.

A frame 4 timesteps later was then explored and the importance map is shown in Figure 27. Here it can be seen that focus is on both the smoke front as well as on and

(a) Frame 5 (b) Importance Map (c) Frame 5 (Match 1)

Figure 24: Comparison of Runs of ‘Runner’ No RAN at Frame 5 using ‘SB’ Masks



(a) Frame 50 (b) Importance Map (c) Frame 50 (Match 1)

Figure 25: Comparison of Runs of ‘Runner’ No RAN at Frame 50 using ‘SB’ Masks

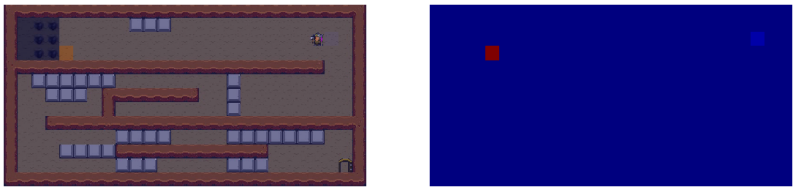
above the player. It was expected that due to the delay in the player’s progress, caused by the unnecessary axe hit, that the focus above the player related to where the player was in the matched frame and this was confirmed. Further, it also shows that the focus on the smoke front is either related to differences in smoke position or the existence of a smoke ‘swirl’.

Further, even later in the gameplay, it was noted that RAN caused one gameplay to fail to break a block which had normally been broken in the ‘Collector’ playstyle. An importance map was examined after this point, and this is shown in Figure 28 alongside the matched frame. Within the importance map, the main focus can be seen on the block that was not broken but focus was also seen on the differences in the smoke. The comparison of these gameplays at different time points suggested that the RISERD method was not only able to identify both differences caused by the stochastic element of the environment but also differences in the player’s actions due to RAN.

In summary, when comparing gameplays of the same playstyle, the main differences have been identified. Both the differences in how the smoke propagated through the level and also errors in the playstyle’s application caused by the introduction of RAN.

Following this, comparisons were made between gameplays that showed different playstyles. The first comparison pair was conducted on gameplays that did not contain any RAN. Early in the gameplay, both playstyles were seen to act in the same way. When looking at an importance map from this initial time period, shown in Figure 29, the focus areas were seen to be on the smoke. Based on the matched frame, this was indeed the only difference at this point.

However, when considering later stages of the gameplay, the playstyles started to deviate. This took the form of the ‘Collector’ playstyle stopping to break a block, whereas the ‘Runner’ playstyle continued towards the exit. When examining an importance map following this deviation, a focus region was seen on the smoke but also on the block that the ‘Collector’ had stopped to break. This importance map is shown in Figure 30.



(a) Frame 3 (b) Importance Map

(c) Frame 3 (Match 1) (d) Frame 4 (Match 2)

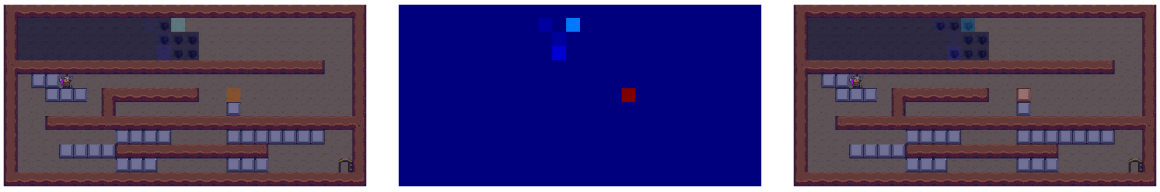
Figure 26: Comparison of Runs of ‘Collector’ 20% RAN at Frame 3 using ‘SB’ Masks

(a) Frame 7 (b) Importance Map (c) Frame 8 (Match 1)

Figure 27: Comparison of Runs of ‘Collector’ 20% RAN at Frame 7 using ‘SB’ Masks

Further, focus was also seen on and behind the player, based on the matched frame, this was due to a difference in player position caused by the ‘Collector’ pausing to break the block. Additionally, as was normally seen, focus was also seen on differences in the smoke position between the frame pair.

The final pair of gameplays examined also used different playstyles but in addition have 20% RAN. Again, a frame early in the gameplay was examined and the importance map is shown in Figure 31. While the optimal path for each playstyle should have been the same up to this point, it can be seen that there was a focus region on the player’s path as well as on the smoke. On examination, the RAN had caused a deviation from the optimal path and hence caused differences in the player’s location, this can be seen based on the matched frame. This difference caused the focus to be on and below the

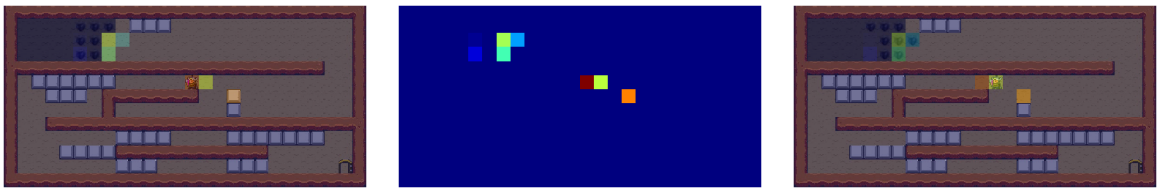


(a) Frame 52 (b) Importance Map (c) Frame 52 (Match 1)

Figure 28: Comparison of Runs of ‘Collector’ 20% RAN at Frame 50 using ‘SB’ Masks

(a) Frame 5 (b) Importance Map (c) Matched Frame (Frame 5)

Figure 29: Comparison of ‘Runner’ and ‘Collector’ Runs each with No RAN at Frame 5 using ‘SB’ Masks



(a) Frame 20 (b) Importance Map (c) Frame 22 (Match 1)

Figure 30: Comparison of ‘Runner’ and ‘Collector’ Runs each with No RAN at Frame 20 using ‘SB’ Masks

player within the importance map.

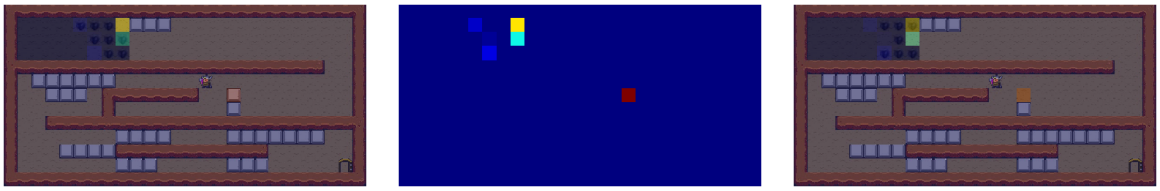
(a) Frame 12 (b) Importance Map (c) Frame 10 (Match 1)

Figure 31: Comparison of ‘Runner’ and ‘Collector’ Runs each with 20% RAN at Frame 12 using ‘SB’ Masks

Next, a frame after the optimal playstyle path deviation was examined. This importance map is shown in Figure 32 alongside the matched frame. As can be seen, in the match frame pair, the player was in the same position. This resulted in two focus regions. Firstly, a focus on the difference in the smoke and secondly a focus on the block which the ‘Collector’ had broken but the ‘Runner’ had not.

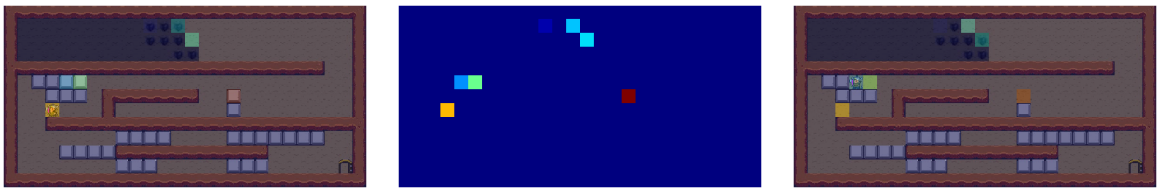
Even later in the gameplay, further playstyle deviation was seen. In this case, the ‘Collector’ mined through the blocks ahead, while the ‘Runner’ breaks the minimum number of blocks needed to run around the block set. A frame during this stage of the gameplays was investigated and the importance map is shown in Figure 33 alongside the matched frame. From these, it can be seen that focus was on both the difference in player location and the additional blocks broken, including the block broken in the previous playstyle deviation. In addition, the difference in the smoke between the frame pair was seen to be a region of focus.

Overall, this analysis has been able to highlight the main regions that differ across the



(a) Frame 25 (b) Importance Map (c) Frame 27 (Match 1)

Figure 32: Comparison of ‘Runner’ and ‘Collector’ Runs each with 20% RAN at Frame 25 using ‘SB’ Masks



(a) Frame 44 (b) Importance Map (c) Frame 49 (Match 1)

Figure 33: Comparison of ‘Runner’ and ‘Collector’ Runs each with 20% RAN at Frame 44 using ‘SB’ Masks

examined gameplays. By first comparing examples of the same playstyle, differences that do not relate to the playstyle have been identified (*e.g.* how the smoke propagates). After which, comparisons were made across the two playstyles, from this several gameplay differences were identified as possible differences in these playstyles. It was noted that if you first discounted the differences, that were seen within playstyles, then the main focus was on certain blocks. The blocks in which focus fell were examined and were often the blocks which the ‘Collector’ playstyle had broken but the ‘Runner’ playstyle had not. This meant that the analysis was able to identify the main visual differences between the two playstyles, although it is recognised that the meaning of this was only known due to an understanding of the playstyles, which is not always available.

Finally, the encoded gameplays were projected onto a 2D plane to explore how each gameplay compared to one another. Given five encoders were trained, the representations for each of these has been projected and one example is shown in Figure 34. The full projection set is shown in Appendix A.1.2.

As can be seen, two clear groups were always seen within each projection space. Interestingly, in some cases, as the RAN level was increased the points that represented a gameplay moved closer together but still remained in separate groups. However, when the two groups were examined further, in many cases a single gameplay was placed in the wrong group. This rogue gameplay was further studied, and it was found that the RAN had resulted in this ‘Runner’ gameplay breaking additional blocks that were not normally broken. Some of these additional blocks were normally broken by the ‘Collector’ playstyle and hence this was likely the cause of this gameplay appearing to be more similar, within the projection space, to the ‘Collector’ gameplays.

Finally, the two previously introduced representation evaluation techniques were

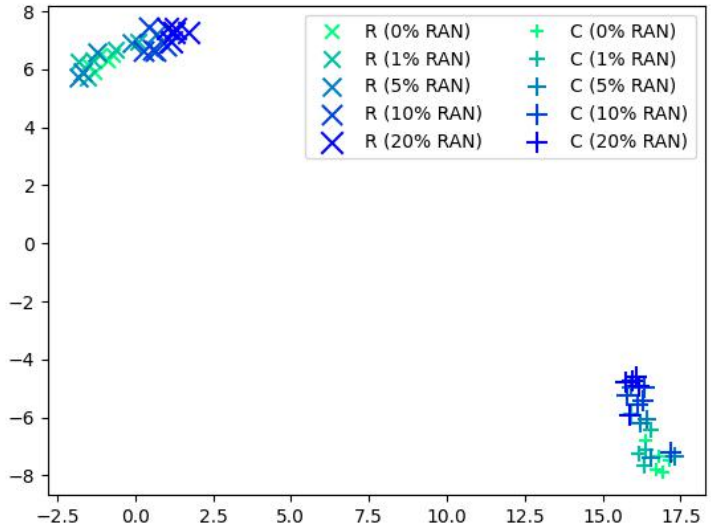


Figure 34: Projection 1 of Blacksmoke Gameplays using the STDIM-VAE

applied to quantify the quality of the representation space. Firstly, the matrix of group average linkage between playstyles was generated and is shown in Figure 35.

As can be seen when comparing the same playstyle and RAN level against itself, the group average linkage was smaller than when different playstyles with the same RAN level were compared. However, it can also be seen that this difference was reduced as the RAN level was increased. This shows that the playstyles were being encoded to different points in the representation space but the distance between them decreases as the RAN level is increased, which reflects what was seen within the projection space. Secondly, when comparing the average silhouette score for all combinations of encoders and metrics, the combination of STDIM-VAE and cosine metric, scores the highest and hence generates the best representation space. This is shown in Table 3.

Table 3: Average Silhouette Scores for BlackSmoke ($\pm$ SD)

		Distance Metric		
		Manhattan	Euclidean	Cosine
Encoder	VAE	0.162 (0.004)	0.181 (0.003)	0.286 (0.004)
	ST-DIM	0.284 (0.011)	0.288 (0.012)	0.482 (0.015)
	STDIM-VAE	0.304 (0.025)	0.304 (0.024)	0.495 (0.036)

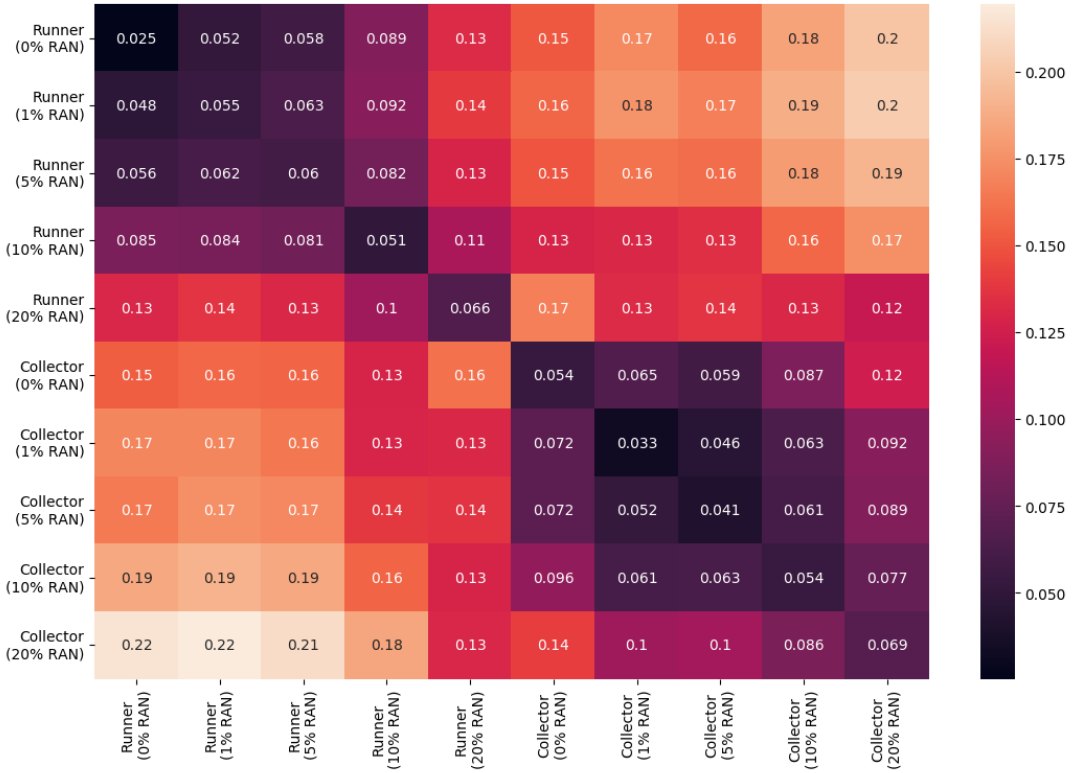


Figure 35: Group Average Linkage Between Playstyles for BlackSmoke

4.4.2 VizDoom

This second case study was conducted using the game 'VizDoom' [61]. This was chosen as it introduced additional challenges not seen in the previous case study, as the game has a three dimensional partially observed environment. For this work, focus was given to the 'Deathmatch' game mode and screenshots of the gameplay can be seen in Figure 36.

(a) (b)

Figure 36: Frame Examples from VizDoom

Similarly, to the previous case study, it was felt desirable to have access to Ground Truth labels for evaluation of the quality of the clusters found. Due to the added complexity within this game, to ensure that the gameplay was human-like, the decision was made to manually collect the 'VizDoom' dataset. When choosing the playstyles

from which to collect data, it was key to ensure that the playstyles were likely to be found within the given game. To ensure that this was the case, consideration was given to the literature around clustering on human data of games of the same genre (*i.e.* 'first-person shooter'). One such example was the analysis shown in the literature on the 'Destiny' game [27]. Within this analysis, two main features were used to describe the found playstyles. The first of which was 'Weapon Choice', as most playstyles preferred either one weapon or a set of weapons. The second feature was 'Player Location', which related not only to where the player was within the game map but also their relative distance to other players within the game. Interestingly each of these features can be linked back to overarching features, that being 'Navigation' and 'Interaction' previously discussed in the literature [101]. For each of these features, a number of possible options were defined and then playstyles were generated by combining these different options. How these were combined can be seen in Figure 37. Additionally, subsets of the whole dataset were shown to allow the evaluation of certain features. This was achieved by having the playstyles within subsets only differ by one of the two previously mentioned features.

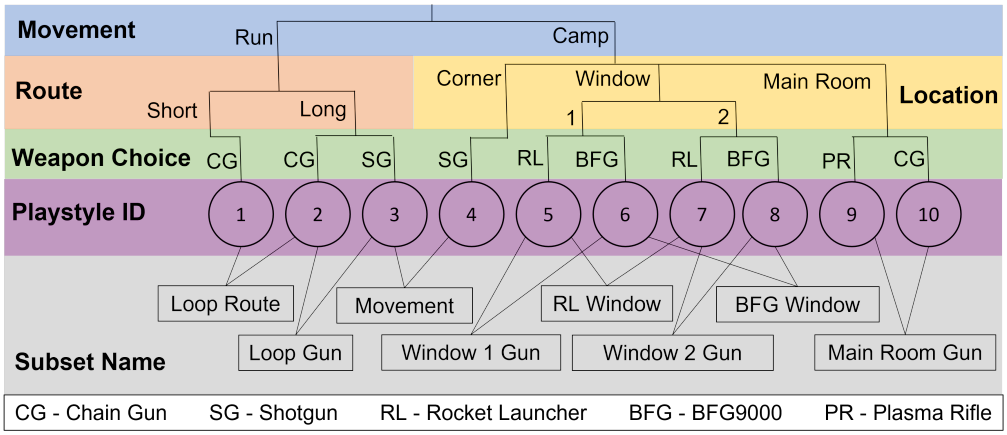


Figure 37: VizDoom Playstyles

Figure 38 shows the game map, with each of the locations and routes used within the defined playstyles marked. Following the definition of these playstyles, the game was played under each of these playstyles to collect corresponding data.

Similarly to the previous case study, a number of additional parameters were also selected for 'VizDoom' and are shown in Table 4. The main difference between these two case studies was the introduction and use of both 'Down Sampling' and 'Frame Limit' parameters. 'Down Sampling' related to the frequency in which frames were taken from the video. This was required as the players' actions took longer to materialise on the screen and hence only every fifth frame was needed. The second parameter 'Frame Limit', limits how many frames were taken from the whole gameplay video. As these gameplay videos contained a large number of frames, it was decided to split off a small initial gameplay sequence to allow faster analysis. This allowed focus on a short time period of up to 400 frames and hence focused on the playstyle instead of the player persona.

Further, the time requirement for each method is shown in Table 5. Interestingly, in this case the early stopping mechanism was activated for the hybrid encoder and

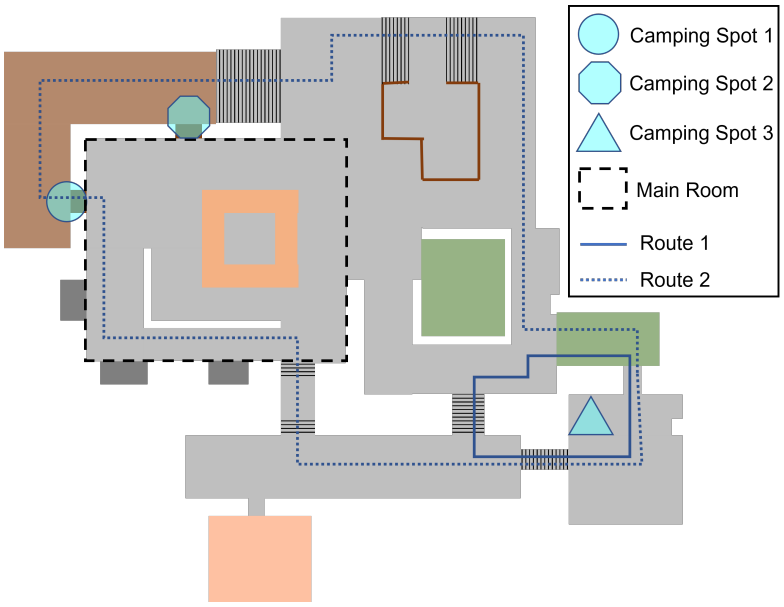


Figure 38: VizDoom Game Map

Table 4: Gameplay Recording Settings for VizDoom

Setting	Value
Image Width	640
Image Height	480
Down Sampling	5
Frame Limit	400

hence required a shorter training time compared to the other encoder methods. The early stopping mechanism was activated if improvement in the loss value was not seen for 15 consecutive epochs.

Table 5: Comparison of Average Encoder Training Times for VizDoom

Encoder	Training Time (m) $\pm$ SD
VAE	28.16 ± 3.98
ST-DIM	142.54 ± 22.59
STDIM-VAE	174.82 ± 28.88

Further, in this case study the loss components were also investigated. The final hybrid loss can be seen in Figure 39, which appeared to regularly converge to approximately 0.248. The loss components can be seen in Figures 40a and 40b.

Comparable conclusions can be drawn, namely that both loss components appeared to be optimised correctly, as per the previous case study. However, it was noted that the difference in the final loss value for the VAE component was larger in this case. It was suggested that this was the case because within this game environment the variation in the screen data was larger, making the correct generation harder, hence likely requiring

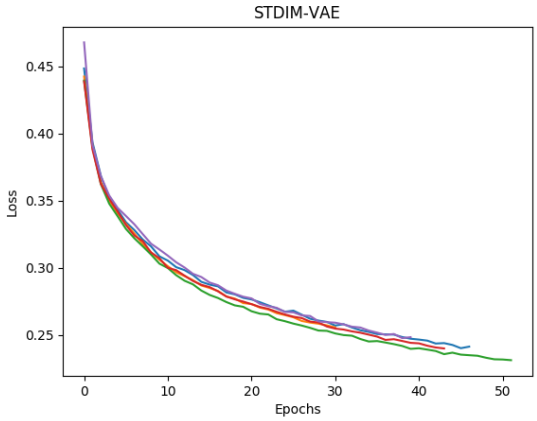
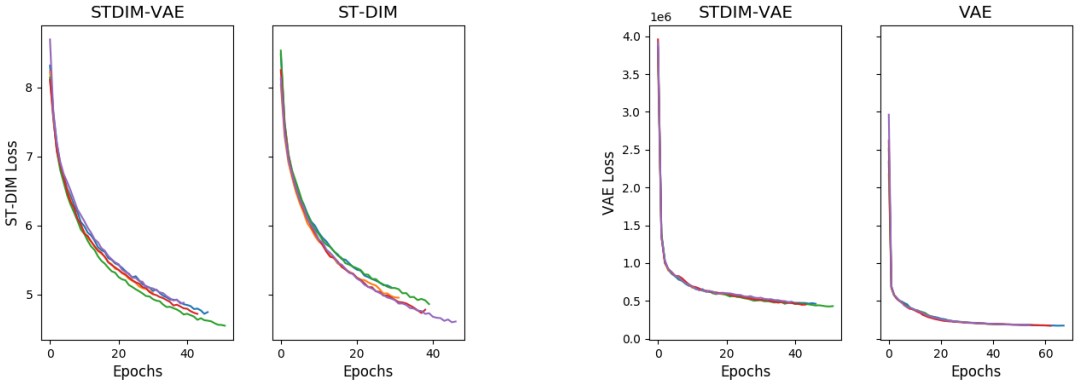


Figure 39: STDIM-VAE Training Loss for VizDoom



(a) ST-DIM Loss Component

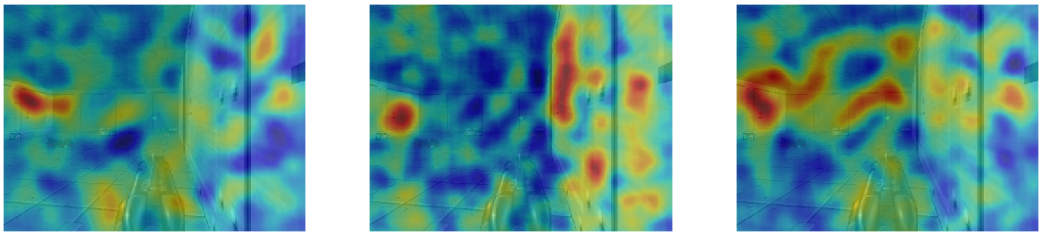
(b) VAE Loss Component

Figure 40: Comparison of Loss Components for the Hybrid and Individual Encoders for VizDoom

a greater amount of the representation space than before. This caused issues when balancing the optimisation of both loss components.

Again, when all the encoders were trained, an investigation was conducted into what each of the encoders had learnt. However, in this case, because the game frame was not constructed by combining different square assets, only 'Literature Based' masks were used in the analysis. When looking at the ST-DIM encoder, there seemed to be a main focus on the wall on the left. Whereas the VAE appeared to focus on both the left and right walls. This is shown in Figures 41a and 41b respectively. However, when exploring the STDIM-VAE, this encoder appeared to focus on a mixture of the areas that each of the individual encoder types focused on, this is shown in Figure 41c. This further supports the idea that the STDIM-VAE was balancing the loss components and hence learning aspects from each encoder component.

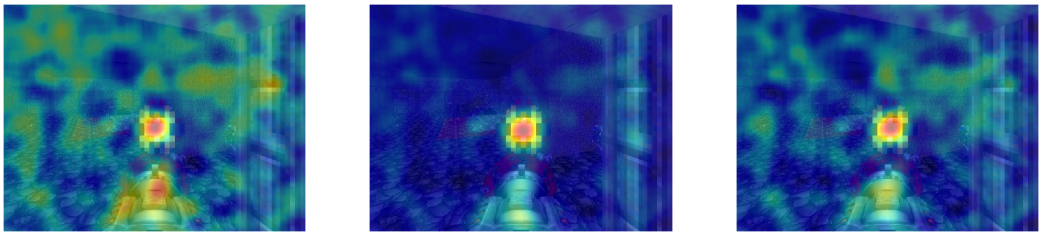
An investigation was also conducted on a later part of the gameplay. This studied a time when the player fired a gun and looked at two sequential frames to see what each encoder type had focused on. For the VAE it was seen that there was a very strong



(a) ST-DIM (b) VAE (c) STDIM-VAE

Figure 41: ‘Corner Camping’ with the ‘Shotgun’ Run 1, Frame 190 Importance Maps

focus on the fired projectile, as shown in Figure 42b. On the other hand, the ST-DIM encoder focused on a mixture of the projectile and the weapon. This is shown in Figure 42a. Additionally, there was a small focus on the wall to the right of the player. This focus on the background had been seen in the analysis of the previous frame as well. In theory, the information about the background could be used to estimate the position of the player in the level which is key to understanding which playstyle was being used. Another key factor of the playstyle was the weapon used, this can be determined by both what weapon the player was holding and the projectile shot. Hence, the hope would be to see a strong focus on the weapon and fired projectile. When examining the STDIM-VAE it was seen that there was indeed a focus on both the weapon and the projectile fired, as well as a smaller focus on the wall to the right of the player. This is shown in Figure 42c.

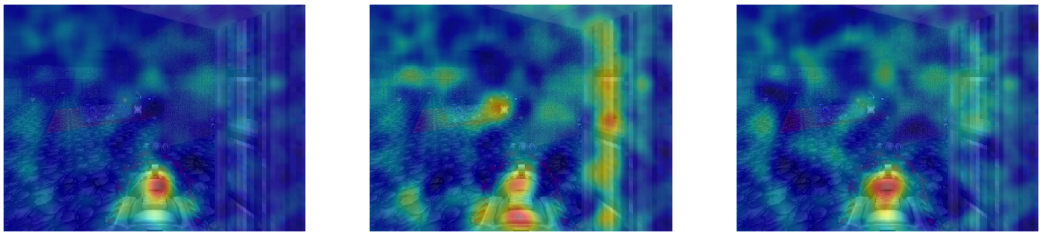


(a) ST-DIM (b) VAE (c) STDIM-VAE

Figure 42: ‘Window Camping’ at ‘Window 1’ with the ‘Rocket Launcher’ Run 1, Frame 132 Importance Maps

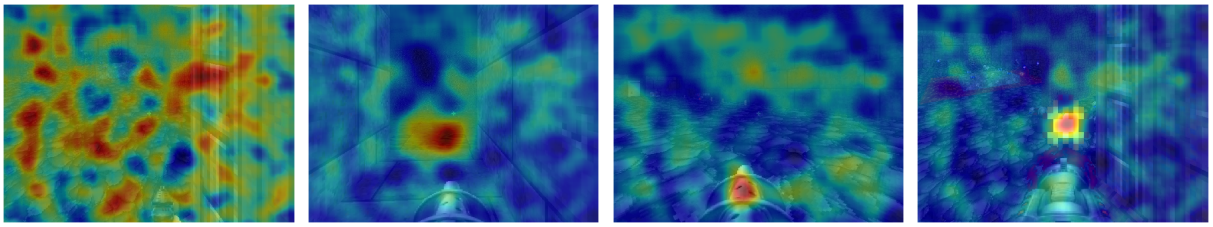
When considering the subsequent frame, the projectile was no longer within the frame and hence the VAE changed focus. While part of this focus was on the weapon, it also included the wall to the right of the player, this is shown in Figure 43b. In contrast, the ST-DIM encoder focused strongly on the weapon, as shown in Figure 43a. A similar focus was seen for STDIM-VAE, in Figure 43c, but a little of the right wall was also the subject of the focus. This was presumably from the VAE part of the hybrid encoder, but with a relatively low intensity.

After comparing the different encoder types, it was decided to conduct further analysis focusing on just the STDIM-VAE in additional situations. Based on this analysis several common focus types were found, examples for which are shown in Figure 44. Further examples for each type are shown in Appendix A.2.1.



(a) ST-DIM (b) VAE (c) STDIM-VAE

Figure 43: 'Window Camping' at 'Window 1' with the 'Rocket Launcher' Run 1, Frame 133 Importance Maps



(a) Broad Frame (b) Player's Surroundings (c) Weapon (d) Fired Projectile

Figure 44: Focus Types seen within VizDoom Importance Maps

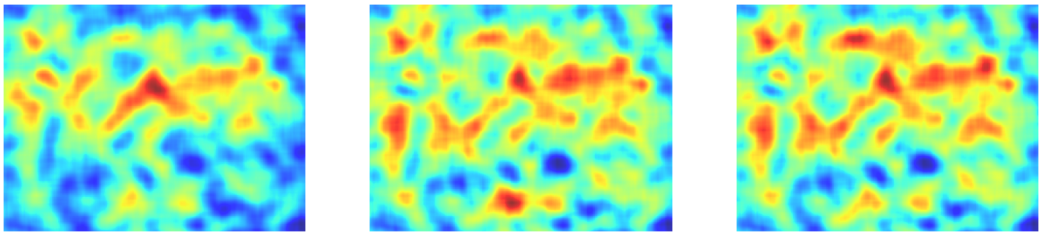
The first of these, appeared to have a interest in the whole frame. This tended to take the form of hotspots over the whole frame, that often encompassed most of the surrounding textures as seen within the frame. This is shown in Figure 44a. The second focus type had a few main areas that were directly ahead of the player. These covered parts of the environment (*e.g.* floor, wall, *etc.*). Figure 44b shows an example of when the focus was on the floor. The third focus type had its main area of focus on the player's weapon. This can be seen over a range of weapons the player holds in different gameplays, with one example shown in Figure 44c. It was noted on occasion focus was seen on both the weapon and fired projectile. This led to a fourth focus type which focused on either the weapon projectile or the flash around the gun muzzle upon firing and was found for a range of weapons from different gameplays. An example of focus on the fired projectile is shown in Figure 44d.

Overall, these focus types suggested that the encoder had the ability to encode both information about the player's surroundings and hence position. In addition, it is felt that encoded information about the weapon the player was using was also being captured. It is believed that the combination of this information should allow successful clustering of the different playstyles.

Following this, focus 'fingerprints' were also generated for the 'VizDoom' gameplays. However, presumably due to larger variability in what had been seen in the frames, a broad interest over the frame was often observed. Further, the idea of unique focus regions per playstyle was less relevant within this case, as the two defining factors were more defined by what was in a given region, namely in the player's hand for 'Weapon Choice' or background for 'Player Location'. In addition, it was seen that key foci on features (*e.g.* muzzle flash and weapon projectiles) related to the 'Weapon Choice'

appeared to be averaged out due to the small number of frames in which they were present in relation to the whole game sequence. Based on this, the focus of the analysis looked more at the shapes of the foci on both where the weapon was in the frame and the background in general, as these shapes most likely related to either the weapon shape or background type being encoded. In order to explore this, two gameplay triplets were chosen, where each, both had a gameplay pair that shared 'Weapon Choice' as well as a different pair that shared 'Player Location'.

The first triplet considered contained one gameplay where the player utilised 'Corner Camping' with a 'Shotgun'. The other two gameplays both traversed the 'Long' route but one used the 'Chain Gun' and the other used the 'Shotgun'. Each of the 'fingerprints' for these gameplays are shown in Figure 45. Firstly, the 'fingerprints' of the gameplays that both used the 'Shotgun' were compared, while a similar focus shape was seen on the centre of the background, in the form of an upside down 'V', other parts of the background had different focus regions. For example, in the 'Long' route version, a strong focus region was seen stretching horizontally across the top of the upside down 'V'. It was noted that when focussing on the bottom middle of the frame, where the weapon was held, there were three focus points which appeared to form a triangle in both 'fingerprints'. These appeared to be located on the left, right and top of the 'Shotgun'. Next, when comparing the gameplays that both used the 'Long' route, more similarities could be seen in the focus regions shape, for example the horizontal region spanning across the top of the upside 'V' is seen in both. However, when you focus on the weapon region, different focus shapes were seen. The combination of these suggested a commonality in the 'Player Location' but a difference in 'Weapon Choice' which was what was observed.



(a) 'Corner Camping' with the 'Shotgun' Run 1

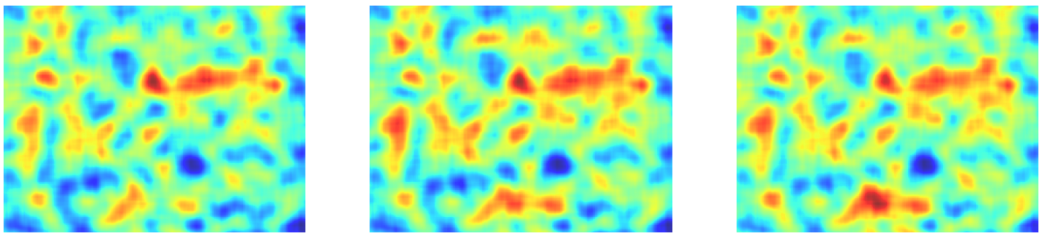
(b) Traversing the 'Long' Route with the 'Chain Gun' Run 1

(c) Traversing the 'Long' Route with the 'Shotgun' Run 1

Figure 45: Comparison of Focus 'Fingerprints' for Different Gameplays

The second triplet of gameplays contained gameplays where they all utilised 'Window Camping'. Two of these gameplays camped at the same window, with one using the 'BFG9000' and the second the 'Rocket Launcher'. The third gameplay also used the 'Rocket Launcher' but camped at a different window. Each of the 'fingerprints' for these gameplays are shown in Figure 46. Again, when comparing the 'fingerprints' where the same weapon had been used, in this case the 'Rocket Launcher', the same shaped focus region was seen over the weapon location. However, in this case, there was a thick rectangular region across the width of the weapon in the frame. After which, the 'fingerprints' of gameplays that camped at the same window were compared. For these,

similarity could be drawn between the shape of the background focus regions, however, different focus regions were seen over the weapon area.



(a) 'Window Camping' at 'Window 1' with the 'BFG9000' Run 1

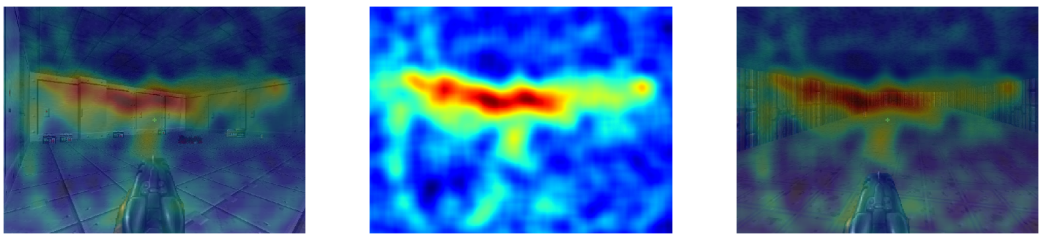
(b) 'Window Camping' at 'Window 1' with the 'Rocket Launcher' Run 1

(c) 'Window Camping' at 'Window 2' with the 'Rocket Launcher' Run 1

Figure 46: Comparison of Focus 'Fingerprints' for 'Window Camping' Gameplays

Next, analysis was conducted to compare gameplays of different playstyles. Given that each playstyle was built from two factors, namely 'Weapon Choice' and 'Player Location', it was decided to compare gameplays that only varied on one of these factors at a time and to explore if focus was found on aspects relating to that given factor. It was noted that for a large proportion of the frames, a broad focus was seen over the whole frame. It was suggested that this was due to more variability in each frame and therefore the VAE component of the encoder would often lead to a broad focus encapsulating all parts of the frame to allow reconstruction. Due to this, it was decided to concentrate more on frames where there were regions of more directed focus.

The first pair of gameplays compared shared the use of the 'Shotgun' but one gameplay 'Corner Camped' while the other traversed the 'Long' route. On several occasions, with one example shown in Figure 47, the focus was on parts of the wall of the room in which the player was camping. When this region was transferred to the matched frame, it fell on the brown wall seen in the matched frame. It was suggested therefore that this focus was encoding information about the player's surroundings, which could be used to infer where on the map the player was located.



(a) Frame 142

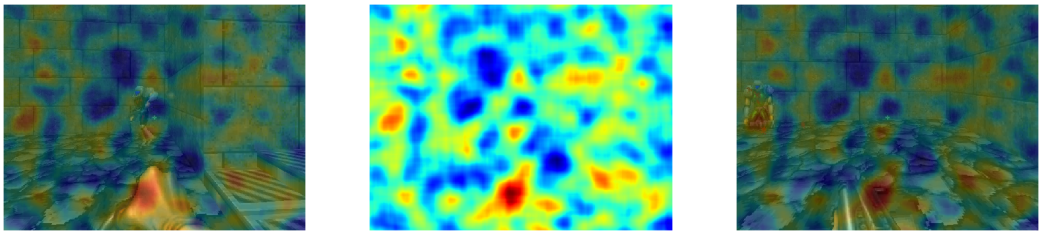
(b) Importance Map

(c) Frame 268 (Match 1)

Figure 47: Comparison of Runs 'Corner Camping' and traversing the 'Long' Route each with the 'Shotgun' at Frame 142

It was noted that occasionally the focus was on the weapon region, even though the 'Weapon Choice' was the same in both gameplays. This is shown in Figure 48. When investigating this example frame further, it was seen that the focus occurred when the

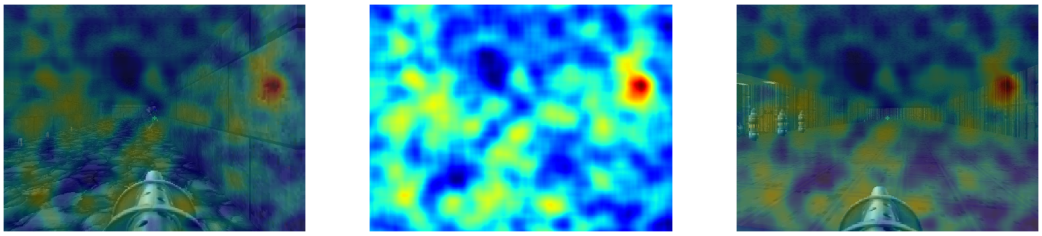
player was reloading their weapon and hence the weapon was held in a different position. This focus region hence made sense as the reload had caused visual differences in the weapon region to the matched frame, as although the ‘Weapon Choice’ was the same, the timing of the reloads was not.



(a) Frame 49 (b) Importance Map (c) Frame 61 (Match 1)

Figure 48: Comparison of Runs ‘Corner Camping’ and traversing the ‘Long’ Route each with the ‘Shotgun’ at Frame 49

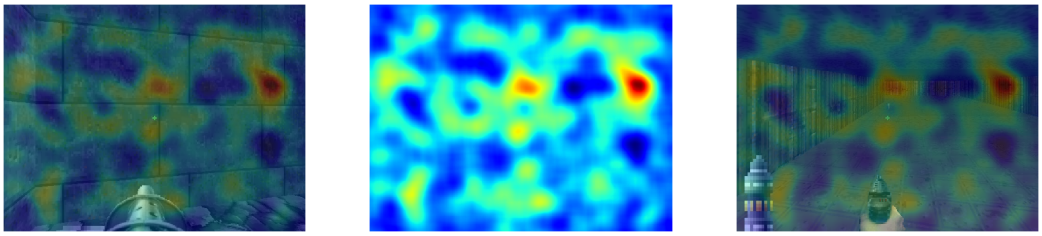
The following gameplay pair also explored the shared ‘Weapon Choice’ characteristic, but this time the weapon selected was the ‘Chain Gun’. In this case, in both gameplays, the player traversed the map, using either the ‘Short’ or ‘Long’ route. Again, throughout the gameplay, foci were seen on different parts of the player’s surroundings, whether that be on the walls as shown in Figures 49 and 50 or the floor as seen in Figures 51 and 52. When each of the example focus regions were transferred to the relevant matched frame, it was noted that often both the colour and texture of the surroundings changed within the region. As before, it was suggested that encoding of these parts of the player’s surroundings could allow the inference of the player’s location based on a combination of the colour and texture of the player’s surroundings.



(a) Frame 179 (b) Importance Map (c) Frame 180 (Match 1)

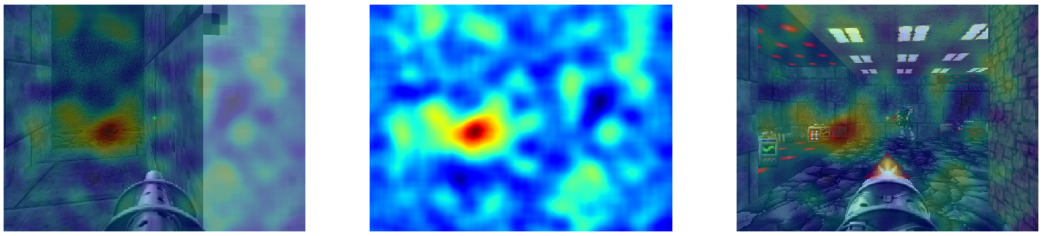
Figure 49: Comparison of Runs traversing the ‘Short’ and ‘Long’ Routes each with the ‘Chain Gun’ at Frame 179

As before foci were also seen on the weapon. When investigating this further they often took one of two forms. Firstly, the focus was on the edge of the weapon, as shown in Figure 53. Although the desired weapon was the same, it was possible that it was not held by the player at all points in the gameplay. In this case, after examining the matched frame, it was seen that the player had just died and respawned with the ‘Pistol’ and hence the weapon held at that point should have been highlighted as a difference, as was the case. Secondly, while the focus was on the weapon, focus was located on the muzzle of the weapon in the frames in which the weapon was fired and hence



(a) Frame 395 (b) Importance Map (c) Frame 386 (Match 1)

Figure 50: Comparison of Runs traversing the ‘Short’ and ‘Long’ Routes each with the ‘Chain Gun’ at Frame 395

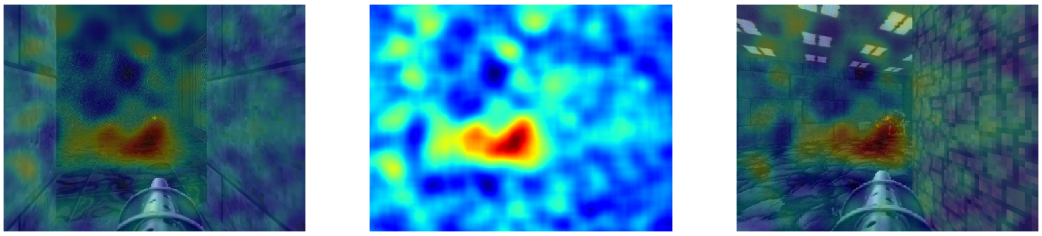


(a) Frame 271 (b) Importance Map (c) Frame 269 (Match 1)

Figure 51: Comparison of Runs traversing the ‘Short’ and ‘Long’ Routes each with the ‘Chain Gun’ at Frame 271

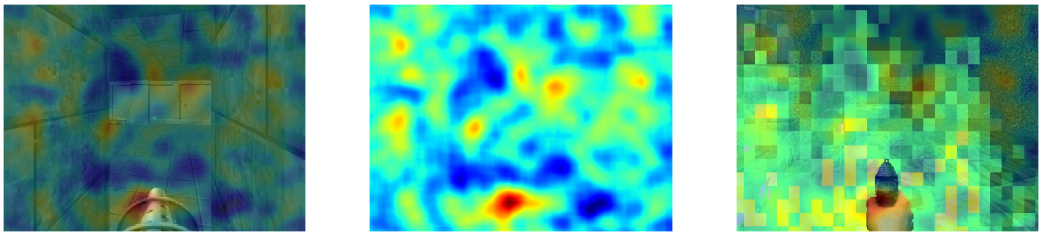
muzzle flash was present, as shown in Figures 54, 55 and 56. As the focus appeared to concentrate on the muzzle flash, differences are expected between gameplays, as the players are likely to shoot at different times even if they are using the same weapon. When investigating the relevant matched frame for each of these examples, it was indeed seen that the weapon was not being fired in each matched frame.

Subsequently a comparison was made between gameplays that used different weapons, either the ‘Chain Gun’ or ‘Shotgun’, but in both cases traversed around the ‘Long’ route. The analysis showed, that even although both use the same route, focus was still on the player’s surroundings as shown in Figures 57 and 58. This was likely to be the case as, even although the route was the same, due to other factors (*e.g.* spawn location) a perfect match in the loop was unlikely. This is shown in the matched frames,



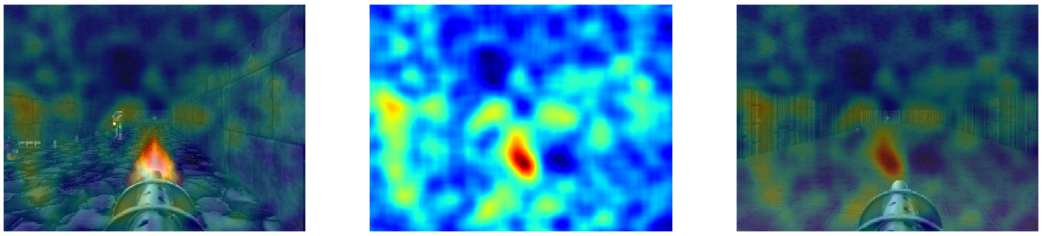
(a) Frame 274 (b) Importance Map (c) Frame 272 (Match 1)

Figure 52: Comparison of Runs traversing the ‘Short’ and ‘Long’ Routes each with the ‘Chain Gun’ at Frame 274



(a) Frame 327 (b) Importance Map (c) Frame 317 (Match 1)

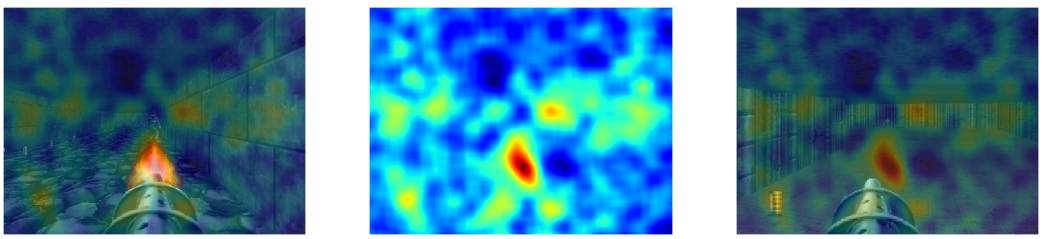
Figure 53: Comparison of Runs traversing the ‘Short’ and ‘Long’ Routes each with the ‘Chain Gun’ at Frame 327



(a) Frame 182 (b) Importance Map (c) Frame 183 (Match 1)

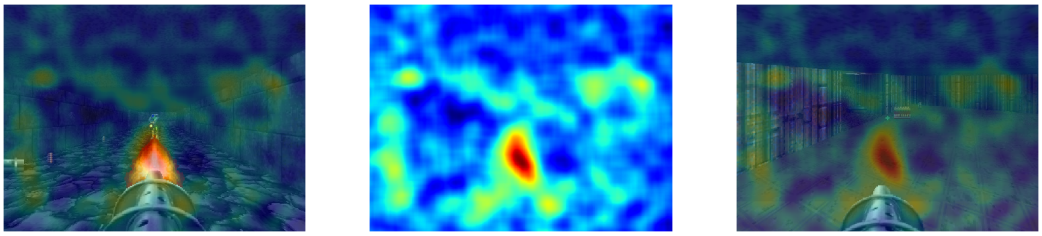
Figure 54: Comparison of Runs traversing the ‘Short’ and ‘Long’ Routes each with the ‘Chain Gun’ at Frame 182

as it shows matches have been made between frames showing the player at different parts of the route.



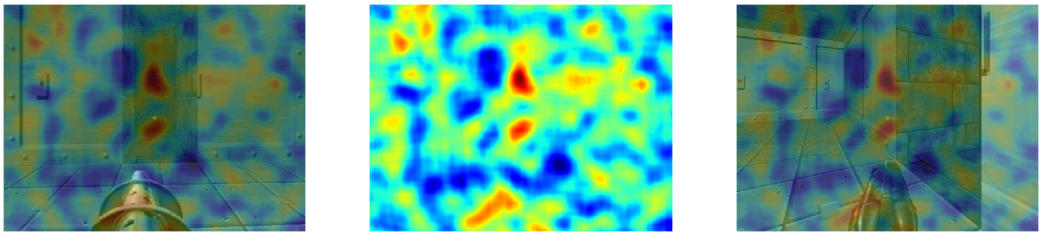
(a) Frame 189 (b) Importance Map (c) Frame 190 (Match 1)

Figure 55: Comparison of Runs traversing the ‘Short’ and ‘Long’ Routes each with the ‘Chain Gun’ at Frame 189



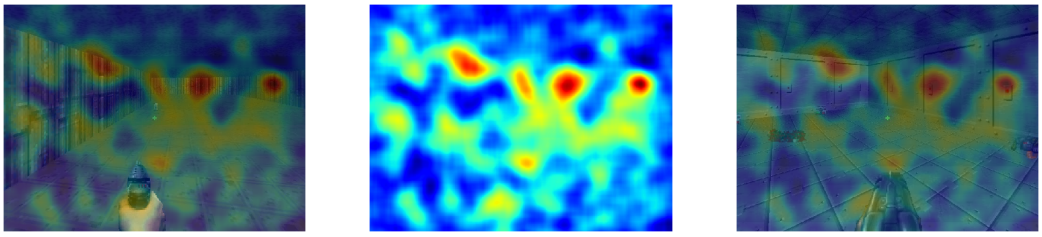
(a) Frame 200 (b) Importance Map (c) Frame 201 (Match 1)

Figure 56: Comparison of Runs traversing the ‘Short’ and ‘Long’ Routes each with the ‘Chain Gun’ at Frame 200



(a) Frame 101 (b) Importance Map (c) Frame 184 (Match 1)

Figure 57: Comparison of Runs using the ‘Chain Gun’ and ‘Shotgun’ each traversing the ‘Long’ Route at Frame 101



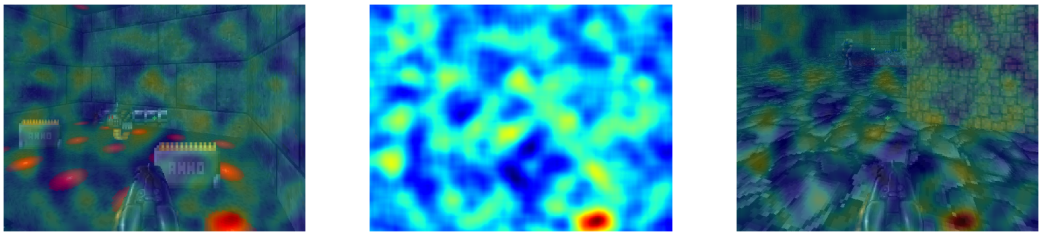
(a) Frame 388 (b) Importance Map (c) Frame 388 (Match 1)

Figure 58: Comparison of Runs using the ‘Chain Gun’ and ‘Shotgun’ each traversing the ‘Long’ Route at Frame 388

Further, patterns on the floor were also areas of focus, as seen in Figure 59. Although, these patterns were centred around the location of one of the weapons, it was thought that these patterns could potentially provide information about the weapon.

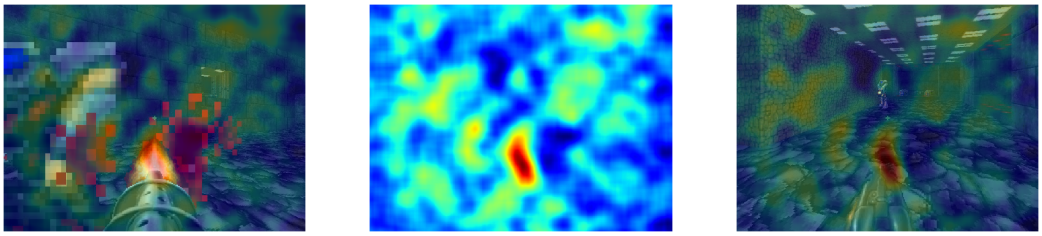
In addition, focus was often seen on the weapon when it was fired, as shown in Figure 60. Information about the firing animation can provide information about the weapon being used, as the animation differs across weapon types.

The next comparison pair also used different weapons but this time either the ‘Chain Gun’ or ‘Plasma Rifle’. Further, these comparisons also shared the ‘Player Location’ characteristic, although this time both playstyles stayed within the ‘Main Room’. In this case, focus was seen on the weapon, not only when it was fired but also at other times as well, as shown in Figures 61 and 62 respectively. It is suggested that these foci should allow the encoding of information that can infer the weapon used. Further,



(a) Frame 24 (b) Importance Map (c) Frame 28 (Match 1)

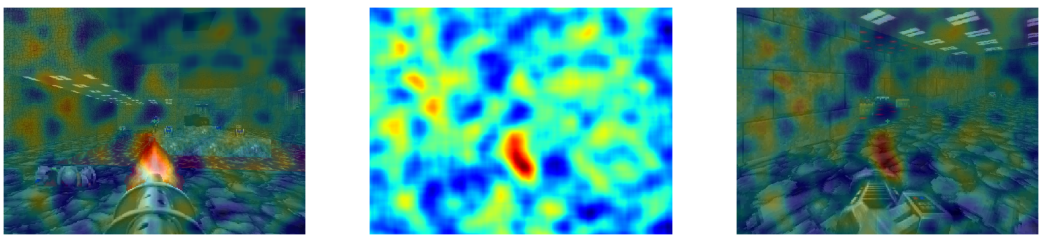
Figure 59: Comparison of Runs using the ‘Chain Gun’ and ‘Shotgun’ each traversing the ‘Long’ Route at Frame 24



(a) Frame 229 (b) Importance Map (c) Frame 308 (Match 1)

Figure 60: Comparison of Runs using the ‘Chain Gun’ and ‘Shotgun’ each traversing the ‘Long’ Route at Frame 229

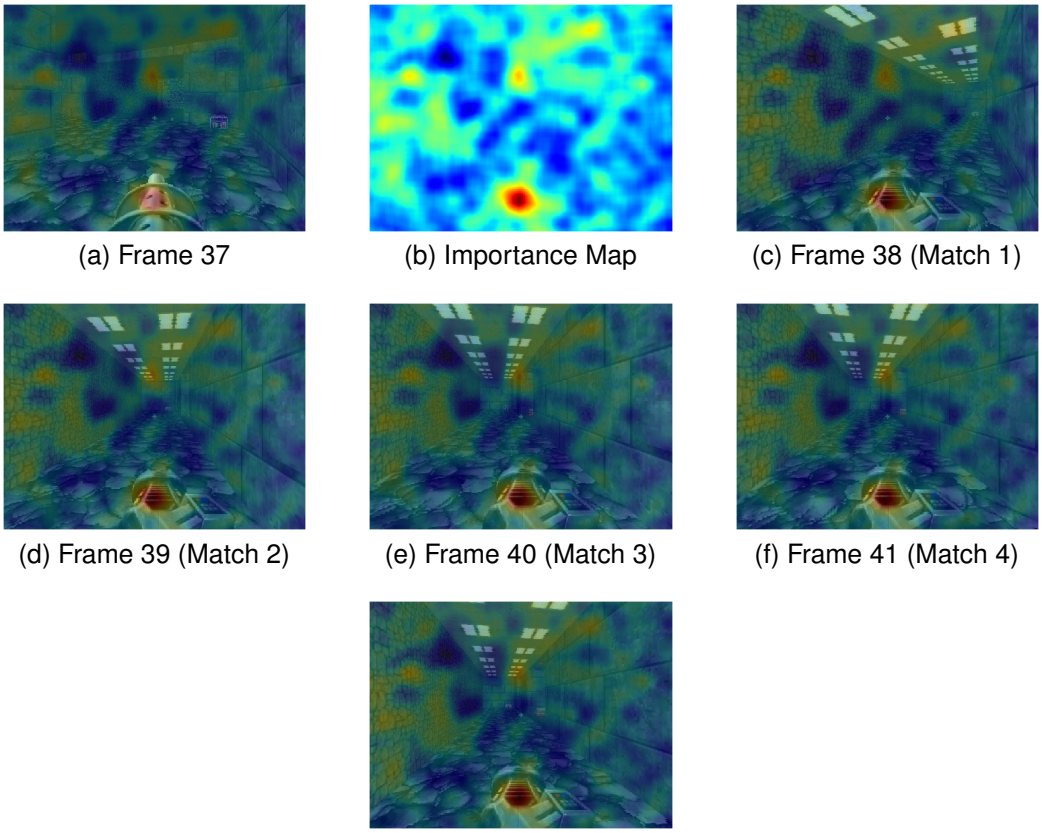
when exploring the matched frames from when the weapon was not fired, it was found that when the region was transferred to the matched frames, it also focused on part of the weapon. Potentially, the similarity of the matched frames can provide information regarding similarity of the weapons used.



(a) Frame 85 (b) Importance Map (c) Frame 90 (Match 1)

Figure 61: Comparison of Runs using the ‘Chain Gun’ and ‘Plasma Rifle’ each that ‘Camps’ in the ‘Main Room’ at Frame 85

Focus regions were also noticed on different parts of the player’s surroundings including the walls as shown in Figures 63 and 64, as well as the floor as shown in Figures 65, 66 and 67. Although, in many cases when the frames were examined, it was found that these foci appeared at a time when the player was not in the ‘Main Room’, however they were most likely heading back to the ‘Main Room’ and hence the surroundings were different. Further, in one case, when the matched frame shown in Figure 64 was examined, it was found that the focus appeared to be on a projectile in



(a) Frame 37 (b) Importance Map (c) Frame 38 (Match 1)

(d) Frame 39 (Match 2) (e) Frame 40 (Match 3) (f) Frame 41 (Match 4)

(g) Frame 42 (Match 5)

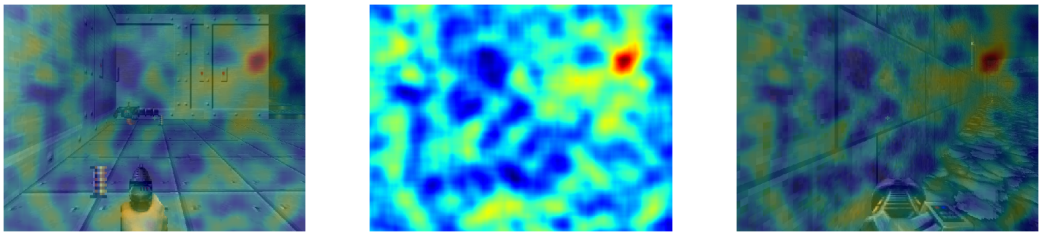
Figure 62: Comparison of Runs using the ‘Chain Gun’ and ‘Plasma Rifle’ each that ‘Camps’ in the ‘Main Room’ at Frame 37

the matched frame instead of the wall, as was originally thought.

Further, at different points in this gameplay, a focus was seen on other assets, for example the enemy model as shown in Figure 68. While this has no direct link to the playstyle, a reason for this difference could be theorised. For the enemy model case, the differences could be caused by the fact that it is down to chance if the player encounters an enemy and hence this would manifest itself as differences between gameplays. However, the matched frame, shows the focus region also aligns with a fired projectile observed in the matched frame. It is suggested that as the region appears to fit better with the projectile shape, that this could be the reason instead of the enemy model as previously thought.

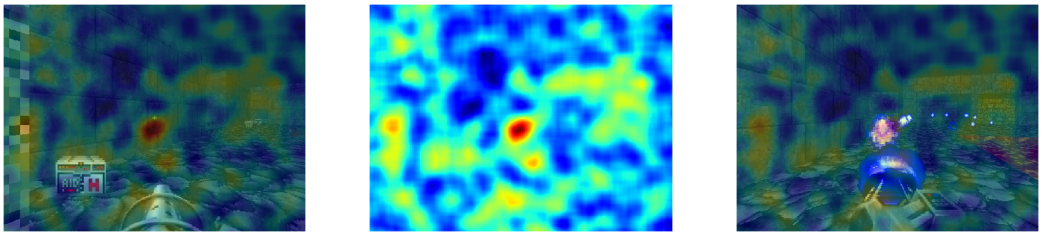
The last two comparisons looked at gameplays that utilised ‘Window Camping’. The first of these camped in the same window but used different weapons, either the ‘BFG9000’ or ‘Rocket Launcher’. Focus could be seen on the player’s surroundings (*e.g.* walls and ceiling), as shown in Figures 69 and 70. Although when considering the relevant matched frames these regions appeared to align with more unique visuals, for example ammunition packages, weapons and ceiling lights. The observation of these assets can potentially narrow down where the player is currently located.

Later in the gameplay comparison, focus can be seen on both the weapon, as shown in Figure 71, as well as on the fired projectiles, as shown in Figure 72. Each of these



(a) Frame 116 (b) Importance Map (c) Frame 111 (Match 1)

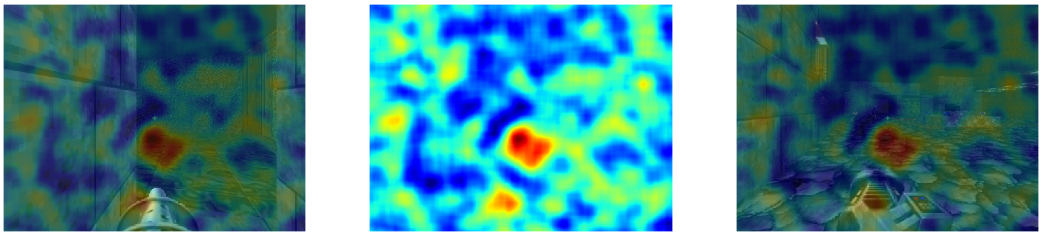
Figure 63: Comparison of Runs using the ‘Chain Gun’ and ‘Plasma Rifle’ each that ‘Camps’ in the ‘Main Room’ at Frame 116



(a) Frame 148 (b) Importance Map (c) Frame 128 (Match 1)

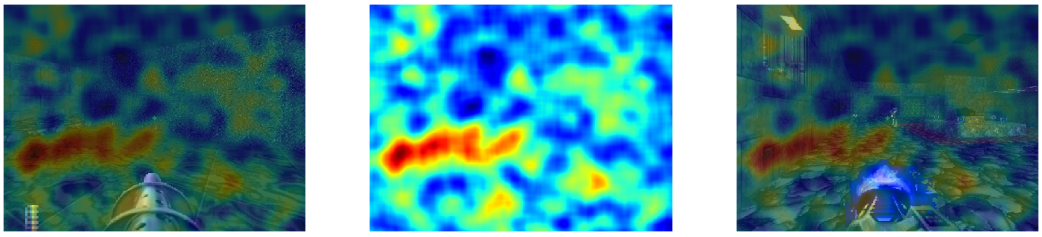
Figure 64: Comparison of Runs using the ‘Chain Gun’ and ‘Plasma Rifle’ each that ‘Camps’ in the ‘Main Room’ at Frame 148

foci contain visual information that can be used to determine the weapon being used.



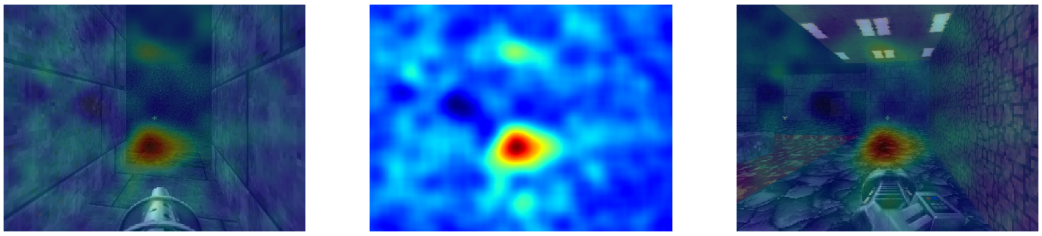
(a) Frame 136 (b) Importance Map (c) Frame 116 (Match 1)

Figure 65: Comparison of Runs using the ‘Chain Gun’ and ‘Plasma Rifle’ each that ‘Camps’ in the ‘Main Room’ at Frame 136



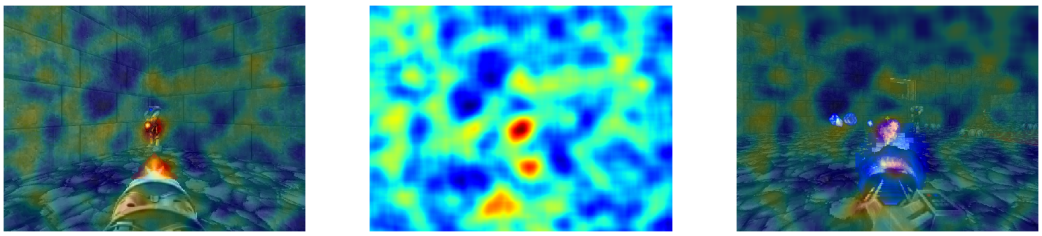
(a) Frame 138 (b) Importance Map (c) Frame 118 (Match 1)

Figure 66: Comparison of Runs using the ‘Chain Gun’ and ‘Plasma Rifle’ each that ‘Camps’ in the ‘Main Room’ at Frame 138



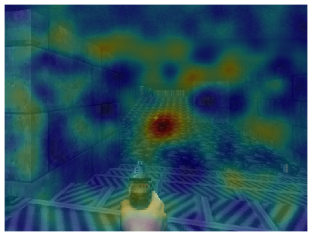
(a) Frame 298 (b) Importance Map (c) Frame 302 (Match 1)

Figure 67: Comparison of Runs using the ‘Chain Gun’ and ‘Plasma Rifle’ each that ‘Camps’ in the ‘Main Room’ at Frame 298

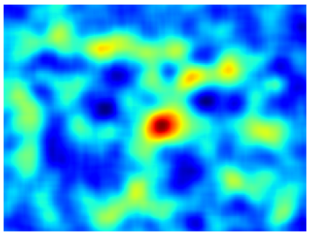


(a) Frame 257 (b) Importance Map (c) Frame 261 (Match 1)

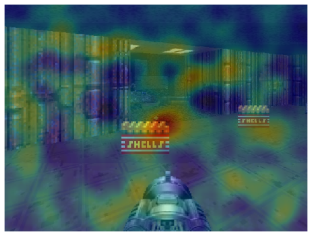
Figure 68: Comparison of Runs using the ‘Chain Gun’ and ‘Plasma Rifle’ each that ‘Camps’ in the ‘Main Room’ at Frame 257



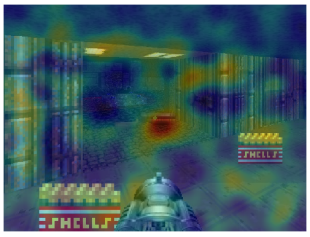
(a) Frame 47



(b) Importance Map

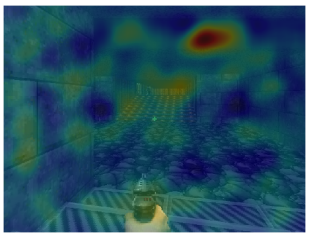


(c) Frame 89 (Match 1)

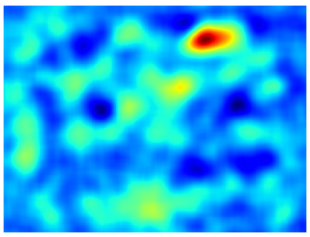


(d) Frame 90 (Match 2)

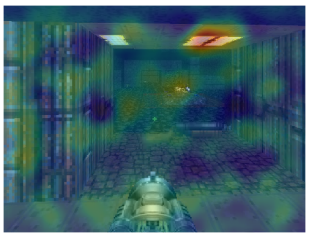
Figure 69: Comparison of Runs using the ‘BFG9000’ and ‘Rocket Launcher’ each utilising ‘Window Camping’ at ‘Window 1’ at Frame 47



(a) Frame 49

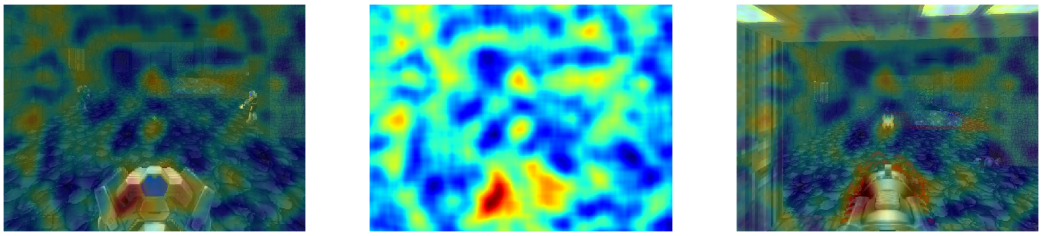


(b) Importance Map



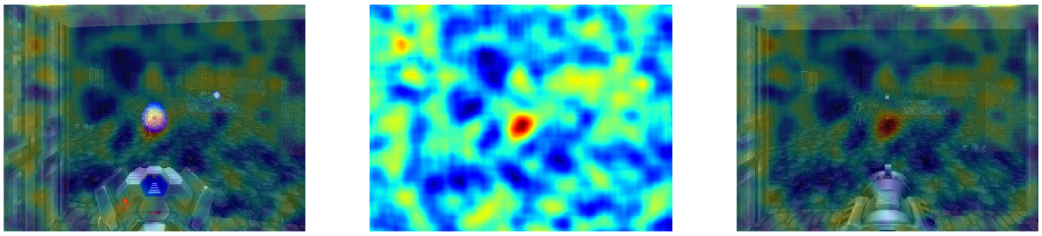
(c) Frame 92 (Match 1)

Figure 70: Comparison of Runs using the ‘BFG9000’ and ‘Rocket Launcher’ each utilising ‘Window Camping’ at ‘Window 1’ at Frame 49



(a) Frame 191 (b) Importance Map (c) Frame 277 (Match 1)

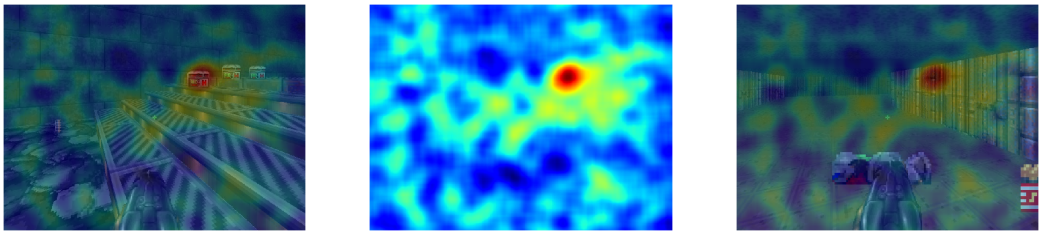
Figure 71: Comparison of Runs using the ‘BFG9000’ and ‘Rocket Launcher’ each utilising ‘Window Camping’ at ‘Window 1’ at Frame 491



(a) Frame 275 (b) Importance Map (c) Frame 324 (Match 1)

Figure 72: Comparison of Runs using the ‘BFG9000’ and ‘Rocket Launcher’ each utilising ‘Window Camping’ at ‘Window 1’ at Frame 275

In addition, in these gameplay comparisons, focus was also seen on other assets, for example a ‘medikit’. An example of an importance map of this is shown in Figure 73. In this case, the focus was seen as the player was approaching the ‘medikit’ item in order to acquire it. The player is only going to need and hence acquire a ‘medikit’ after having taken damage, mostly commonly from an encounter with an enemy, and hence these differences made sense. Alternatively, when looking at the matched frame, the focus fell on the brown wall and hence this could be an alternative reason for this difference.

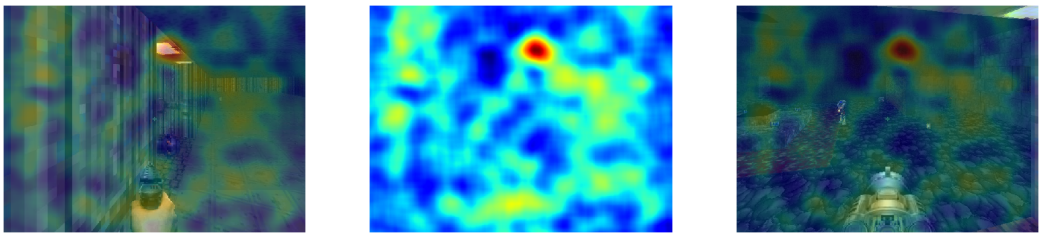


(a) Frame 366 (b) Importance Map (c) Frame 367 (Match 1)

Figure 73: Comparison of Runs using the ‘BFG9000’ and ‘Rocket Launcher’ each utilising ‘Window Camping’ at ‘Window 1’ at Frame 366

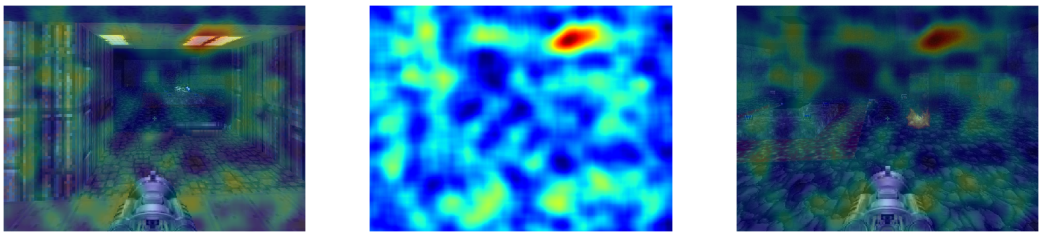
The second ‘Window Camping’ comparison compared gameplays that used the same weapon but camped at different windows. Interestingly, the gameplay focus was

seen to be on the ceiling lights above each of the possible camping windows, as shown in Figure 74 and Figure 75. Potentially, the fact that both of these were seen, could be used to predict which window the player had camped at.



(a) Frame 61 (b) Importance Map (c) Frame 65 (Match 1)

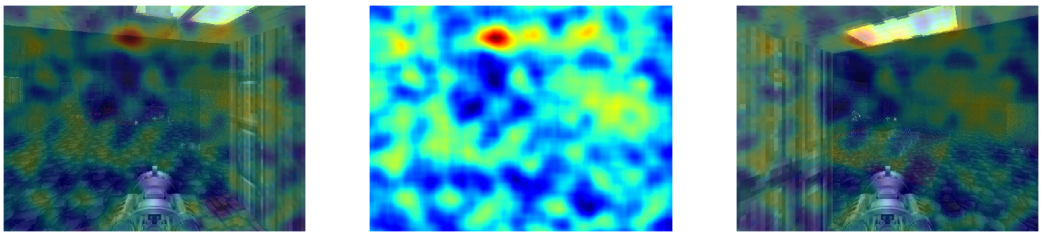
Figure 74: Comparison of Runs utilising 'Window Camping' at 'Window 1' and 'Window 2' each with the 'Rocket Launcher' at Frame 61



(a) Frame 92 (b) Importance Map (c) Frame 96 (Match 1)

Figure 75: Comparison of Runs utilising 'Window Camping' at 'Window 1' and 'Window 2' each with the 'Rocket Launcher' at Frame 92

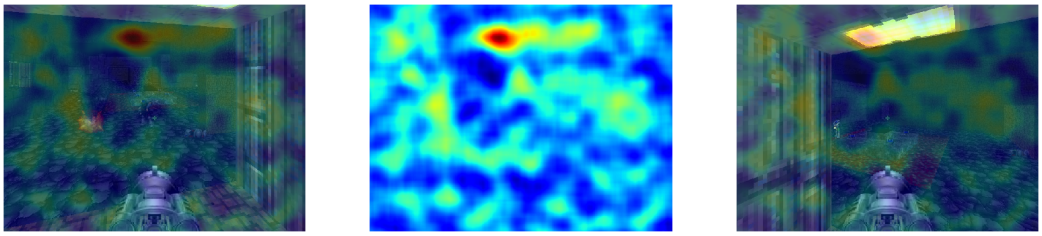
Further, foci on the player's surroundings were occasionally seen, as shown in Figure 76 and Figure 77. Although, for each of these cases when examining the relevant matched frame, the focus was seen on the ceiling light. As suggested before, the observation of these lights can help narrow down the possible locations of the player.



(a) Frame 312 (b) Importance Map (c) Frame 296 (Match 1)

Figure 76: Comparison of Runs utilising 'Window Camping' at 'Window 1' and 'Window 2' each with the 'Rocket Launcher' at Frame 312

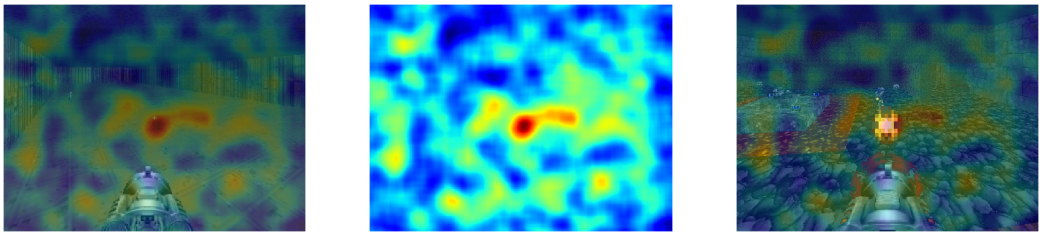
Another environmental surrounding type based focus appeared to be seen on the floor, as shown in Figure 78. However, when the matched frame was examined, the focus region appeared over a fired projectile in the matched frame. Given that this focus



(a) Frame 337 (b) Importance Map (c) Frame 323 (Match 1)

Figure 77: Comparison of Runs utilising ‘Window Camping’ at ‘Window 1’ and ‘Window 2’ each with the ‘Rocket Launcher’ at Frame 337

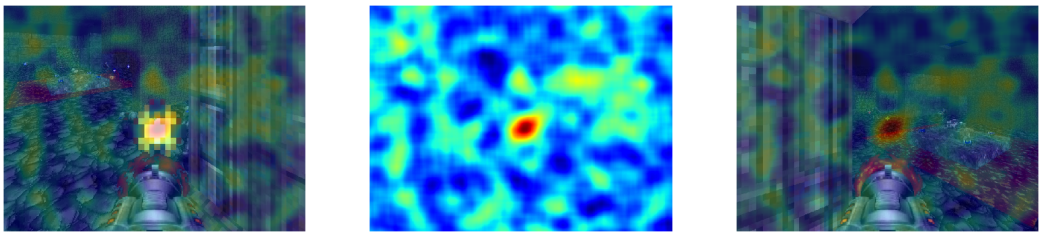
fitted tightly around the projectile it was expected that this was the true reason for the difference.



(a) Frame 70 (b) Importance Map (c) Frame 74 (Match 1)

Figure 78: Comparison of Runs utilising ‘Window Camping’ at ‘Window 1’ and ‘Window 2’ each with the ‘Rocket Launcher’ at Frame 70

Further examples of projectile focus can be seen in Figure 79 and Figure 80. This focus on projectiles makes sense, as although the weapon was the same and hence the fired projectiles were the same, the time that they were fired was different.

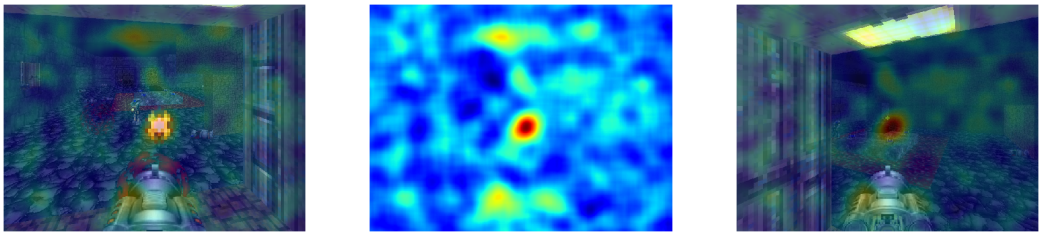


(a) Frame 141 (b) Importance Map (c) Frame 147 (Match 1)

Figure 79: Comparison of Runs utilising ‘Window Camping’ at ‘Window 1’ and ‘Window 2’ each with the ‘Rocket Launcher’ at Frame 141

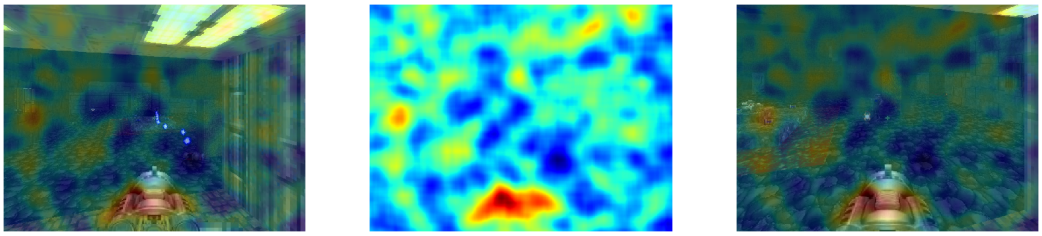
At various points the focus was also seen on the weapon as shown in Figure 81 and Figure 82.

The relevant matched frames were investigated to look for a reason for this, given that the same weapon was meant to have been used. In both cases, the ‘Rocket Launcher’ was used in both frames of each matched pair. However, when examined



(a) Frame 339 (b) Importance Map (c) Frame 325 (Match 1)

Figure 80: Comparison of Runs utilising ‘Window Camping’ at ‘Window 1’ and ‘Window 2’ each with the ‘Rocket Launcher’ at Frame 339



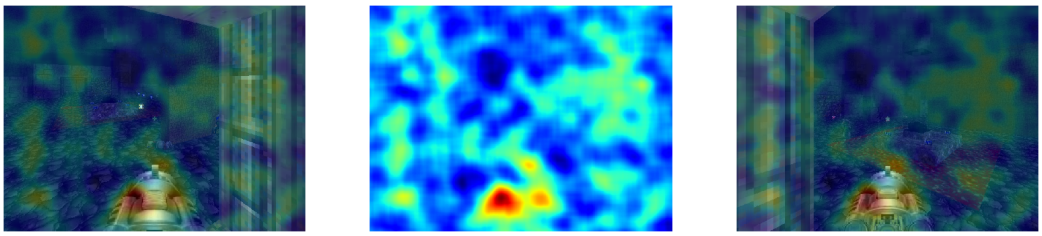
(a) Importance Map (b) Importance Map (c) Frame 100 (Match 1)

Figure 81: Comparison of Runs utilising ‘Window Camping’ at ‘Window 1’ and ‘Window 2’ each with the ‘Rocket Launcher’ at Frame 96

further it was found that weapon movement had been caused by the weapon having recently been fired, hence this focus made sense.

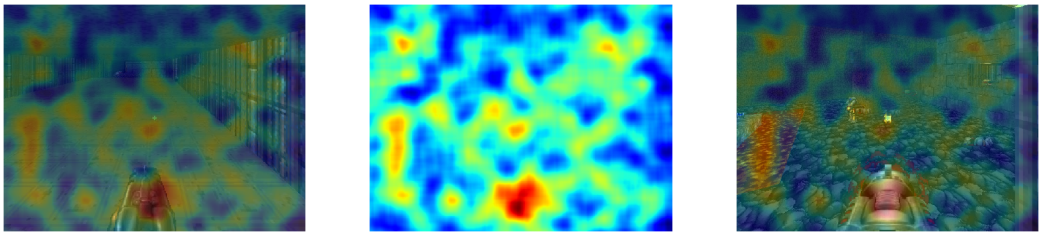
An additional instance of weapon focus was also investigated and the importance map can be seen in Figure 83. In this case a different weapon was being held by the player, as shown in the matched frame. Although the desired weapon had been acquired, it had appeared to have run out of ammunition and hence another weapon had been swapped to by the player while more ammunition was being searched for.

Finally, each of the gameplays were projected onto a 2D plane. In this case, more variability was seen in these projections over the encoder types, than seen in the previous case study. Firstly, the projections from the VAE were investigated. One example is shown in Figure 84 and the full set of projections are in Appendix A.2.2.



(a) Frame 134 (b) Importance Map (c) Frame 138 (Match 1)

Figure 82: Comparison of Runs utilising ‘Window Camping’ at ‘Window 1’ and ‘Window 2’ each with the ‘Rocket Launcher’ at Frame 134



(a) Frame 377 (b) Importance Map (c) Frame 364 (Match 1)

Figure 83: Comparison of Runs utilising ‘Window Camping’ at ‘Window 1’ and ‘Window 2’ each with the ‘Rocket Launcher’ at Frame 377

When considering the playstyles that utilised ‘Window Camping’, it can be seen that these were grouped by weapon choice (*i.e.* represented by different colours in the projection). Further, it can be seen that all playstyles that used the ‘Chain Gun’ (*i.e.* orange in colour in the projection) are grouped together. Although, it can be seen that a further point representing the use of the ‘Shotgun’ (*i.e.* blue in colour in the projection) is on the edge of this group. However, this was likely to be the case, as the playstyle used in the gameplay shared the same ‘Player Location’ as other points within the ‘Chain Gun’ group. On the whole, it can be seen that each group normally shared the ‘Weapon Choice’ characteristic. However, for the ‘Shotgun’ playstyles (*i.e.* blue in colour in the projection), apart from the single gameplay previously mentioned, were physically separate even though they shared the same ‘Weapon Choice’. This made sense as they had a different ‘Player Locations’ and hence were considered to be different playstyles. It was noted that this may suggest that in these cases the encoded representations contain more information about the ‘Weapon Choice’ than the ‘Player Location’.

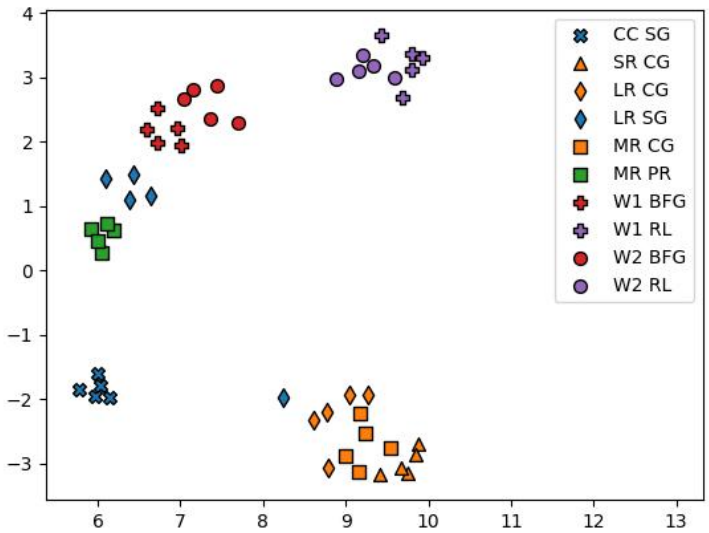


Figure 84: Projection 1 of VizDoom Gameplays using the VAE

Following this, the ST-DIM projections were investigated. One of these projections

is shown in Figure 85, with the full set in Appendix A.2.2. A similar projection pattern was seen for the playstyles utilising 'Window Camping', although in some cases there appeared to be a slightly clearer split within the group when looking at different 'Player Locations'. Further, groups were observed based on only 'Player Location'. For example, gameplays using the 'Long' route (*i.e.* diamond in shape in the projection) were grouped together even when they used either the 'Chain Gun' (*i.e.* orange in colour in the projection) or 'Shotgun' (*i.e.* blue in colour in the projection). However, there were also groups based on 'Weapon Choice'. For example, eight gameplays using the 'Chain Gun' (*i.e.* orange in colour in the projection) are grouped together. This group contained two different 'Player Locations', namely 5 that traversed the 'Short' route (*i.e.* triangle in shape in the projection) and 3 where the player stayed in the main room (*i.e.* square in shape in the projection). Further, it was noted, that the two other gameplays that stayed within the main room (*i.e.* square in shape in the projection) and used the 'Chain Gun' (*i.e.* orange in colour in the projection) were projected between this group and another group made up of gameplays that stayed in the main room (*i.e.* square in shape in the projection) but used the 'Plasma Rifle' (*i.e.* green in colour in the projection). Based on these observations, while there was still a focus on 'Weapon Choice' a further interest in 'Player Location' appeared more prominent than within the VAE projections.

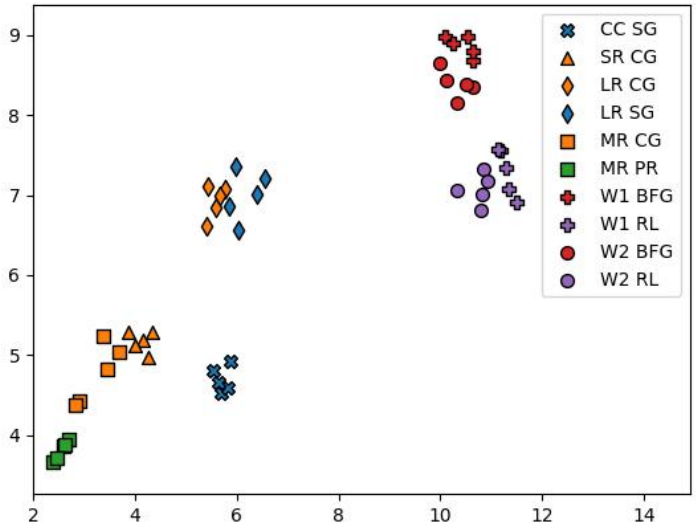


Figure 85: Projection 1 of VizDoom Gameplays using the ST-DIM

Following this, the projections of the STDIM-VAE were explored and one example is shown in Figure 86. Again, the full set of projections are shown in Appendix A.2.2. Similar patterns, to those seen in the ST-DIM projections, were observed. However, there appeared to be more alignment across these groups. While each group in itself, normally shared at least one characteristic, how they were normally aligned meant that the side adjacent to a close group in the projection space, shared the second trait with that group. Although, this was seen in the ST-DIM projections, it appeared clearer for the STDIM-VAE projections. For example, when focusing on the 'Window Camping'

playstyles, there were two main groups based on the ‘Weapon Choice’ (*i.e.* represented by colour in the projection). It was seen that when exploring the points on the edge closest to the adjacent group, these gameplays often shared the same ‘Player Location’ (*i.e.* represented by shape in the projection) as seen on the edge of the other group. Further, when considering ‘non-camping’ playstyles, a similar pattern was often seen. Firstly, often at one corner of the projection space, there was a group of gameplays that all stayed in the main room (*i.e.* rectangle in shape in the projection), although some gameplays used the ‘Plasma Rifle’ (*i.e.* green in colour in the projection) while others used the ‘Chain Gun’ (*i.e.* orange in colour in the projection) but these were on opposite sides of the group. If you moved from the group at the side of the ‘Chain Gun’ gameplays, the next group contained gameplays that all used the ‘Chain Gun’ (*i.e.* orange in colour in the projection) but either stayed in the main room (*i.e.* rectangle in shape in the projection) or traversed the ‘Short’ route around the level (*i.e.* triangle in shape in the projection). The main room gameplays were usually encountered first, as they were most similar to the previous group. If you then continued from this group in the direction of the gameplays traversing the ‘Short’ route, you then encountered another group. This group contained gameplays that traversed the ‘Long’ route (*i.e.* diamond in shape in the projection), although parts of these routes were shared and hence their similarity. However, this group contained two ‘Weapon Choices’, either ‘Chain Gun’ (*i.e.* orange in colour in the projection) or ‘Shotgun’ (*i.e.* blue in colour in the projection). Again, the gameplays most similar to the previous group were encountered first, namely the gameplays using the ‘Chain Gun’.

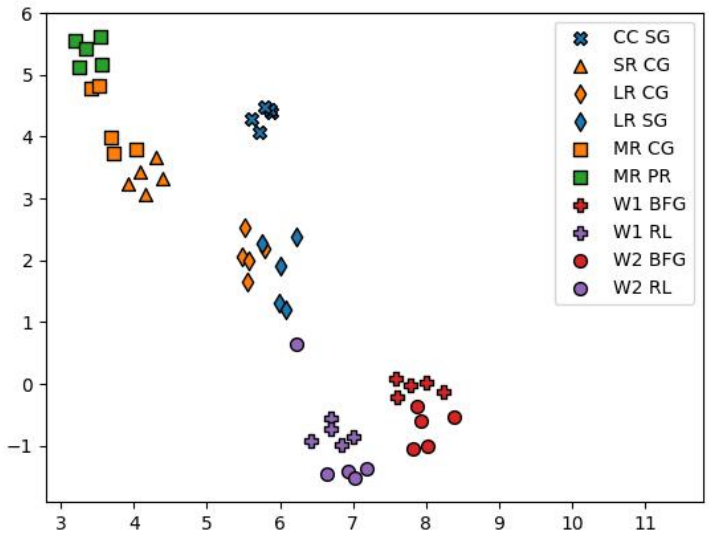


Figure 86: Projection 1 of VizDoom Gameplays using the STDIM-VAE

Similarly to the previous case study, the two representation evaluation techniques were applied to quantify the quality of the representation space. When examining the group average linkage between playstyles, shown in Figure 87, a lower value can be seen on the main diagonal indicating a lower group average linkage between each

playstyle and itself. A further trend can also be observed, namely when two playstyles share one of the factors used to define the playstyle, this often resulted in a lower group average linkage between the playstyles. These two observations show that not only are videos of the same playstyle being encoded close together but also that they were aligned so that playstyles that shared a playstyle defining attribute are also closer together. This further reinforces the pattern noticed within the projection space.

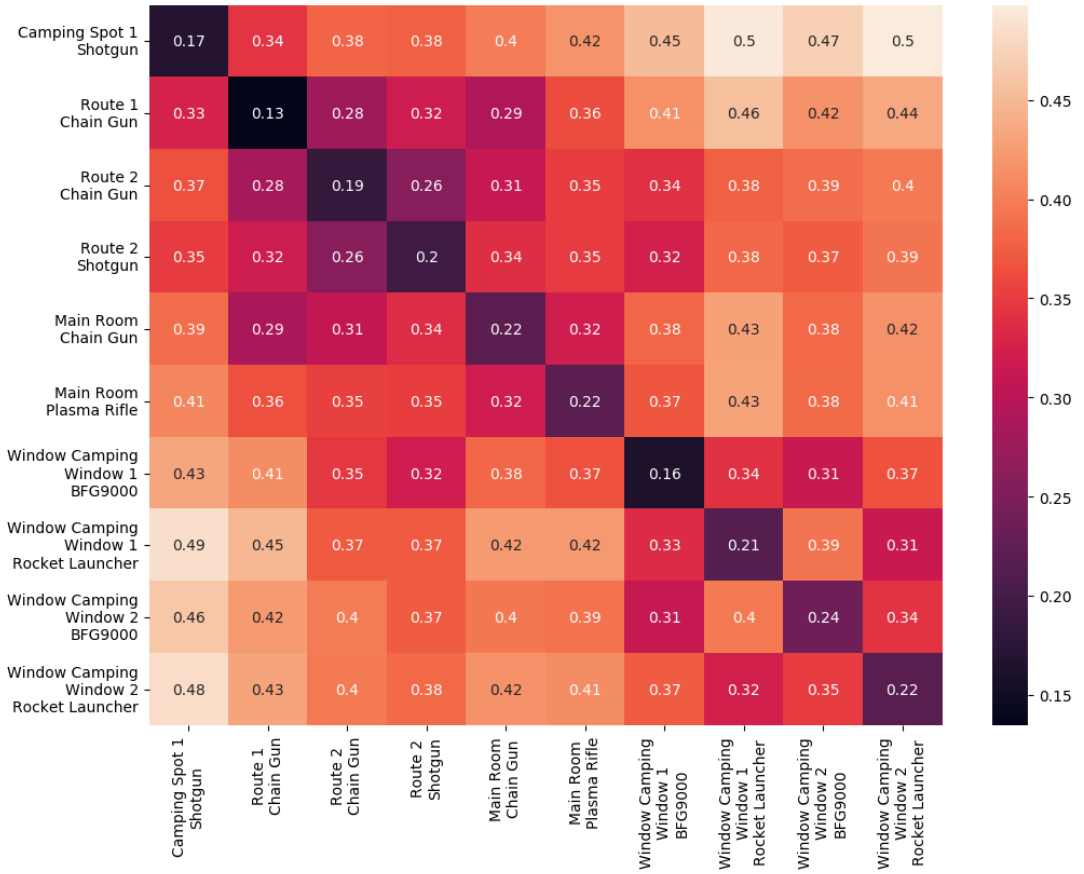


Figure 87: Group Average Linkage Between Playstyles for VizDoom

The average silhouette score defined encoder metric combinations were also compared and are shown in Table 6. Once again, the combination of the STDIM-VAE and cosine metric produced the best score. However, the score was notably lower than that observed in the previous case study. This potentially reflects the expected increased difficulty of clustering playstyles within this game, given that it is a 3D, partially observed environment.

Table 6: Average Silhouette Scores for VizDoom (± SD)

		Distance Metric		
		Manhattan	Euclidean	Cosine
Encoder	VAE	0.006 (0.002)	0.007 (0.002)	0.058 (0.004)
	ST-DIM	0.097 (0.004)	0.098 (0.004)	0.173 (0.005)
	STDIM-VAE	0.097 (0.006)	0.099 (0.006)	0.175 (0.007)

4.4.3 Hitman

The third case study focused on a 'stealth' game that was designed based on the 'Hitman' series. However, this new game was converted into a grid-based environment with the player only seeing a section of the game space at any given time. This was done so the environment was partially observed with a small discrete action space. A base game level was designed and is shown in Figure 88.

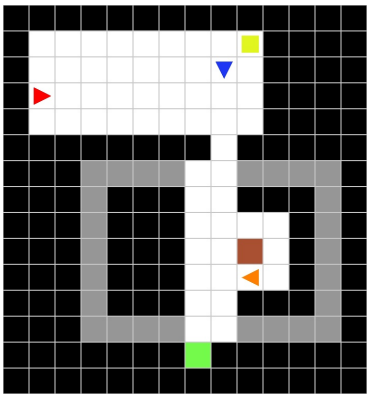


Figure 88: Base Hitman Level

In this newly designed 'stealth' game the player was represented by a blue triangle and is given the target of reaching the green goal square. However, to be successful the player must avoid being caught by the enemy NPCs when on route to the goal. The NPCs would catch the player if they ended a timestep in the 'line of sight' of an enemy while not being camouflaged. In this game there were two types of enemies. The first enemy type patrolled an area by walking back and forth. These enemies were represented by red triangles. The second enemy type remained stationary at a given location and are depicted as orange triangles.

The player had three possible options in order to navigate passed enemies. The first method was with the use of camouflage. Within the level there was a camouflage package represented by a yellow square and if the player moved to that location, they put on a disguise. This is shown to the player by their character triangle changing from blue to yellow. Once in this state, NPCs no longer could catch the player when in 'line of sight'. The second option was to 'takedown' the enemy. The player could achieve this by landing on the same square as the enemy, without being caught on any previous step. In these cases, the enemy was removed from the game. The third option was to simply attempt to navigate passed the enemy without being caught.

Further, a number of visual choices were made with links back to 'stealth' game

genre. Firstly, brown squares could be inserted within a room as physical objects, where NPCs and the player could not pass through these objects. The aim of these was to emulate objects within a room that the player could hide behind. The second was to set a square to a light grey colour. This aimed to represent a vent system, as these were commonly used in 'stealth' games in order to allow the player to sneak around NPCs.

The next step was to define a set of playstyles for the designed game. The 'stealth' game genre was chosen when initially designing the game as previous literature had explored valid playstyles in this game genre [101]. From this literature, three playstyles were chosen. These are referred to as 'Camouflage', 'Takedown' and 'Stealth'. Each of these can be categorised based on the combination of two features characterising how the player plays the game, namely 'Number of Takedowns Performed' and the 'Use of Camouflage'. An additional tracked feature monitored if the player reached the goal square. This was because, although the playstyle used was key, the player should also be able to complete the level. The specific feature values for each playstyle can be seen in Table 7.

Table 7: Playstyle Definition via Game Variables

Playstyle	Camouflage	Takedowns	Completed
Camouflage	1.00	0.00	1.00
Takedown	0.00	2.00	1.00
Stealth	0.00	0.00	1.00

Below, each of these playstyles will be described, as well as a description of how the player with that playstyle would approach completing the base level of the designed game.

The first playstyle considered was called 'Camouflage' and was based on the 'Role-play' style in the literature. The 'Roleplay' style uses disguises to blend into the environment and gain access to restricted areas. Within this designed game, this would translate to the player acquiring the yellow camouflage package and then sneaking passed both enemies.

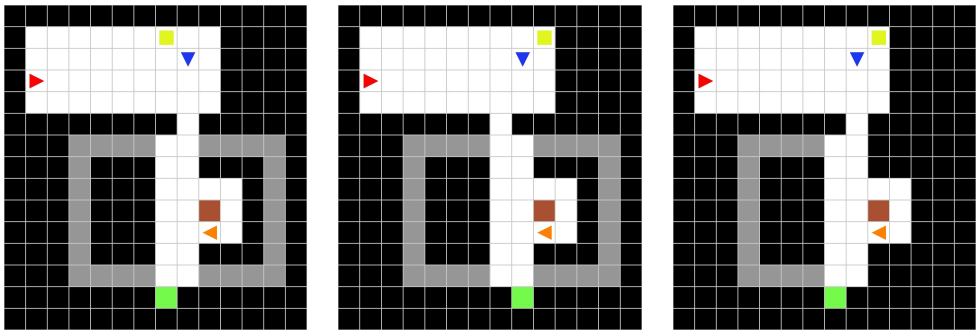
The second playstyle called 'Takedown' was based on the 'Brawn' playstyle. In the literature the 'Brawn' playstyle aimed to show off the character's physical attributes. This commonly takes the form of climbing, the use of close quarters combat, *etc.* In the designed game, this playstyle resulted in the player preferring taking down enemies in order to get passed them.

The final playstyle was called 'Stealth' which was based on the similarly named style in the literature. This was mainly defined by the player always staying in the trademark black suit. This meant that the player does not use disguises and hence must rely on the use of stealth to reach the required restricted areas. Within the designed game this meant that the player would not use the yellow camouflage package, in addition it was stipulated that the player should not takedown any enemies.

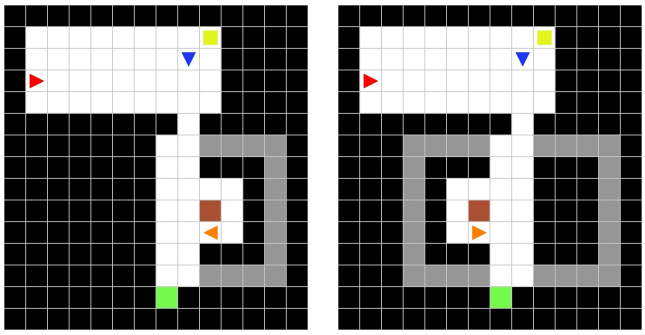
To generate data for each of these three playstyles discussed above, standard RL techniques were used on a custom reward function. It was found that each of the playstyles could be produced by giving rewards based on the following events: acquiring the camouflage package, taking down enemies, the use of vents, reaching the goal

square and being caught.

In addition to the previously introduced base level, several level variants were also created. These level variants can be seen in Figure 89.



(a) Move Camouflage (b) Move Exits (c) Left Vent Only



(d) Right vent Only (e) Flip Room

Figure 89: Hitman Level Variants

Each of these were designed to challenge certain aspects of each playstyle. For example, moving the camouflage package, so that the 'Camouflage' playstyle needed to identify where it was and move to it instead of just repeating an action sequence that would normally result in finding the camouflage package.

Following the game design, playstyles being defined, and gameplays collected, the next step was to train the encoders. As previously mentioned, some game specific parameters needed to be chosen and are shown in Table 8. In addition, the average time to train each encoder type can be seen in Table 9. It was noted that the required training times for this game were substantially lower than the training times for the other two games considered. This was to be expected due to the smaller image size of the gameplays, which allowed faster computation by the network.

After encoder training had concluded, the loss graphs were examined to check for successful optimisation. Firstly, the STDIM-VAE loss was investigated, and this is shown in Figure 90. Again, throughout training, all the runs appeared to converge to a similar value, of approximately 0.296.

Next, comparisons were drawn between the optimisation of the components of the STDIM-VAE loss and optimisation of an encoder that only contained a single component *i.e.* either the ST-DIM or VAE elements. From the two components, the ST-DIM component was first explored, and the loss graphs are shown in Figure 91a. As can

Table 8: Gameplay Recording Settings for Hitman

Setting	Value
Image Width	150
Image Height	150
Down Sampling	None
Frame Limit	None

Table 9: Comparison of Average Encoder Training Times for Hitman

Encoder	Training Time (m) $\pm$ SD
VAE	0.31 $\pm$ 0.01
ST-DIM	0.75 $\pm$ 0.01
STDIM-VAE	0.82 $\pm$ 0.01

be seen, even within the STDIM-VAE architecture, the ST-DIM component was still optimised to a similar level to that seen for the ST-DIM architecture.

The VAE component was then considered, and the loss graphs are shown in Figure 91b. It was noted that the overall shape of these two loss graphs were broadly similar. It can be seen that there was an initial fairly quick reduction in the loss value before it plateaued out, followed by a further fairly quick decrease which then levelled out one final time. However, the number of epochs to get over this plateau was slightly higher in the case of the STDIM-VAE. Further, the convergence point, appeared to be slightly higher for the STDIM-VAE. Both of these observations are in keeping with the fact that the STDIM-VAE network was attempting to balance the optimisation of both the ST-DIM and VAE components.

Once an encoder of each type had been trained, analysis was conducted to explore any differences in what each encoder had encoded. This was achieved by applying the RISER method to each encoder for key frames using an example of each playstyle.

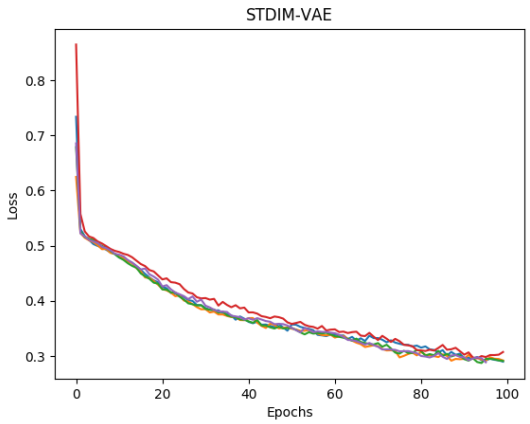
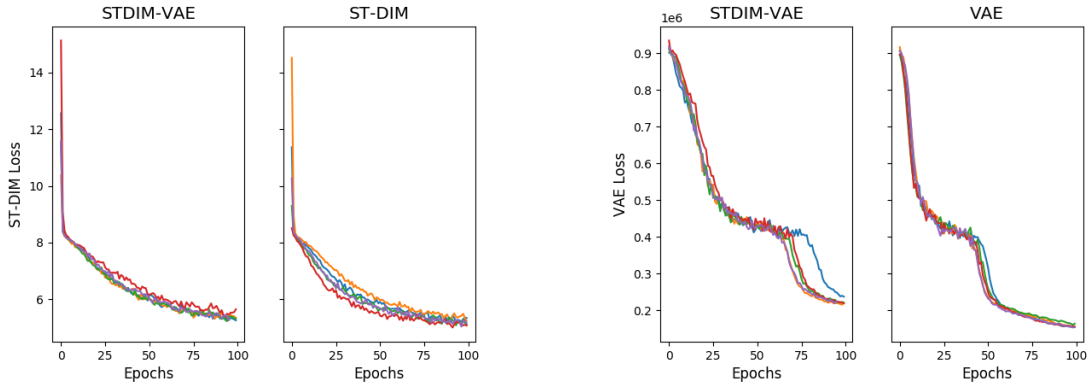


Figure 90: STDIM-VAE Training Loss for Hitman



(a) ST-DIM Loss Component

(b) VAE Loss Component

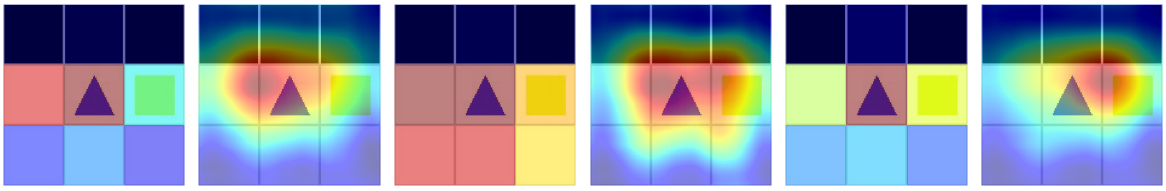
Figure 91: Comparison of Loss Components for the Hybrid and Individual Encoders for Hitman

The ‘Single Block’ mask type was applied to gain an understanding of which portions of the frame were of interest to the encoder. Then the ‘Literature Based’ mask type allowed a more refined understanding of the parts of the asset in the portion of the frame that was of interest to the encoder, as it was potentially thought possible to extract a full understanding of a given asset from only a fraction of it.

The analysed frames were first taken from a gameplay of the ‘Camouflage’ playstyle. The first of these frames was taken before the camouflage had been acquired but was in view of the player and the importance maps for the ‘Single Block’ and ‘Literature Based’ mask types are shown in Figure 92. For the ‘Single Block’ mask type, it can be seen that in all the encoder types there was a focus on the centre, where the player was located, but this expanded out differently depending on the encoder type. For the ST-DIM architecture, the focus only expanded to the middle left side of the frame. Whereas for the VAE, the main focus expanded to the middle left, bottom left and bottom middle of the frame. On top of this a secondary focus appeared on the middle right and bottom right of the frame. For the STDIM-VAE the main focus was only contained in the centre, with a secondary focus on the middle right and a tertiary focus on the middle left of the frame.

The ‘Literature Based’ mask type was then used to further explore the foci. For the ST-DIM architecture, the focus centre was on a point between the centre and middle left squares and hence appeared to be more interested in the player rather than the camouflage package. Whereas the VAE focus expanded outwards from the centre to cover part of each asset square. On the other hand, for the STDIM-VAE the focus was between the centre and middle right squares of the frame and hence there appeared to be interest in both the player and the camouflage package.

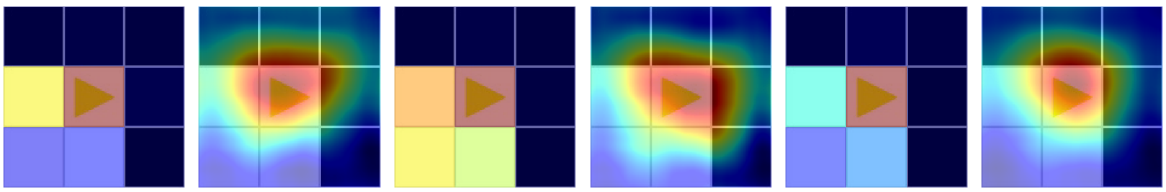
The next frame considered occurred after the player had acquired the camouflage. The importance maps for each mask type can be seen in Figure 93. In this case, for all encoders, the main focus was on the player, given the player’s icon colour had changed this logically made sense. Interestingly, when looking at the ST-DIM importance map, a secondary focus was seen on the middle left square of the frame. This combined with the previous focus discussed, suggested that the ST-DIM may have a regularly occurring



(a) ST-DIM 'SB' Masks | (b) ST-DIM 'LB' Masks | (c) VAE 'SB' Masks | (d) VAE 'LB' Masks | (e) STDIM-VAE 'SB' Masks | (f) STDIM-VAE 'LB' Masks

Figure 92: 'Camouflage' Run 1, Frame 1 Importance Maps

interest to the left of the player. In the case of the VAE, a secondary focus was seen on the middle left, bottom left and bottom middle squares of the frame. Whilst this was not exactly the same as the last frame, it covered all squares with assets that were not walls. Given the VAE was trained to replicate the frame image from the representation, it appeared that by default a region was considered as a wall and hence focus was only required on assets that were not as such. It was noted that when considering the STDIM-VAE architecture only the focus on the player was prominent. Within the focus of the 'Literature Based' mask type, this related to a tighter focus region around the player compared to the other two encoder types.



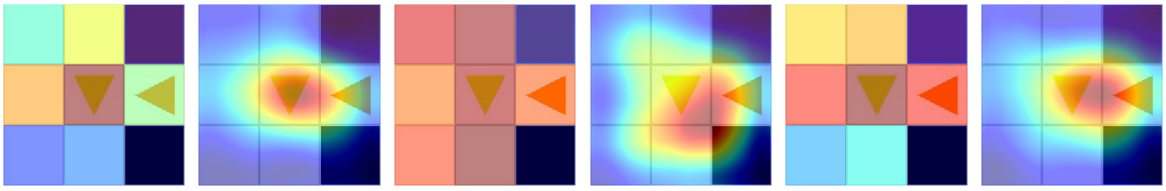
(a) ST-DIM 'SB' Masks | (b) ST-DIM 'LB' Masks | (c) VAE 'SB' Masks | (d) VAE 'LB' Masks | (e) STDIM-VAE 'SB' Masks | (f) STDIM-VAE 'LB' Masks

Figure 93: 'Camouflage' Run 1, Frame 2 Importance Maps

The final frame taken from the 'Camouflage' playstyle gameplay showed the player sneaking past the second NPC. The importance maps for this frame are shown in Figure 94. It can be seen that the ST-DIM architecture mainly focused on the player but also had some focus on the squares to the left and above the player. Whereas the VAE focus was on all assets that were not walls or a box. The STDIM-VAE, on the other hand, had a focus region horizontally across the middle of the frame. Based on the 'Literature Based' mask type analysis, it can be seen that the ST-DIM focused more on the player. Whereas the VAE was also not only interested in the NPC but in addition focused on other parts of the frame. While the STDIM-VAE focused more tightly between the player and the NPC.

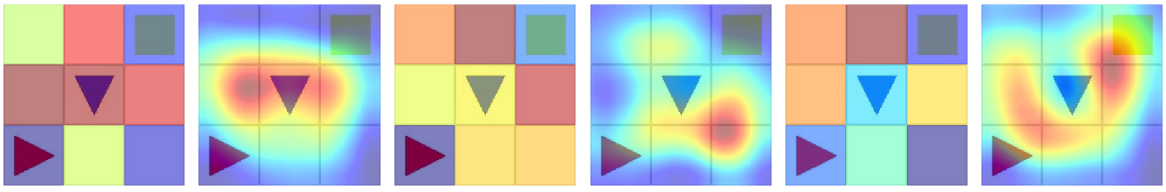
Following this, analysis was conducted on the frames from a gameplay of the 'Takedown' playstyle. The two frames chosen showed the player lining up to perform a takedown on an NPC. Interestingly, before the taking down of the first NPC, none of the encoders had a strong focus on the NPC, as shown in the importance maps in Figure 95.

However, when lining up to perform a takedown on the second NPC, all the encoder



(a) ST-DIM 'SB' Masks | (b) ST-DIM 'LB' Masks | (c) VAE 'SB' Masks | (d) VAE 'LB' Masks | (e) STDIM-VAE 'SB' Masks | (f) STDIM-VAE 'LB' Masks

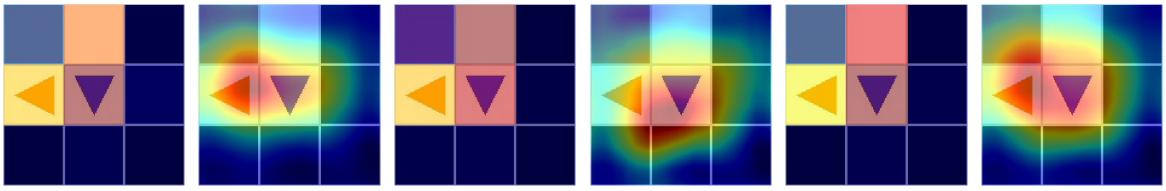
Figure 94: 'Camouflage' Run 1, Frame 12 Importance Maps



(a) ST-DIM 'SB' Masks | (b) ST-DIM 'LB' Masks | (c) VAE 'SB' Masks | (d) VAE 'LB' Masks | (e) STDIM-VAE 'SB' Masks | (f) STDIM-VAE 'LB' Masks

Figure 95: 'Takedown' Run 1, Frame 6 Importance Maps

types had a focus on this NPC, as shown in Figure 96. Although their main focus was still on the player, the focus region also expanded upwards to the square above.

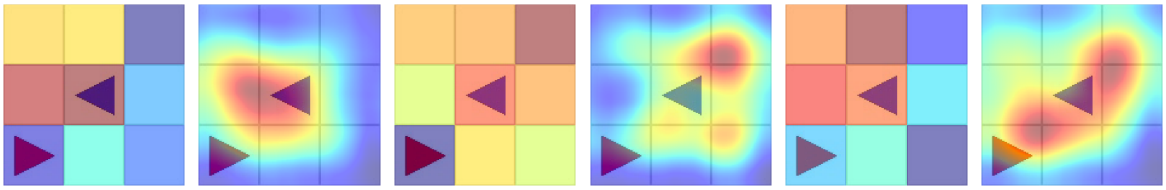


(a) ST-DIM 'SB' Masks | (b) ST-DIM 'LB' Masks | (c) VAE 'SB' Masks | (d) VAE 'LB' Masks | (e) STDIM-VAE 'SB' Masks | (f) STDIM-VAE 'LB' Masks

Figure 96: 'Takedown' Run 1, Frame 18 Importance Maps

Subsequently, frames taken from a 'Stealth' playstyle gameplay were explored. In the first frame, the player was waiting for the NPC to pass, so that they could themselves sneak passed. Once again, all the encoders appeared to have minimal interest in the NPC, as shown in Figure 97. Given that visually, apart from the player orientation and the camouflage package being in view, the frame appeared very similar to what had been seen previously when the player was lining up to perform a takedown of the first NPC in the 'Takedown' playstyle. The similarity in this lack of interest makes consistent sense.

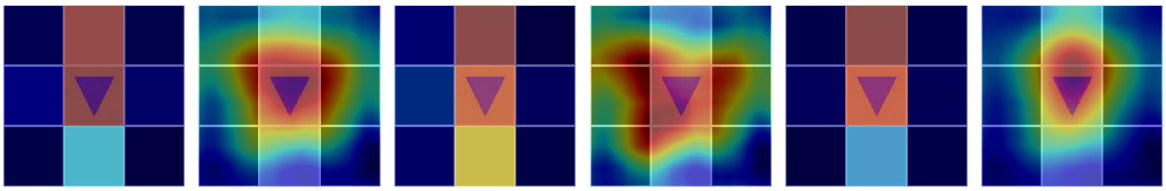
The final frame considered shows the player moving through a vent and the importance maps are shown in Figure 98. In all cases the main focus was on the player and the square above the player. Interestingly for the VAE architecture, the bottom middle square was also an area of focus, this meant again that all regions with assets that were not walls nor boxes were areas of focus for this encoder type. Further, when foci from



(a) ST-DIM 'SB' Masks | (b) ST-DIM 'LB' Masks | (c) VAE 'SB' Masks | (d) VAE 'LB' Masks | (e) STDIM-VAE 'SB' Masks | (f) STDIM-VAE 'LB' Masks

Figure 97: 'Stealth' Run 1, Frame 3 Importance Maps

the 'Literature Based' mask type were explored, it was seen that for the STDIM-VAE, that the focus region appeared tighter around the player than the other two encoder types.



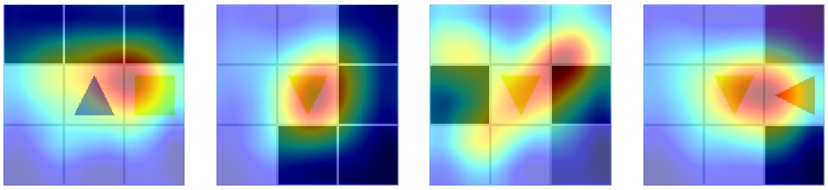
(a) ST-DIM 'SB' Masks | (b) ST-DIM 'LB' Masks | (c) VAE 'SB' Masks | (d) VAE 'LB' Masks | (e) STDIM-VAE 'SB' Masks | (f) STDIM-VAE 'LB' Masks

Figure 98: 'Stealth' Run 1, Frame 22 Importance Maps

Overall, based on the above results, it appeared that the ST-DIM architecture was more focused on the player than the NPCs or camouflage package in many situations. Whereas the VAE focus was regularly interested in any regions with an asset that were not walls or boxes and hence often had focus regions on the NPCs and the camouflage package, as well as the player. The STDIM-VAE, while not having a wide focus across large parts of the frame, regularly had focus regions on the player, NPCs and the camouflage package. This in itself showed promise, as these elements allowed a good understanding about which playstyle was being demonstrated in a given gameplay.

The next step was to conduct further analysis on the STDIM-VAE using the 'Literature Based' mask type. In the following section, example importance maps will be shown for each of the playstyles. Firstly, a gameplay demonstrating the 'Camouflage' playstyle was examined and a sample of the frames with their importance maps overlaid are shown in Figure 99.

It can be seen in Figure 99a, prior to acquiring the camouflage package, the encoder focus appears to be split between the player and the camouflage package. Given one of the defining factors of the playstyle, was the use of camouflage, this is encouraging. Examining a frame later in the gameplay, for example Figure 99b, the focus was seen to be heavily on the player. In this case the player was camouflaged, hence shaded yellow in the frame, so this focus would allow the determination of the player's current camouflage status. It was further noted that this focus on the player was also seen when the player did not have camouflage. In both cases this focus should be useful in order to determine the playstyle. Looking even later in the gameplay, when the player was

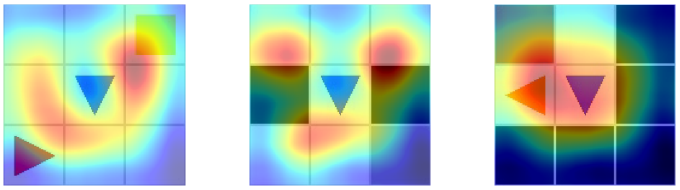


(a) Frame 1 (b) Frame 5 (c) Frame 7 (d) Frame 12

Figure 99: ‘Camouflage’ Run 1, Importance Maps using ‘LB’ Masks

moving between rooms, as shown in Figure 99c, it can be seen that the focus spreads out to the rest of the frame. This is important to ensure the representation did not just contain information about the player but also the player’s surroundings. This is shown further in Figure 99d, where it can be seen that the encoder was also interested in the position of NPCs in the game level.

After this a gameplay using the ‘Takedown’ playstyle was explored. Similarly, to before, a sample of the frames and importance maps are shown in Figure 100.



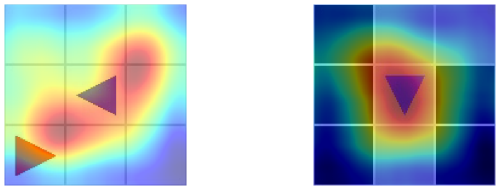
(a) Frame 6 (b) Frame 9 (c) Frame 18

Figure 100: ‘Takedown’ Run 1, Importance Maps using ‘LB’ Masks

Figure 100a further shows the encoder’s interest in the player’s surroundings, which in this case included both the NPCs and camouflage. This meant that it appeared that the existence of the camouflage package was encoded, not just if it had been acquired. Next a frame showing the player moving between rooms was considered, as seen in Figure 100b. Similarly, to what had been seen for the camouflage gameplay, the encoder seemed to spread its focus to the rest of the room. However, in this case, it was noted that the focus on the player was greatly reduced. This was potentially caused by the fact that the player was not camouflaged and hence less interesting to the encoder. The last frame considered in relation to this gameplay is shown in Figure 100c. Once again, when the player approaches an NPC, albeit from a different direction, the encoder focuses on the position of both the player and the NPC.

Following this, a gameplay of the stealth playstyle was examined, and samples of the frames and importance maps are shown in Figure 101.

As can be seen in Figure 101a, the encoder focused on the player and NPC within the frame. It can be seen in Figure 101b, that when the NPC was no longer present, the focus was purely on the player. This was beneficial for two reasons. Firstly, this focus would most likely result in the encoding of the colour of the player icon. This was useful as it could provide a visual representation as to whether the camouflage package had been acquired. Secondly, the background behind the player could provide information about where the player was within the environment. In this case, the background behind

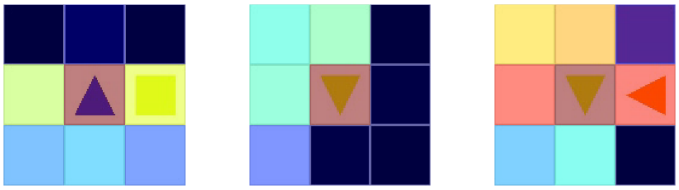


(a) Frame 3 (b) Frame 20

Figure 101: ‘Stealth’ Run 1, Importance Maps using ‘LB’ Masks

the player was grey and hence shows that the player was in a vent. This was again expected to be useful information to allow the separation of playstyles.

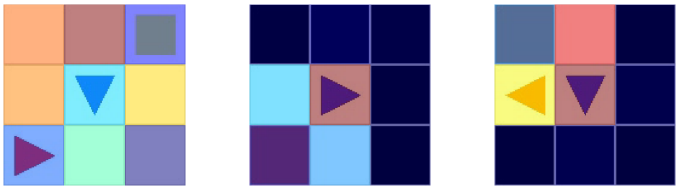
Finally, the ‘Single Block’ mask type was investigated. Similarly, to before, firstly a ‘Camouflage’ playstyle gameplay was examined. It can be seen that there was commonly a focus on the player throughout all of the three frames shown in Figure 102. Although, when the camouflage package or NPC was also in view, there was often also focus on these object/characters, as shown in Figures 102a and 102c respectively. However, in both these cases there also appeared to be a focus to the left of the player as well.



(a) Frame 1 (b) Frame 5 (c) Frame 12

Figure 102: ‘Camouflage’ Run 1, Importance Maps using ‘SB’ Masks

Next a ‘Takedown’ gameplay was considered. Although, as can be seen in Figure 103a, the focus was not always on the camouflage package or NPC, even when they were in view. Further, the main focus was not on the player. Although later in the gameplay focus returned to the player as can be seen in Figure 103b. Additionally, a secondary focus was seen on the second NPC when it came into view, as shown in Figure 103c, but the main focus was still on the player and the area directly above the player.

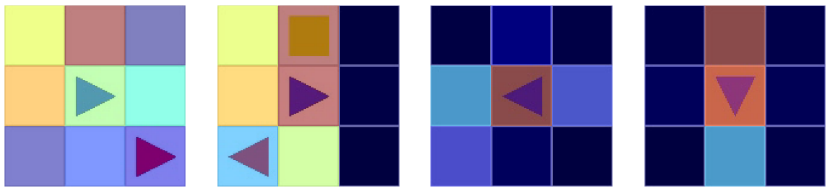


(a) Frame 6 (b) Frame 15 (c) Frame 18

Figure 103: ‘Takedown’ Run 1, Importance Maps using ‘SB’ Masks

Finally, a ‘Stealth’ playstyle gameplay was investigated. In this case focus was

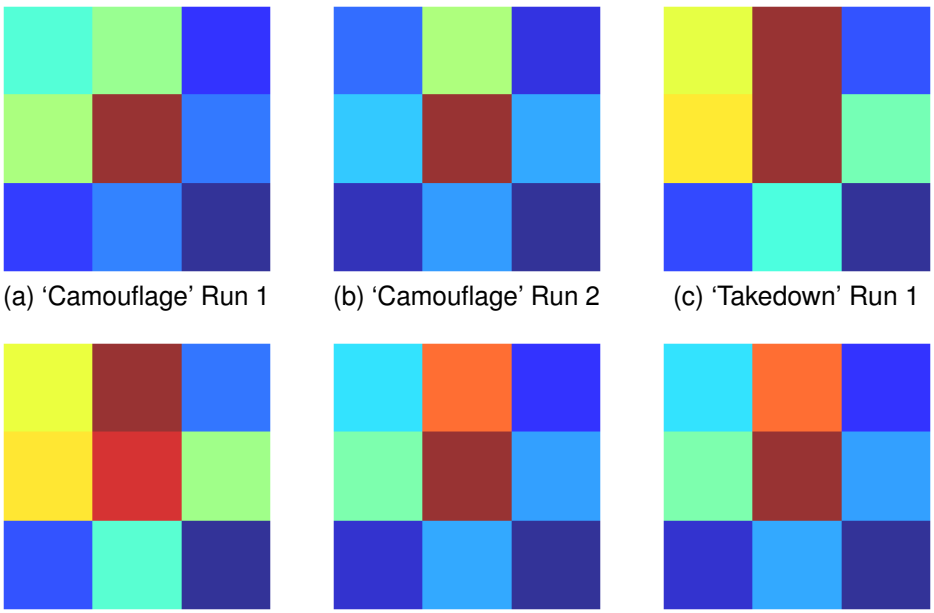
occasionally seen above the player as shown in Figure 104a. Later in the gameplay, the focus fell on the camouflage package, although this happened after the camouflage was directly above the player on screen, which is shown in Figure 104b. Even later in the gameplay, once the player had entered the vent, focus switched to either just the player or the player and above the player as shown in Figures 104c and 104d.



(a) Frame 7 (b) Frame 9 (c) Frame 18 (d) Frame 22

Figure 104: 'Stealth' Run 1, Importance Maps using 'SB' Masks

Following this, a focus 'fingerprint' was calculated for each gameplay by combining the foci from each frame. Initially, this was done using the 'Single Block' mask type and the resulting focus 'fingerprints' are shown in Figure 105 for two gameplays of each playstyle. Firstly, the two gameplays of the 'Camouflage' playstyle were considered. In these two 'fingerprints' the main focus was on the centre of the frame. However, there was some variability in the position of the secondary foci. In both cases, one of these region(s) was on the middle top of the frame, although in one of these examples there was a focus on the middle left as well.



(a) 'Camouflage' Run 1 (b) 'Camouflage' Run 2 (c) 'Takedown' Run 1

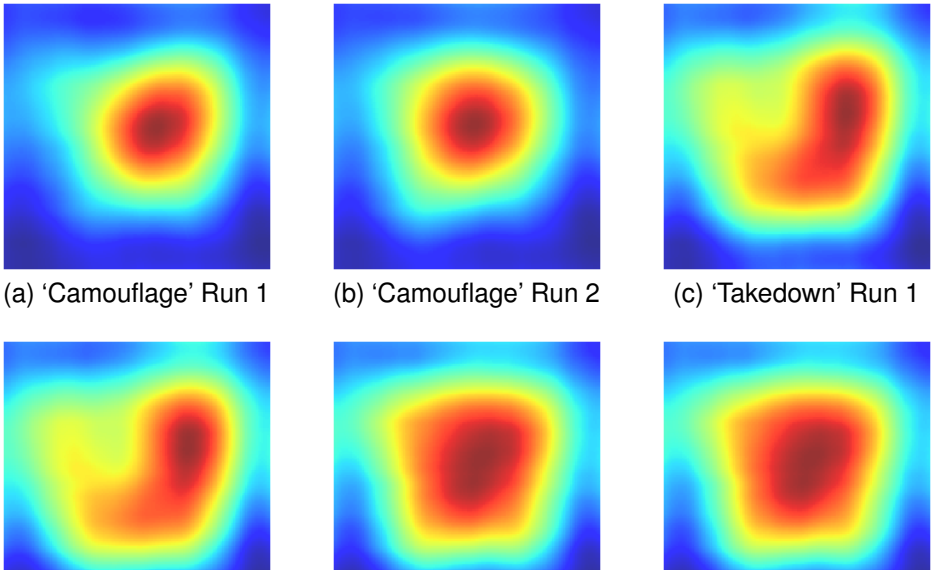
(d) 'Takedown' Run 2 (e) 'Stealth' Run 1 (f) 'Stealth' Run 2

Figure 105: Comparison of Focus 'Fingerprints' across Playstyles using 'SB' Masks

Next, the two 'fingerprints' for gameplays of the 'Takedown' playstyle were investigated. In this case both focus 'fingerprints' showed the same pattern, namely that the

main focus was on the centre and top middle of the frame. In addition, two secondary focus areas were positioned on the middle left and top left of the ‘fingerprints’. Finally, the ‘Stealth’ playstyle gameplays ‘fingerprints’ were explored. As can be seen, in both ‘fingerprints’, the main focus areas were on the centre as well as the top middle of the ‘fingerprints’.

Further, focus ‘fingerprints’ were also generated with the ‘Literature Based’ mask type. With this mask type, common patterns were also seen and are shown in Figure 106. For the ‘Camouflage’ playstyle, these ‘fingerprints’ contained a circular focus region centred around a point close to the middle of the frame. Potentially, for this playstyle the main focus was on the player, due to the change in the colour of the player icon, as in this playstyle camouflage was being used. Whereas, with the ’Takedown’ playstyle the ‘fingerprints’, each looked like a backwards ’L’. Potentially, this focus ‘fingerprint’ was caused by the combination of the focus being between the player and camouflage package in the initial frames with the focus moving to the player later in the gameplay. Finally, when considering the ‘Stealth’ playstyle ‘fingerprints’, these had a more rectangular shaped focus in the centre of the frame. It is suggested that this was due to focus being on the background of the player’s location, as when the player was in a vent the background was a light grey colour.



(a) ‘Camouflage’ Run 1 (b) ‘Camouflage’ Run 2 (c) ‘Takedown’ Run 1

(d) ‘Takedown’ Run 2 (e) ‘Stealth’ Run 1 (f) ‘Stealth’ Run 2

Figure 106: Comparison of Focus ‘Fingerprints’ across Playstyles using ‘LB’ Masks

Overall, in each case the ‘fingerprints’ shared common focus regions across gameplays of the same playstyle. This suggested that information was being encoded which was playstyle dependent and hence should allow the playstyles to be split based on their learnt representations.

Following this, analysis comparing different gameplays was conducted. Three pairs of gameplays were chosen to compare gameplay examples of each playstyle. In all cases, early in the gameplay, an equal focus of 1 was seen over the whole frame.

An example from each gameplay pair is shown in Figures 107, 108 and 109. On examination, this showed that the matched frames were perfect replications of one another.

(a) Frame 5 (b) Importance Map (c) Frame 5 (Match 1)

Figure 107: Comparison of Runs of ‘Camouflage’ at Frame 5 using ‘SB’ Masks

(a) Frame 2 (b) Importance Map (c) Frame 2 (Match 1)

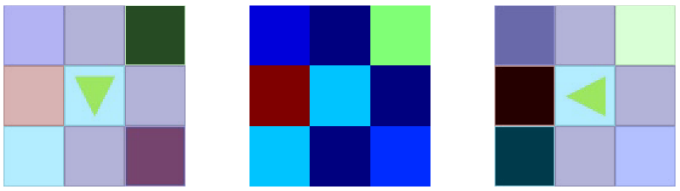
Figure 108: Comparison of Runs of ‘Takedown’ at Frame 2 using ‘SB’ Masks

(a) Frame 4 (b) Importance Map (c) Frame 4 (Match 1)

Figure 109: Comparison of Runs of ‘Stealth’ at Frame 4 using ‘SB’ Masks

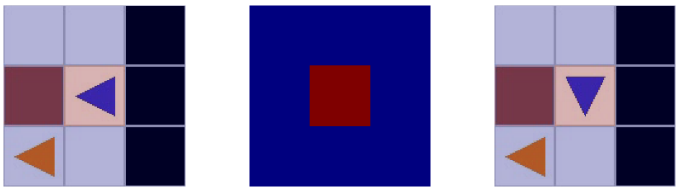
However later in the gameplay, for two out of three of these comparisons, the method choosing the action caused an action deviation in the gameplay. This caused an uneven focus over the frame, with a focus on the left of the player for the ‘Camouflage’ pair and on the player for the ‘Takedown’ pair. These importance maps are shown in Figure 110 and Figure 111. Based on the matched frame for the ‘Camouflage’ pair, it appeared that the focus to the left of the player was likely caused by a wall that would not normally be seen at this point. Although, walls are also seen or not seen in other locations but in these cases a focus area was not observed and hence the mere existence of the walls was not deemed important on its own. It is suggested that it may be the position of the walls that are important in whether a focus region was created. Further for the ‘Takedown’ pair, based on the matched frame shown, the only difference was the player

orientation. Presumably, this was the reason for the focus on the player and hence suggested that the orientation of the player had been encoded.



(a) Frame 10 (b) Importance Map (c) Frame 10 (Match 1)

Figure 110: Comparison of Runs of 'Camouflage' at Frame 10 using 'SB' Masks



(a) Frame 17 (b) Importance Map (c) Frame 15 (Match 1)

Figure 111: Comparison of Runs of 'Takedown' at Frame 17 using 'SB' Masks

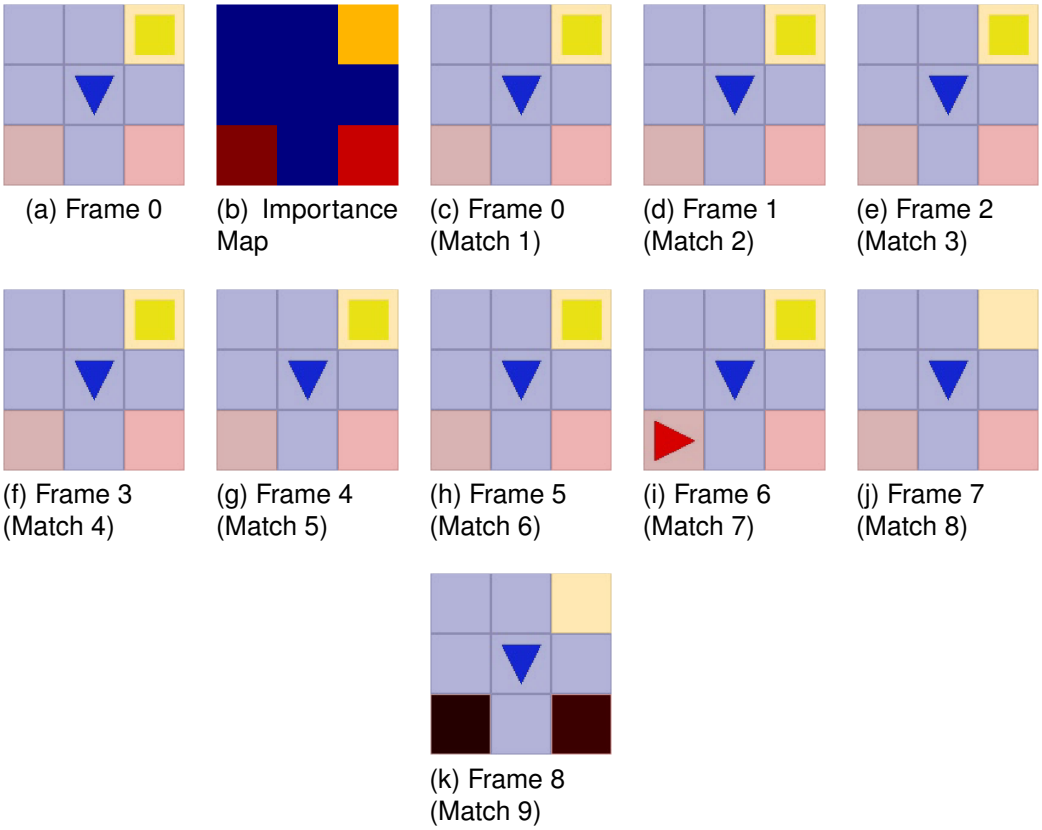
It should be noted that when considering frames later in the gameplay for the 'Stealth' playstyle pair, an equal focus of 1 was seen over the frame. An example of this is shown in Figure 112. This was because no action choice deviation was seen between these two gameplays and hence the matched frame was a perfect replica of the frame considered.

(a) Frame 28 (b) Importance Map (c) Frame 28 (Match 1)

Figure 112: Comparison of Runs of 'Stealth' at Frame 28 using 'SB' Masks

Following this, comparisons were made between gameplays demonstrating different playstyles. The first of these pairs compared the 'Camouflage' and 'Takedown' playstyles. On the first frame, as shown in Figure 113, the main focus was on the bottom left and right corners of the frame. This information can potentially help identify where the player was in the game level (*e.g.* when moving through a 'door' or 'vent', as these regions of the frame would be black). Further, a secondary focus region was seen on the camouflage package, which was a key item for determining the playstyle used. When the matched frames were explored, the reasons for each of the focus regions became

clear. Firstly, for the bottom corner foci, in one of the matched frames these were indeed walls. Further, in another matched frame, an NPC was present in the bottom left and hence this was potentially why the bottom left of the frame was deemed slightly more important than the bottom right. As for the focus on the camouflage package in the top right corner of the frame, this can be explained by the fact that in two of the matched frames the camouflage package was not present in this location.



(a) Frame 0 (b) Importance Map (c) Frame 0 (Match 1) (d) Frame 1 (Match 2) (e) Frame 2 (Match 3)

(f) Frame 3 (Match 4) (g) Frame 4 (Match 5) (h) Frame 5 (Match 6) (i) Frame 6 (Match 7) (j) Frame 7 (Match 8)

(k) Frame 8 (Match 9)

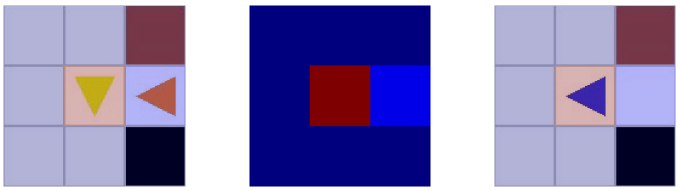
Figure 113: Comparison of ‘Camouflage’ and ‘Takedown’ Runs at Frame 0 using ‘SB’ Masks

Later in the gameplay sequence, after the camouflage had been acquired, the focus was regularly on the player, as shown in Figure 114. This was often the case even when an NPC was visible, as shown in Figure 115. This player based focus could be useful as the player’s camouflage state could be extracted based from the player icon colour.

Next, comparisons were drawn between gameplays showing the ‘Camouflage’ and ‘Stealth’ playstyles. Once again, at multiple times in the gameplay, the focus was on the player. This could happen prior to the camouflage package being acquired and an example of an importance map is shown in Figure 116. When examining the matched frames, this focus appeared to be caused by different player orientations seen in the matched frames, this also implies that player orientation is being encoded. Further, in the importance map a small amount of interest was seen at the top right and bottom right corners of the frame. The top right focus appeared to be caused by the fact that regularly the camouflage package was not seen at this location in the matched frames. Whereas with regard to the bottom right focus, in the matched frames an NPC was often

(a) Frame 9 (b) Importance Map (c) Frame 17 (Match 1)

Figure 114: Comparison of ‘Camouflage’ and ‘Takedown’ Runs at Frame 9 using ‘SB’ Masks



(a) Frame 12 (b) Importance Map (c) Frame 20 (Match 1)

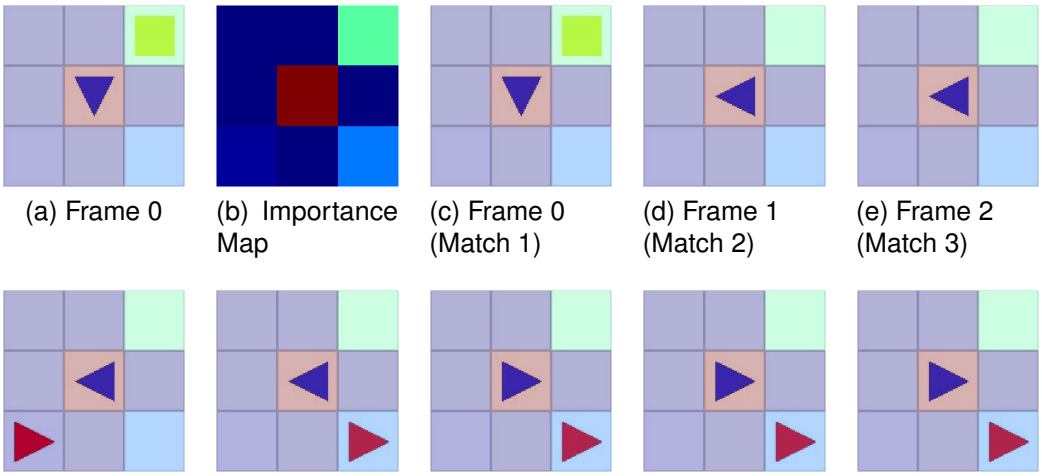
Figure 115: Comparison of ‘Camouflage’ and ‘Takedown’ Runs at Frame 12 using ‘SB’ Masks

seen at that location.

A further importance map demonstrating player focus can be seen in Figure 117. Although in this case, based on the matched frame, it was clear that the reason for this focus was the colour change in the player icon, caused by the use of camouflage, as the rest of the frame was the same.

Following this, gameplays demonstrating the ‘Takedown’ and ‘Stealth’ playstyles were examined. Early on, it was seen that an equal focus of 1 was observed over the whole frame, as shown in Figure 118. This was because in this case, the player was waiting to takedown the NPC and had not moved, hence a perfect match to the initial frame was possible which resulted in the observed importance map.

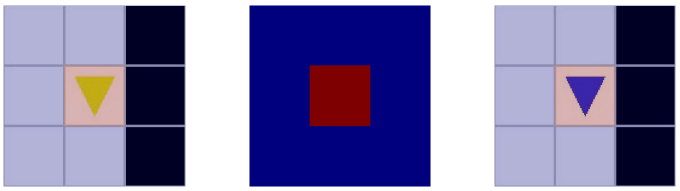
Later in the gameplay, after the first NPC came into view, a focus region appeared on the NPC as shown in Figure 119. When the matched frame was examined, it was seen that a match had been made to the initial frame from the ‘Stealth’ gameplay and hence the only visual difference was the presence of the NPC.



(a) Frame 0 (b) Importance Map (c) Frame 0 (Match 1) (d) Frame 1 (Match 2) (e) Frame 2 (Match 3)

(f) Frame 3 (Match 4) (g) Frame 4 (Match 5) (h) Frame 5 (Match 6) (i) Frame 6 (Match 7) (j) Frame 7 (Match 8)

Figure 116: Comparison of ‘Camouflage’ and ‘Stealth’ Runs at Frame 0 using ‘SB’ Masks

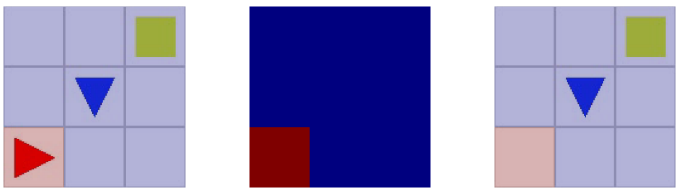


(a) Frame 3 (b) Importance Map (c) Frame 10 (Match 1)

Figure 117: Comparison of ‘Camouflage’ and ‘Stealth’ Runs at Frame 3 using ‘SB’ Masks

(a) Frame 4 (b) Importance Map (c) Frame 0 (Match 1)

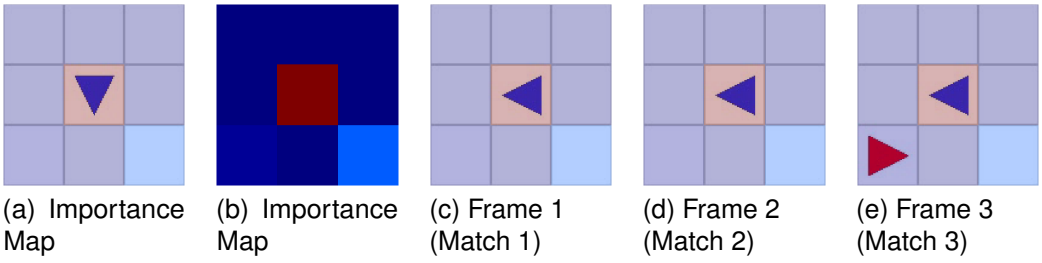
Figure 118: Comparison of ‘Takedown’ and ‘Stealth’ Runs at Frame 4 using ‘SB’ Masks



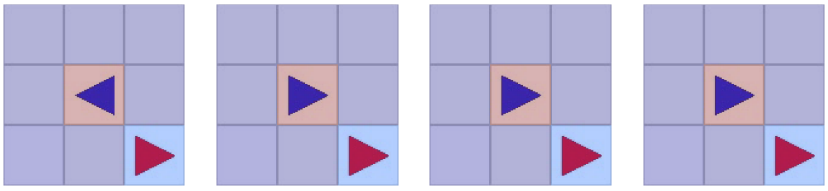
(a) Frame 6 (b) Importance Map (c) Frame 0 (Match 1)

Figure 119: Comparison of ‘Takedown’ and ‘Stealth’ Runs at Frame 6 using ‘SB’ Masks

However, once the NPC had been takedown and hence not in sight, the main focus switched back to the player with a small interest in the bottom right of the frame, as seen in Figure 120. Based on the matched frames, it can be seen that the player based focus appeared to originate from the different player orientation seen in the matched frames. Further, the small interest region in the bottom right corner of the frame appears to be caused by the common presence of the NPC in that location in the matched frames.



(a) Importance Map (b) Importance Map (c) Frame 1 (Match 1) (d) Frame 2 (Match 2) (e) Frame 3 (Match 3)



(f) Frame 4 (Match 4) (g) Frame 5 (Match 5) (h) Frame 6 (Match 6) (i) Frame 7 (Match 7)

Figure 120: Comparison of ‘Takedown’ and ‘Stealth’ Runs at Frame 7 using ‘SB’ Masks

Finally, as the player approached the second NPC, in order to perform the takedown, a focus region appeared on the NPC. This is shown in Figure 121. Although, this NPC focus may be amplified based on it contrasting with what was seen in the region in the matched frame (*i.e.* black wall). This was further supported by the fact that the other visual changes, for example a brown box being switched for a black wall or a black wall being switched with a grey vent, while having some focus, were potentially of less interest due to the lower contrast.

(a) Frame 18 (b) Importance Map (c) Frame 23 (Match 1)

Figure 121: Comparison of ‘Takedown’ and ‘Stealth’ Runs at Frame 18 using ‘SB’ Masks

Finally, the encoded gameplays were projected onto a 2D plane. One projection example is shown in Figure 122 and the full set is shown in Appendix A.3.1. From these, it can be seen that a set of playstyle gameplays were projected far from the other two sets. When investigated, it was found that this was the ‘Camouflage’ playstyle set

that was some distance from the other two sets. Logically, this makes sense as the use of camouflage changed the colour of the player icon and hence caused a large visual change throughout the gameplay. When considering the other two sets, they both avoided the use of camouflage and hence it would be expected that they would appear to be more similar. When further examined, there still appeared to be a clear split between these latter two playstyles. Finally, the raw distance matrix was examined. Here it was found that there was still a noticeable difference in the distance between gameplays with the same playstyle and the distance between gameplays of different playstyles. However, this difference was larger when considering the distance to a 'Camouflage' playstyle gameplay. This most likely explained the result in the projection space.

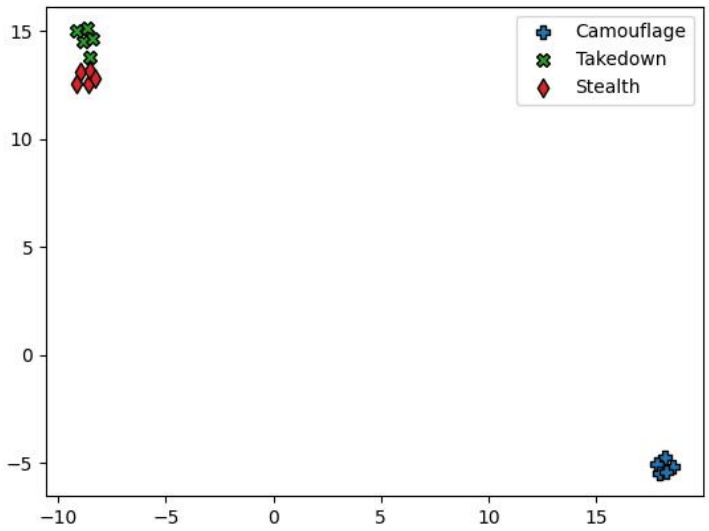


Figure 122: Projection 1 of Hitman Gameplays using the STDIM-VAE

Finally, the two representation evaluation techniques were used to quantify the quality of the representation space. When considering the group average linkage between playstyles, shown in Figure 123, a notably lower value was seen when comparing a playstyle to itself than when comparing it to different playstyles. This shows that videos of the same playstyle were being encoded close together and far away from videos of other playstyles.

Then when comparing the average silhouette score, although the combination of STDIM-VAE and cosine metric produces a high score, it is slightly lower than ST-DIM and the cosine metric. This can be seen in Table 10.

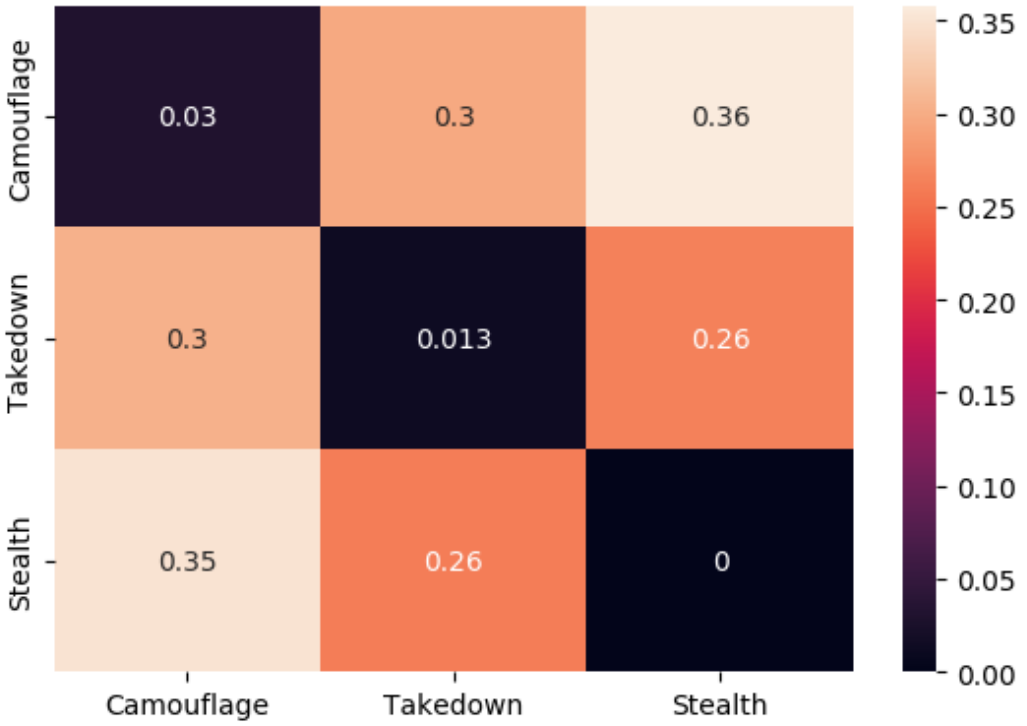


Figure 123: Group Average Linkage Between Playstyles for Hitman

Table 10: Average Silhouette Scores for Hitman (± SD)

		Distance Metric		
		Manhattan	Euclidean	Cosine
Encoder	VAE	0.903 (0.001)	0.903 (0.001)	0.899 (0.003)
	ST-DIM	0.917 (0.002)	0.918 (0.002)	0.935 (0.003)
	STDIM-VAE	0.912 (0.002)	0.912 (0.002)	0.924 (0.004)

4.5 Conclusion

In this chapter, a new encoder architecture, namely the STDIM-VAE has been introduced and investigated using several case studies. Within each case study, encoders were trained, and the learnt representation spaces explored.

After the initial comparative analysis of the different encoder types, the main analysis focused on the STDIM-VAE. It has been shown that when investigating representations of individual frames, key features of the playstyles were often points of focus. For example, in the 'BlackSmoke' gameplays, focus was often on either the smoke or blocks that had been broken. Whereas, for the 'VizDoom' gameplays, focus was in the relevant frames occasionally seen on the weapons, muzzle flashes and weapon projectiles. In the 'Hitman' gameplays, interest was regularly seen on the player, as well as NPCs and the camouflage package when they were in view.

Further, when comparisons were made within a playstyle, the focus regions could often be explained based on either RAN or other stochastic elements of the game. In

addition, when comparing across playstyles, focus regions not seen within a playstyle could often explain differences between the playstyles being compared. However, this was only able to be confirmed due to known playstyle definitions. Although, it is suggested that this analysis could potentially be used to explain unknown playstyles, however it was noted that in some of the cases considered this was hard to visually extract.

Finally, each gameplay was projected onto a 2D plane to more easily investigate how gameplays compared against each other. This regularly revealed that gameplays grouped together shared the same playstyle or at least a common characteristic was shared that defined the playstyle.

In summary, it appears that the STDIM-VAE has been able to learn a representation space that contained playstyle relevant features. This was considered a promising result, suggesting clustering within the learnt representation space will allow successful clustering of different playstyles and this will be investigated in the next chapter.

5 Clustering Encoded Playstyles

This chapter focuses on the clustering of playstyles from gameplay videos. Initial investigation showed that comparing gameplays by comparing frames directly, had limited success when attempting to cluster playstyles accurately. The reason for this was that even minor changes in the gameplay frames, while having little to no impact on the playstyle, could result in large distances between frames, which caused large values within the distance matrix and hence resulted in poor clustering. It is hence suggested that the transformation of frames to an intermediate representation before a comparison is made, could be key to successful playstyle clustering. Given that the unsupervised representation learning technique (STDIM-VAE), introduced in Section 4.1, aimed to encode the relevant information about a given playstyle, it is felt that it would be an appropriate method to use to generate this representation space. In addition, within this chapter, this method will also be compared to alternative unsupervised representation learning methods with several case studies presented.

After the encoding network had been trained, the sequence of frames could be transformed into a sequence of representation vectors. Following this transformation, clustering could then be applied to the representation sequences. In this chapter, a novel clustering algorithm will be introduced and baselined against a number of classical clustering techniques in order to highlight its possible use within this type of application. In the next section, this novel technique called Reference-Based Clustering will be introduced together with a discussion on the procedure that it uses to cluster data.

5.1 Reference-Based Clustering

As mentioned above, after transforming the frame sequences into representation sequences, clustering could then be applied. There were a couple of decisions that needed to be taken about how to achieve this clustering effectively. Firstly, due to the data still being in a time series form, following the transformation, a time series based method to generate a distance matrix must be used. Based on the reviewed literature, it was decided to use Dynamic Time Warping as this allowed temporal shifting within the gameplay. The next choice for consideration related to which metric should be used when comparing two representation vectors. Three possible choices were investigated namely manhattan, euclidean and cosine. These decisions allowed the calculation of a distance matrix that contained the similarity between all gameplays in the dataset.

However, on investigation of this distance matrix, the distances between gameplays of the same playstyles were not always small. These comparisons were possible as the Ground Truth (GT) labels are available. On further study, an interesting property was found. Although, the direct comparison value between gameplays of the same playstyle was not always small, if you compared their distances to another gameplay, the distances to this other gameplay were often similar in magnitude.

The question arose as to how this relative distance property could be utilised in order to cluster the gameplay data. It was hypothesised that if the distance to common reference points were similar and if these distances were plotted against one another for all reference points, then all these points should lie on the line $y = x$. To explore this, an artificial dataset was generated. This dataset contained 5 gaussian clusters with

each cluster comprising of 20 data points, each of which had 256 features. A sample of how these mentioned plots would look is shown in Figure 124, which shows a sample of the comparisons between points from two of the five clusters. Further, the sample shown is a comparison of all combinations of two points from each of these clusters, where a point from cluster x is identified by C_x. The distance to other reference data points is plotted and shown as blue crosses and the calculated line of best fit for these points is plotted as an orange dashed line.

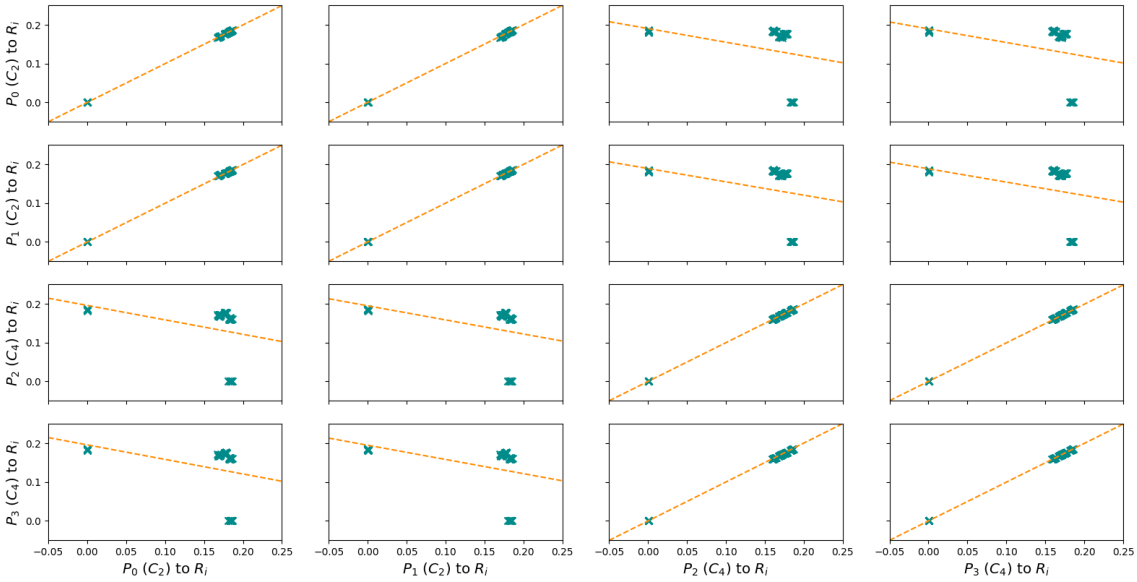


Figure 124: Comparison of Gradient Graphs between Two Examples of Two Clusters

The next step was to extract a single value to describe the similarity between two given data points, it was therefore decided to extract the gradient of the line of best fit and use this as that value. If the given points being compared were the same, then all plotted points should lie on the line $y = x$, it is therefore expected that a gradient value of 1 would infer perfect similarity. Whilst it would be possible to have gradients with values higher than 1, it was assumed that the higher the gradient the more similar the two data points would be. Figure 125 shows a heatmap of the gradients when comparing all the data, namely 100 points, organised by cluster label.

The next step was to start grouping the data points based on these gradient values. It was decided to first join data points by linking each data point to the data point with which it shared the highest gradient value. This assumed all clusters have at least two data points. To visualise this, a graph network was used by first populating it with a node for each data point and then joining each data point to the point it shared with the highest gradient value, by an edge that was weighted based on that gradient value. The generated subgraphs for this artificial dataset are shown in Figure 126.

When examining the generated subgraphs, two main properties were noted. Firstly, in the subgraphs there was often one node where the other nodes appeared to emanate, an example of this property is shown in Figure 126c. Secondly, if there were two high degree nodes, then these were often linked by a short single link chain, an example of this property is shown in Figure 126a.

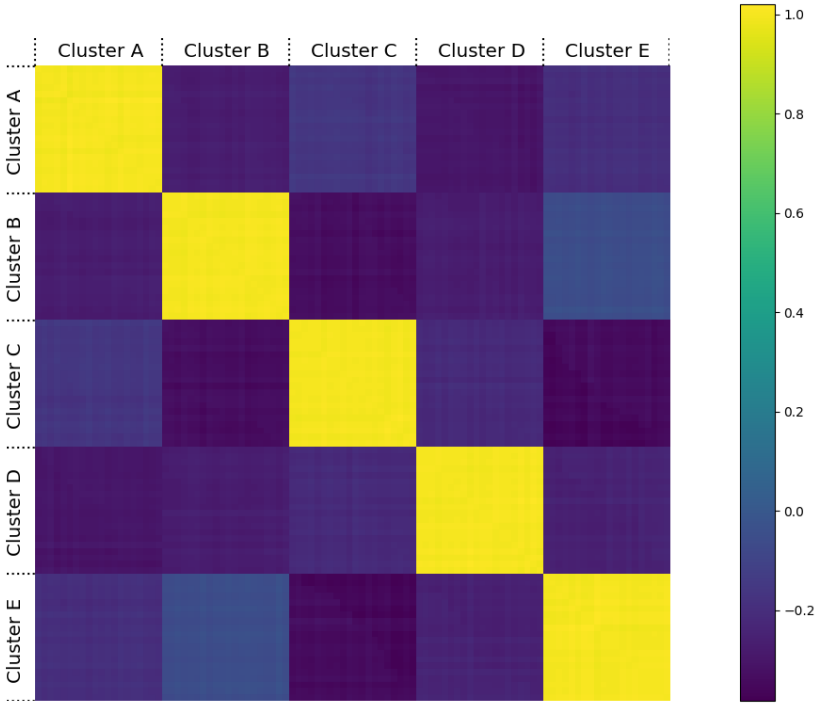
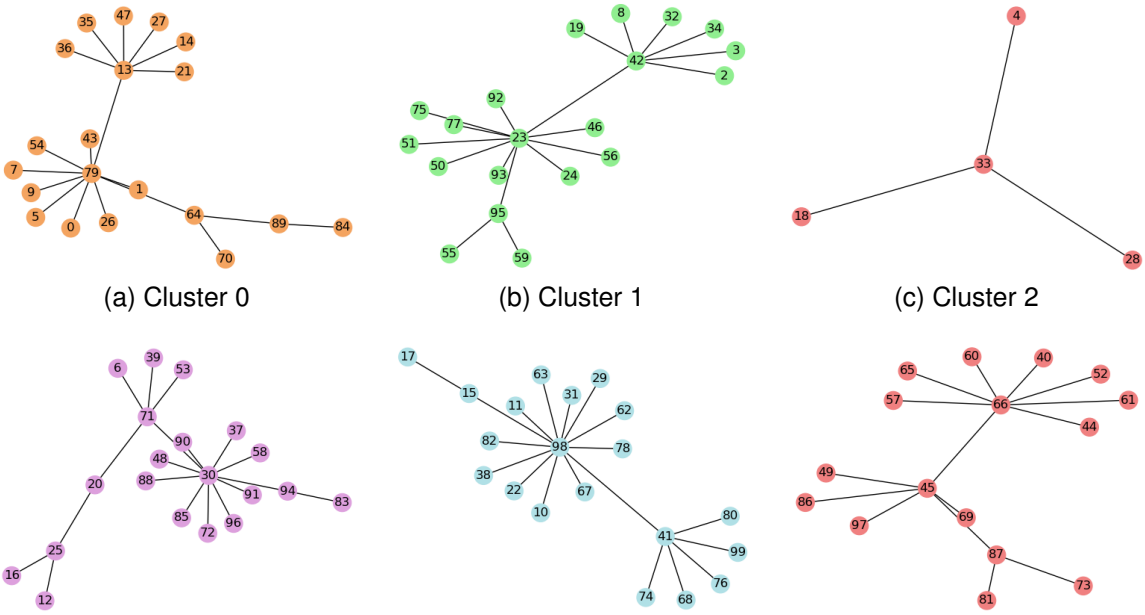


Figure 125: Gradient Heatmaps

When considering this second property, a question arose around whether these two high degree nodes were indeed from the same cluster, or if they should be split. To account for this, the next step was to trace a path between these two high degree nodes. However, it was decided that in order to allow this, the degree of both nodes must be above 3. The value of 3 was chosen, as a value of 1 meant that the node was at the end of the graph, *e.g.* node 5 in Figure 126a. Whereas a value of 2 meant that the node was in the middle of a line, *e.g.* node 15 in Figure 126e. This meant that having a value of 3 resulted in the graph splitting, this can be seen for example in node 25, in Figure 126d. However, this splitting was deemed not to be significant enough to suggest that the node was from a different cluster and hence the degree of the nodes must be above 3. The shortest path was then traced between these two high degree nodes and the edge values on this path were then considered, where the edge values had previously been set to the gradient values. Several conditions were defined to ascertain if and where this cut should occur. Firstly, if the edge value was too low, namely under 0.7, or conversely if it was too high, that is above 1.3, a potential cut was marked. Given that a perfect match would result in an edge value of 1 it was decided that the thresholds should be set at equal distances from this perfect matched value. Based on some experimentation, the aforementioned thresholds were chosen. However, if multiple edges broke these conditions, then the edge value that was most different from the threshold was kept for each condition. In situations where both conditions were broken at some point on the path, the system prioritised the cutting of the edge over 1.3. If a cut was made, this then produced two subgraphs. This process was then repeated on each subgraph recursively until no more cuts were made. An example is shown in Figure 127, in this case the two highest degree nodes were 79 and 13 with degrees of 10 and 7 respectively.



(a) Cluster 0 (b) Cluster 1 (c) Cluster 2

(d) Cluster 3 (e) Cluster 4 (f) Cluster 5

Figure 126: Initial Cluster Graphs

This met the first condition and hence the shortest path was traced between these two high degree nodes, this is shown by the blue dotted line in Figure 127. In this example, the path contained only one edge which did not meet any of the conditions marking a potential cut and hence no cut was made.

Up to this point, all decisions on whether to join or split data points had been based solely on point to point comparisons. The final task was to explore the similarity between whole subgraphs and then decide as to whether to merge them or not. To achieve this, the average gradient between nodes in each subgraph was calculated. If this value was above 0.8, this value being chosen through experimentation, then those two subgraphs were merged. If any merging had occurred, the average gradients were recalculated to check if any further merging was required. This was then repeated until no more merging occurred. Figure 128 shows the final set of subgraphs for the artificial dataset. In this figure each dashed region shows which subgraphs have been merged in the final step. Additionally, the values marked on the dashed arrows in Figure 128 show the average gradient between the subgraphs, used to decide whether to merge or not. Each set of merged subgraphs was defined as a separate cluster.

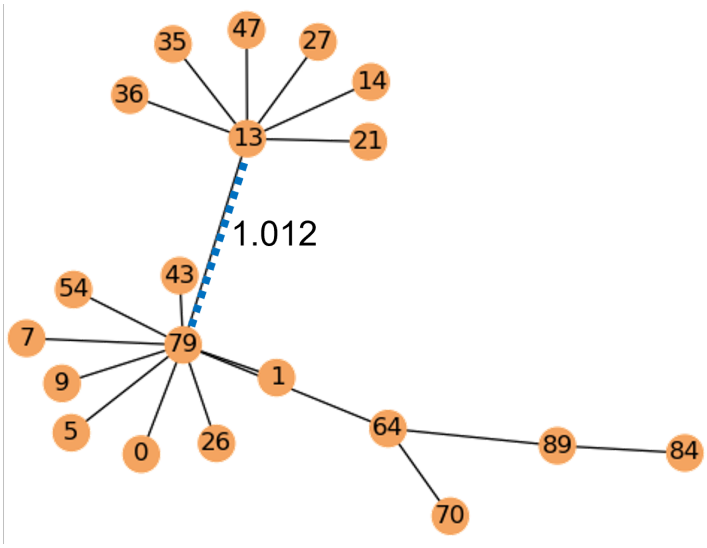


Figure 127: Cluster 0 (With possible cut location shown)

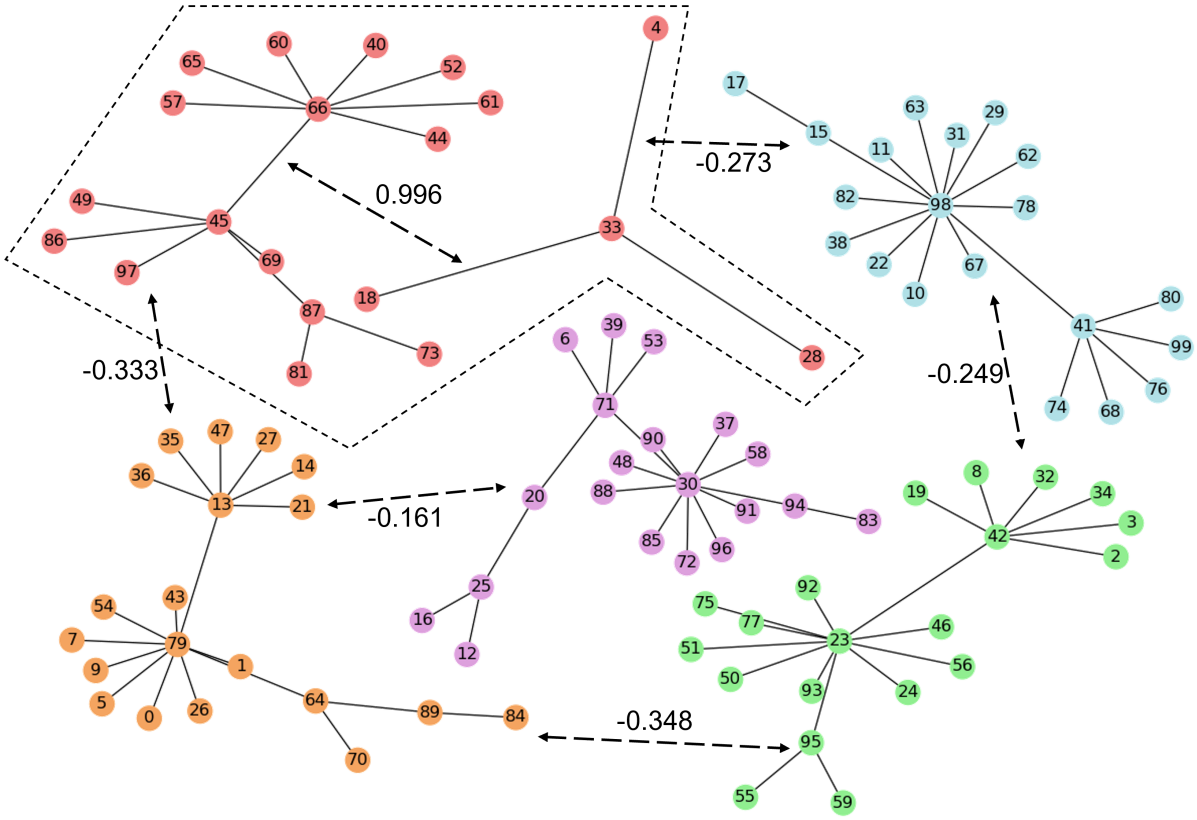


Figure 128: Final Clusters

5.2 Case Studies

The technique introduced above, called Reference-Based Clustering, was tested against previously established methods across a number of case studies. Each of the case studies investigated the quality of the clustering of different standard clustering algorithms (*e.g.* Hierarchical Clustering). They also explored the data sources used, including a range of learnt representation spaces and game variable combinations. For the game variables, clustering was applied to both the whole series and also just to the end values. After clustering was applied, Adjusted Mutual Information (AMI) was used to evaluate the quality of the found clusters. AMI allowed the evaluation of the quality of clusters given the predicted and true cluster labels. A score of 1 meant perfect clustering had been achieved, which meant that the predicted and true clusters were the same. Whereas a value of 0 meant that the quality of the clusters was equal to what would have been expected if data was randomly separated. For each case study a wide range of setups of different data sources, clustering algorithms and metrics were evaluated.

For the ease of presentation, a subset of the results will be shown within each case study section, with the full set of results shown in Appendix B.

5.2.1 Black Smoke

The first case study focused on the game 'Black Smoke', previously introduced in Section 4.4.1. Firstly, the aim was to gain an understanding of how well clustering could be applied based on game variables. For the game 'Black Smoke', three game variables were chosen 'x', 'y' and 'b'. The first two variables related to the player's current 'x' and 'y' positions within the game level respectively. Whereas the third variable 'b' was a running count of the number of blocks broken.

Hierarchical Clustering was then applied to different combinations of the game variable series data, using DTW to pre-compute the relevant distance matrix under different metrics. The next step was to then apply clustering to only the end values of the game variables. This also allowed the introduction of additional standard clustering methods (*i.e.* Kmeans and GMMs). A subset of the results is shown in Figures 129 and 130 for the 0%, 1%, 5% and 10% RAN datasets.

It can be seen that most of the experimental setups studied, that utilised the series game variable data, were able to achieve perfect clustering for these low RAN datasets. Whereas for the setups that utilised only the end values of the game variables, it can be seen with the full result set that clustering on the combination of the game variables 'x' and 'y' failed to cluster perfectly. This logically made sense as on completion, the player ends in the same location assuming level completion. This showed that for some game variables having the whole data series was important for successful clustering. Although some game variables, namely 'b', were able to cluster perfectly given just the end value.

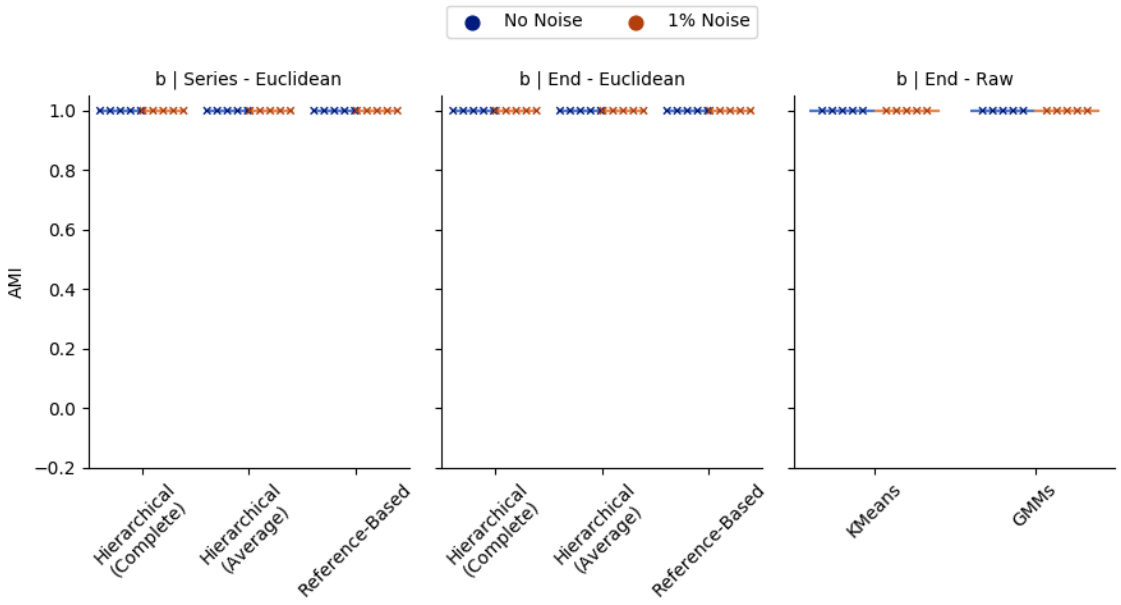


Figure 129: Game Variables Clustering on 0% and 1% RAN Datasets

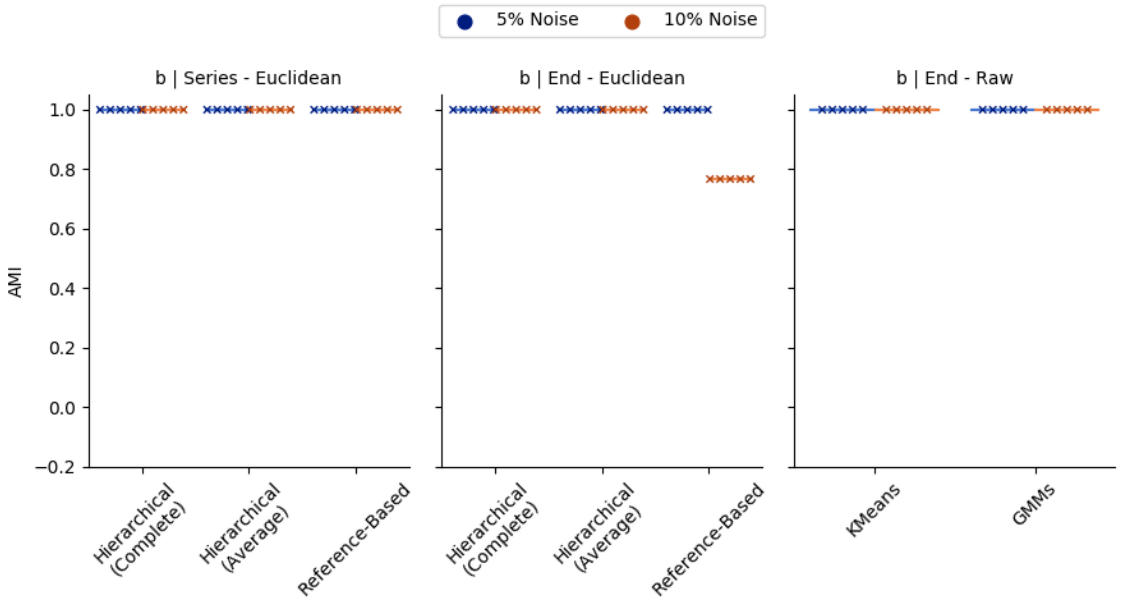


Figure 130: Game Variables Clustering on 5% and 10% RAN Datasets

The final step for the game variable analysis for this case study was to look at the 20% RAN dataset and the dataset containing all RAN levels. For the 20% RAN dataset, a drop off in quality was seen but it was noted that some setups were still able to cluster perfectly. Further, quality drops were also observed for the all RAN dataset, but it was noted that for this dataset, the Reference-Based Clustering algorithm often performed best. The key result subset is shown in Figure 131.

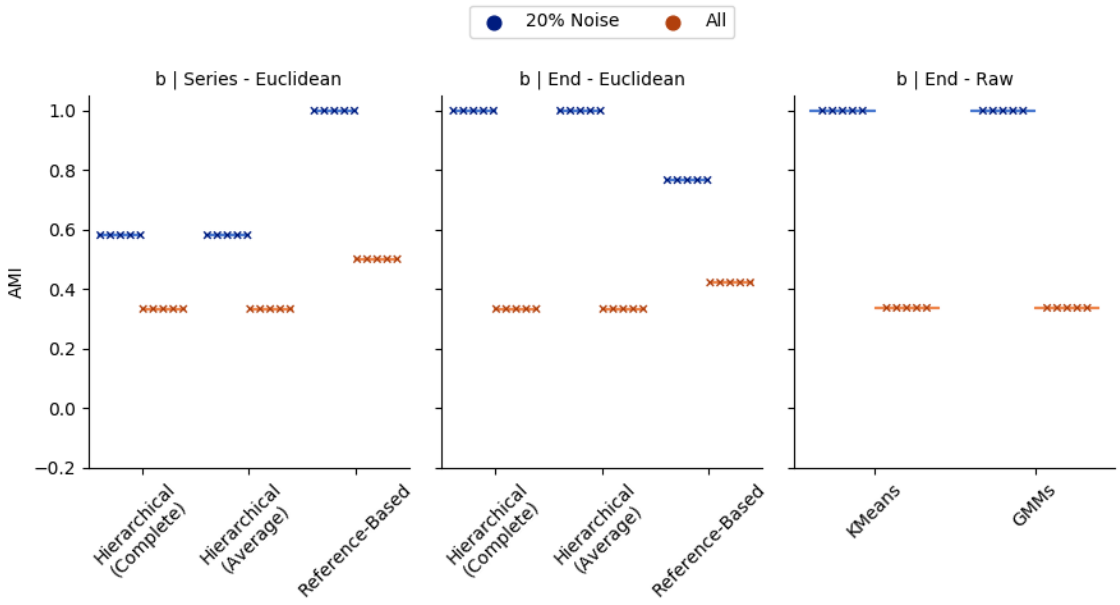


Figure 131: Game Variables Clustering on 20% and All RAN Datasets

After applying clustering to the game variables, the next step was to cluster based on the learnt representation spaces. This was done for both the hybrid encoder (STDIM-VAE) setup, as well as each of the individual encoder architectures used within the hybrid encoder. This allowed comparisons to be drawn between how each individual encoder worked and how these compared to the combined hybrid encoder architecture.

When looking at the lower RAN datasets, perfect clustering was often possible. However, on some occasions, Reference-Based Clustering performed slightly worse. This is shown, in the result summary, in Figures 132 and 133. On examination of the found clusters using the hybrid encoder, namely STDIM-VAE, with the cosine metric and Reference-Based Clustering method, it was found that additional cluster(s) often were identified, resulting in a higher number of clusters than existed in the Ground Truth labels. This was possible because the Reference-Based Clustering technique was not told how many clusters existed, while the other methods considered, utilised this information. On investigating the clusters for the two runs from the ‘No Noise’ dataset where perfect clustering was not achieved, it can be seen that an additional cluster was generated by splitting the second true cluster into two. Further for the ‘5% Noise’ dataset, on the occasions that perfect clustering was not achieved, similarly to before, one of the clusters was split into two. However, the playstyle cluster which was split depended on the particular training run. Finally, for the ‘10% Noise’ dataset, two additional clusters were generated by splitting each of the true clusters into two forming

a total of four clusters. It should be noted, that in all these cases all of the clusters were still pure. This meant that they all contained only one playstyle.

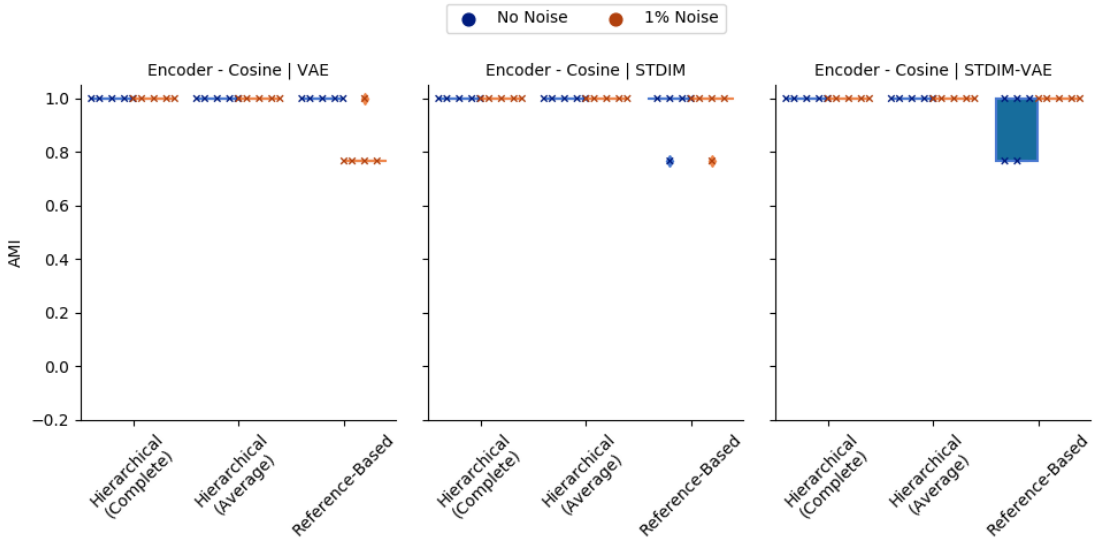


Figure 132: Representation Clustering on 0% and 1% RAN Datasets

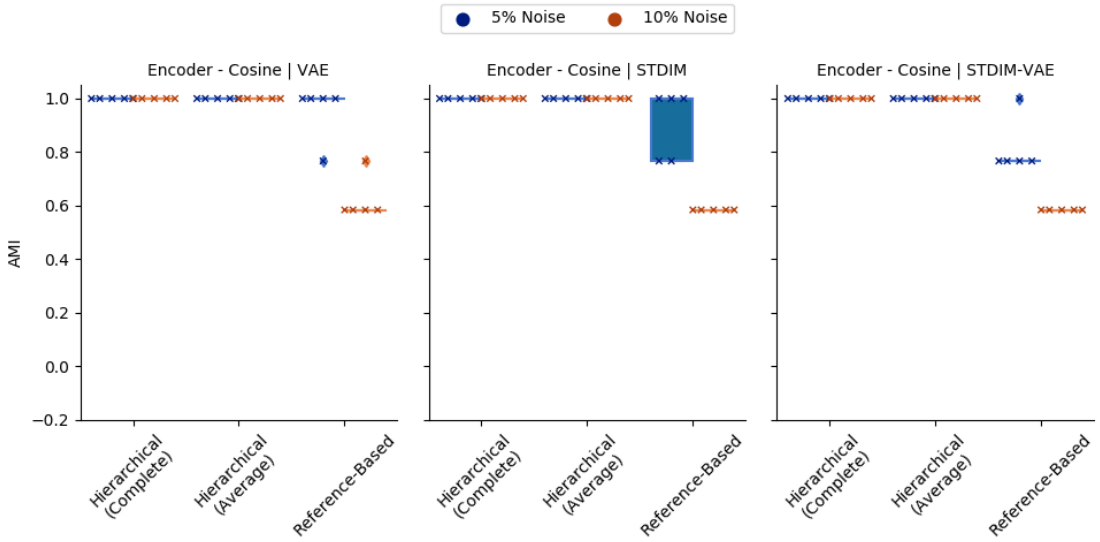


Figure 133: Representation Clustering on 5% and 10% RAN Datasets

Similarly, to game variable clustering, a drop off in clustering quality was seen on the 20% RAN dataset. It was also noted that in the all RAN dataset, the Reference-Based Clustering method performed the best. A subset of the results is shown in Figure 134.

Finally, Table 11 provides a summary of the clustering quality for different approaches experimented with, within the BlackSmoke case study. This brings together the best results from clustering on frame representations, game variable series and end game variable values. Within the frame representation case, this includes the best result when

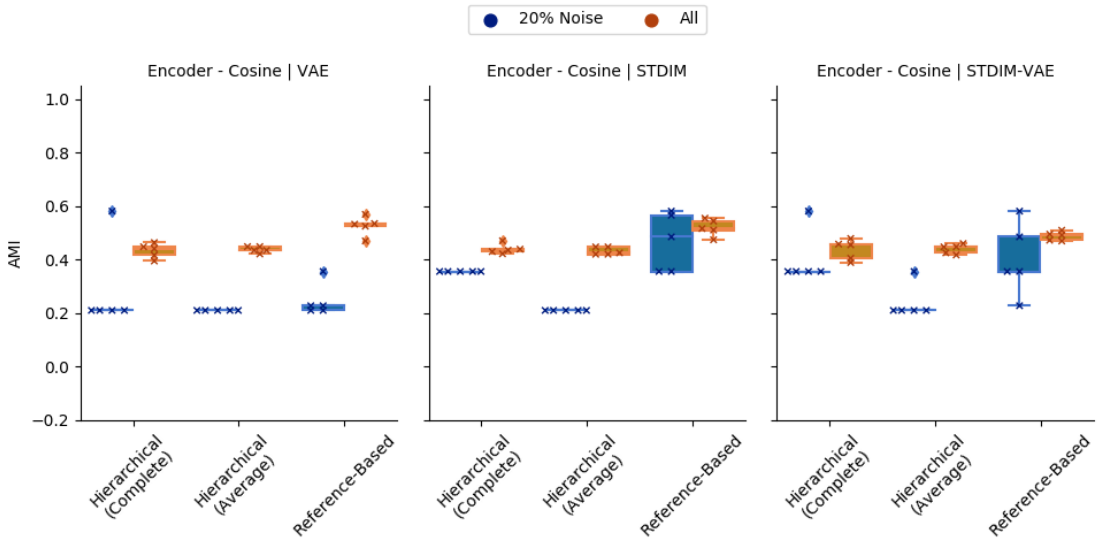


Figure 134: Representation Clustering on 20% and All RAN Datasets

using both, one or none of the proposed techniques. As regards the game variable clustering case, the best results are presented when both using and not using the proposed clustering algorithm. When multiple setups provide the same performance for a given category, one example setup is shown in the table.

Table 11: BlackSmoke Clustering Quality Summary

Data Source	Distance Metric	Clustering Algorithm	AMI
ST-DIM	Manhattan	Reference-Based	0.565 (±0.024)
STDIM-VAE	Euclidean	Reference-Based	0.564 (±0.020)
x,y (Series)	Cosine	Reference-Based	0.515 (±0.000)
VAE	Euclidean	Hierarchical (Average)	0.454 (±0.000)
STDIM-VAE	Cosine	Hierarchical (Average)	0.440 (± 0.016)
b (End)	Euclidean	Reference-Based	0.422 (±0.000)
x,y,b (Series)	Cosine	Hierarchical (Complete)	0.348 (±0.000)
b (End)	Raw	GMMs	0.337(±0.000)

As can be seen in the summary table, the combination of the STDIM-VAE and ‘Reference-Based’ clustering performs second best, however the difference between this and the best setup is small. Further, it can be seen that the top three setups use at least one of the proposed techniques. In addition, when considering clustering on game variables, the use of ‘Reference-Based’ clustering improves the clustering quality when applied to either the series or end values of the game variables.

5.2.2 VizDoom

The second case study focused on the game ‘VizDoom’, previously discussed in Section 4.4.2. Similarly, several game variables were selected based on the previously

mentioned literature [27]. These were as follows: ‘gunKP’, ‘dis’ and ‘dam’. The first of these ‘gunKP’ related to a group of variables that reflected the proportion of kills that were achieved with each weapon. The second ‘dis’ represented the average kill distance. The final variable, ’dam’ recorded how much damage the player had taken.

As discussed previously in Section 4.4.2, when the set of ‘VizDoom’ playstyles were introduced, each playstyle was defined based on two main factors. These were ‘Weapon Choice’ and ‘Player Location’. From the overall dataset, a number of subsets were created to focus on each of these factors. Firstly, focus was given to the ‘Weapon Choice’ factor.

The first two subsets that focused on the ‘Weapon Choice’ were called ‘Loop Gun’ and ‘Main Room Gun’ both of which had the ‘Player Location’ fixed to either a given route or room. This ensured that the main difference between the playstyles was which weapon was being used. This allowed the evaluation of the methods’ ability to split playstyles that only differed by the weapon used.

Clustering on the game variables was then applied to these two datasets, this included clustering on the whole series and the end values. A subset of these results are shown in Figure 135, with the full result set in Appendix B.2.

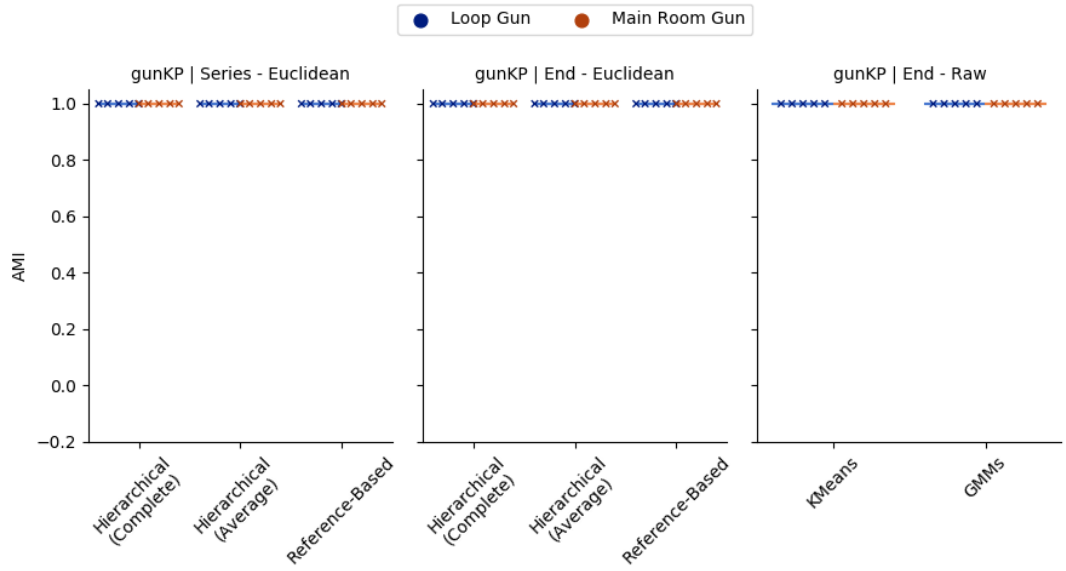


Figure 135: Game Variable Clustering on ‘Loop Gun’ and ‘Main Room Gun’ Datasets

As would be expected, given that the playstyles differed based only on gun choice, the ‘gunKP’ game variable set allowed perfect clustering for both the series and end value data. However, when examining the weapon used for each kill, it was found that there was one gameplay utilising the ‘Main Room Plasma Rifle’ playstyle, within the ‘Main Room Gun’ dataset, where a kill was also achieved using the ‘Pistol’. As all other runs only ever achieved kills with the preferred weapon, this difference was not sufficient to cause issues when clustering. Equally, the ‘dis’ and ‘dam’ game variable combination performed poorly. More interestingly, the combination of all game variables also performed badly. This suggested that not only do you need to select good game variables, but the introduction of bad game variables can nullify any good variables

chosen.

The next step was to apply clustering on the learnt representation spaces on the datasets previously described. A subset of these results can be seen in Figure 136. Overall, it can be seen that the cosine metric appeared to be the best metric in order to obtain high quality clustering.

When focusing on this metric, it was shown that for all encoder types, Hierarchical Clustering (with both complete and average linkage) was able to perfectly cluster the 'Main Room Gun' dataset. On the other hand, Reference-Based Clustering only performed perfectly for the VAE, whereas the ST-DIM encoder performed worst. Interestingly, however, when looking at the hybrid encoder there was only a small drop in cluster quality, caused by one bad run. When examining the clusters from this bad run, it was observed that an additional cluster had been found by splitting the first cluster into two and hence all the clusters were still pure.

When considering the 'Loop Gun' dataset, it was still seen that the VAE performed best. Although perfect clustering was not found in this case, it was noted that both the ST-DIM and STDIM-VAE architectures did not generally perform as well when compared to either the VAE on this dataset or themselves on the previous dataset.

Looking at these two datasets it was hypothesised that the VAE was better than the ST-DIM encoder at splitting playstyles that differed based on the weapon used.

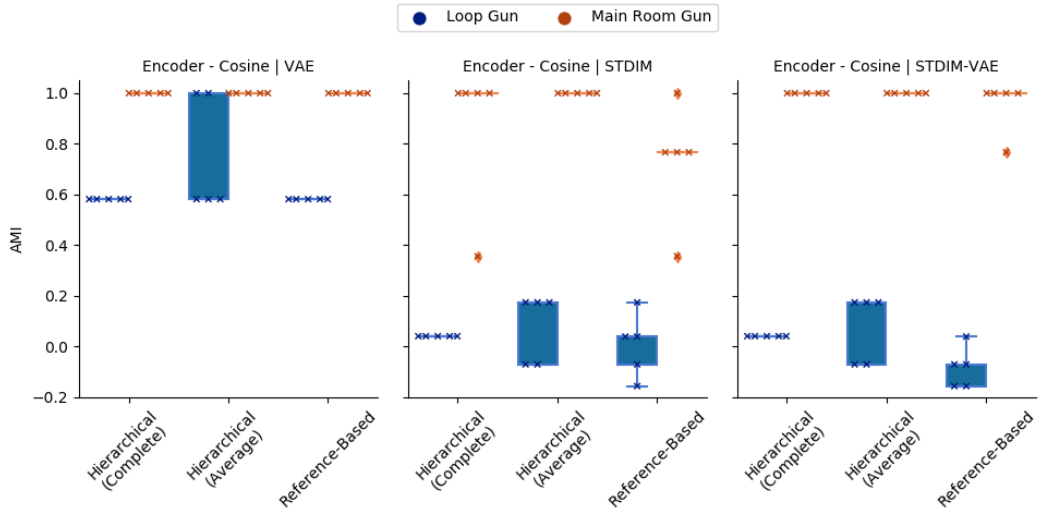


Figure 136: Representation Clustering on 'Loop Gun' and 'Main Room Gun' Datasets

The next two subsets considered were called 'W1 Gun' (Window 1 Gun) and 'W2 Gun' (Window 2 Gun). Similarly, to the previous two subsets, the player's location was fixed to a given window position with the main difference being the weapon used. Although some positional differences were introduced as the player needed to collect their preferred weapon.

Once again clustering was applied to both the data series and the end values of the game variables. A subset of the resultant evaluation for each of these can be seen in Figure 137. The outcomes were found to be similar to those seen in the analysis of the previous two datasets. That being that the 'gunKP' game variable set was able to cluster well and that clustering on 'dis' and 'dam' resulted in bad quality clusters.

Additionally, the combination of all game variables also performed poorly. This further showed that how picking bad game variables can result in bad clustering even if good game variables are also present.

Generally, when considering both datasets, it appeared that the use of the series data results in better quality clusters when compared to end value clustering. On examination of the game variable data, many of the gameplays achieved a kill with another weapon, alongside the kills achieved with the preferred weapon. It was noted that this additional weapon varied based on the playstyle. For example, for the 'Win 1 BFG' playstyle from the 'W1 Gun' dataset, four out of five of the runs achieved a single kill with the 'Shotgun'. This kill appeared to often happen when the player was collecting more 'ammo' for the 'BFG' and hence usually occurred later in the gameplay. Only the series data retained this temporal information. Hence this was expected to allow better matching earlier in the gameplay sequence and therefore allowed better clustering overall.

Additionally, better clustering was regularly achieved on the 'W2 Gun' dataset compared to the 'W1 Gun' dataset. In a similar manner this can also be explained by the game variable data. In 'W2 Gun' all gameplays achieved kills with the preferred weapon, but occasionally kills were also achieved with another weapon as well. Whereas, for the 'Win 1 RL' (Window 1 Rocket Launcher) playstyle from the 'Win 1 Gun' dataset, although when the 'Rocket Launcher' was used, no kills were achieved in two of the gameplays. This hence made it harder to cluster that dataset.

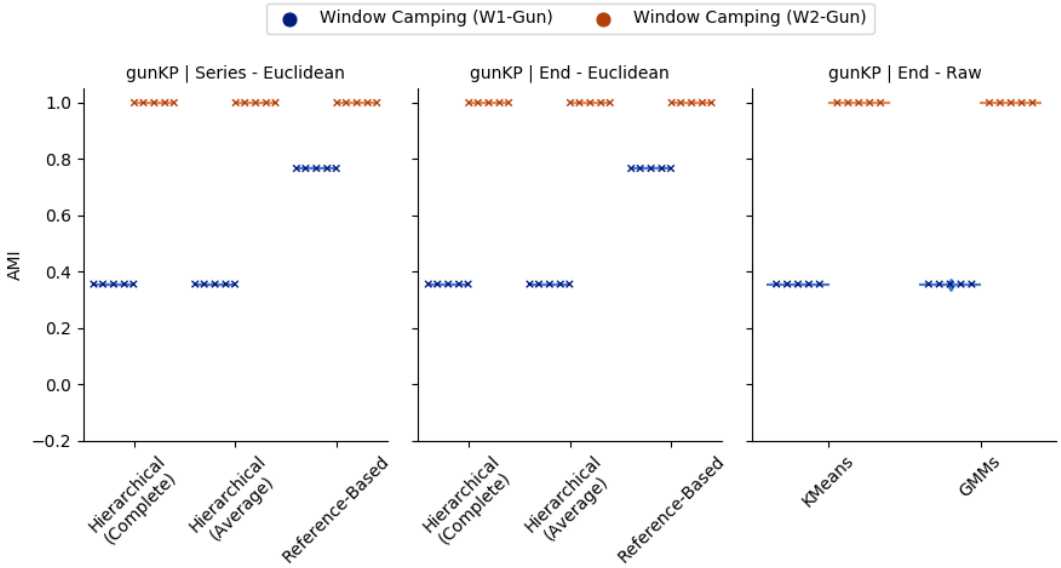


Figure 137: Game Variable Clustering on 'Window 1 Gun' and 'Window 2 Gun' Datasets

A subset of the results for clustering of these datasets within the representation spaces are shown in Figure 138. In these cases, only the combination of the VAE, cosine metric and Hierarchical Clustering (using complete linkage) was able to achieve perfect clustering on both datasets. This further suggested that the VAE is better at clustering based on weapon choice.

When examining the combination of the cosine metric and Reference-Based Cluster-

ing, an interesting pattern was observed. When considering the ‘W1 Gun’ dataset, the VAE outperformed the ST-DIM encoder. Further, it was noted that the STDIM-VAE cluster quality fell between the quality values of the VAE and ST-DIM architectures. On the other hand, when considering the ‘W2 Gun’ dataset, the ST-DIM encoder outperformed the VAE. Although, once again the STDIM-VAE architecture cluster quality fell between these quality values as before. This showed that the STDIM-VAE balanced each part of its architecture well and hence performed relatively well on both datasets. Further analysis was then conducted on the clusters found using the STDIM-VAE and the reason for the cluster quality dropping was found. In each of these cases, Reference-Based Clustering had split one of the clusters into two, which had caused the quality to drop, however this still meant that each cluster was pure.

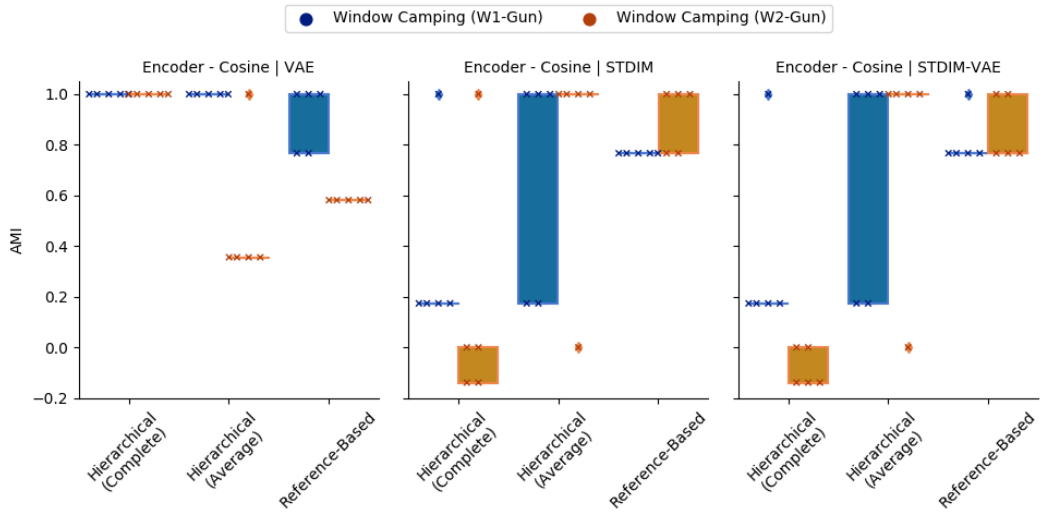


Figure 138: Representation Clustering on ‘Window 1 Gun’ and ‘Window 2 Gun’ Datasets

The subsets up to this point had aimed to explore which methods were able to split playstyles based on ‘Weapon Choice’. However, there was another factor, namely ‘Player Location’ that defined a playstyle. The following subsets focused on playstyles that could be split by this factor alone. The first two subsets where this was true were the ‘Loop Route’ and ‘Movement’. In the first of these, the same weapon was used by both playstyles, namely the ‘Chain Gun’, however the route taken around the map differed. The second of these, namely ‘Movement’ also used the same gun, but in this case it was the ‘Shotgun’, with one playstyle camping in a corner and the other playstyle exploring the map. These datasets allowed the analysis to evaluate the methods’ ability to split playstyles based on ‘Player Location’.

For these two datasets, all the selected game variables resulted in bad clustering. A subset of these results is shown in Figure 139. Given that logically none of these variables encapsulated the defining factor, namely the player location this poor clustering makes sense. This showed that correct game variable selection was extremely important to achieve good quality clusters. However, in order to select the correct game variables, it must be understood which playstyles were expected to be encountered. This meant that these methods were unlikely to discover the presence of new playstyles, that were not anticipated.

Further, the selection of bad game variables could lead to a misunderstanding about the kind of playstyles that existed. For example, in the ‘Movement’ dataset, two of the gameplays for the ‘Corner Camping Silver Room Shotgun’ playstyle do not achieve any kills. If clustering was based on the ‘gunKP’ game variable set, it is likely that these two gameplays would be identified as a separate playstyle that does not achieve any kills. However, in reality the same playstyle is being used and a different factor caused no kills to be achieved.

In theory, a game variable encapsulating the player’s current location within the level was likely to allow the clustering of the playstyles. However, it was likely that the ability to extract this information would require access to the code base and development of systems to extract it. This adds a further cost to game variable based methods, compared to those that can cluster based on visual information alone.

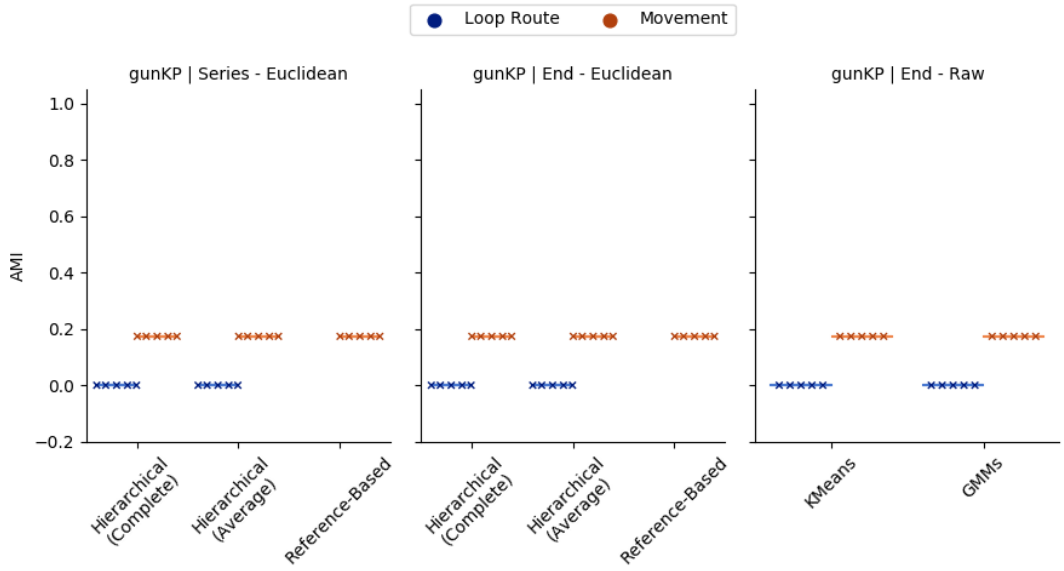


Figure 139: Game Variable Clustering on ‘Loop Route’ and ‘Movement’ Datasets

Following the game variable clustering above, clustering was then applied to the datasets using the learnt representation spaces and a subset of the results is shown in Figure 140. It can be seen that for all encoder types, when using the cosine metric, that Hierarchical Clustering (using complete linkage) achieved perfect clustering across both datasets.

However, when considering the combination of the cosine metric and Reference-Based Clustering on the ‘Loop Route’ dataset, only ST-DIM achieved perfect clustering. Although it should be noted that both VAE and STDIM-VAE performed reasonably well, with only a few bad runs out of the five conducted. Here the VAE had one bad run compared to the two bad runs for the STDIM-VAE, However, when these bad runs for the hybrid encoder STDIM-VAE were explored, it was found that the drop in quality was caused by the splitting of the second cluster into two. It should be noted that all the clusters were still pure. Further, when looking at the ‘Movement’ dataset, all encoder types achieved perfect clustering. These two findings suggested that the ST-DIM encoder may be better at splitting playstyles based on a player’s location.

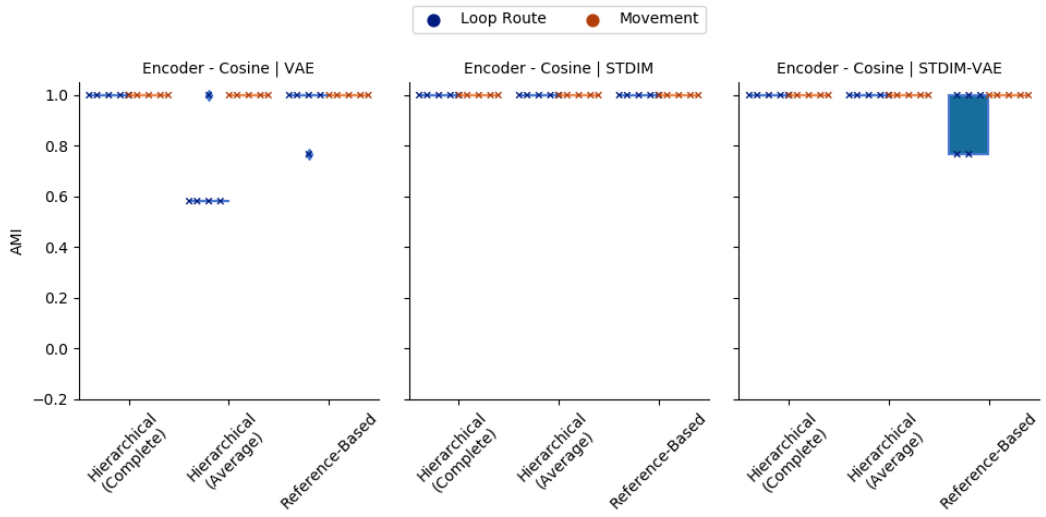


Figure 140: Representation Clustering on 'Loop Route' and 'Movement' Datasets

The final two subsets, namely 'BFG Window' and 'RL Window' also evaluated the methods' ability to split by 'Player Location'. Within both these subsets, the main difference was which window was being used for camping. Although the difference between these subsets was which weapon was used, that being the 'BFG9000' or the 'Rocket Launcher' respectively. The resultant subset for these datasets can be seen in Figure 141.

Similarly, on the whole, the clustering of game variables does not perform that well based on the fact that none of the available game variables encapsulates the player's location. However, in a few cases a better quality of clustering can be seen when using the Reference-Based Clustering technique. Further and conversely to before, in some cases clustering on the end value of the game variables performed better compared to the series data. Although, this was not the case for the 'gunKP' variable set, as perfect clustering was possible on the series data. When the game variable data was investigated for the 'BFG Win' dataset, it was found that four out of five 'Win 1 BFG' gameplays achieved one kill with the 'Shotgun'. Cluster splitting on this information alone would result in fairly good clusters, although the factor for causing this split was incorrect. However, as the use of series data, allowed perfect clustering, it was likely that the speed/timing of kills allowed the splitting of the one 'Win 1 BFG' gameplay that did not achieve a 'Shotgun' kill.

After game variable clustering, clustering was then applied to each of these datasets using the learnt representation spaces. A subset of the results is shown in Figure 142. Interestingly, it appeared to be easier to cluster the 'BFG Window' subset than the 'RL Window' subset, as multiple different experimental setups were able to achieve perfect clustering on this subset. Although it should be noted that all these setups, either use the STDIM-VAE architecture and/or Reference-Based Clustering. While perfect clustering was not found for the 'RL Window' subset, the highest quality clusters were often found using Reference-Based Clustering.

When focusing on the combination of the cosine metric and Reference-Based Clustering, a couple of points were noted. Firstly, for both datasets ST-DIM outperformed the

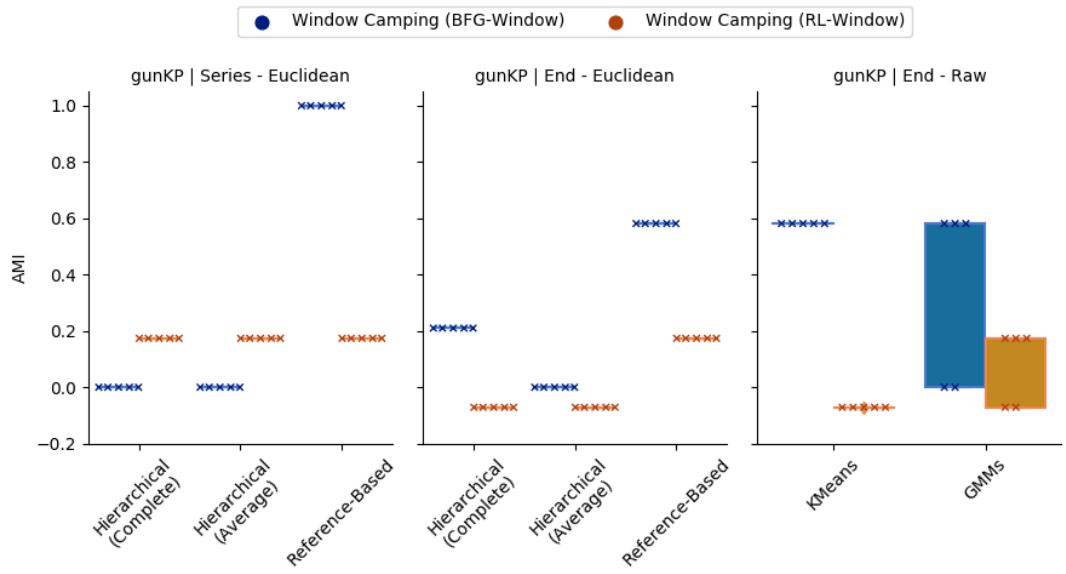


Figure 141: Game Variable Clustering on 'BFG9000 Window' and 'Rocket Launcher Window' Datasets

VAE. This further suggested that the ST-DIM encoder was better at splitting playstyles based on player location. Secondly, STDIM-VAE outperformed both VAE and ST-DIM architectures. Further, when examining the clusters using the hybrid encoder STDIM-VAE and Reference-Based Clustering, it was found that the first cluster was split in two and hence this meant that all clusters were pure.

The final test was to attempt to cluster a dataset that contained all of the playstyles. This meant that the methods must be able to split by a combination of the 'Weapon Choice' and 'Player Location'.

This was first attempted using game variables. It was shown that the 'gunKP' game variable set performed best, a subset of the results for these variables is shown in Figure 143. Further, the quality of the clusters, on the whole, were similar for both series and end values for game variable clustering.

The next step was to cluster using the learnt representation spaces. A subset of the results is shown in Figure 144, it was noted that when looking at the combination of the cosine metric and Reference-Based Clustering, that STDIM-VAE outperformed on average both the VAE and ST-DIM architectures on the whole dataset. Further, the combination of the cosine metric, STDIM-VAE and Reference-Based Clustering outperformed on average all other setups considered.

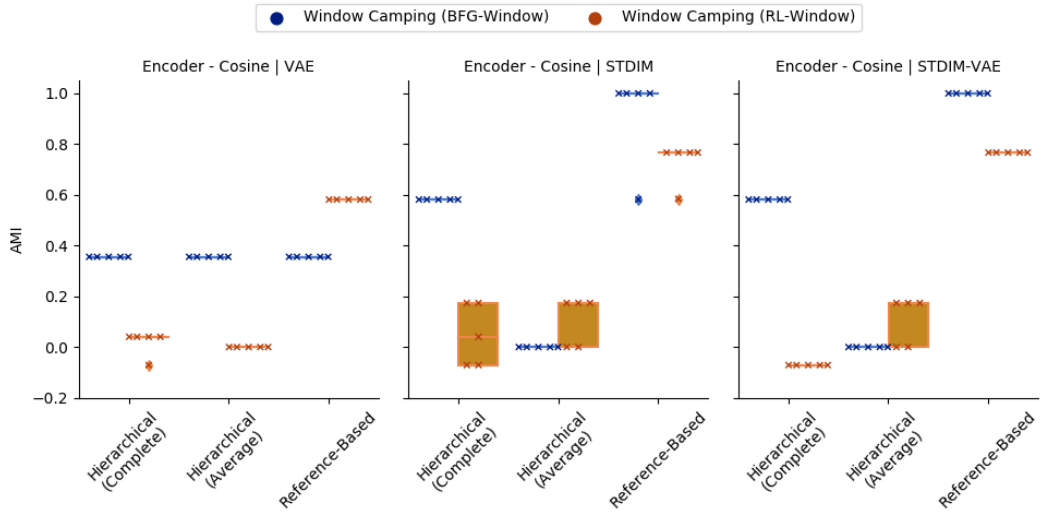


Figure 142: Representation Clustering on ‘BFG9000 Window’ and ‘Rocket Launcher Window’ Datasets

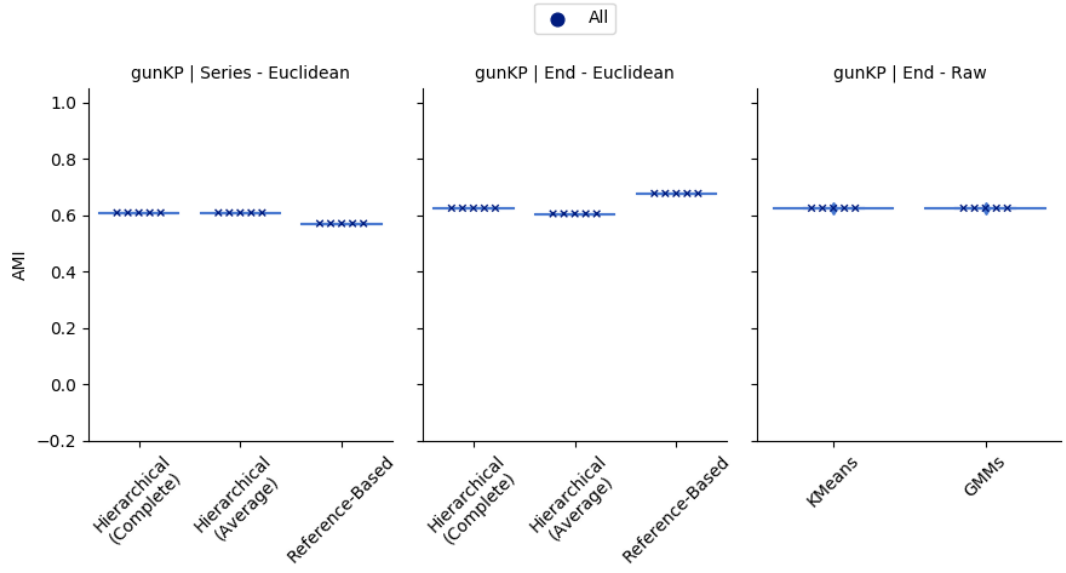


Figure 143: Game Variable Clustering on Full Dataset (Ground Truth = Gun and Position)

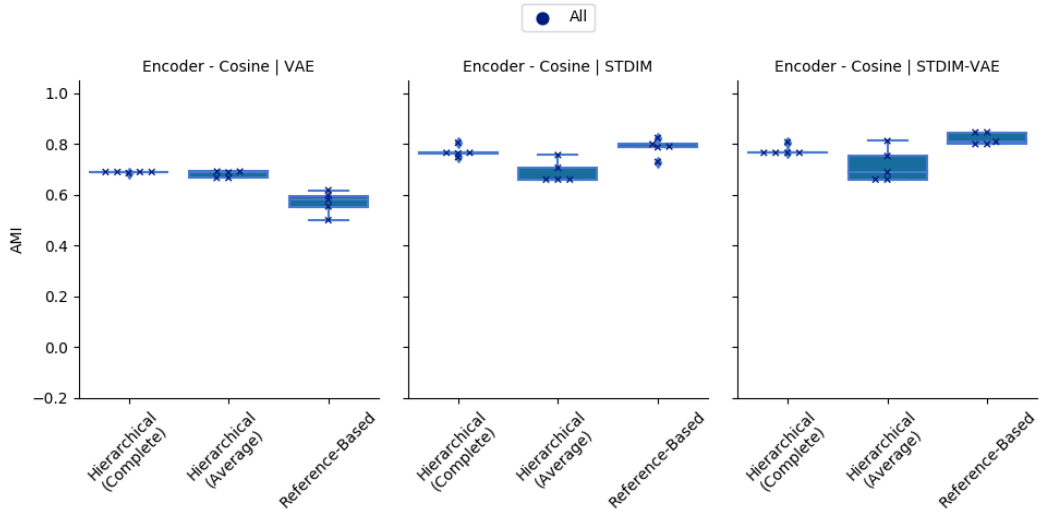


Figure 144: Representation Clustering on Full Dataset (Ground Truth = Gun and Position)

It was suspected that game variable clustering was more heavily focused on the weapon choice of the playstyles. To evaluate this the AMI scores were recalculated with the Ground Truth labels based on either just the weapon used or the player's location. A subset of the results for this can be seen in Figure 145. Focusing on the 'gunKP' game variable set, it can be seen that the AMI score was higher for the gun only Ground Truth labels when compared to both the position only Ground Truth labels and the combined Ground Truth labels. This showed that indeed the game variable clusters were based more heavily on the weapon choice. However, it was noted that perfect clustering was not achieved. This was believed to be down to the difference in the intended playstyle compared to the executed playstyle. Although, a certain weapon may be used within a gameplay, if a kill was not achieved, then this was not reflected in the 'gunKP' game variable set.

This confirmed that poorly selected game variables can result in missing playstyles from within the clusters. While these found playstyle clusters made sense, they failed to find the additional factors that defined a playstyle.

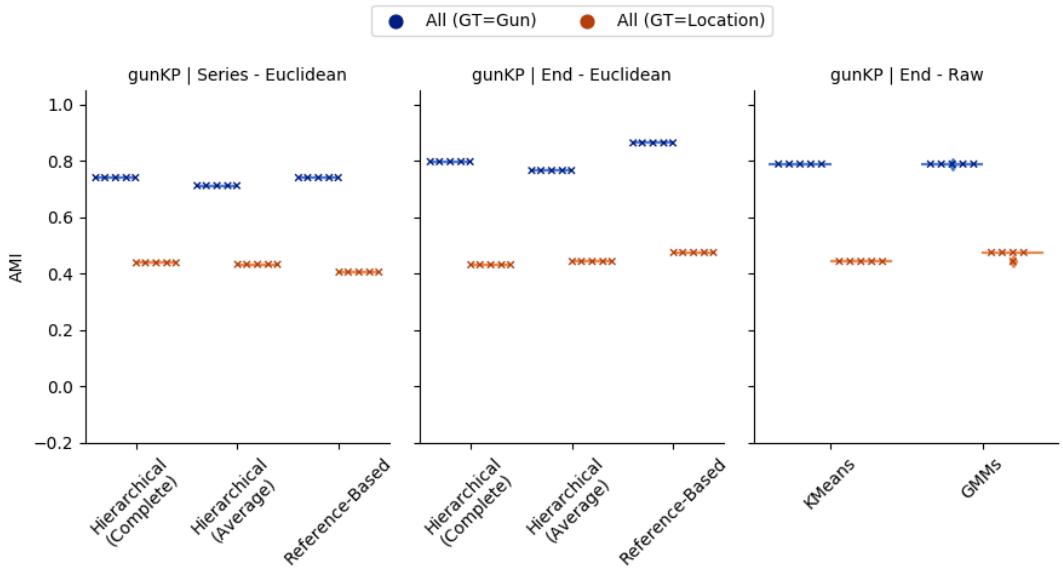


Figure 145: Game Variable Clustering on Full Dataset (Ground Truth = Gun or Position)

This re-evaluation under different Ground Truth labels was also conducted for representation clustering and the resultant subset can be seen in Figure 146. In this case for the combination of the cosine metric and reference base clustering, the VAE performed better under the gun only Ground Truth labels. Whereas the ST-DIM encoder performed better under the position only Ground Truth labels. This further reinforced the idea that the VAE was better at retaining information about the gun, while the ST-DIM encoder better retained information about the player's location.

Finally, Table 12 provides a summary of the clustering quality for different approaches experimented with, within the VizDoom case study. This brings together the best results from clustering on frame representations, game variable series and end game variable values. Within the frame representation case, this includes the best result when using both, one or none of the proposed techniques. As regards the game variable clustering

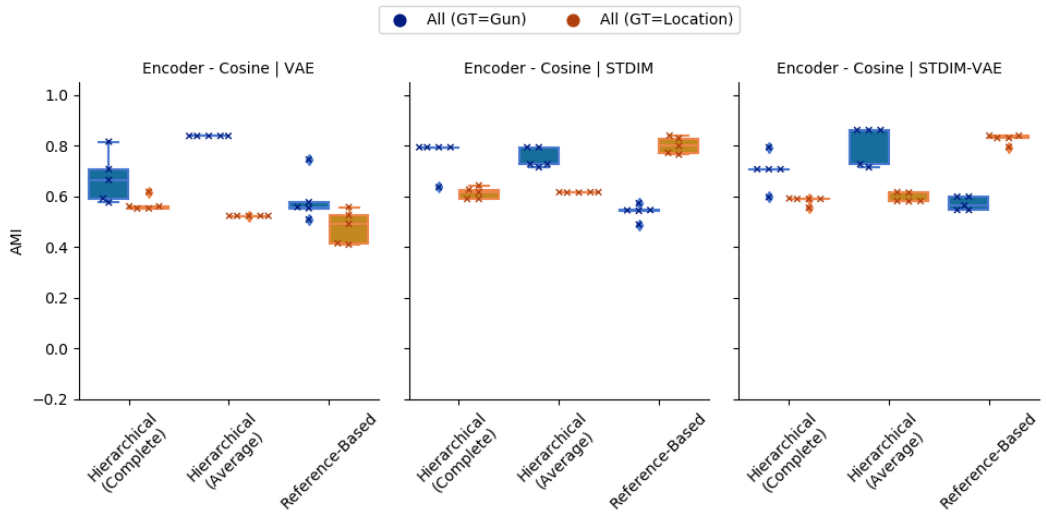


Figure 146: Representation Clustering on Full Dataset (Ground Truth = Gun or Position)

case, the best results are presented when both using and not using the proposed clustering algorithm. When multiple setups provide the same performance for a given category, one example setup is shown in the table.

Table 12: VizDoom Clustering Quality Summary

Data Source	Distance Metric	Clustering Algorithm	AMI
STDIM-VAE	Cosine	Reference-Based	0.820 ($\pm$0.022)
STDIM-VAE	Cosine	Hierarchical (Complete)	0.774 ($\pm$0.017)
ST-DIM	Euclidean	Reference-Based	0.772 ($\pm$0.032)
ST-DIM	Cosine	Hierarchical (Complete)	0.770 ($\pm$0.019)
gunKP (Series)	Manhattan	Reference-Based	0.678 ($\pm$0.000)
gunKP (End)	Euclidean	Reference-Based	0.678 ($\pm$0.000)
gunKP (Series)	Manhattan	Hierarchical (Complete)	0.661 ($\pm$0.000)
gunKP (End)	Euclidean	Hierarchical (Complete)	0.625 ($\pm$0.000)

When examining the summary table, it can be seen that the combination of the STDIM-VAE and ‘Reference-Based’ clustering performs best. Further, the use of ‘Reference-Based’ clustering improves clustering quality when applied to both game variable series and end game variable values.

5.2.3 Hitman

The final case study focused on a stealth game. For this game, four main game variables were chosen. The first two were ‘x’ and ’y’ which related to the player’s location within the level. The third variable ‘c’ took the value of either 1 or 0 which showed if the player was currently camouflaged or not and finally ’k’ counted the number of enemies the player had taken down.

Game variable clustering was then applied to different game variable combinations in the same way as in the previous two case studies. A subset of the game variable clustering is shown in Figure 147.

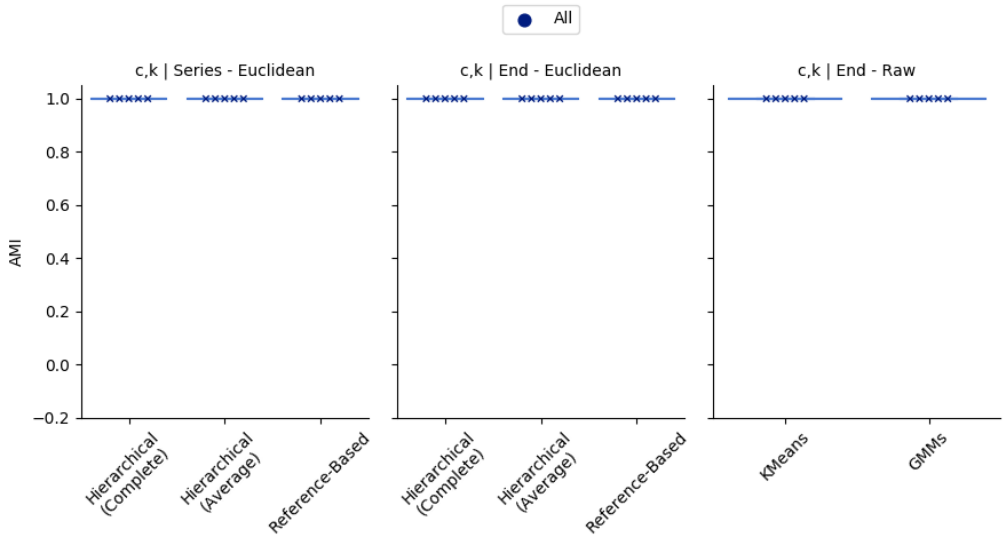


Figure 147: Game Variable Clustering for Full Dataset

It can be seen that when clustering the series data, all combinations resulted in perfect clustering. However, when considering only the end values, the 'x', 'y' combination failed to find perfect clusters. This once again makes logical sense, as was the case for the game 'Black Smoke', as on completion, the end position of the player was always the same. This further showed that for some game variables the whole series of values was required not just the end values.

The final step in this analysis was to evaluate clustering on the learnt representation spaces. A subset of the generated results is shown in Figure 148, and it was noted that all encoder experimental setups were able to cluster perfectly.

Finally, Table 13 provides a summary of the clustering quality for different approaches experimented with, within the Hitman case study. This brings together the best results from clustering on frame representations, game variable series and end game variable values. Within the frame representation case, this includes the best result when using both, one or none of the proposed techniques. As regards the game variable clustering case, the best results are presented when both using and not using the proposed clustering algorithm. When multiple setups provide the same performance for a given category, one example setup is shown in the table.

As can be seen in the summary table, for all considered categories, at least one setup is able to perform perfect clustering.

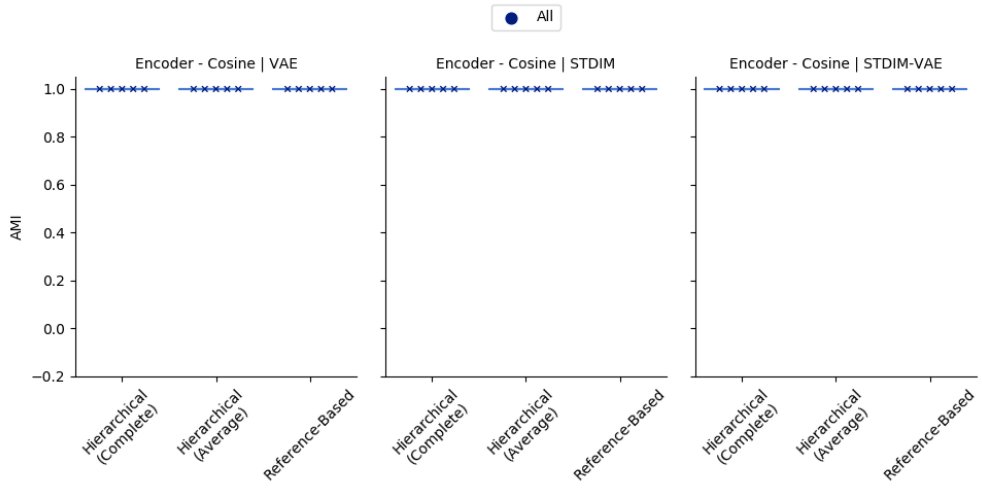


Figure 148: Representation Clustering for Full Dataset

Table 13: Hitman Clustering Quality Summary

Data Source	Distance Metric	Clustering Algorithm	AMI
STDIM-VAE	Cosine	Reference-Based	1.000 (±0.000)
STDIM-VAE	Cosine	Hierarchical (Average)	1.000 (±0.000)
ST-DIM	Cosine	Reference-Based	1.000 (±0.000)
ST-DIM	Cosine	Hierarchical (Average)	1.000 (±0.000)
c,k (Series)	Euclidean	Reference-Based	1.000 (±0.000)
c,k (Series)	Euclidean	Hierarchical (Average)	1.000 (±0.000)
c,k (End)	Euclidean	Reference-Based	1.000 (±0.000)
c,k (End)	Euclidean	Hierarchical (Average)	1.000 (±0.000)

5.3 Conclusion

The ability to successfully cluster playstyles from player data has many possible applications, as previously discussed. However, this was commonly achieved using game variable clustering. It has been shown within this chapter that this approach can run into many issues. Not only are systems required to extract and record the game variables, but a certain level of game knowledge would be required to select a good set of game variables. As would be expected, selecting bad game variables leads to bad clustering. However, picking a mixture of good and bad game variables also resulted in poor clustering.

Additionally, it has been shown that picking an incomplete set of game variables can result in certain aspects of the playstyle being missed. This meant that a human bias would be introduced in the game variable selection process, as this controlled what kind of playstyles were likely to be discovered.

On top of game variable selection, decisions were also needed to be made regarding the use of the whole data series or just the end values. As the whole series provided additional information, this often resulted in better clustering, however in some cases

this was not true. Thus, a further decision would be needed to be made around the game variables, where a bad decision can cause a reduction in clustering quality.

To overcome these problems, in this chapter it has been suggested that unsupervised representation learning techniques could be used to generate representation vectors for each frame to be used, instead of game variable vectors. These can be produced without human input and hence remove the human bias of selecting game variables.

Reference-Based Clustering has then been applied to these representation spaces to cluster playstyles. Overall, it appeared that the VAE and ST-DIM architectures each learn different aspects of the playstyle and hence each performed well on different subsets of the data. It was also shown that the use of the STDIM-VAE hybrid encoder allowed the representation to contain a mixture of both types of information and therefore often performed better on the whole dataset, especially when both aspects were required to divide the playstyles.

In conclusion, the hybrid encoder STDIM-VAE can be used to generate a useful representation space, for which good quality clusters can be found using Reference-Based Clustering. In two out of the three case studies explored, the combination of the STDIM-VAE and Reference-Based Clustering found the best quality clusters. In the other case study, while this was not true, it was observed that there was only a small difference in the clustering quality. Further, the setup that found the best quality clusters in this case study still used Reference-Based Clustering.

6 Imitating Playstyles

This chapter focuses on the development of a new technique that looks to learn to imitate different playstyles. In previous chapters, it has been shown that the combination of the STDIM-VAE and DTW matching, allows the creation of a metric that indicates the similarity of the playstyles within two gameplay videos. In the following sections a new method is introduced namely, Dynamic Time Warping Imitation (DTWI) [36], that utilised this metric to reward the system in order to learn to play with a given playstyle.

6.1 Dynamic Time Warping Imitation

As previously discussed, the distance between gameplays, calculated using Dynamic Time Warping on the learnt (STDIM-VAE) representation space, indicated the degree of similarity of the playstyles shown between two gameplays. Dynamic Time Warping Imitation aimed to learn a policy that created agent trajectories with the smallest distance to one of the demonstrations provided. The overall system architecture for DTWI is shown in Figure 149 with the purpose of each of these elements discussed below.

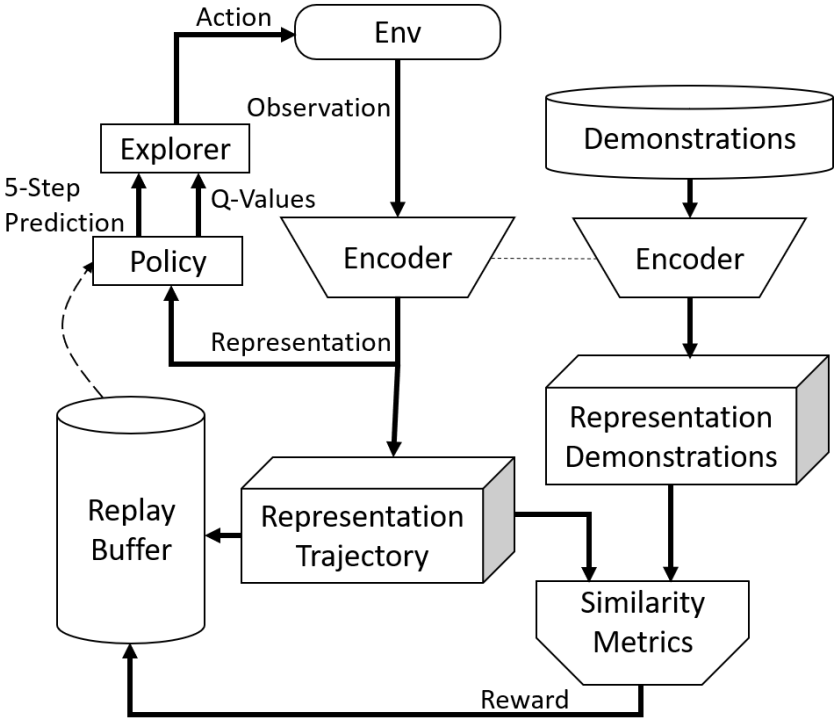


Figure 149: DTWI Architecture

6.1.1 Encoder

The first part of the system discussed is the encoder. The encoder was pre-trained using an unsupervised representation learning technique, in this case STDIM-VAE [35]. The learning system was rewarded based on the similarity of trajectories within this learnt

representation space. The STDIM-VAE architecture was chosen so that the system was rewarded based on the similarity of the playstyles shown.

In the system, the encoder which was used had two purposes. Firstly, on initialisation the encoder was used to transform the demonstration set into the representation demonstrations, these were then stored for later use. This was undertaken because the comparisons were performed in the representation space and hence pre-converting these meant that a transformation was not required for each comparison.

The second purpose of the encoder was to transform the observation, that was received from the environment, into a representation vector. This representation was then passed to the policy where the choice was made on which action should next be performed. This was also stored in the representation trajectory store. This was required as the episode needed to end before the comparison with the representation demonstrations could be conducted.

6.1.2 Similarity Metric

On completion of each episode, the representation trajectory store now contained the representations for the whole episode. After this, the trajectory representation sequence was compared against the representation demonstrations. Ideally, to find the true best match, the representation trajectory would be compared to all the demonstrations stored. However, to calculate this distance for all demonstrations, particularly if many demonstrations were provided, would make the process very slow. To help reduce this time requirement, a different approach was used to select an approximated best demonstration, before full matching was applied to calculate the distance. This approach first compared the initial representation of the representation trajectory with the first representation for each of the demonstrations. At this point the demonstration(s) with the smallest distance to the representation trajectory was (were) selected. If this extracted a set containing more than one demonstration, the previous process was repeated for the representation next in the sequence until only one demonstration was extracted. As mentioned, the full DTW matching process was then performed between the representation trajectory and the selected demonstration. This process then returned two results that were utilised to calculate a similarity score between the two trajectories. The first of these was simply a cost matrix showing distances between all combinations of the representations of the agent trajectory and the demonstration. The second returned result, expressed the matching path taken through the sequences. Figure 150 shows how the path, depicted as a dotted line, could be projected onto the cost matrix.

The next step was to decide on a method to extract a reward signal for the transitions between two sequential states. It was decided to explore two types of score values for each state and also investigate ways to combine these values to generate the reward score. These two score values were a state similarity score and a mismatch score. The state similarity score was defined as the average similarity to all the demonstration states that the trajectory state had been matched to on the found path. To do this, firstly the dissimilarity scores that were returned in the matrix needed to be converted into similarity scores. Given that the distance metric used in this work was the cosine metric, this maximum distance was 2. This meant that a similarity score could be generated

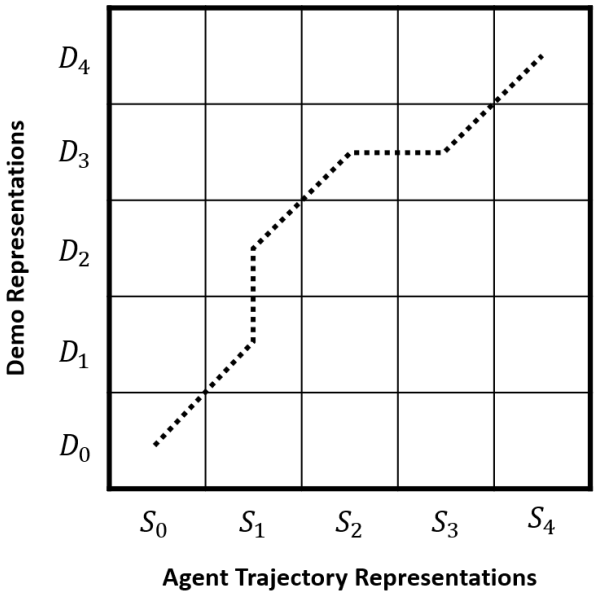


Figure 150: DTW Matching

by calculating the value $2 - dissimilarity$. Once this was calculated, the state similarity scores were found by compressing the path onto the trajectory. This meant that for the S_0, the state similarity score was simply the similarity between S_0 and D_0. Whereas the state similarity score for S_1 was the average similarity of S_1 and D_1 and S_1 and D_2.

On the other hand, the mismatch score looked to evaluate whether the trajectory was making progress through the demonstration. In order to achieve this, the index of both the trajectory representation and demonstration representation was monitored along the found path. Progression along the demonstration was shown by the increment of both the index of the trajectory representation and demonstration representation, this can be seen in Figure 150, as the path going diagonally into the next square.

Initially, only the state similarity value was extracted. However, the introduction of additional trajectory states, beyond the number of demonstration states, allowed the addition of further rewards. Although, these further states would most likely cause a worse overall match. This led to the extraction of the mismatch score. Based on the fact that the cosine metric was used, the maximum additional reward for a further state would again be 2 and hence the penalty for the mismatch should be greater than 2. It was decided to penalise the state by 2.1, if any mismatches were found.

Three different combinations of these score values were used as the reward in the experimental imitation of the ‘Camouflage’ playstyle in the Hitman game ‘Base’ level. An initial set of algorithm parameters was selected as shown in Table 14 and only the reward signal components were changed. For each of these combinations, three runs were conducted to evaluate the performance of each. In Figure 151, abbreviations were used to identify each of these reward components. The first two of these were CS and NS which represented the current state similarity and next state similarity. The final two abbreviations were CM and NM which represented the current state mismatch and next state mismatch. When multiple scores are listed on the figure, the reward was the sum of all of these scores.

Table 14: DTWI Parameters Fixed During Experiments

Parameter	Value
Learning Steps	50000
Encoder Layers Frozen	5 (All Layers)
Network Head Architecture	Two Dense Layers of 100 Neurons
Batch Size	32

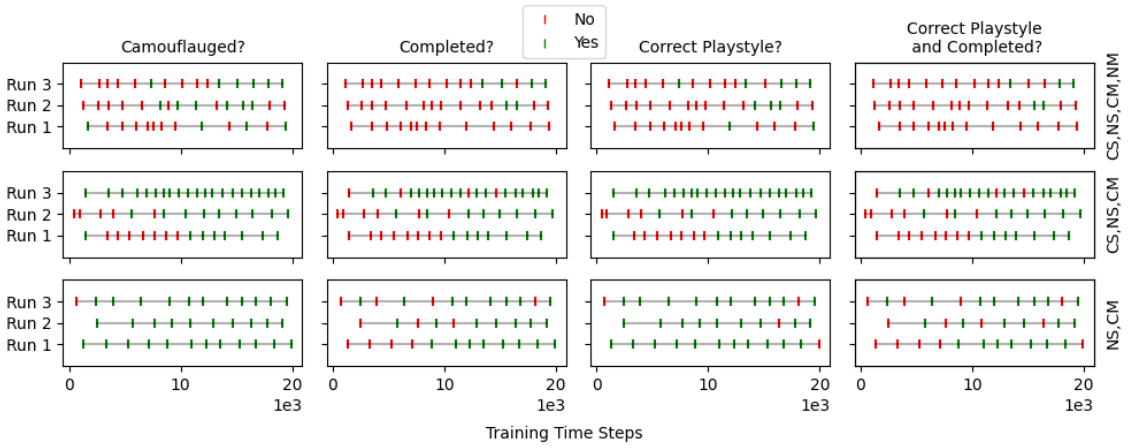


Figure 151: Comparison of Metrics for Different Q-Value Reward Components

Firstly, when looking at these options it can clearly be seen that the 'CS,NS,CM,NM' setup performed worst compared to the other setups considered. Next, the other two setups were examined. When investigating the 'Camouflaged' metric, it can be seen that the 'NS,CM' setup appeared to be quicker to reliably acquire the camouflage package and hence a similar reliability was seen in the use of the correct playstyle, as can be seen by the 'Correct Playstyle' metric. After this the 'Completed' and 'Completed with Correct Playstyle' metrics were examined and in this case it can be seen that more oscillation between success and failure was evident. Based on the quicker trend to collect the camouflage package, it was decided to define the reward, for the transition between S_x and S_{x+1}, as a combination of the next state (S_{x+1}) similarity and the current state (S_x) mismatch. This ensured not only that the trajectory moved away from the current state, but also towards the next state in the demonstration.

6.1.3 Policy

In order to generate the full agent representation trajectory, a policy must be defined to decide which action to take, given the current situation. As previously mentioned, the raw frame observation was first passed through the encoder to transform it into a representation. A policy was then trained to select the best action based on the representation observation. The policy network took the form of two dense layers each with 100 neurons, as shown in Figure 152.

Initially, the policy aimed only to predict the Q-value of each possible action. However,

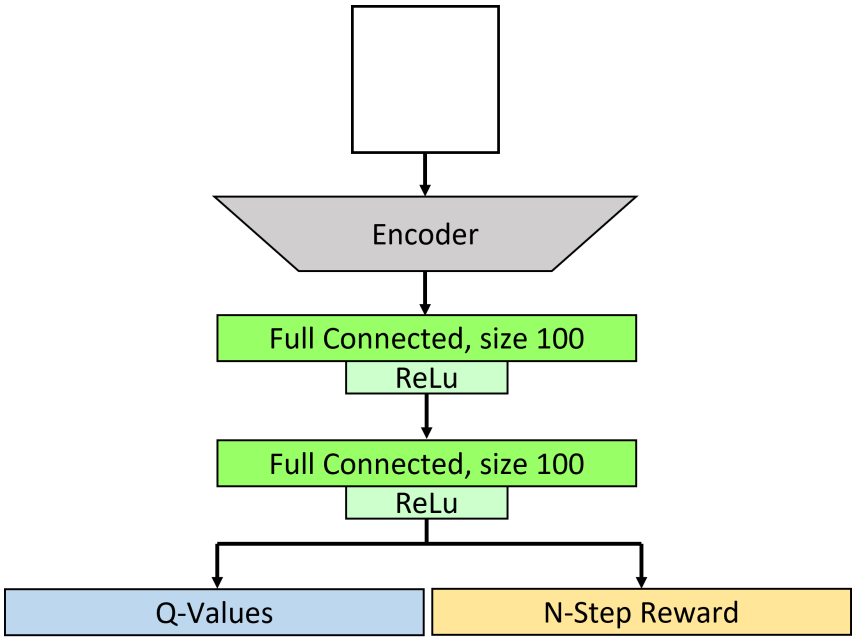


Figure 152: Policy Architecture

auxiliary predictions were also experimented with, as it was considered that these could be integrated into the explorer, which is explained below, to help speed up convergence. The auxiliary prediction settled on was a 5-step reward, which was the discounted sum of the next five rewards. However, the end reward was repeatably used to ensure five rewards were always used in this calculation. This was because the maximum reward per step was 2 and hence repeating the last reward, if required, ensured that the maximum theoretical reward was always possible. These experiments firstly used the same reward components for the 5-step reward as that chosen for the single step reward. It was decided to explore other reward options, as these predictions served a different purpose and hence would likely benefit from different reward components. Once again, three runs were completed for each component combination and the results can be seen in Figure 153.

Based on this investigation, it was decided to set the auxiliary reward components to be only the current state similarity score ('CS'). This was chosen, as although the training timesteps required for each method to regularly find the camouflage were only slightly better, the improvement for level completion was significantly better. This improvement was also seen for the completion with the correct playstyle. The final step in this process was to evaluate the performance of the method with and without the auxiliary reward predictions. Again, three runs were completed for each setup and the results are shown in Figure 154.

On examination, it can be seen that as before the speed with which the camouflage was reliably acquired was similar in both methods. However, there was a significant difference when considering how often the agent completed the level. It can be seen that the N-Step method outperformed the Q-Value method. This pattern was also seen when examining the ability to complete the level with the correct playstyle.

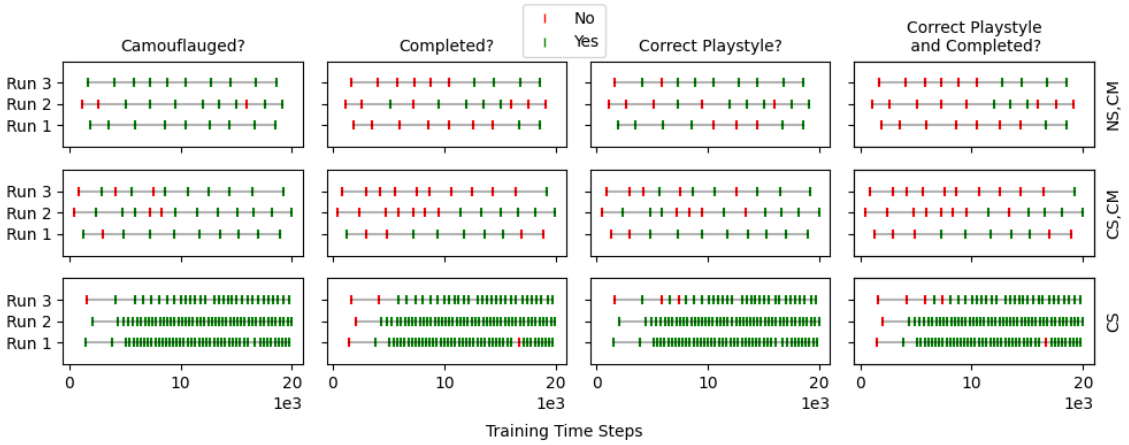


Figure 153: Comparison of Metrics for Different N-Step Reward Components

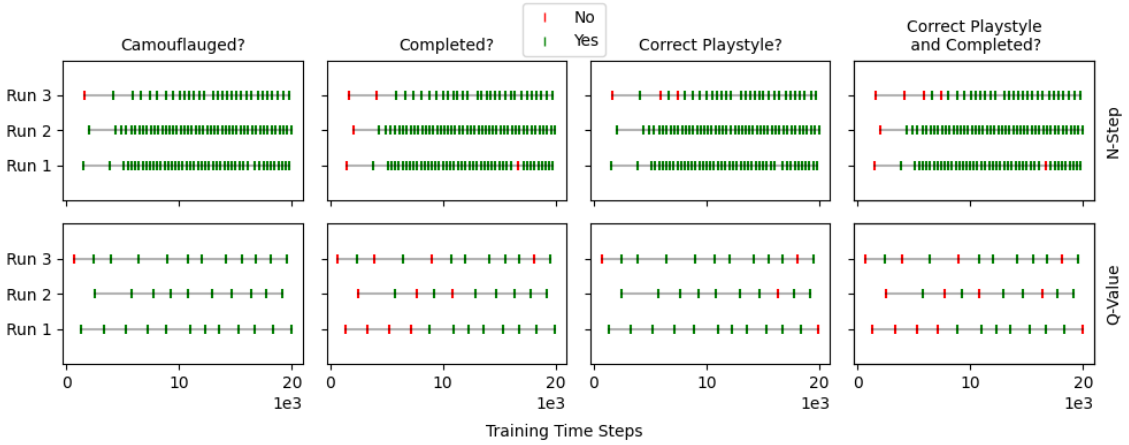


Figure 154: Comparison of Metrics for the Best Q-Value and N-Step Reward Components

6.1.4 Explorer

The final version of the policy network made two predictions, namely Q-values and N-step rewards. The explorer unit received both values and aimed to adaptively choose an ε value, to balance exploration and exploitation. As discussed above, the definition of the reward signal meant that there was a maximum reward of 2 per step and hence a maximum theoretical value for the 5-step reward, if a perfect match was achieved, could be calculated. Further, repeated use of the last reward when the episode terminated before five steps, ensured that this value was always obtainable at all points in the episode. It was hence expected that the reward prediction would converge to the theoretical maximum reward throughout training, and this can be seen in Figure 155. Although, it should be noted that this convergence happened quicker for timesteps earlier in the episode. This made sense, as the agent must successfully imitate all the previous timesteps to arrive at a situation from which the next five actions can be correctly copied.

As the reward prediction converged towards the theoretical maximum reward, it was

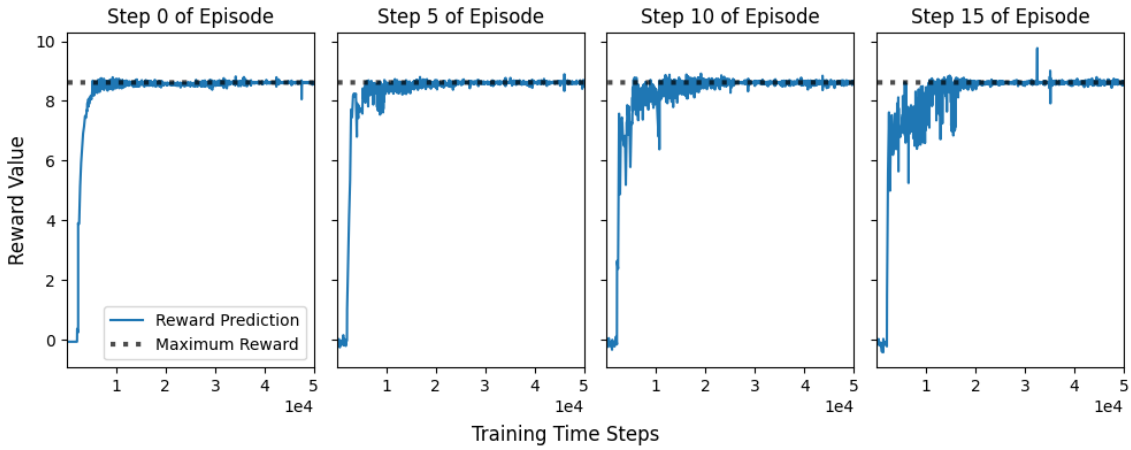


Figure 155: Reward Prediction at Different Points in the Episode

expected that the agent had become better at imitating the next five steps and hence the level of exploration could be decreased. To achieve this, a prediction based ε value was chosen based on a score value. This score value evaluated the difference between the maximum score, R_{max}, the prediction score N_a for action a which is the action with the highest Q-value, as shown in Equation 13. A set of conditions were then set out which translated this score value into an ε value, as shown in Equation 14.

$$s = \left| \frac{N_a}{R_{max}} - 1 \right| \quad \text{where} \quad a = \underset{x}{argmax}(Q_x) \tag{13}$$

$$\varepsilon_p = \begin{cases} 0.8, & \text{if } s > 0.9 \\ 0.7, & \text{if } 0.7 < s \leq 0.9 \\ 0.6, & \text{if } 0.5 < s \leq 0.7 \\ 0.5, & \text{if } 0.4 < s \leq 0.5 \\ 0.4, & \text{if } 0.3 < s \leq 0.4 \\ 0.2, & \text{if } 0.2 < s \leq 0.3 \\ 0.1, & \text{if } 0.1 < s \leq 0.2 \\ 0.05, & \text{if } 0.05 < s \leq 0.1 \\ 0.0, & \text{otherwise} \end{cases} \tag{14}$$

In addition to the prediction based ε, an ε value was also calculated, that linearly decayed throughout training. Once each of these values was calculated the final ε value was chosen based on the following steps. Firstly, the ε value was held high, at 0.9, for the first n steps of training. This aimed to encourage more exploration at the beginning of the training process. Following this, for all future training steps, the lower value between the prediction based ε and linearly decayed ε was chosen. Generally, the prediction based ε decayed quicker. However, the aim was to imitate a playstyle on an altered version of the environment and hence a perfect match was not always expected to be possible. In this case, the linearly decayed ε ensured that the ε values continually decayed through the training. Although, this could be done at a quicker rate if the 5-step reward predictions had a high enough quality and hence this allowed a

quicker shift to exploitation. An example of how the ε value decayed throughout the training process is shown in Figure 156. This ε value was then used within the ε greedy approach, to select the next action to take.

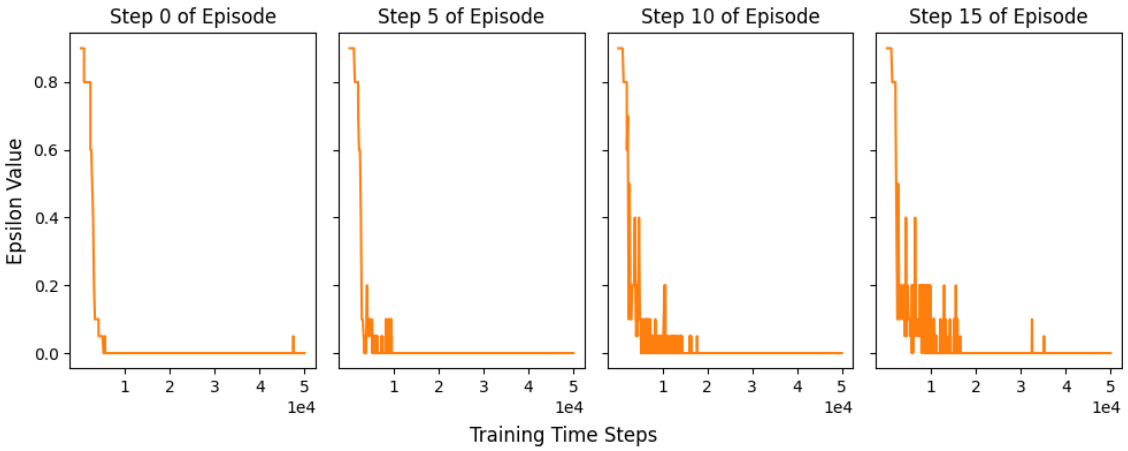


Figure 156: Epsilon Value at Different Points in the Episode

As can be seen, throughout the training process, the reward predictions converge to the theoretical maximum reward of $\sum_{n=0}^{N-1}(\gamma^n \times 2)$ and hence the ε value converged towards zero. This allowed the agent to exploit what had been learnt.

6.1.5 Replay Buffer

After an episode had been completed, the whole episode would be contained in the representation trajectory store. At this point the reward values were calculated using the similarity metric. These scores were then paired with each transition and added to the replay buffer. Samples could then get pulled from the buffer and used to update the policy using a standard RL approach.

6.1.6 Hyperparameter Tuning

Prior to the evaluation of each of these methods, a set of parameters required to be tuned. In this section the process used to achieve this tuning is discussed and an example of tuning the DTWI algorithm will be used to illustrate each step. The first step was to identify all the parameters that needed to be tuned and decide on suitable values that these parameters could take. The type of values these parameters could take, depended on the type of parameter, whether the parameter was categorical, binary, discrete or continuous. An example for each of these parameter types for the tuning of the DTWI algorithm can be seen in Table 15.

After all the possible parameter values had been defined, the next step was to apply tuning. To achieve this the Weights & Bias platform (wandb) [11] was used to tune using a Bayesian based search strategy. This strategy used a Gaussian Process (GP) to model the relationship between the algorithm's parameter set and the metric being optimised. In this case, the metric was the similarity between the agent and demonstration trajectories. This model was then used to iteratively choose parameter

Table 15: Examples of Parameter Types and Values for DTWI

Parameter Name	Parameter Type	Parameter Values
Batch Accumulator	Categorical	Mean or Sum
Double Network	Binary	Yes or No
Epsilon Min Steps	Discrete	0 or 10 or 20
Gamma	Continous	Between 0.9 and 0.997

sets to attempt to improve this metric. The tuning process used 3 agents that ran in parallel and performed 120 runs in total to tune the parameters. Throughout each of these runs, at regular intervals, the agents generated a trajectory without exploration, and this was compared to the demonstration set provided. The similarity score was then logged. On the completion of tuning, the runs were sorted based on these logged similarity values. Initially, they were sorted based on the last logged similarity. If however multiple runs shared the same last similarity value, then the previous similarity log value was used to further sort the runs. This comparison process was repeatedly used until all runs had an individual rank. Based on these rankings, the top five performing parameter sets were chosen.

For each of the 5 highest ranking runs, the parameters were extracted and rounded to 4 significant figures. For each of these rounded parameter sets, a further 3 runs were conducted. This was done to check the stability of training under each of the parameter sets. The parameter sets were then re-ranked based on the average similarity of these repeated runs. Again, if multiple parameter sets ended with the same similarity score, the previous similarity logs were used to further refine the ranking of the parameter sets. Table 16 shows the end similarity of each of these runs for each parameter set to 3 significant figures, as well as the average similarity for the DTWI algorithm. Based on these results the parameter set 1 was chosen.

Table 16: Final Similarity Values for Parameter Refinement for DTWI

Parameter Set Number	Similarity			
	Run 1	Run 2	Run 3	Average
1	0.000746	0.000729	0.000746	0.000740
2	0.00229	0.00232	0.000735	0.00178
3	0.334	0.000775	0.000746	0.112
4	0.000775	0.0122	0.000785	0.00458
5	0.441	0.0880	0.0617	0.197

As previously mentioned, this process was conducted for all the algorithms used and the final parameter sets for all of the algorithms can be found in Appendix C. Further, the final similarity for all the chosen parameter sets was examined and are shown in Table 17. This was to ensure that all chosen parameter sets resulted in correct imitation, this can be identified by a low distance score and on examination this was indeed the case for all algorithms.

Table 17: Comparison of Final Similarity Value for each Method after Parameter Refinement

Method	Parameter Set Number	Average Similarity of Repeats
BC	4	0.000733
GAIL-T	1	0.000730
GAIL	4	0.000759
GAIFO	2	0.000742
GAIFO-S	2	0.00525
DTWI	1	0.000740

6.2 Hitman - Case Study

In this section, the results are presented when attempting to imitate each of the playstyles on the different level variants. When designing each of the level variants, the aim was to challenge a certain playstyle and hence this analysis focused on these playstyle / level variant pairs. Figure 157 shows the success level for the imitation of these pairs and it can be seen that generally Dynamic Time Warping Imitation outperforms all other tested methods in these situations.

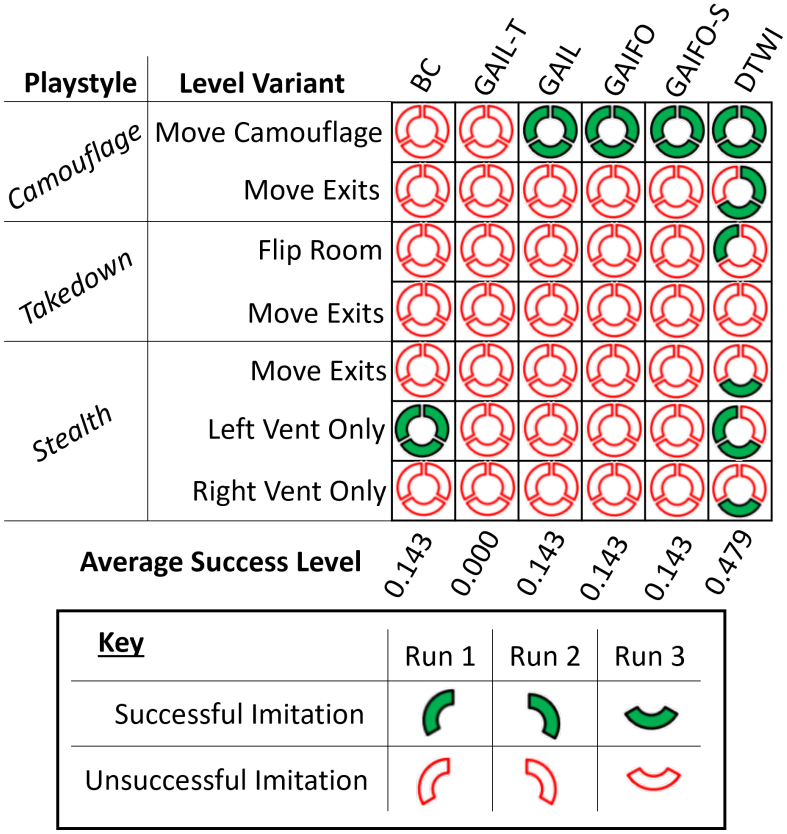


Figure 157: Overview of Imitation Success

As can be seen, when averaged over these playstyle / level variant combinations,

the DTWI outperformed all other considered methods. In the following sections these combinations will be explored in more detail. This was achieved by first looking at tables that displayed the three game variables, namely 'Camouflage', 'Takedowns', and 'Completed'. Three training runs were undertaken for each method and the averages are shown in the Tables 21 through 27. In addition, the Ground Truth (GT) values for the game variables are shown at the top of each table, for the playstyle being considered. Further, two graphs are shown to display, firstly how successful each imitation was at maintaining the playstyle and secondly if the agent can maintain the playstyle and complete the level.

6.2.1 'Base' Level

The initial experiments attempted imitation of the playstyle on the same level that the demonstrations were collected. This aimed to establish a baseline on how well each method could imitate these playstyles when the demonstration and imitation levels were the same. The end game variables after each imitation are shown in Table 18 to 20 for each of the different playstyles.

Table 18: Game Variables for Imitating 'Camouflage' Playstyle on 'Base' Level

Method	Game Variables ($\pm$SD)		
	Camouflage	Takedowns	Completed
GT	1.00	0.00	1.00
BC	1.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00
GAIL-T	1.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00
GAIL	1.00 ± 0.00	0.00 ± 0.00	0.67 ± 0.47
GAIFO	1.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00
GAIFO-S	1.00 ± 0.00	0.00 ± 0.00	0.67 ± 0.47
DTWI	1.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00

Table 19: Game Variables for Imitating 'Takedown' Playstyle on 'Base' Level

Method	Game Variables ($\pm$SD)		
	Camouflage	Takedowns	Completed
GT	0.00	2.00	1.00
BC	0.00 ± 0.00	2.00 ± 0.00	1.00 ± 0.00
GAIL-T	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIL	0.00 ± 0.00	1.00 ± 0.82	0.00 ± 0.00
GAIFO	0.00 ± 0.00	0.33 ± 0.47	0.00 ± 0.00
GAIFO-S	0.00 ± 0.00	2.00 ± 0.00	0.00 ± 0.00
DTWI	0.00 ± 0.00	1.67 ± 0.47	0.67 ± 0.47

On examination of the 'Camouflage' playstyle game variables, it can be seen that all methods were able to both acquire the camouflage package and avoid taking down any Non-Player Characters (NPCs). However, unlike all the other methods considered,

Table 20: Game Variables for Imitating 'Stealth' Playstyle on 'Base' Level

Method	Game Variables ($\pm$SD)		
	Camouflage	Takedowns	Completed
GT	0.00	0.00	1.00
BC	0.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00
GAIL-T	0.00 ± 0.00	0.00 ± 0.00	0.33 ± 0.47
GAIL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIFO	0.00 ± 0.00	0.33 ± 0.47	0.00 ± 0.00
GAIFO-S	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
DTWI	0.00 ± 0.00	0.00 ± 0.00	0.33 ± 0.47

both GAIL and GAIL-S were not able to complete this level on every run, although, they do complete the level on two out of the three runs. Interestingly, when considering the GANs based methods, the two methods that do not complete on every run, do not provide the next state as an input to the discriminator.

Next, when considering the game variables for the 'Takedown' playstyle, it was seen that all methods avoided acquiring the camouflage. Further, all but the GAIL-T method performed some takedowns, but not always the two takedowns shown in the demonstrations. It was noted that only BC and DTWI could complete this level, with BC completing in all three runs and DTWI only completing twice.

Similarly, for the 'Stealth' playstyle, all methods were able to avoid acquiring the camouflage package. Additionally, most methods could avoid taking down any NPCs, although GAIFO performed one takedown on one run. In this case, only BC, GAIL-T and DTWI were able to complete the level. BC completed on all three runs, whereas the other two methods only completed once.

After examining the game variables, the next step was to evaluate whether the playstyle was correctly used and also whether it was used to complete the level. Figure 158 shows both of these metrics and from this it can be seen that the most successful imitation was seen for the 'Camouflage' playstyle. Two possible reasons for this have been identified or indeed it could be that a combination of these reasons resulted in successful imitation. Firstly, parameter optimisation was conducted on this playstyle / level pair and hence this suggested that potentially the selection of parameters could be dependent on the playstyle being imitated. Secondly, that the 'Camouflage' playstyle was easier to imitate. This was thought to be the case as once the camouflage package had been acquired, the player could simply walk through the rest of the level. On the other hand, the other two playstyles required timing and hence skill, to either takedown or avoid NPCs for the 'Takedown' and 'Stealth' playstyles respectively. Further, it was shown that BC was the only method able to always imitate all playstyles successfully. This shows that BC is still a good candidate when two conditions are met, namely the imitation and demonstration levels are the same and the environment is deterministic.

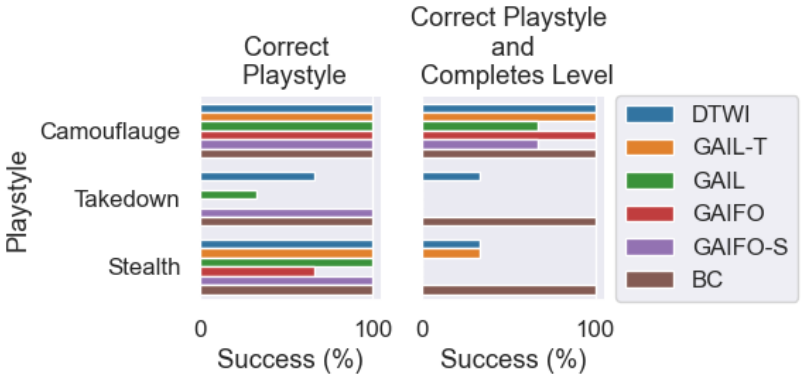


Figure 158: Success of Imitation on 'Base' Level

6.2.2 'Move Camouflage' Level Variant

The next step was to investigate the various methods' abilities to imitate the playstyles on different level variants. The first level variant explored was the 'Move Camouflage'. In this investigation, focus was given to the imitation of the 'Camouflage' playstyle, as this was the playstyle that the level variant was designed to challenge. Within this level variant, the camouflage package was moved by two squares and hence the player must recognise the camouflage package location and move to collect it, instead of just following the action sequence shown in the demonstrations. The end game variable values for imitation of this playstyle level / variant pair can be seen in Table 21.

Table 21: Game Variables for Imitating 'Camouflage' Playstyle on 'Move Camouflage' Level Variant

Method	Game Variables ($\pm$SD)		
	Camouflage	Takedowns	Completed
GT	1.00	0.00	1.00
BC	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIL-T	0.33 ± 0.47	0.00 ± 0.00	0.00 ± 0.00
GAIL	1.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00
GAIFO	1.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00
GAIFO-S	1.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00
DTWI	1.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00

Based on the end game variables, a few observations were noted. Firstly, BC never acquired the camouflage package, and on further investigation it was found that the agent just copied the action sequence from the demonstrations. Not only did this result in the camouflage package not being acquired, but the player was also caught by the enemy in 4 steps. Further, the GAIL-T method only acquired the camouflage package on one of the three runs. However, even on that occasion when the camouflage was collected, the level was not completed. Apart from these two methods, all other methods considered could find and collect the camouflage package and additionally, they also completed the level.

Further, Figure 159 shows the ability of each method to imitate the playstyle, as well as completing the level while using that playstyle.

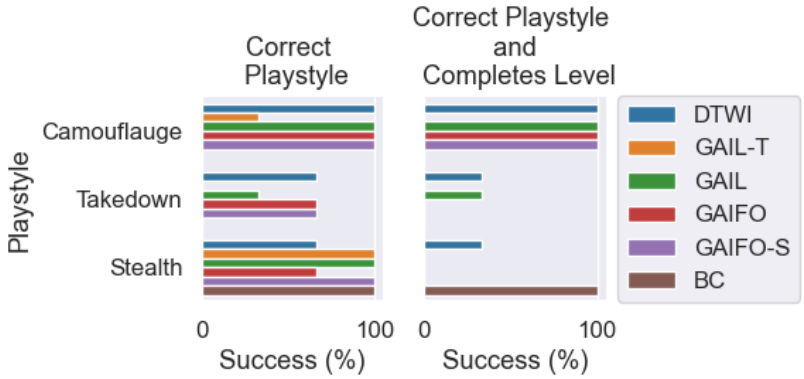


Figure 159: Success of Imitation on 'Move Camouflage' Level Variant

Interestingly, it can be seen that only DTWI was able to correctly imitate a playstyle and complete the level at least once for all of the playstyles considered. Further, it was seen that BC was only able to successfully imitate the 'Stealth' playstyle and complete the level. This meant that even for a playstyle where the camouflage was not required, such as the 'Takedown' playstyle, merely moving the camouflage package can cause this method to fail to imitate the playstyle correctly. This showed that while BC was a good method for imitating when the demonstrations were collected from the same environment from where the imitation had taken place, small changes in the environment could cause BC to fail to imitate correctly.

6.2.3 'Flip Room' Level Variant

The next level variant considered was the 'Flip Room' variant. In this level the second room was flipped so that it expanded out to the left instead of to the right. This changed the path the player had to take to get behind the second NPC and hence challenged the 'Takedown' playstyle most, as it was designed to do. The end game variables for this playstyle level variant combination can be seen in Table 22.

Table 22: Game Variables for Imitating 'Takedown' Playstyle on 'Flip Room' Level Variant

Method	Game Variables (±SD)		
	Camouflage	Takedowns	Completed
GT	0.00	2.00	1.00
BC	0.00 ± 0.00	1.00 ± 0.00	0.00 ± 0.00
GAIL-T	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIL	0.00 ± 0.00	0.33 ± 0.47	0.00 ± 0.00
GAIFO	0.00 ± 0.00	0.33 ± 0.47	0.00 ± 0.00
GAIFO-S	0.00 ± 0.00	1.00 ± 0.00	0.00 ± 0.00
DTWI	0.00 ± 0.00	1.33 ± 0.47	1.00 ± 0.00

It can be seen that GAIL-T was unable to perform any takedowns. On further examination, with the exception of the DTWI method, all other methods never performed more than 1 'Takedown'. This was most likely because of one of two reasons, either the agent struggled to get behind the second NPC or became 'confused' once behind the NPC as it had to attack from a different direction. However, when examining the DTWI results, it was found that on one run both takedowns were performed, while on the other two runs only one takedown had been performed. Although, more interestingly, it was found that this second 'Takedown' was performed on the second NPC and that the agent had been able to get behind this NPC and perform the takedown.

Interestingly, from the game variables, it can be seen that the DTWI method always completed the level. However, when evaluating the correctness of the playstyle, as shown in Figure 160, only one run completed with the correct playstyle. This suggested that there was some inconsistency in how the DTWI method had imitated this playstyle. As previously mentioned, on two of these runs, only a single takedown had been performed, although the level was completed in both of these runs. This meant that the playstyle was not exactly being maintained, however it was still similar.

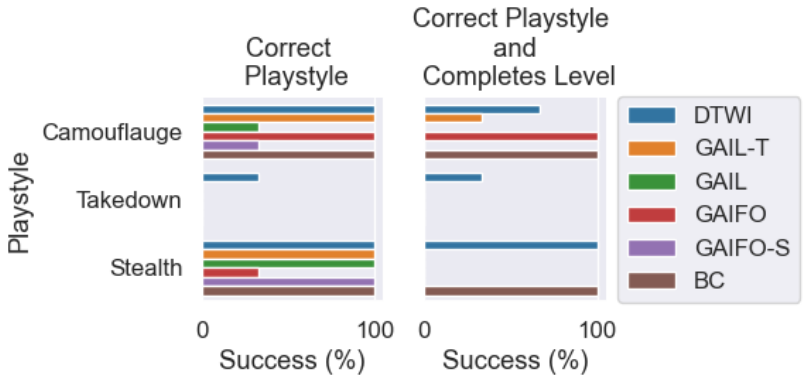


Figure 160: Success of Imitation on 'Flip Room' Level Variant

6.2.4 'Move Exits' Level Variant

Here the 'Move Exits' level variant was explored. In this case, the room exit, and level exit were both shifted by a single square. These changes were designed to challenge all the playstyles equally. The end game variables for all the playstyles for this level variant can be seen in Table 23 to 25.

Firstly, for the 'Camouflage' playstyle, while all DTWI runs were able to acquire the camouflage, only two of them were also able to complete the level. For the 'Takedown' playstyle, it can be seen that only one DTWI run successfully completed the level and the average number of takedowns was 1 across the three runs. On examination it was found that when the level was completed, only one takedown had been performed. Whereas, on a different run 2 takedowns were performed, however the level was not completed. Finally, for the 'Stealth' playstyle only the DTWI algorithm successfully completed this level. Overall, only the DTWI ever completed this level variant for any of the playstyles.

Table 23: Game Variables for Imitating ‘Camouflage’ Playstyle on ‘Move Exits’ Level Variant

Method	Game Variables (±SD)		
	Camouflage	Takedowns	Completed
GT	1.00	0.00	1.00
BC	1.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIL-T	1.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIFO	1.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIFO-S	0.33 ± 0.47	0.00 ± 0.00	0.00 ± 0.00
DTWI	1.00 ± 0.00	0.33 ± 0.47	0.67 ± 0.47

Table 24: Game Variables for Imitating ‘Takedown’ Playstyle on ‘Move Exits’ Level Variant

Method	Game Variables (±SD)		
	Camouflage	Takedowns	Completed
GT	0.00	2.00	1.00
BC	0.00 ± 0.00	1.00 ± 0.00	0.00 ± 0.00
GAIL-T	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIL	0.00 ± 0.00	0.67 ± 0.94	0.00 ± 0.00
GAIFO	0.00 ± 0.00	0.67 ± 0.94	0.00 ± 0.00
GAIFO-S	0.00 ± 0.00	0.33 ± 0.47	0.00 ± 0.00
DTWI	0.00 ± 0.00	1.00 ± 0.82	0.33 ± 0.47

Following this, the correctness of the playstyle and combination of the correct playstyle and its ability to complete the level were evaluated and the results are shown in Figure 161.

As can be seen, only DTWI was able to complete the level while using the correct playstyle. However, this was only achieved for two out of the three runs for the ‘Camouflage’ playstyle and one out of the three runs for the ‘Stealth’ playstyle. Although, for the ‘Takedown’ playstyle, one of the runs completed the level but here the playstyle was not correctly maintained throughout and hence showed the same inconsistency discussed for the previous level variant, namely not performing both takedowns.

Table 25: Game Variables for Imitating ‘Stealth’ Playstyle on ‘Move Exits’ Level Variant

Method	Game Variables (±SD)		
	Camouflage	Takedowns	Completed
GT	0.00	0.00	0.00
BC	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIL-T	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIFO	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIFO-S	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
DTWI	0.00 ± 0.00	0.00 ± 0.00	0.33 ± 0.47

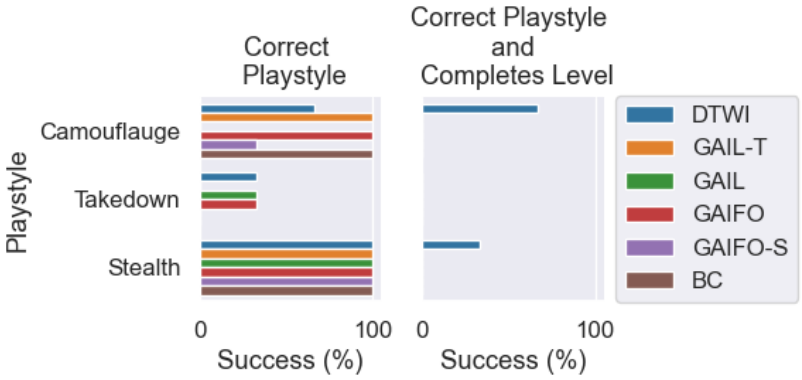


Figure 161: Success of Imitation on ‘Move Exits’ Level Variant

6.2.5 ‘Left Vent Only’ Level Variant

The ‘Left Vent Only’ level variant was then investigated. In this level the right vent system had been removed. Given that only the ‘Stealth’ playstyle used vents, the focus of this investigation was on the imitation of this playstyle. However, given that all the provided demonstrations for this playstyle used the left vent, it was expected to have little effect on the imitation. The end game variable values for the ‘Stealth’ playstyle are shown in Table 26.

In this case, a minor edit *i.e.* removing the right vent, from the level still allowed BC to complete the level and Figure 162 shows that this was also with the correct playstyle. This showed that only certain edits caused problems for BC. When considering the other methods that were able to imitate the playstyle on the ‘Base’ level, GAIL-T dropped from one completion to no completions. However, DTWI increased from one completion on the ‘Base’ level to two completions on this level variant.

Table 26: Game Variables for Imitating ‘Stealth’ Playstyle on ‘Left Vent Only’ Level Variant

Method	Game Variables (±SD)		
	Camouflage	Takedowns	Completed
GT	0.00	0.00	1.00
BC	0.00 ± 0.00	0.00 ± 0.00	1.00 ± 0.00
GAIL-T	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIFO	0.00 ± 0.00	0.33 ± 0.47	0.00 ± 0.00
GAIFO-S	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
DTWI	0.00 ± 0.00	0.00 ± 0.00	0.67 ± 0.47

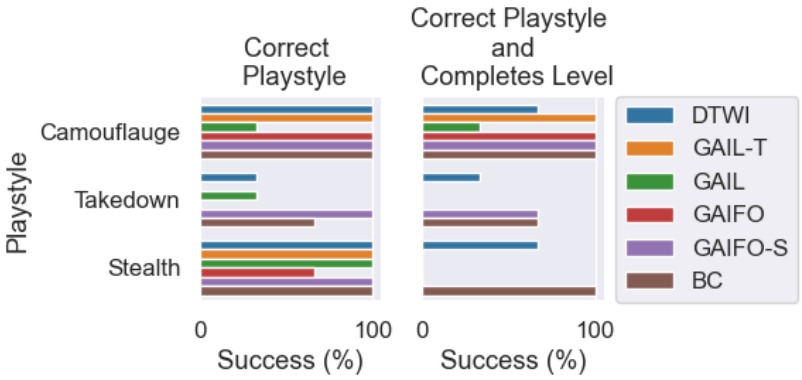


Figure 162: Success of Imitation on ‘Left Vent Only’ Level Variant

6.2.6 ‘Right Vent Only’ Level Variant

The last level variant considered was ‘Right Vent Only’. In this case, it was the left vent that was removed. Once again, as only the ‘Stealth’ playstyle used vents, the imitation of this playstyle was the focus. However, as previously mentioned, all the ‘Stealth’ playstyle demonstrations used the left vent and hence the method must find the right vent and then identify that using it would be consistent with the playstyle shown in the demonstrations. The end game variable values for the ‘Stealth’ playstyle are shown in Table 27.

For this level variant, only the DTWI method was able to complete the level. However, this was only achieved in one out of the three runs. Based on Figure 163, on this run the correct playstyle was also used. Out of interest, when the BC runs were examined, it was found that the player just stood in front of the location from where the left vent had been removed. This showed that the method was more likely to copy the action sequence from the demonstrations, even when the environment had been altered.

Table 27: Game Variables for Imitating ‘Stealth’ Playstyle on ‘Right Vent Only’ Level Variant

Method	Game Variables (±SD)		
	Camouflage	Takedowns	Completed
GT	0.00	0.00	1.00
BC	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIL-T	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIL	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIFO	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GAIFO-S	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
DTWI	0.00 ± 0.00	0.00 ± 0.00	0.33 ± 0.47

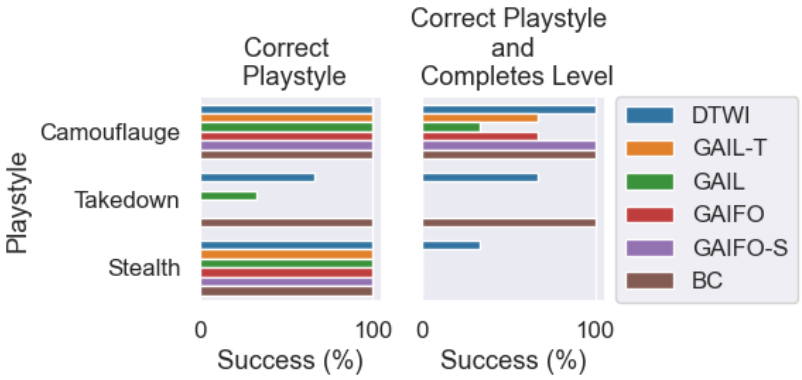


Figure 163: Success of Imitation on ‘Right Vent Only’ Level Variant

6.3 Conclusion

In the previous chapter, it has been shown that clustering techniques can extract a set of demonstrations that have the same playstyle. The next task was to train an agent to play with the playstyle shown within the particular demonstrations, even when the level that the imitation was attempting differed from the level from that which the demonstrations had been collected.

It has been shown that many common Imitation Learning techniques struggle to imitate playstyles successfully even when only minor alterations were made to the level. This issue arose for BC as this technique aimed to learn a policy to predict the action to be taken based on the current state. The change in the level altered the observation fed into the policy and hence the predicted actions were no longer correct.

On the other hand, other approaches used as a baseline, trained a policy by rewarding it based on a discriminator’s ability to identify the transitions as real or fake. Although, what was contained in the transition depended on the approach taken. These methods ran into an issue, as the discriminators were likely to give bad reward signals to transitions displayed during the playstyle. This was because the discriminators could identify the agent’s trajectories as fake based on changes in the level alone.

To solve this issue a new technique, the DTWI algorithm, has been suggested. This

method utilises the metric previously discussed to evaluate the similarity of the playstyle and rewards the policy based on it. This allowed the imitation of the playstyle on levels that differed from the demonstration level.

In conclusion, when examining the imitation of the playstyle where level variants were designed to challenge the playstyle, it has been shown that the DTWI algorithm outperformed all other methods that were used to baseline this method.

7 Applications Beyond Games

Up to this point, all the developed methods have been solely applied within the video game domain. However, it was expected that the proposed methods would have further applications beyond just games. Within this chapter, the previously developed unsupervised representation learning method was applied to the field of creativity. This will take the form of training a Spatiotemporal DeepInfomax Variational Autoencoder (STDIM-VAE), based on a set of videos that captured different models that have been created by members of the public. The idea being that the learnt representation space would contain key information about the model, and this could hence allow the automatic evaluation of the novelty of an individual model when compared to a set of created models [34].

7.1 Creativity Evaluation

Within the creativity field, research is still ongoing in the development of methods to evaluate creativity. The design of these evaluation techniques could be beneficial for many differing groups, for example within the creative industries or educational environment, *etc.* Although, research has led to many techniques assessing creativity across the 'four P's' (person, process, product and press) [60, 63].

It was noted however that humans were often required within the evaluation process. For example, the 'gold standard' in psychology for the evaluation of creative products used human experts [83]. Further, humans were commonly used as judges to evaluate computational creativity [63]. However, the use of humans within the evaluation process can lead to several issues. Firstly, the use of humans makes any evaluation, time intensive [63, 58] and hence these approaches are difficult to scale to large datasets. Further, having a small number of experts can lead to problems due to low levels of inter-rater reliability, although this can be improved by the inclusion of more experts. However, this naturally, amplifies the time requirement issue.

While more scalable evaluation techniques of human creativity have been developed (*e.g.* self-reports), the quality of these were often worse than human evaluations and hence were referenced as poor-but-scalable. Whereas human based evaluations were classified as good-but-expensive [60]. This meant that currently compromises are often required between the quality of the evaluation and the ease of its measurement.

This has led to discussion within the creativity community for the need to develop techniques for automated evaluation of creative products, as this would aim to be both good-and-scalable. While some research has been conducted to develop methods to evaluate human creativity [42, 59, 32, 114], research generally focused more on computational creativity [63].

Further, research in the automation of creativity evaluation has led to the experimental use of computational formalisations of creativity (*e.g.* novelty) [63]. In these cases, where works to be evaluated were not created digitally, it was hence necessary to transform these into a digital representation. From these representations, common creativity metrics (*e.g.* novelty) could then be extracted. It is suggested in this work that the STDIM-VAE could serve this purpose, as it allows a simple video of the created work (*e.g.* a model) to be transformed into a representation.

7.2 Interpretation Methods

When examining the use of the encoder within the video game domain, several analysis techniques were defined in Section 4.2. These analysis techniques were also used within this creativity domain, however due to this change, a couple of additional analysis techniques were also used and are discussed below.

7.2.1 Representation Transition

Firstly, particularly within the creativity domain, the aim was to allow interpolation to model designs that had not been seen during training. This was hoped to be achieved by having a smooth transition between models within the representation space and hence movement within this space, in different directions, should result in the alterations of different attributes of the model. To examine this, Representation Transition was used. In this analysis, two models were chosen and encoded into the representation space. The difference between these two representations was then calculated. Following this, sequential steps were taken to step between these two model representations. This resulted in a sequence of representations that would transition between the two models. Each of these representations was then fed into the decoder, which was trained as part of the VAE architecture of the hybrid encoder, to attempt to reproduce an image of the model for a given representation. If the desired smooth transition was present, then the reproduced image models should show a gradual morphing between these two models.

7.2.2 Novelty Detection

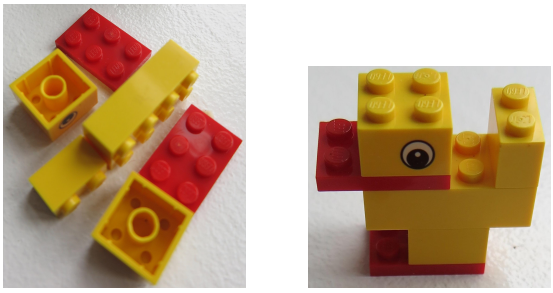
The second group of new analysis techniques introduced within this chapter looked to extract a score to evaluate the novelty of a model compared to the overall model set. This scoring could then be used as the automatic novelty rating system. Two different scoring metrics have been defined based on two different ideas. Firstly, ‘cluster-based novelty’, in this case standard clustering was applied to the models. Each of these clusters would then group together similar models, and hence models within small clusters, or even clusters containing a single model, could be thought of as having some degree of novelty within the dataset. Equally the inverse would be true, meaning that models within big clusters could be thought of as less novel.

Secondly, ‘distance-based novelty’ was also explored. It was expected that the closer two models were together in the representation space, the more similar they would be and hence a novel model would be more distant from all other models. To evaluate this, a score was calculated as the average distance from the model in question to all other models in the set.

7.3 Case Study - LEGO Duck Task

In order to explore this idea of automatic novelty evaluation, a task had to be chosen. It was decided to use the LEGO Duck Task, which henceforth will be called the ‘Duck Task’, in which participants were provided with six LEGO pieces and asked to ‘Build a Duck for Science!’. The six pieces provided to participants are shown in Figure 164a

and an example model from these six pieces can be seen in Figure 164b. As can be seen, the provided set contains: one 2×4 yellow brick, one 1×2 yellow brick, two 2×2 yellow bricks (one of which had an eye printed on it) and two 2×3 red plates.



(a) (b)

Figure 164: Six Lego Pieces (a) and Example Duck Model (b)

This creativity task was chosen for a number of reasons. Firstly, it was a fairly easy task to explain and could be understood across many different cultures. Secondly, the task was repeatable, this meant that knowing the solution did not affect future build attempts. Finally, the use of six LEGO pieces balanced the complexity of the task output, as these outputs were limited to the same six pieces but the combination of these pieces allowed for a wide range of models to be produced, as shown in Figure 165.

Figure 165: Duck Model Examples

It should be noted that the duck dataset was collected with collaborating partners at the 'Interacting Minds Centre' from Aarhus University, Denmark. This data was collected at an Open Science Event, where members of the public were asked to 'Build a Duck for Science!'. After the models were produced, they were then photographed from multiple angles. This was achieved by placing the duck on a *Foldio 360* turntable to allow the duck to be rotated. Further, a *Foldio 2* photography light tent was used to keep the light conditions constant. An iPhone with the *Foldio 360 app* (orangemonkey.com/app) was then used to take the required photos as the models were rotated through 360 degrees. This setup can be seen in Figure 166. These images were then stitched together to create a video for each model duck.

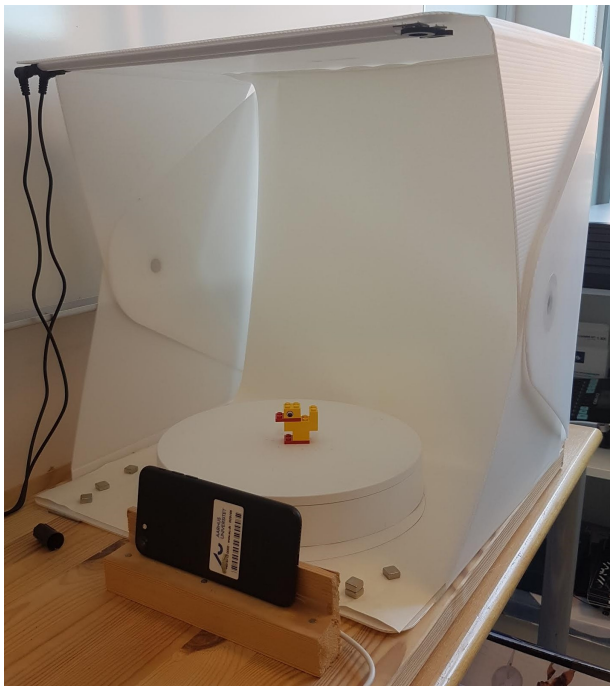
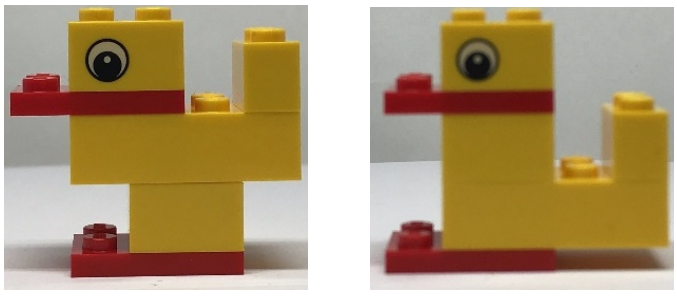


Figure 166: 3D Photograph Setup

Once collected, an initial review of the model designs within the dataset was conducted to gain an understanding of the designs that existed. This was required to allow future analysis. On reviewing, two duck model designs were commonly seen, these will be referenced as a ‘Standing Duck’ and a ‘Sitting Duck’, and are shown in Figures 167a and 167b respectively. When examined further, the identical model was seen a total of 9 times for the ‘Standing Duck’ and 9 times for the ‘Sitting Duck’ from a total population of 162 model ducks. However, in addition to these exact replicas, there also existed models that had only minor differences from the two aforementioned designs. Although, there appeared to be more of these for the ‘Standing Duck’ model, where there were thought to be more alternatives, such as shifting the placement of the eyes, legs, feet, *etc.*



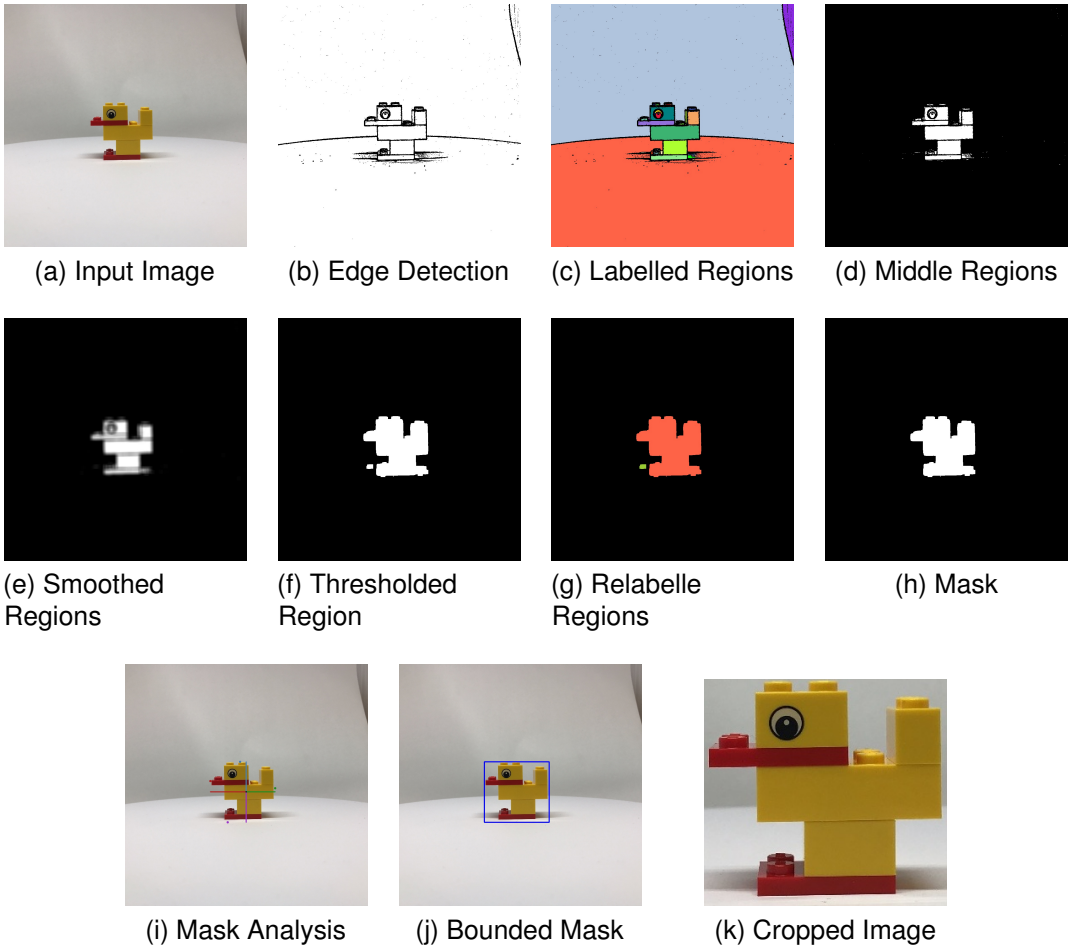
(a) ‘Standing Duck’ (b) ‘Sitting Duck’

Figure 167: Common Duck Model Designs

7.3.1 Image Pre-processing

Prior to using the duck model videos to train the hybrid encoder, pre-processing was applied. This was done for two main reasons. Firstly, the images were cropped to maximise the space the duck occupied within the frame, this also reduced the image size, but not its resolution. This hence reduced the compute requirement of the network, allowing faster training. Secondly, specific cropping was performed to align the models in the images to ensure that any dissimilarity found was due to changes in the model and not caused by image shifts from model placement. To achieve both of these aims, a multi step process was applied across the whole video set and is explained below. Additionally, an example of these steps is demonstrated on Duck 0 Frame 0, the raw input image being shown in Figure 168a.

Firstly, analysis was conducted at a frame by frame level to find an approximation of the size of the bounding box of each duck in each frame. In each frame, a colour-based edge detection method [22] was applied and then a threshold of 20 set. The output of this can be seen in Figure 168b. After which Connected Component Labelling (CCL) [109] was applied to label each of the regions, this is shown in Figure 168c. Next, the edge of each image was scanned, and a record made of all regions that touched the side of the image. Given that the duck was always fully contained within the image, then all these edge regions can be discarded as they did not contain any part of the duck. All the remaining regions were marked as possible sections of the duck, this is shown in Figure 168d. Following this, convolution was performed using a 20 by 20 filter populated with ones. This filter simply summed the pixels within the region, that were marked as possible parts of the duck model. The aim here was to merge the regions that contained each of the LEGO pieces. The resulting image was then remapped to the 0, 255 range to allow the visual representation to be shown and is shown in Figure 168e. The image was then re-thresholded but this time with a threshold of 10, this result is shown in Figure 168f. Next, these possible regions were re-labelled using CCL and shown in Figure 168g. The duck mask was then taken as the largest labelled region, as seen in Figure 168h. From this mask, the Centre of Mass (CoM) of the mask was calculated and recorded along with the distance from this CoM point to the edge of the mask, as shown in Figure 168i. These distances were then used to calculate the bounding box, as shown in Figure 168j. Further, Figure 168k shows how this image looked when cropped. This whole process was repeated for each frame from each video. An overall bounding box for the video set was found by taking the maximum distance in each direction. This overall bounding box was then used to crop each image from each video by aligning the centre of the bounding box with centre of the frame mask.



(a) Input Image (b) Edge Detection (c) Labelled Regions (d) Middle Regions

(e) Smoothed Regions (f) Thresholded Region (g) Relabelle Regions (h) Mask

(i) Mask Analysis (j) Bounded Mask (k) Cropped Image

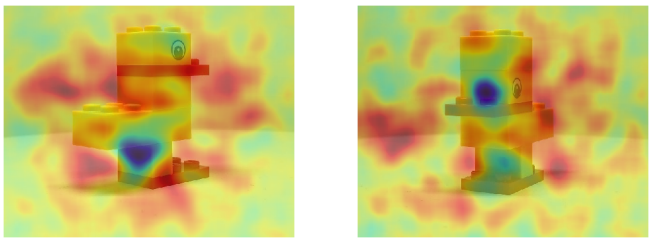
Figure 168: Image Pre-processing Steps

7.3.2 Explaining Duck Model Representations

Once the encoder had been trained for the duck dataset, the previously introduced, representation analysis techniques were applied. In this section the results of RISER analysis will be discussed and in the following section the RISERD analysis will be explored. The aim of this section was to investigate which parts of the duck models were being encoded into the representation space. This investigation led to the discovery of a variety of focus types. Interestingly, it was found that the type of focus could change through the image sequence of a model, as the model was rotated. One of the less interesting focus types just took a broad focus on the whole frame. Several examples of this focus type can be seen in Figure 169.

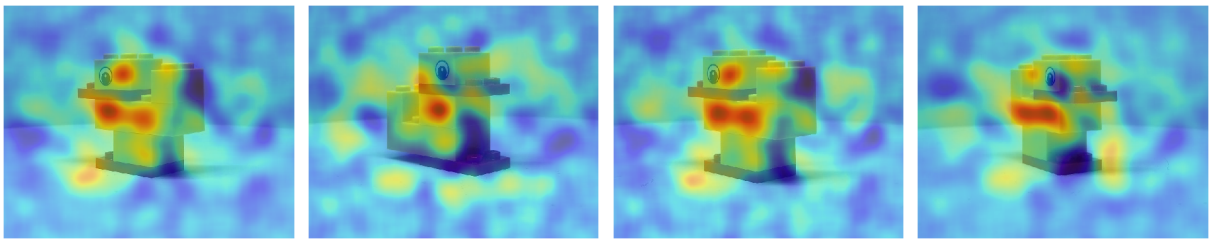
However, in some cases the focus was more on the central bricks of the duck model as shown in Figure 170. This information, even on its own, can provide a certain level of understanding about how similar any two models were.

It was recognised that this information could also be obtained by a different focus type which focused more on the edge of the duck model. However, this was often the top side of a brick, where the cylinder connector points were located, and hence allowed the extraction of the brick's position. Examples of this can be seen in Figure 171.



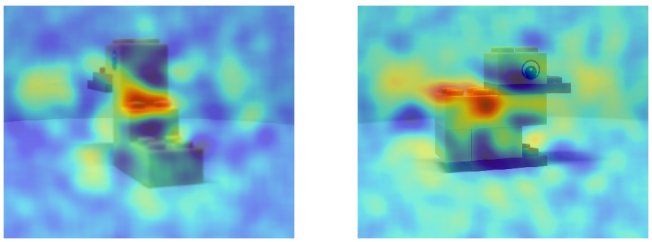
(a) Duck 35 Frame 9 (b) Duck 92 Frame 20

Figure 169: Importance Map Examples That Focus on the Whole Frame



(a) Duck 0 Frame 3 (b) Duck 25 Frame 15 (c) Duck 96 Frame 3 (d) Duck 124 Frame 16

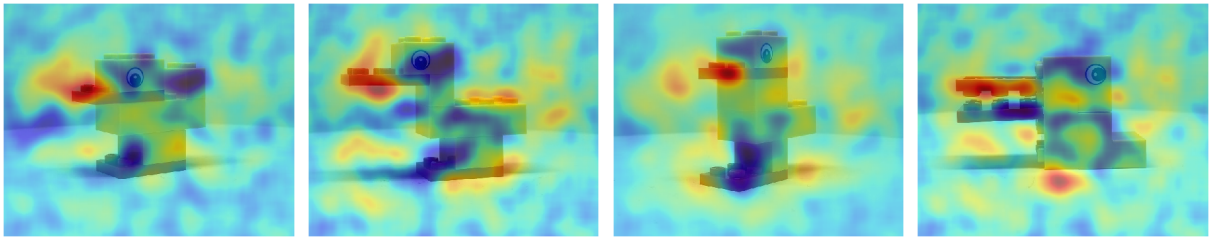
Figure 170: Importance Map Examples That Focus on Bricks



(a) Duck 9 Frame 5 (b) Duck 95 Frame 11

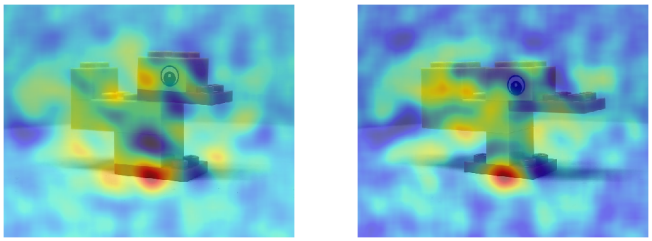
Figure 171: Importance Map Examples That Focus on the Edge of the Model

Further, in certain situations the focus was mainly on a certain anatomical part of the duck model. Based on visual inspection of these models, three of these anatomical parts were suggested, namely the duck's beak, feet and wings. Examples of each of these focus regions can be seen in Figure 172, 173 and 174 respectively.



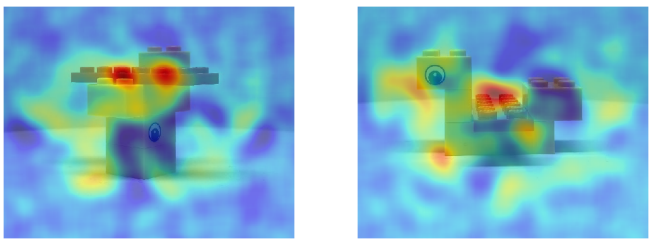
(a) Duck 0 Frame 22 (b) Duck 26 Frame 23 (c) Duck 75 Frame 21 (d) Duck 97 Frame 0

Figure 172: Importance Map Examples That Focus on the Duck's Beak



(a) Duck 1 Frame 14 (b) Duck 63 Frame 14

Figure 173: Importance Map Examples That Focus on the Duck's Feet

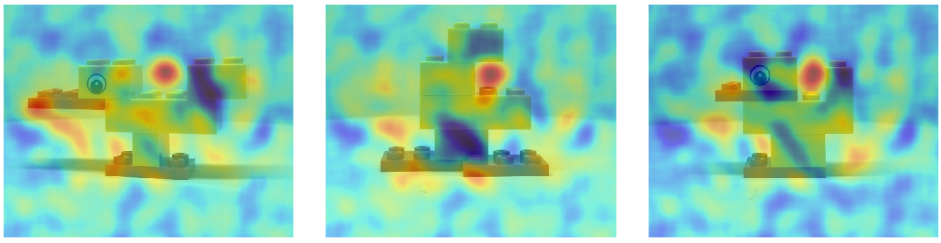


(a) Duck 43 Frame 21 (b) Duck 52 Frame 0

Figure 174: Importance Map Examples That Focus on the Duck's Wings

All focus types up to this point have directly focused on parts of the duck model. However, a couple of focus types appeared to be more interested in the negative space, *i.e.* where the model was not. This for example can take the form of focusing on a gap between two bricks, as shown in Figure 175. Similarly, when the focus was on the central bricks of the duck model, this gap position could provide a basic understanding of the shape of the duck and hence its similarity to other models.

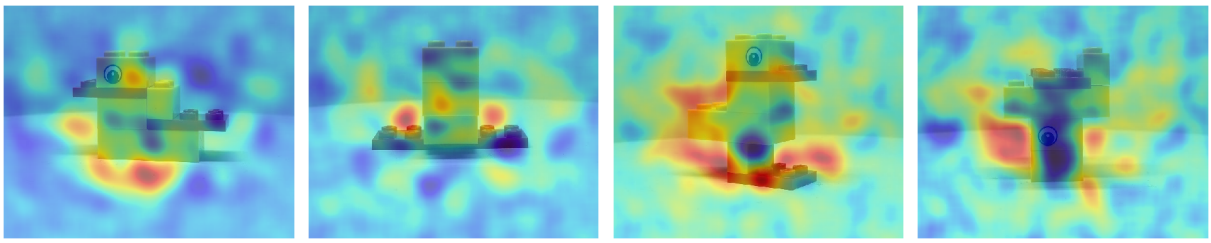
Although, this single negative space point could not provide a full understanding of the duck model shape, in some cases this shape appeared to be fully extracted, as the



(a) Duck 70 Frame 1 (b) Duck 94 Frame 0 (c) Duck 96 Frame 0

Figure 175: Importance Map Examples That Focus on Gaps

focus regions provided a rough bounding of the model. A few examples of this focus type are shown in Figure 176.



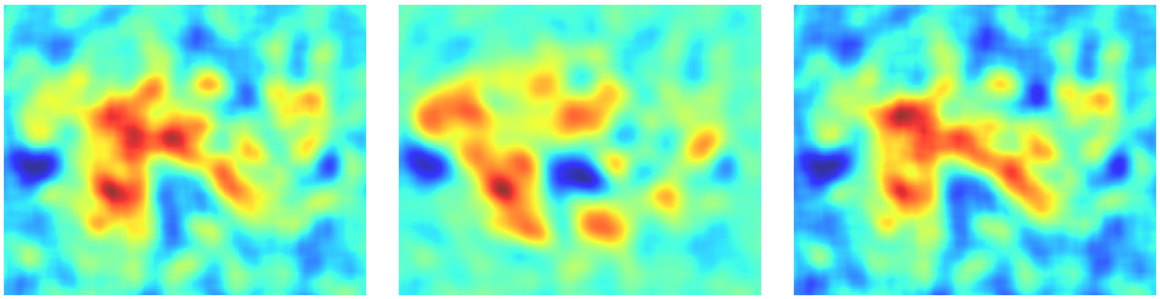
(a) Duck 18 Frame 1 (b) Duck 20 Frame 18 (c) Duck 22 Frame 14 (d) Duck 43 Frame 0

Figure 176: Importance Map Examples That Focus on Outline

After focus maps were calculated for each frame, an average focus map was calculated for each model. The aim here was to create a focus 'fingerprint' which would show what was important in the frame to the representation for a given model. It was decided to explore the focus 'fingerprints' for the two most common model designs. It was expected that for each re-creation of the same design, the focus 'fingerprint' would be similar.

Firstly, the 'Standing Duck' models were considered and an example set of 'fingerprints' is shown in Figure 177, with the full 'fingerprint' set shown in Appendix D. On reviewing these 'fingerprints' a common pattern was often seen. This coincided with firstly a rectangular focus region in the middle left side of the frame. This focus region then expanded out from the upper right side horizontally and then continued diagonally down towards the bottom righthand corner. Overall, the main focus region was shaped like a 'n', but the intensities varied from model to model.

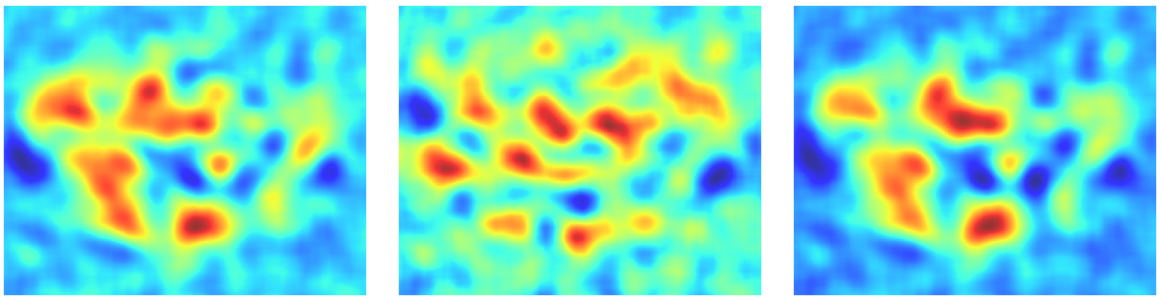
Next, the 'Sitting Duck' models were considered. An example set of generated 'fingerprints' for each of these can be seen in Figure 178, with the full set shown in Appendix D. Although in these cases, there was more variability in the 'fingerprints', the majority still had a common pattern. This pattern contained a focus region spanning outwards towards the upper left corner from slightly left of the horizontal midpoint. A further two regions spanned horizontally from close to the top and bottom of this region, however the region from the bottom tended to be shorter. The ends of each of these two regions were then joined diagonally by an additional region, this was often with a lower intensity compared to the other regions. Overall, these four regions appeared to



(a) Duck 0 (b) Duck 13 (c) Duck 71

Figure 177: Example Focus ‘Fingerprints’ for ‘Standing Ducks’

combine together to form an ‘upside down isosceles trapezoid’.



(a) Duck 3 (b) Duck 25 (c) Duck 33

Figure 178: Example Focus ‘Fingerprints’ for ‘Sitting Ducks’

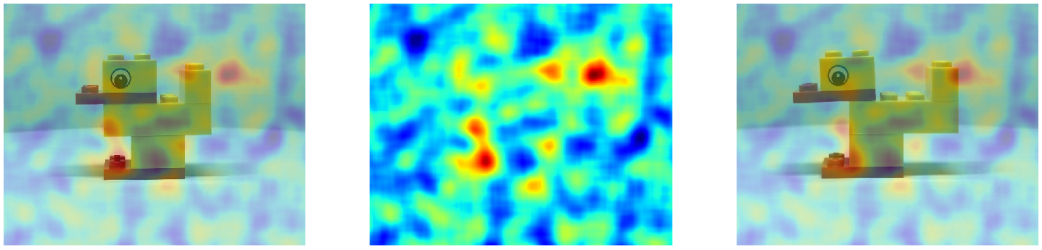
Based on both of these investigations it was found that identically constructed models often resulted in similar model ‘fingerprints’. This suggested that similar details were being extracted from each model and hence were likely to have similar representations, this was the aim when comparing similar models.

7.3.3 Explaining the Differences between Duck Models

In this section several duck pairs were investigated to extract the differences between the models, based on the encoder. On occasion, this analysis produced a result that suggested large parts of the frame differed between models. This was expected to be a result of comparisons where at least one of the models had encoded most of the frame, as discussed in the previous section. However, the more interesting cases only focused on a few regions of the model. On the other hand, this was most likely observed when each of the models had a focus on only a few selected features. In this case, the ‘focus regions’ found within this analysis, should identify key parts of the model that differed to the comparison model. Below a few of these cases are explored.

Firstly, a comparison was conducted between Duck 0 (‘Standing Duck’) and Duck 1. When visually comparing these two models, one minor difference was noted. This was that the body of Duck 1, represented by the 2×4 brick, was shifted backwards. When examining the generated importance map, shown in Figure 179, two main areas

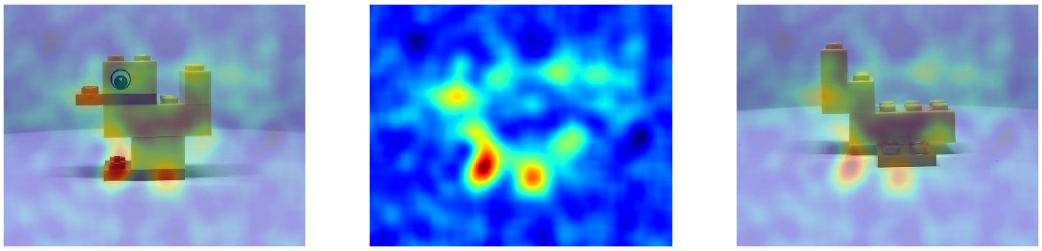
of focus are shown. These were on the top of the front of the duck's feet and behind the duck's tail. Each of these regions showed the changes in the model due to the body shifting, where the feet were shifted forward and the tail was shifted backwards. The area on the feet showed where the feet were on Duck 0 but it was noted that the feet were not at this location on Duck 1. Whereas the area behind the tail showed where part of the tail on Duck 1 had been repositioned. There was a further secondary focus on the front of the duck's body, this was on an area of the duck's body that had shifted compared to Duck 1. Overall, the main areas highlighted were parts of the model that had changed between the two compared models.



(a) Frame 0 (b) Importance Map (c) Frame 0

Figure 179: Comparison of Duck 0 Frame 0 against Duck 1 Frame 0

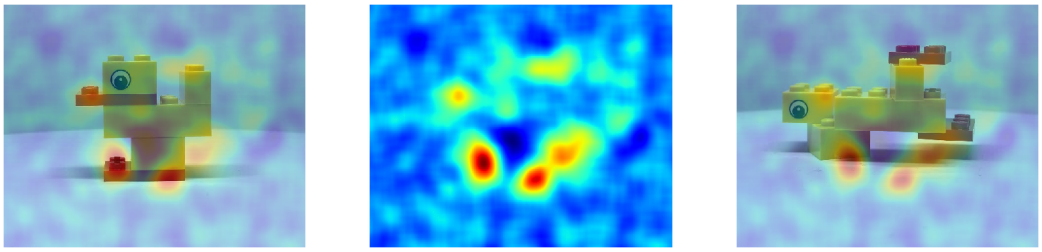
The next two comparisons looked to compare Duck 0 ('Standing Duck') against what were thought to be more visually looking novel model designs. In this case Duck 0 was compared against Duck 20 and Duck 58, and the resulting importance maps can be seen in Figure 180 and Figure 181 respectively. Once again, in both cases, the main focus area was seen to be on the front of the duck's feet. Although, in these cases, another focus area was also noticed not only on the duck's feet but particularly on the bottom of the back of the feet. When investigating where these areas fell in the other models, it was found that they fell in parts of the frame that did not contain any of the model or at least fell on the model's edge, this meant that only part of the area contained the model. This showed that the areas identified do indeed show parts of the model that differ and hence it is suggested that information about the feet and potentially their placement is contained in the representation.



(a) Frame 0 (b) Importance Map (c) Frame 0

Figure 180: Comparison of Duck 0 Frame 0 against Duck 20 Frame 0

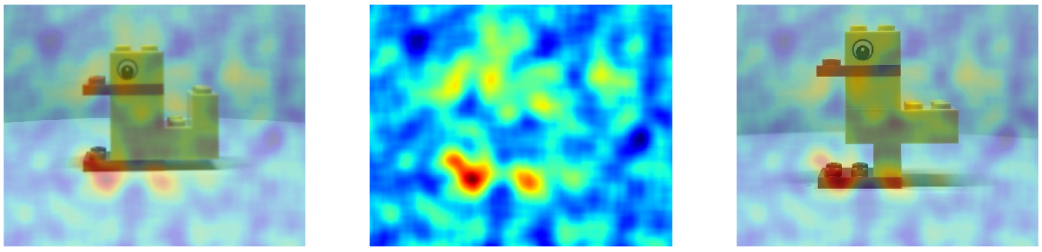
Following this, comparisons were made with Duck 3 ('Sitting Duck'). Firstly Duck 3 was compared to Duck 22, where the 1×2 brick had been placed on top of a 2×3 red



(a) Frame 0 (b) Importance Map (c) Frame 0

Figure 181: Comparison of Duck 0 Frame 0 against Duck 58 Frame 0

plate, as presumably the duck's legs, resulting in a taller model being generated. When this comparison was conducted, the main focus area in the importance map was on the top and below the front of the duck's feet, as shown in Figure 182. Further, a secondary focus was seen below the middle of the feet. When investigating where these foci fell on Duck model 22, it was found that these were either where the duck's feet were located in Duck 3 or Duck 22. This showed that this difference had been captured and further suggested that information about the duck's feet was being encoded.



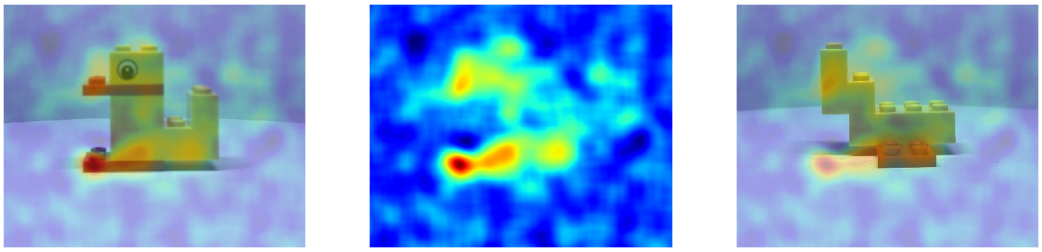
(a) Frame 0 (b) Importance Map (c) Frame 0

Figure 182: Comparison of Duck 3 Frame 0 against Duck 22 Frame 0

Further, comparisons were made between Duck 3 ('Sitting Duck') against more novel looking models, here Ducks 20 and 56 were selected. The importance maps for these comparisons are shown in Figure 183 and Figure 184 respectively. Once again, as can be seen in the importance maps, the main focus was on the duck's feet. This again seemed like an important difference that was highlighted in both comparisons, as in these cases the feet position was different.

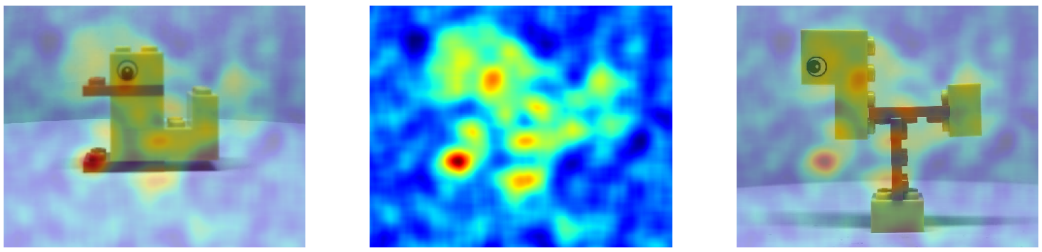
Finally, two comparisons were made between model pairs, each of which only had minor brick or plate placement changes between them. The first of these compared Duck 5 to Duck 13, the importance map is shown in Figure 185. It was noted that the difference between these two models was that a 2×3 red plate had been shifted on the model. When looking at the importance map, the main focus was on the front of the 2×3 red plate which had been shifted. This hence meant that the analysis had been able to identify the main differing factor between these models.

The second comparison was made between Duck 26 and Duck 27, the importance map is shown in Figure 186. In this case, the 1×2 yellow brick had been moved on the model, this resulted in the head portion of the duck model having been lowered. Also, a 2×3 red plate had been shifted. Interestingly, in this case, the main focus was on



(a) Frame 0 (b) Importance Map (c) Frame 0

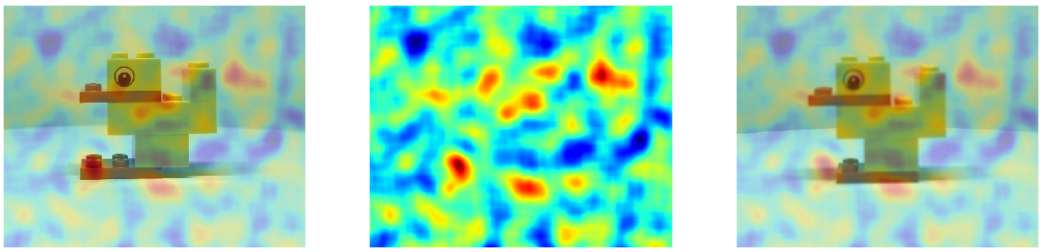
Figure 183: Comparison of Duck 3 Frame 0 against Duck 20 Frame 0



(a) Frame 0 (b) Importance Map (c) Frame 0

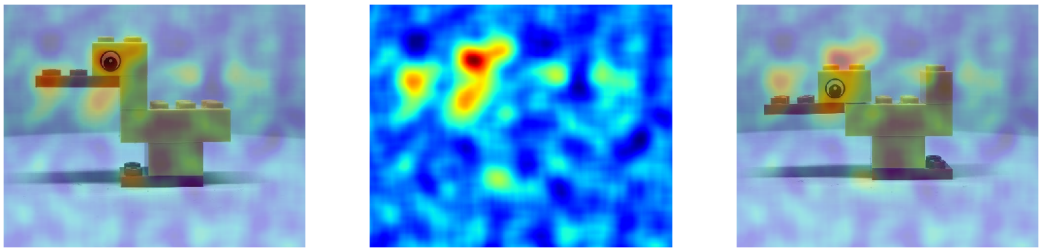
Figure 184: Comparison of Duck 3 Frame 0 against Duck 56 Frame 0

the duck's head, as well as a region just below the head. An additional focus area was also found on the duck's beak. These areas were most likely caused by the head being lowered between the models. This suggested that this factor had been identified in the representation. As regards the second factor, namely the shifting of the 2×3 red plate, there appeared to be a small amount of interest on the front of the duck's feet. Although, this was at a much lower intensity compared to the interest on the duck's head.



(a) Frame 0 (b) Importance Map (c) Frame 0

Figure 185: Comparison of Duck 5 Frame 0 against Duck 13 Frame 0



(a) Frame 0 (b) Importance Map (c) Frame 0

Figure 186: Comparison of Duck 26 Frame 0 against Duck 27 Frame 0

7.3.4 Duck Model Projections

The projection of the duck representation space onto a 2D plane is shown in Figure 187. Within this projection, the colour of the points represents the cluster label of each point found by applying Hierarchical Clustering for 20 clusters. This aimed to check that generally the clusters within the full representation space were still within close proximity within the projection space. Seven regions were identified that appeared to group closely together and have been subsequently marked on the projection. The ducks from each of these regions were then visually inspected to explore if there were common traits that could be identified.

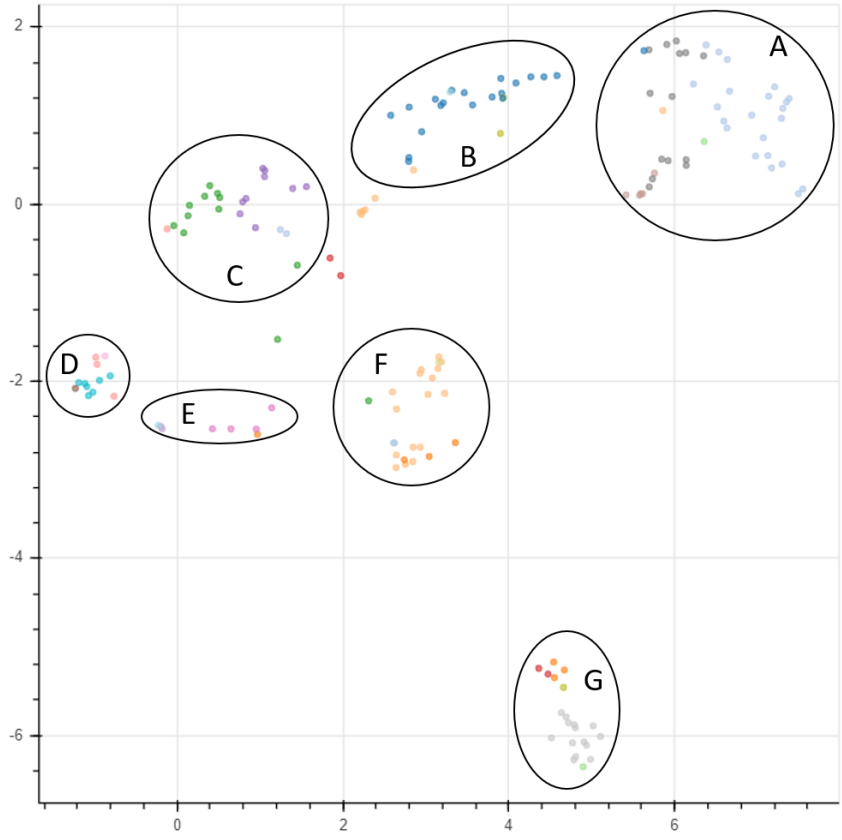
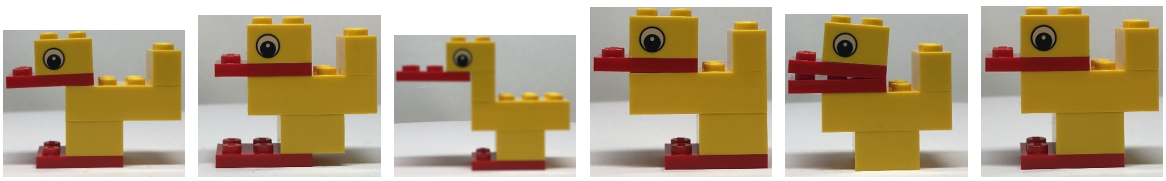


Figure 187: UMAP of Lego Duck Representations

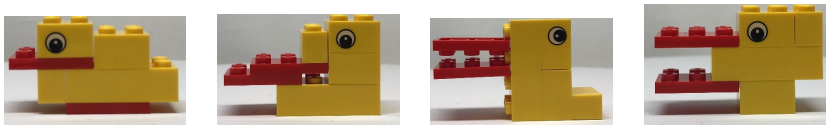
Firstly, region A was investigated for common factors in these models. Generally, it appeared that the models in this region attempted to create a model of the whole duck, an example set of models from this region is shown in Figure 188. Further, it was seen that the models commonly shared a trait in how the four parts of the duck's anatomy had been created. Commonly, what was assumed to be, the duck's head was constructed by placing a 2×2 yellow brick on top of a 2×3 red plate. Whereas a 1×2 brick was often placed on the back of the 2×4 brick, to presumably, represent the duck's tail. The duck's legs appeared to be created by connecting a 2×2 brick beneath a 2×4 brick, however the position of the connection on the 2×4 brick occurred at different places. Finally, it appeared that the duck's feet were represented by a red 2×3 plate which was connected to the bottom of the model. When reviewing this whole region, it was found that all the models with the 'Standing Duck' design were found in this region.



(a) Duck 1 (b) Duck 5 (c) Duck 10 (d) Duck 41 (e) Duck 61 (f) Duck 158

Figure 188: Duck Examples from Region A

Next, region C was examined and examples of models from this region are shown in Figure 189. In this case, the models' overall shape appeared to be more rectangular. This often took the form of focusing on just modelling the duck's head and in these situations the two red 2×3 plates, presumably, were used to represent the duck's beak.

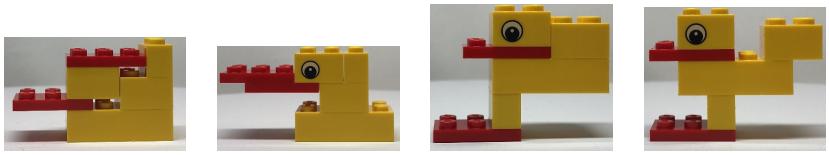


(a) Duck 15 (b) Duck 23 (c) Duck 97 (d) Duck 110

Figure 189: Duck Examples from Region C

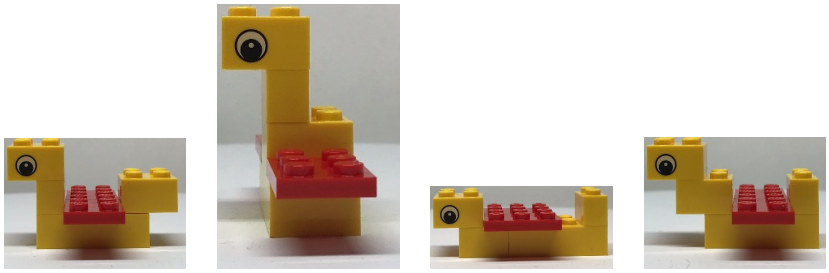
Following this, region B was explored. In this region, there were no clear common features observed. Instead, this region appeared to be a transition region between regions A and C. Two example models from the left side of this region, the side closest to region C, can be seen in Figure 190a and Figure 190b. Whereas, two example models from the right side of this region, the side closest to region A, can be seen in Figure 190c and Figure 190d. In these cases, the models from region B shared similarities with the closest regions, either A or C.

Next, models in region D were investigated and an example set of these models is shown in Figure 191. It was found that these models would often be built up using the 2×4 brick as the base of the model and then connected on top of this would be the two red 2×3 plates. Visually, it was assumed that this appeared to be modelling the wings of the duck.



(a) Duck 122 (b) Duck 159 (c) Duck 130 (d) Duck 148

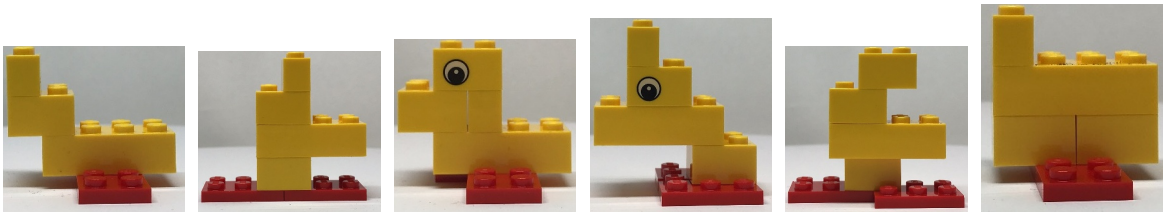
Figure 190: Duck Examples from Region B



(a) Duck 50 (b) Duck 69 (c) Duck 84 (d) Duck 114

Figure 191: Duck Examples from Region D

Following this, region E was explored, and model examples are shown in Figure 192. In this case, the common factor appeared to be how the models used the two red 2×3 plates. In this region they tended to be used to create a base, which appeared to represent the duck's feet. The yellow bricks were then built up from this base, often diagonally. Based on visual inspection, this looked like a duck with its head under the water and its tail up in the air.

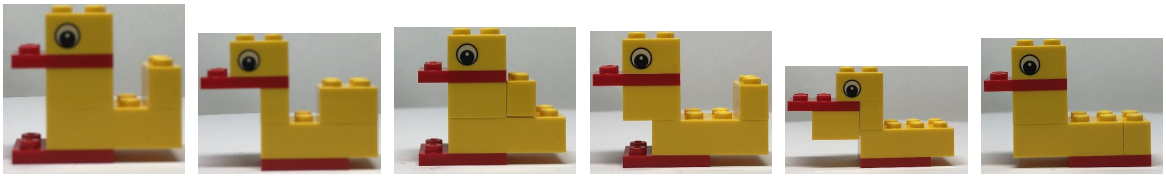


(a) Duck 20 (b) Duck 28 (c) Duck 62 (d) Duck 90 (e) Duck 94 (f) Duck 120

Figure 192: Duck Examples from Region E

Following this, models within region F were considered. Example models from this region are shown in Figure 193. In these cases, again it can be seen that the use of placing a 2×2 yellow brick on top of a 2×3 red plate, to presumably represent the duck's head, was common as was also seen in region A. However, in this case, it was also observed that a 2×3 red plate was often placed under the 2×4 yellow brick which appeared to show the duck's feet. This gave the appearance that the duck was sitting down, consequently all the models with the 'Sitting Duck' design were found in this region. Further, when examining the height of the models, the maximum height was often made by the combination of 3 bricks and 2 plates.

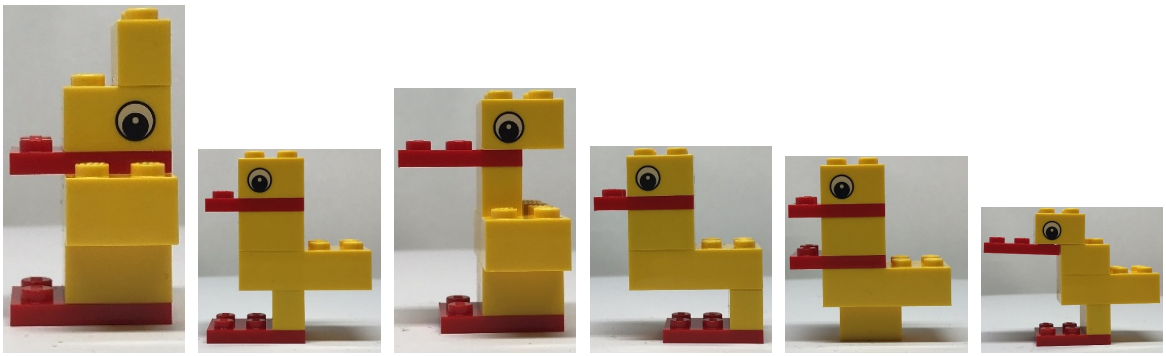
Finally, region G was investigated, and an example model set is shown in Figure 194.



(a) Duck 3 (b) Duck 11 (c) Duck 49 (d) Duck 57 (e) Duck 78 (f) Duck 127

Figure 193: Duck Examples from Region F

While a 2×3 red plate under a 2×2 yellow brick was still commonly used to, presumably, represent the duck's head, and a 2×3 red plate was presumably used to represent the duck's feet. In this region, there was more variability in what the 2×3 red plate representing the feet were attached to, for example 2×1, 2×2 or 2×4 yellow bricks. Further, when examining the height of the models, the maximum height was often found by the combination of 4 bricks and 2 plates. This meant that the models in region G were often taller than the models in region F.



(a) Duck 6 (b) Duck 22 (c) Duck 24 (d) Duck 45 (e) Duck 66 (f) Duck 76

Figure 194: Duck Examples from Region G

7.3.5 Duck Model Representation Transitions

Next analysis was conducted on the learnt representation space more directly. As previously mentioned, this took the form of selecting several duck pairs and then generating a set of representations that transition between these two representations. The trained decoder was then used to reconstruct the image of the model from the representation. This aimed to assess if moving within the representation space resulted in the gradual changing of different properties of a model. Initially, model pairs were chosen with only small design differences, however, latterly model pairs were chosen to explore the transition between more visually dissimilar models.

Firstly, the transition between Duck 0 to both Duck 1 and 3 were considered, as shown in Figure 195. In this figure a transition between a commonly observed 'Standing Duck' (Duck 0) to a similar looking duck but with the duck's head and legs shifted forward (Duck 1), as well as the other commonly seen 'Sitting Duck' (Duck 3) are shown. As can be seen, as the representation shifted between these two duck pairs, the relevant

features also gradually shifted. This can be seen as either the head and legs slowly shifting, or the body being positioned lower down, depending on the transition.

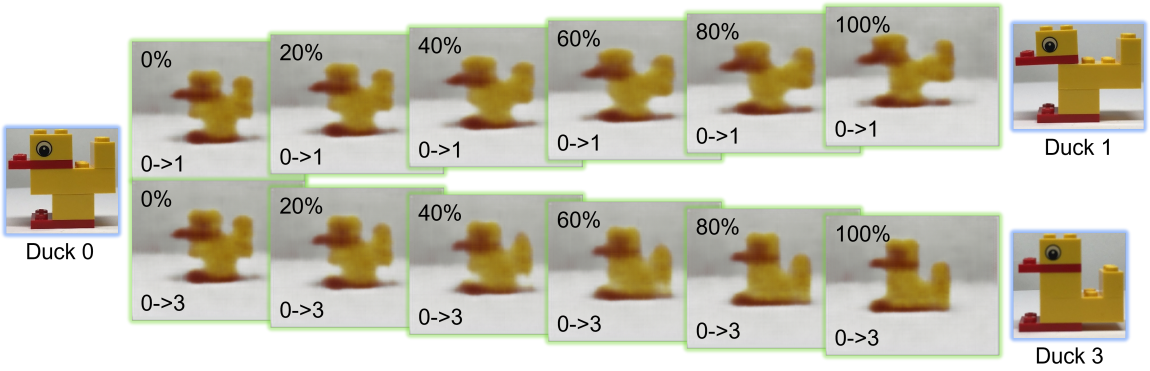


Figure 195: Transitioning from Duck 0 to Duck 1 and Duck 3

Next, the transition between Duck 3 and Duck 22 was explored. This is shown in Figure 196. Once again, a slow morphing was seen, with the body slowly rising and the tail disappearing.

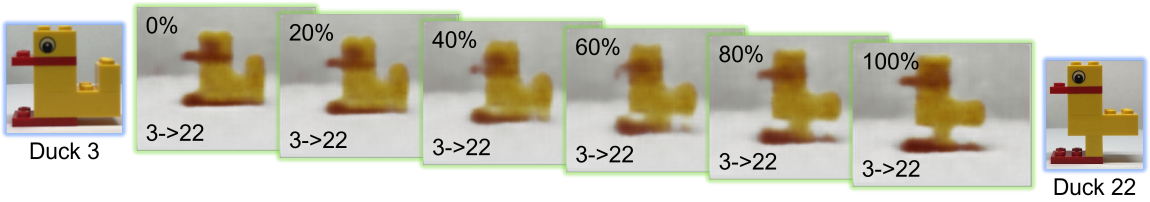


Figure 196: Transitioning from Duck 3 to Duck 22

Further, this was also observed for the transition between Duck 26 and Duck 27, as shown in Figure 197. However, in this case the duck's neck had shrunk while the tail grew.

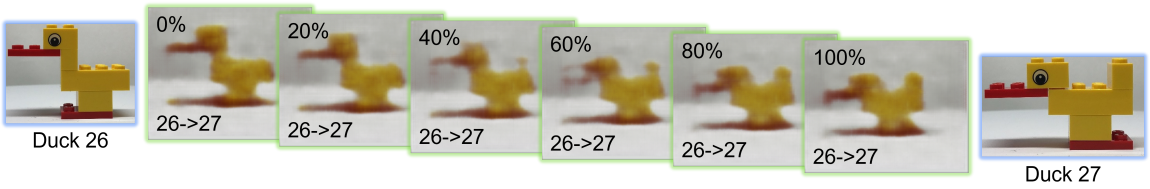


Figure 197: Transitioning from Duck 26 to Duck 27

Up to this point, the transitions investigated focused on more common looking ducks. However, below the transitions between common and more visually novel duck models were explored. This took the form of transitions between two common duck models, namely the 'Standing Duck' and 'Sitting Duck', morphing into more novel model designs. The first investigation looked at the transition to a design which appeared to represent the duck's tail when the duck was presumably diving under the water. These transitions are shown in Figure 198. Once again, it can be seen that there was a gradual morphing

between the two model designs, however the reconstruction images from more novel models tended to have less sharp edges. This was most likely due to the fact that during training there were fewer examples of this type of model compared to the other two more common models.

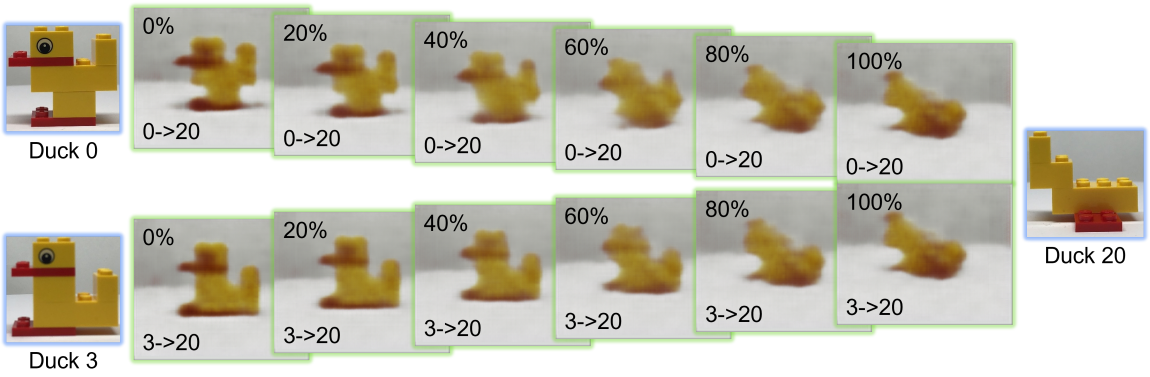


Figure 198: Transitioning from Duck 0 and Duck 3 to Duck 20

Next, the transitions to two other novel looking models were considered. Again, the transition was taken from the 'Standing Duck' and 'Sitting Duck' but this time transitioning to Duck 56, which showed an unusual use of the red plates, or Duck 58, which appeared to show a duck swimming, see Figures 199 and 200 respectively.

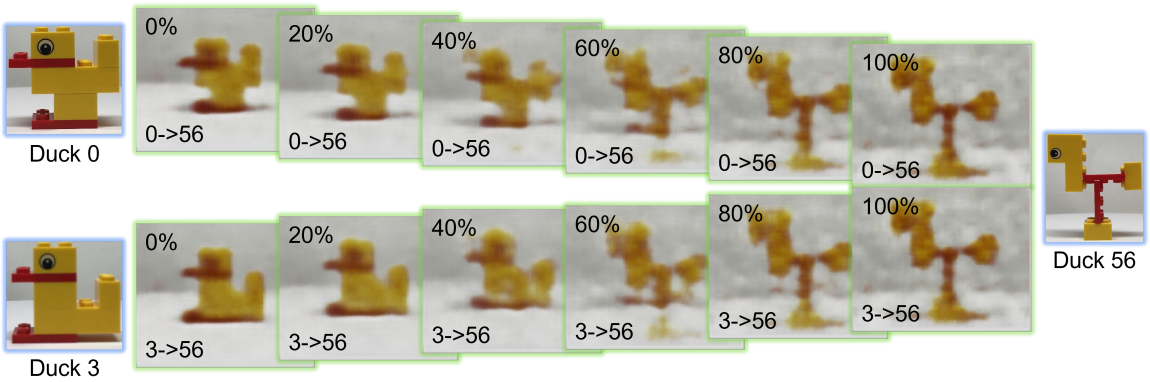


Figure 199: Transitioning from Duck 0 and Duck 3 to Duck 56

Similarly, when transitioning to Duck 20, the reconstruction images had edges with reduced sharpness but also at stages through the transition the reconstruction images looked incomplete *i.e.* there were gaps seen in the brick reconstruction. This was most likely due to the fact that these model designs were considered even more novel. Further, this was not considered a major issue as the middle point of this transition looked less 'duck like' and would be hard to recreate and hence their establishment within the representation space was less important. On the other hand, the other midway transition points between more common ducks generally could still be recognised as ducks.

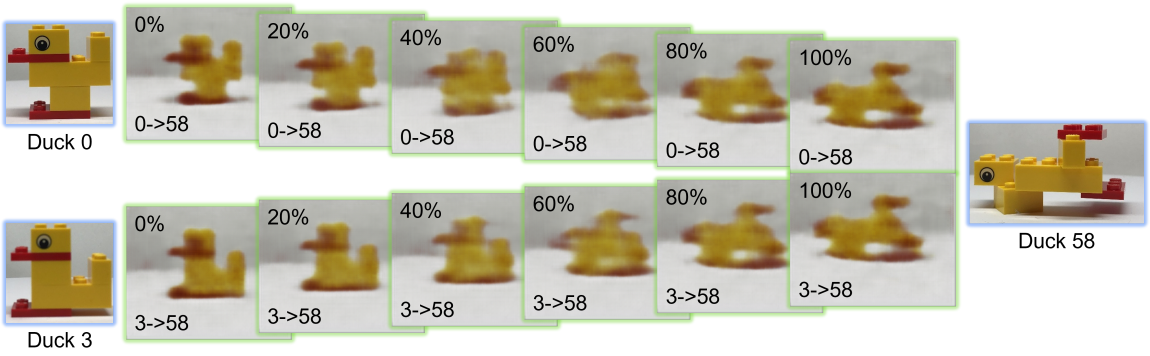
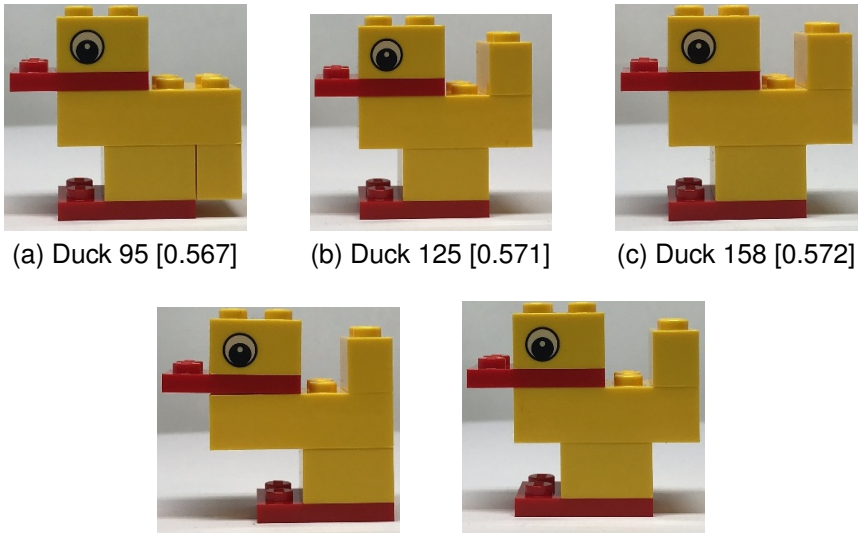


Figure 200: Transitioning from Duck 0 and Duck 3 to Duck 58

7.3.6 'Distance-Based Novelty' Detection

As previously suggested, one novelty metric that could be considered was to take the average distance in the representation space from the duck in question, to all other ducks being considered within the dataset. Within this section, the five highest and five lowest distance value ducks were examined, as these should relate to the most novel and most common ducks respectively. Firstly, the five ducks with the lowest average distance were considered and are shown in Figure 201.



(a) Duck 95 [0.567] (b) Duck 125 [0.571] (c) Duck 158 [0.572]

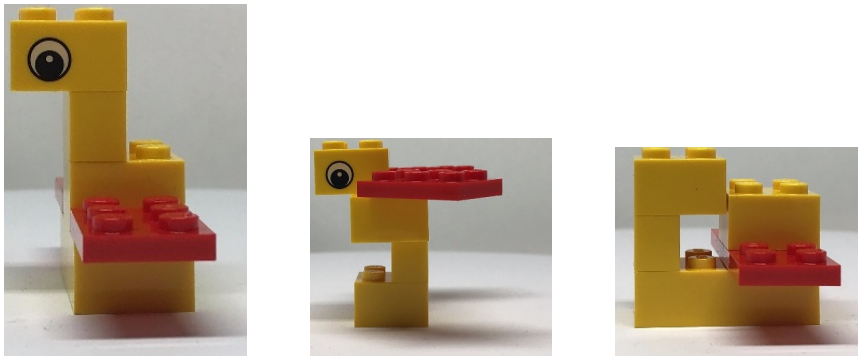
(d) Duck 41 [0.577] (e) Duck 96 [0.579]

Figure 201: Common Ducks Based on Average Distance Shown in Brackets

When examining the five most common ducks based on the average distance metric, it can be seen that three out of the five were of the 'Standing Duck' model design and the other two ducks differed by a single brick shift. Either by shifting the 1×2 yellow brick below the 2×4 yellow brick instead of it being on top or shifting the 2×2 yellow brick to the back of the bottom of the 2×4 yellow brick instead of it being in the middle of the bottom. Interestingly, the model deemed most common has the second of these alterations, potentially this allowed it to sit in between the 'Standing Duck' model and

the ‘Sitting Duck’ model and hence have a lower average distance.

Next, consideration was given to the five duck models with the highest average distance metric and these are shown in Figure 202. It is suggested that models furthest away from all other models within the representation space can be thought of as more novel. These models were then examined in relation to the rest of the model dataset and each of these showed interesting characteristics. For example, these characteristics showed the use of placing LEGO pieces at an angle or presumably using the two 2×3 red plates to represent the duck’s wings. Further, each of these models looked very different when compared to the two commonly seen models within the dataset, namely the ‘Standing Duck’ and the ‘Sitting Duck’.



(a) Duck 69 [0.869] (b) Duck 86 [0.857] (c) Duck 48 [0.849]

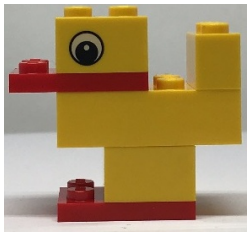
(d) Duck 58 [0.844] (e) Duck 157 [0.838]

Figure 202: Novel Ducks Based on Average Distance Shown in Brackets

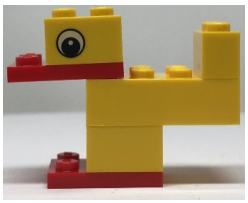
7.3.7 ‘Cluster-Based Novelty’ Detection

Next, cluster size was investigated as a metric to evaluate novelty. Standard clustering was applied, and the biggest and smallest clusters were extracted and examined. Firstly, consideration was given to the largest cluster. This cluster contained a total of 26 duck models and an example of five of these can be seen in Figure 203. On examination of this cluster, many of the duck models shared similarities with the ‘Standing Duck’ which was one of the most replicated model designs. In addition, many models within this cluster had only small changes to the standard design. However, it was seen that there were some models that deviated in more significant ways from this design.

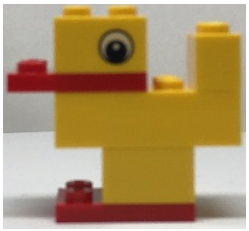
Finally, the smallest clusters were examined. Firstly, there were three single model clusters and these are shown in Figures 204a, 204b and 204c. When considering each of these models, based on an understanding of the whole dataset, these models do indeed look novel and further they all differed from one another. In addition to the single model clusters, there were two clusters having a model population of two. Within Figure



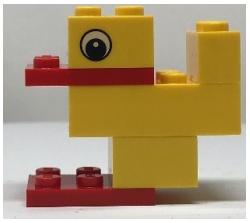
(a) Duck 0 [26]



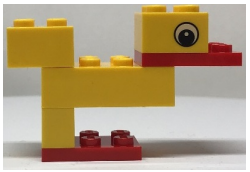
(b) Duck 1 [26]



(c) Duck 4 [26]



(d) Duck 5 [26]

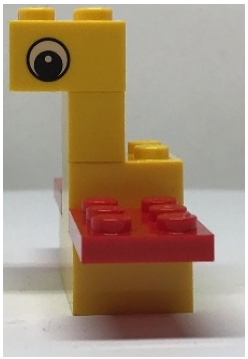


(e) Duck 7 [26]

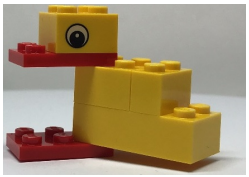
Figure 203: Common Ducks Based on Cluster Size Shown in Brackets

204 a cluster identifier (*i.e.* a or b) has been added alongside the cluster size to allow the distinction of which models were within the same cluster. The first cluster of two models is shown in Figures 204d 204e. These also seemed novel compared to the whole dataset. However, a similarity can be seen between these two models, namely that the vertical placement of the 2×3 red plate by wedging it into the top of another piece. The models in the second of these clusters are shown in Figure 204f 204g. Again, when compared to the whole dataset, these models appear to be novel but also shared some similarities between themselves. In this case the similarity arose through the placement of the red plates on the bottom of the model and then building, what is presumably, the duck's tail upwards.

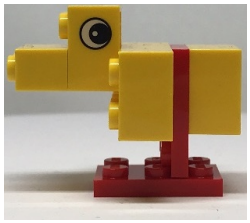
(a) Duck 58 [1]



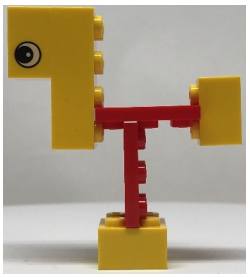
(b) Duck 69 [1]



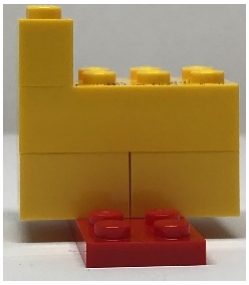
(c) Duck 154 [1]



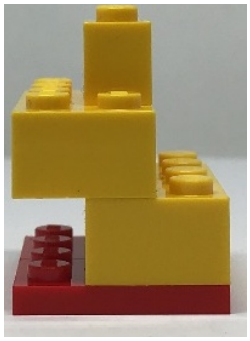
(d) Duck 8 [2a]



(e) Duck 56 [2a]



(f) Duck 120 [2b]



(g) Duck 150 [2b]

Figure 204: Novel Ducks Based on Cluster Size Shown in Brackets

7.4 Conclusion

Within this chapter, the idea was to explore if the developed methods from video game playstyle clustering could be utilised in other domains. In this particular case, investigation of the STDIM-VAE system was explored to ascertain whether it could be used to evaluate novelty within the creativity field. To investigate this, the LEGO duck task was used, where participants were asked to create a duck model from a set of LEGO pieces. Images of these models were then recorded, and these were used to train the encoder. This resulted in a set of representations for each model. Analysis was then conducted on the generated representation space.

Firstly, analysis focused on individual representations to extract and understand what was being encoded. It was found that the encoder often either focused on certain anatomical parts of the duck model (*e.g.* beak, feet, wings, *etc.*) or the overall model shape. The model shape itself was encoded in a number of different ways. This was achieved either in a direct manner by encoding brick positions or the edge of the model. Alternatively, this information was encoded by focussing on the negative space which indirectly encoded the model shape.

Secondly, analysis was conducted to compare representations in order to explore not only what was different between the models but also what was key to the representation. In these cases, the highlighted focus regions in the importance map should suggest the differences in the models and on visual inspection these often showed parts of the model that had changed. This suggests that the representation contained relevant

information about the model allowing the identification of the differences.

Following this, representations were considered at a model level, where each model was made up of a sequence of representations by encoding the image of the model taken at different angles. These sequences were projected onto a 2D plane using UMAP, in order to explore how each of the models compared to one another. From these projections, a number of regions were identified and investigated. Based on visual inspection, within each region, a common trait was often found. These common traits often related to how part of the duck's anatomy was modelled with a given set of bricks and plates. This suggests that the encoder was encoding human-legible traits into the representation and hence resulted in groups based on these traits within the projection.

Following this, the representation space was investigated further by taking pairs of models and encoding the first frame of each into the representation. A sequence of representations was then generated by stepping between each of them and then each of these representations generated a duck image using the trained decoder. This showed how the decoded image often gradually shifted between the two model designs. This further showed how moving in a certain direction within the representation space resulted in changes in a given feature (*e.g.* neck or leg height).

The above results suggested that relevant model features had been encoded into the representation. Hence, a couple of potential novelty metrics were proposed based on these representations. These metrics were applied to the whole dataset and the most novel and common models for each metric were extracted. When visually examined, the common models were often designs that appeared multiple times within the dataset or contained minor alterations to those models and hence were notionally common. On the other hand, those marked as the most novel, not only appeared novel based on the unique use of the bricks and plates but also regularly were the only instance of that design within the dataset.

Overall, it has been shown that the encoder designed for playstyle clustering in video games has other possible use cases in other domains. In particular, it has been shown that the STDIM-VAE is a viable approach to convert images of models to a representation space, from which novelty evaluation can be conducted.

8 Conclusion and Future Work

In this final chapter the main takeaways of this thesis will be drawn together by reflecting on the overarching hypothesis of this thesis, as well as its three individual components. Further, the limitations of this work will be discussed, which will help identify possible avenues for future work. Finally, a few closing thoughts will be provided covering this area of research going forward.

8.1 Conclusion

To conclude, each of the individual hypothesises were reflected upon.

Firstly, 'Is it possible to learn a representation space that contains the key features that define a playstyle?'. In Chapter 4, a new hybrid architecture, namely STDIM-VAE, was introduced in order to generate a representation space from gameplay videos. Analysis of this representation space has determined that playstyle relevant features have indeed been encoded into the representation and hence this has allowed the extraction of differences between the playstyles in many cases.

Secondly, 'Is it possible to cluster within the aforementioned representation space to create a separate cluster for each playstyle?'. Spurred on in the knowledge that playstyle relevant features had been encoded, in Chapter 5 a novel clustering algorithm called Reference-Based Clustering was applied to extract playstyle clusters. On evaluation of the cluster quality, it was found that a combination of the STDIM-VAE and Reference-Based Clustering techniques allowed a fully automated system to extract playstyle clusters from a set of gameplay videos.

Finally, 'Is it possible to train agents to accurately imitate human playstyles based on a set of demonstrations extracted by clustering?'. Within Chapter 6 a new imitation method, namely Dynamic Time Warping Imitation, has been shown to allow the imitation of different playstyles based only on example playstyle videos selected via clustering within the learnt representations space. Further, it has been shown that this algorithm was able to learn the playstyle even in game level variations which differed from the level from which the demonstrations were collected.

By combining the methods introduced in each of these chapters, a three-step process can be constructed in order to train agents with different playstyles based on a set of gameplay videos. This hence demonstrated that 'A set of agents can be trained to act with the diverse set of playstyles shown within a demonstration set that contains only raw gameplay videos'.

In addition, in Chapter 7 it has been further shown that these methods have applications beyond the video game domain. For example, from the learnt representation space, metrics could be defined to extract an understanding about the novelty of model designs.

8.2 Limitations

It is noted that the contributions in this thesis do indeed have some limitations and the most significant of these are discussed below.

In regard to the proposed STDIM-VAE architecture, although it has been shown that playstyle relevant features have been encoded, the case studies that were considered spanned a limited number of games. While the three games chosen aimed to cover different game properties (*e.g.* 2D/3D, grid-based, *etc.*), it is felt that the current work has not conclusively proven that the method would generalise to any chosen game. However, there is nothing in the proposed architecture or the results that would suggest issues with this generalisation. In addition, the representation analysis performed was limited to a subset of the full dataset, although again nothing suggests that the findings would not hold over the whole dataset. The hybrid architecture was defined by combining encoder types that encoded different features, as it was believed that a combination of these features would be required to successfully group playstyles. However, in the cases where only one type of feature is required, it is expected that the encoder element that focuses on this feature type may be a more suitable approach.

Similar limitations are relevant to the clustering method, namely Reference-Based Clustering, as again the clustering method was applied to the same small number of games. Similarly, to the situation above, nothing suggests that this technique would not generalise to other cases. However, it is noted that the algorithm design was built on the assumption that each cluster contained at least two data points and hence the introduction of noise data points may cause issues in this approach.

Next, when considering the proposed imitation method called DTWI, the set of games was further reduced to a single game. Given that this was a grid-based game, the success of this approach has only been shown for this particular case. Again, nothing within the algorithm development was specifically built around this type of game and hence it is not expected that issues would arise when applying this method to other games.

Further, the motivation behind the work in this thesis focused on the idea of improving human AI interaction which naturally must be performed in a multi-agent system. This work has not only focused on a single-agent system but also the learning of a single playstyle. This choice was made to allow a focus on the system's ability to learn a single playstyle in isolation before introducing the additional complexities of both multi-agent systems and multi-style learning. However, currently this would limit the applications to where this approach could be utilised. It is thought that alterations could be made to the current algorithm to transform its suitability to both multi-agent systems and multi-style learning.

Finally, regarding the evaluation of model novelty, analysis was only conducted based on visual inspection of the dataset. While these initial investigations provided promising results on the suitably of using the learnt representation space in order to extract metrics (*e.g.* novelty) within the creativity field, further consideration is required to compare the ratings to human expert scores to fully validate its applicability in this field.

8.3 Future Work

In this section possible avenues for future work will be discussed relating to the various areas of the research within this thesis.

8.3.1 Encoding and Clustering

Naturally, the first avenue for future work with regards to encoding and clustering would be to apply the developed techniques to additional datasets to validate the proposed methods. This could take two possible forms. Firstly, exploration of datasets for different levels using the games that have already been used within this work. Alternatively, different games could be used, datasets collected, and the methods suggested within this thesis then applied. This second option would seem most fruitful in examining these new proposed methods' abilities to fully generalise to different games and not just focus on different levels of the same games. Further, it is recognised that the encoder analysis has only focused on a subset of the gameplays for each of the games investigated. Future work could expand this analysis to all the gameplays to ensure that the analysis is consistent across all of the data.

8.3.2 Imitation

With regards to the imitation section of this work, a beneficial avenue of further exploration in validating the proposed method, would be to attempt to use the imitation algorithm to imitate different playstyles on other games. Additionally, it would be useful to further refine the algorithm, as during the hyperparameter tuning certain parameters were fixed, based on relevant literature or initial experimentation, to aid faster optimisation.

Below a few initial experiments are discussed which have been conducted to explore potential areas for possible algorithm refinement in regard to these parameters.

The first of these experiments explored the effect of changing the n-step value, which was fixed at 5 for the imitation work previously discussed. Initially, experiments were conducted on the 'base' level with the agent attempting to imitate the 'Camouflage' playstyle. During training the agent's ability to collect the camouflage package and complete the level was recorded. Figure 205 shows how different n-step values affect the agent's ability to collect the camouflage package and complete the level. Given the n-step value controls how far into the future the auxiliary reward predicts, and that exploration is reduced as the predictions nears the maximum theoretical reward, it was expected to take longer to converge. This increase in convergence time as the n-step value was increased is due to the fact that the agent had to act accurately over a longer time period to collect samples with a reward close to the maximum value. When comparing runs, based on the timesteps required for the agent to collect the camouflage package, in all tests with an n-step value of 5, 10 and 15 this trend was seen. However, when examining runs with an n-step value of 20, the timestep requirements dropped to a similar value needed when the n-step value was 5. Further research is required to fully understand why this was seen.

The agent's ability to complete the level in these experiments was also investigated under the different n-step values. Again for values 5, 10 and 15, the timesteps required increased. Further, with a value of 20 the timesteps again dropped but not by the same degree. This value was less than that required for 15 but more than that required for 10. While this value had been calculated based on only the two runs in which success was seen, these initial results suggest that a lower n-step value allows quicker convergence,

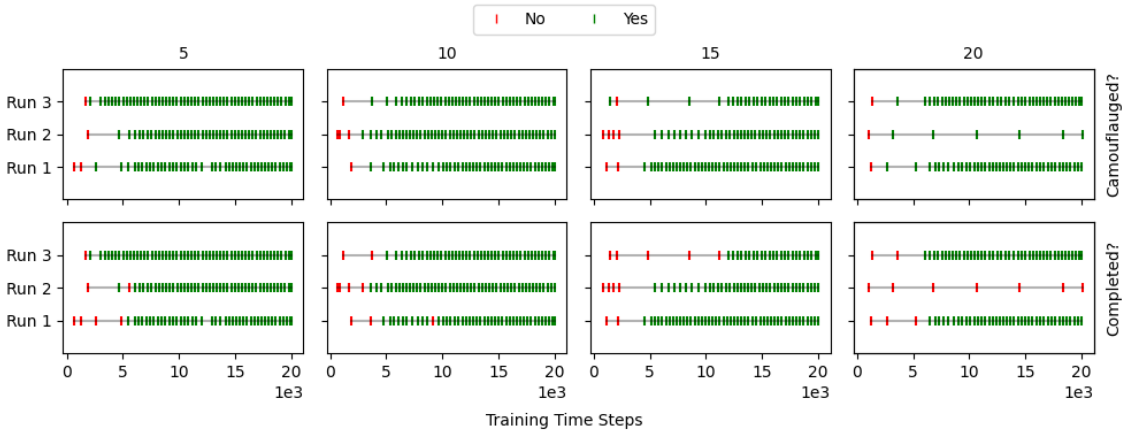


Figure 205: Comparison of the Effect of Different N-Step Values on Different Metrics while Imitating the ‘Camouflage’ Playstyle on the ‘Base’ Level

possibly because predictions over a shorter time span are easier and hence allowed the prediction based epsilon value to decay quicker.

Although, it is speculated that faster convergence based on lower n-step values is reliant on the fact that a perfect match is possible at every timestep, this in itself relies on the fact that the imitation and demonstration levels are the same. It is thought that on variations of the level, a higher n-step value may be required. This higher n-step value should allow the agent to explore initially bad matching paths which may then converge back to the demonstrated path. This idea was born out of the results of initial experiments conducted on the ‘right vent only’ level variant with varying n-step values, which are shown in Figure 206. Interestingly, in this set of experiments, only one run sees regular success. It is noted that in this run the n-step value was set to 15. When examining the alteration made to the level, it was observed that the forced path change deviates from the demonstrated path by approximately 15 timesteps. This could be a coincidence, particularly as regular success was only seen on one of the runs with the n-step value set to 15. However, this does make some degree of logical sense, as the n-step value essentially controls how far down a path the agent must explore before the path is considered bad. The combination of both of these initial investigations led to the following hypothesis, a low n-step value would be suitable in unchanged levels, as a perfect match is always possible and hence an agent acting ‘greedily’ based on the next step reward should find the correct path. Although, in variations of the level, a higher n-step value maybe be required in order to allow the agent to further explore a path before discarding it to ensure it does not re-join the desired path. Future work is hence required to explore this idea further.

Additional experiments were also conducted on a second parameter, namely prioritisation error components, which were fixed during the parameter optimisation in this thesis. Although prioritisation based on q-value error was an option within the parameter optimisation phase, only one of the top five parameter sets used prioritisation. However, after repeat runs were conducted, the top parameter set did not use prioritisation. The use of q-value based error is common within the literature, however in these experiments it was considered as to what other error values could be used to control the prioritisation.

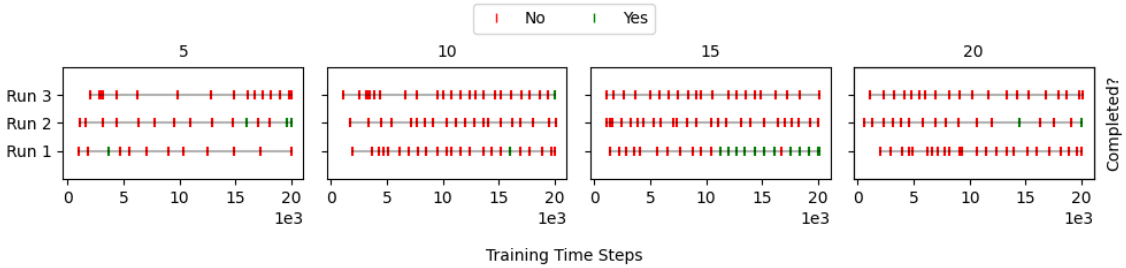


Figure 206: Comparison of the Effect of Different N-Step Values on the Ability to Complete the Level on the 'Right Vent Only' Level while Imitating the 'Stealth' Playstyle

Given the auxiliary reward prediction alongside the q-value prediction, the reward error was considered as another possible option. Experiments were conducted using both the q-value and the reward based error as well as a combination of both. The results for these experiments can be seen in Figure 207 for both the agent's ability to acquire the camouflage package and complete the level.

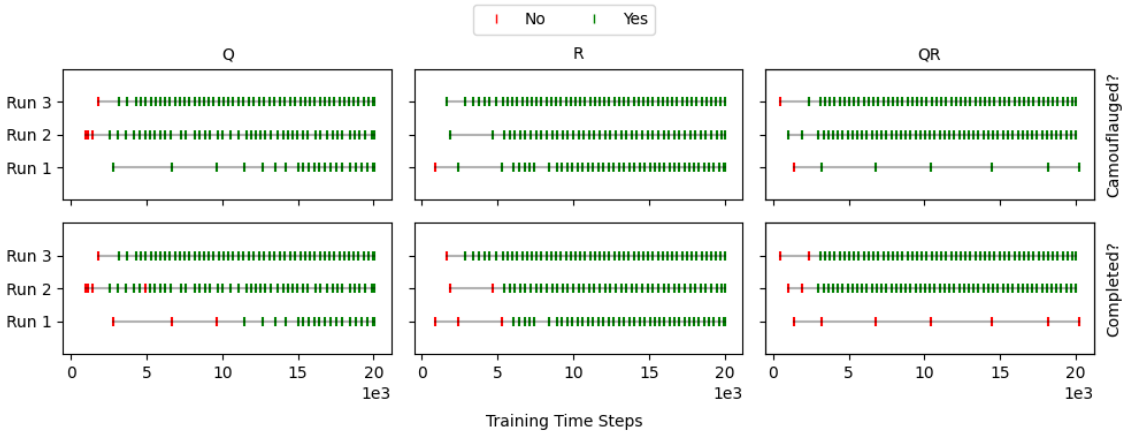


Figure 207: Comparison of the Effect of Different Prioritisation Error Components on Different Metrics while Imitating the 'Camouflage' Playstyle on the 'Base' Level

Initial examination of these results, suggests that potentially prioritisation based on reward prediction error may converge quicker than the standard q-value error prioritisation. However, further experimentation is required to fully validate this suggestion.

Further, in all experiments up to this point in this work, a pure (*i.e.* containing one playstyle) demonstration set had been provided. Although, for the Hitman game this was valid given that prefect clustering had been achieved with Reference-Based Clustering, it is possible in other use cases that an impure demonstration set could be provided. Due to this it would be interesting in future work to explore how the purity of the demonstration set affects the probability of successful imitation. Under the current algorithm, rewards are based on the best matching demonstration and hence if this was an incorrect playstyle, the imitation would likely fail. To account for this, a reward could be potentially provided based on the average of a set of close matches, however, this would increase the compute time.

In addition, when developing the imitation method, it was decided to focus on single-modal imitation (*i.e.* learning one playstyle at a time). However, it is hypothesised that there is likely to be an overlap in how different playstyles act, with potentially common underlying skills. It is hence suggested that if these were learnt concurrently, the time required to learn all playstyles would be less than learning each individually. A few initial experiments were conducted to explore this idea to better understand if future work would be fruitful in this area. It was decided to introduce additional network heads for each of the playstyles that required to be learnt, however, this naturally led to a larger network. This increase in network size could be mitigated by introducing an additional input to control the playstyle. Further, a simple sharing mechanism was introduced. Here a playstyle was selected at random, and data was collected based on that selected playstyle. This data was then stored with a marker that identified that playstyle. The data was then also compared to the demonstrations of the other playstyles, if a good match (*i.e.* over 0.8) was found, the data was also added for that playstyle. This meant that if by accident a different playstyle was performed, then the data was stored for the performed playstyle as well as the target playstyle. Further, if commonality exists between playstyles, then the successful learning of the skill for one playstyle will result in experiences that can reinforce the skill for another playstyle.

Results can be seen below when imitating each possible playstyle combination, *i.e.* evaluating each combinations' ability to successfully imitate each playstyle. This can be seen in Figure 208. For each setup a total on $20,000$ timesteps were given per playstyle being learnt. A few key observations can be made from these results. Firstly, it is clear that the timesteps required to learn the playstyles does not double or triple as the playstyles being learnt increases. This further suggests that there is commonality that can be shared when learning the playstyles together to help reduce the time for the learning process. However, it is also noted that as the number of playstyles increases, more fluctuation is seen between successful and unsuccessful playstyle imitation. Potentially, further refinement to one playstyle may have a negative effect on the other playstyle(s) and hence it is suggested that this issue needs to be further investigated.

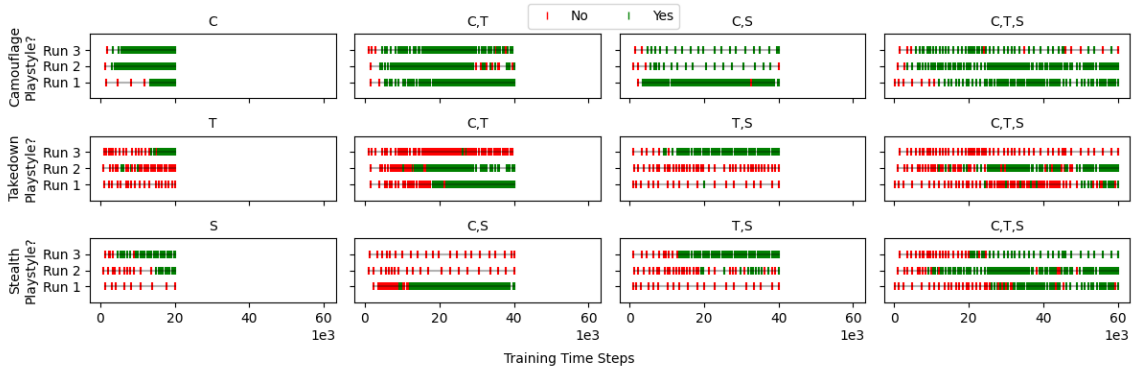


Figure 208: Comparison of the Effect of Learning Different Playstyle Sets on the Ability to Complete the Level using Different Playstyles on the 'Base' Level

Throughout training, a count was recorded showing how much 'off style' data (*i.e.* data stored for a different playstyle compared to the one for which it had been generated)

had been stored in the replay buffer and this can be seen in Figure 209. The purpose of this was initially to investigate if data was being shared across playstyles, markers were set at $20,000$, $40,000$ and $60,000$ timesteps to allow understanding of how much additional data was being stored. Although, an additional inference can be made by comparing how much additional data was stored based on the playstyle set being learnt. Here the greater commonality found between the playstyles being learnt should result in more 'off style' data being stored. It can be seen that regularly for the 'Camouflage', 'Stealth' combination, that the amount of data stored was below $5,000$, compared to approximately $12,000$ to $15,000$ seen for the other two playstyle sets. It is also noted that for the set that contained all the playstyles, the count finished at about $30,000$, this showed that by sharing data across playstyles approximately 50% extra data was stored by the end of training.

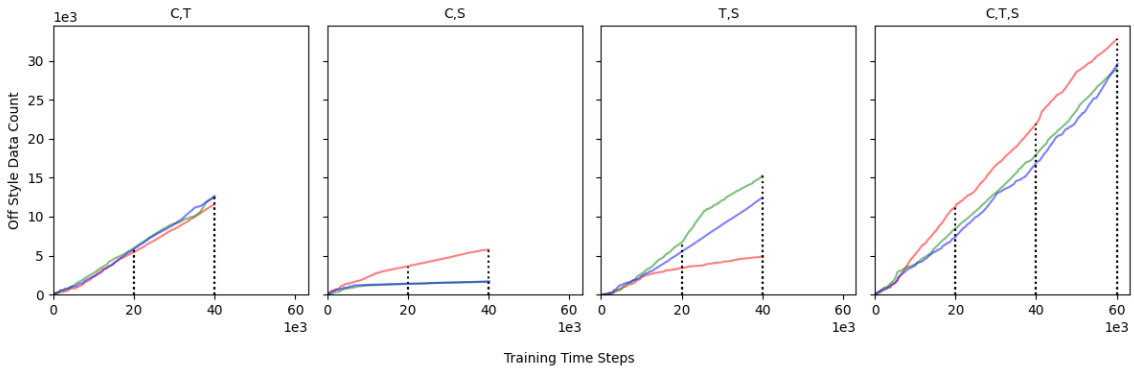


Figure 209: Comparison of the 'Off Style' Data Counts for Different Playstyle Sets on the 'Base' Level

Further, training under different playstyle sets was also conducted on the 'Right Vent Only' level variant and the results are shown in Figure 210. In this case similar observations were seen when considering the learning of the 'Camouflage' playstyle.

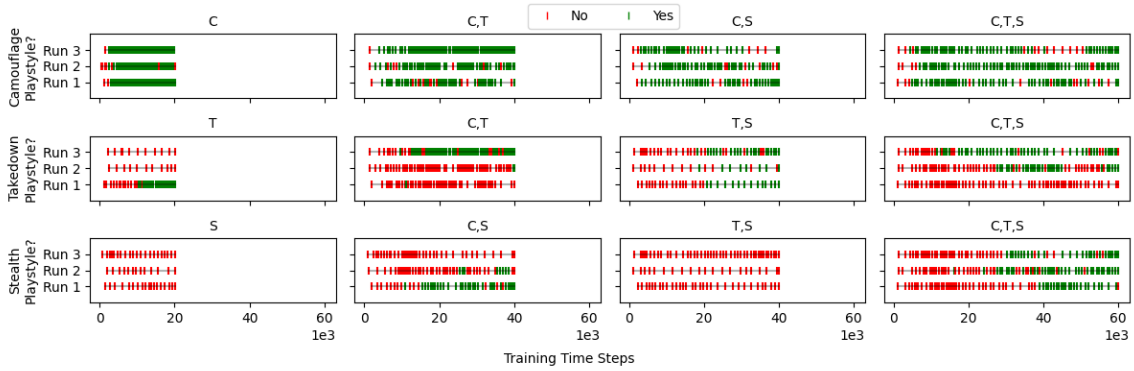


Figure 210: Comparison of the Effect of Learning Different Playstyle Sets on the Ability to Complete the Level using Different Playstyles on the 'Right Vet Only' Level

However, when considering the learning of the 'Takedown' playstyle, it can be seen that two out of the three runs never successfully learnt to play with the 'Takedown'

playstyle when learning the playstyle on its own. Although, when learning multiple playstyles in parallel, in most cases, successful playstyle usage was seen at least once, however the frequency of this varied greatly.

Finally, when considering the 'Stealth' playstyle in two playstyle sets, namely 'Stealth' alone and the combination of 'Stealth' and 'Takedown', playstyle success was never seen. This showed that the combination of playstyles can help learning but is most successful when compatible playstyles were grouped together.

During the 'Right Vent Only' experiments the 'off style' data counts were also recorded and are shown in Figure 211. Although the general trends seen for the 'Base' level are still present, overall there appeared to be a drop in how much 'off style' data was stored. On reflection, given the level alternation, the quality of matches was likely to be reduced and hence less data would cross the threshold to be stored.

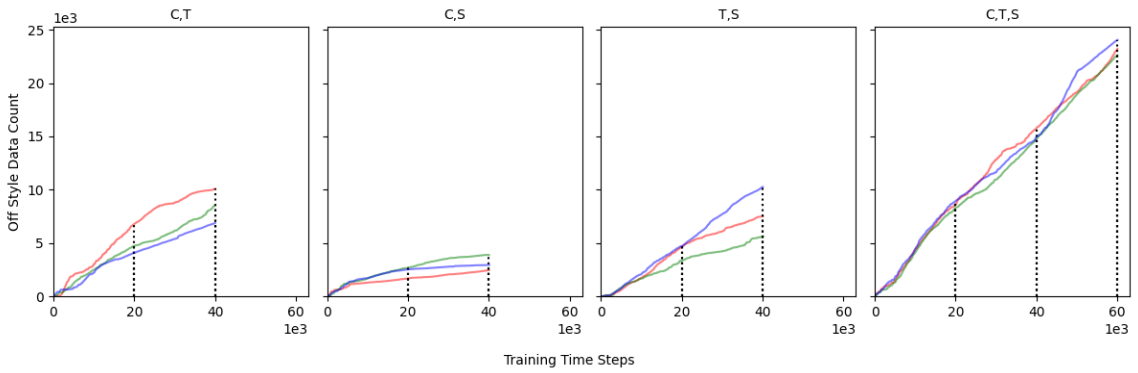


Figure 211: Comparison of the 'Off Style' Data Counts for Different Playstyle Sets on the 'Right Vent Only' Level

Based on these initial investigations, it is suggested that learning playstyles in parallel can help reduce the required timesteps to learn all playstyles. However, further work is still required to fully validate this idea.

8.3.3 Multi-Agent Training

Within this thesis, all agents have been trained within a single agent system. However, as the ultimate goal was to use trained agents with the aim of improving human agent interaction, then the agents would be required to act within a multi-agent system. It is hence felt that an area of future work would be to investigate how to transform these developed techniques to be applicable within multi-agent systems. A few initial thoughts about how this could be achieved are discussed below.

Firstly, a multi-agent game would be required. Some consideration has been given to how the 'Hitman' game could be transformed to meet this purpose. An additional player could be added to the game fairly easily, however some of the game mechanics would need to be altered to require collaboration for each playstyle. For the 'Camouflage' playstyle the camouflage package could be hidden behind a door that required each player to press a different switch at the same time in order to open it. For the 'Takedown' playstyle, the NPCs could be arranged to require synchronised takedowns in order to

avoid detection. Finally, for the 'Stealth' playstyle, mechanics could be introduced to require assistance from the other player to gain access to the vent.

After introducing a suitable game, alterations would be required to the learning system. Due to the player centric view, an input is not required to identify which player is being controlled. Although it would likely be beneficial for the colour of the controlled agent to be the same for each agent. This would hence require the player colour to be flipped between agents, as each player should be identified by a different colour.

When training, it is felt that a Population Based Reinforcement Learning approach may be suitable, however this population may not only be across generations but also playstyles within generations. Although, systems may need to be introduced to account for the fact that all playstyle combinations may not allow successful task completion.

Further, once a collection of different playstyles has been trained, the next step would be to train a single policy that can succeed no matter which learnt playstyle the single agent policy had been paired to. In theory, if the learnt set of playstyles accurately represent the human playerbase playstyles, then the agent should be able to collaborate with a wide range of human players.

Additionally, a better way to evaluate collaboration with regard to player experience may be beneficial. Although, previous literature has discussed possible metrics for evaluating collaboration, for example 'completion time' as here it was assumed that better collaboration should result in faster task completion or 'failure rate' as poor collaboration may also lead to task failure. However, it is expected that when evaluating collaboration through the lens of player experience, these metrics may need to be altered. For example, while a greatly superior AI agent may lead to a faster 'completion time', it is suspected that this may lead to the human player feeling a degree of inadequacy as they are not able to contribute to task completion and hence this potentially may lead to a poorer player experience. Further, it is theorised that a binary 'success' or 'failure' consideration may miss key ideas about how 'success' or 'failure' arose. Firstly, it is expected that failure close to completion is likely to lead to a very different player experience compared to that when failing straight away. In addition, the cause of the 'failure' may affect the player experience, if the player has no control over this cause, this would likely lead to frustration and hence a poorer player experience. Both of these ideas would not be extracted, based on the previously discussed metrics from the literature which shows why further research in this area may be beneficial. Further, from the points above, a better idea of how an AI agent should act to create the best player experience can be drawn.

Based on these initial thoughts, it is suggested that an AI agent should be of a similar skill level or possibly slightly higher, to allow each to contribute but also to try and avoid task failure being caused by the AI agent. To this end, potentially a defined collaboration based player experience metric could be used to aid in the training of collaborative agents.

8.3.4 Beyond Games Applications

Based on an initial metric previously discussed, it has been suggested that 'model design novelty' can be evaluated based on the learnt representation of a model. However, this initial novelty analysis was only conducted based on a visual inspection of the dataset.

To fully explore this, a model dataset would need to also contain human expert ratings of common metrics (*e.g.* novelty, creativity, *etc.*). A dataset with these metrics alongside the videos of the models were collected by collaborating partners, of different types of transport, to allow this future research. A range of initial investigations was conducted to explore the fruitfulness of this area for future work.

One investigation explored the construction of metrics based on a single operator (*i.e.* mean, max or median) applied over various nearest neighbour sets of different sizes for different representation sizes and down sampling values. These metrics were then correlated against different human expert rating scores, with the main aim to establish if there existed an automatic evaluation system that related to standard human expert rating scores. A few example results are shown and discussed below.

Firstly, training and evaluation based on the whole dataset was considered, here the automated metrics were correlated against human expert scores across different criteria. This was conducted both against raters who were given the model titles produced by the participants and those who were not. Example results for both 'How creative is the transportation?' and 'How novel is the transportation?' can be seen in Figures 212, 213, 214 and 215. Interestingly, it is noted that a stronger correlation was seen for ratings in which the raters were given the model titles.

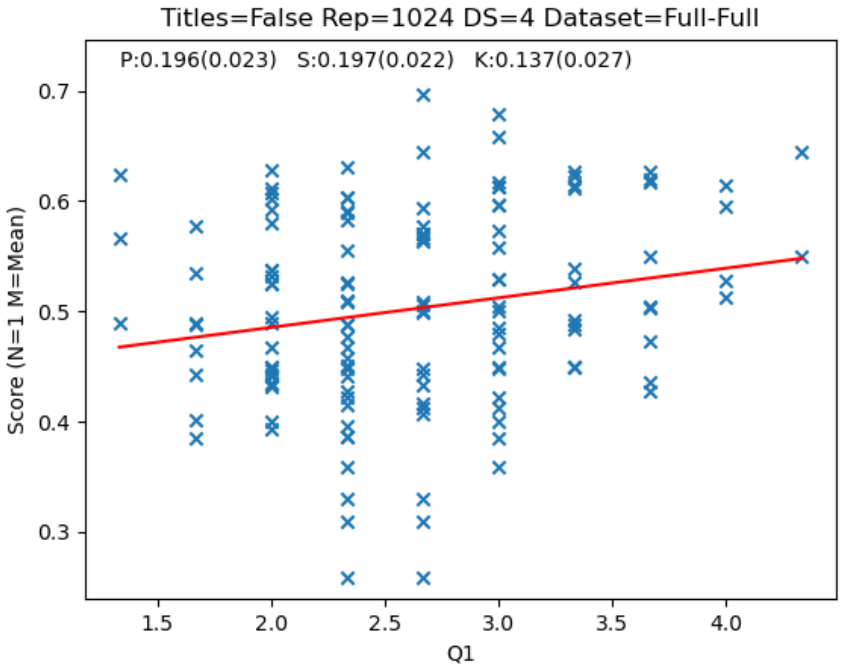


Figure 212: Correlation of the Automatic Rating System Against Expert Creativity Scores of the Full Dataset When the Titles Are Not Known

Further, investigations were also conducted on the novelty metric based on a subset of the data in which the models were identified as a car. In these cases, new encoders were trained on this subset and the results can be seen in Figures 216 and 217. Although, in this investigation a stronger correlation was seen with the expert ratings when the model title was not known to the rater. It is further noted that when comparing

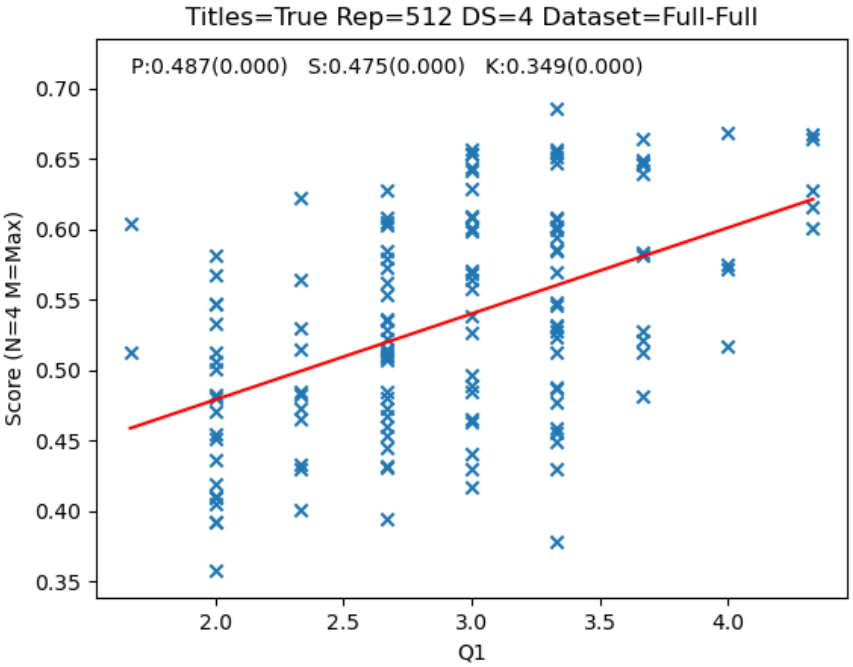


Figure 213: Correlation of the Automatic Rating System Against Expert Creativity Scores of the Full Dataset When the Titles Are Known

the best correlations, a stronger correlation can be seen in the car subset (*i.e.* subset of transport models identified as a car from their title) compared to the full dataset. It is suggested that this is because the automated system may struggle to compare across transport types. For example, if a model of a car and rocket were compared, humans can easily infer that the rocket is a more novel transport type. Whereas the automated system is likely to focus on the novelty in how the pieces were put together, than what they have created.

These initial findings further reinforce the idea that the learnt representation spaces can be used for automatic evaluation of different metrics which normally require human expert raters. In addition, the dataset also contains the result of the standard Alternative Uses Test (AUT) applied to the participants. Further, the dataset also contains a total of five different models from each participant, and it is hence suggested that the combination of how these models fall within the representation space may allow evaluation of the participants themselves and not just the models. To this end, a few initial experiments were conducted to explore possible metrics such as the area or perimeter of the convex hull caused by the five models created by a participant, once each was projected on to the projection space. This was based on the idea that a more creative participant would most likely create a set of models that span a greater area within the projection space. Initial results correlating these automated metrics against both the flexibility and unusual metrics can be seen in Figures 218 and 219. Based on these results it is suggested that the combination of multiple model representations from one participant can allow an evaluation of the participant themselves.

Overall, these initial investigations have shown that common metrics (*e.g.* novelty,

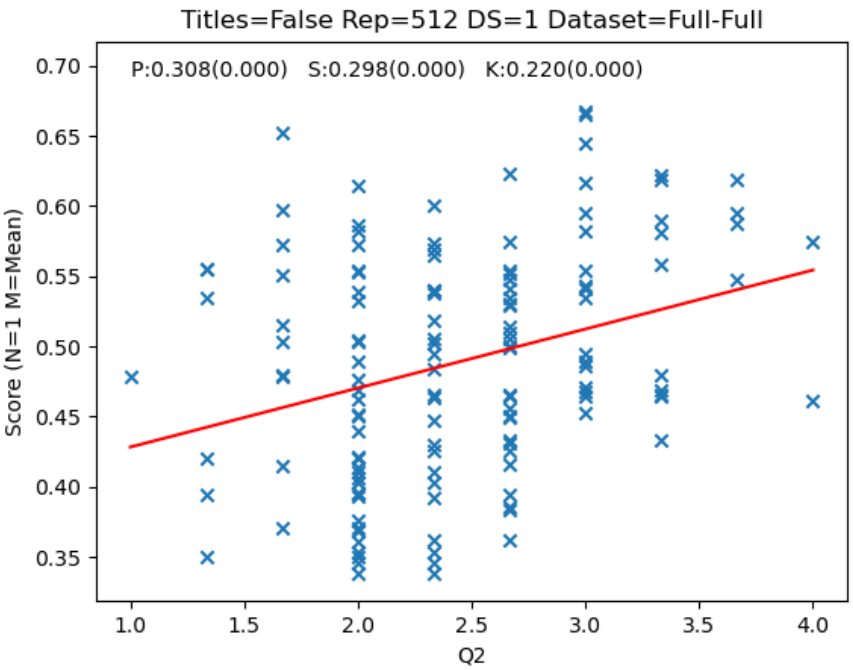


Figure 214: Correlation of the Automatic Rating System Against Expert Novelty Scores of the Full Dataset When the Titles Are Not Known

flexibility, *etc.*) can be correlated to metrics defined on the learnt representation space. However, further work is still required in order to fully validate these ideas and also finally settle on a scoring definition of each common metric.

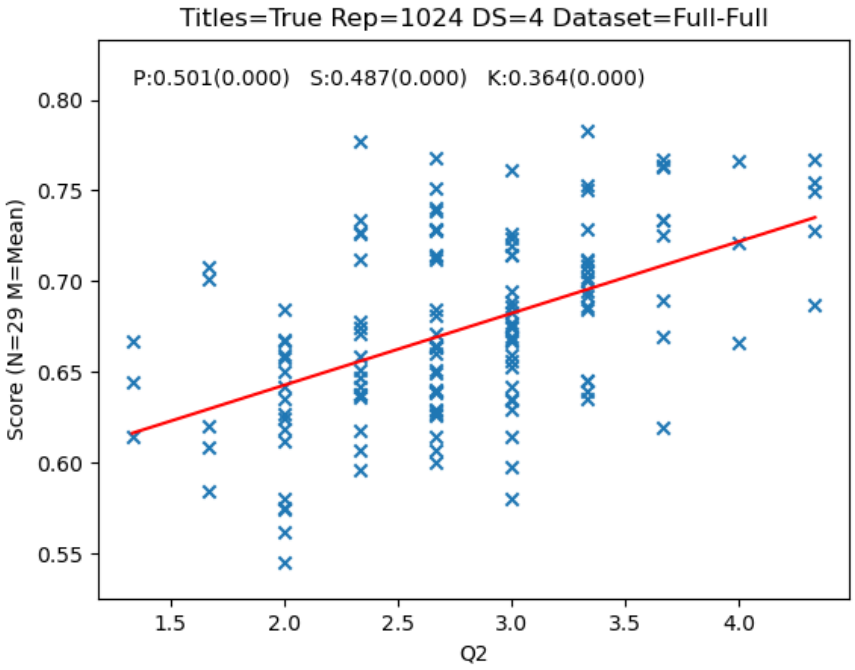


Figure 215: Correlation of the Automatic Rating System Against Expert Novelty Scores of the Full Dataset When the Titles Are Known

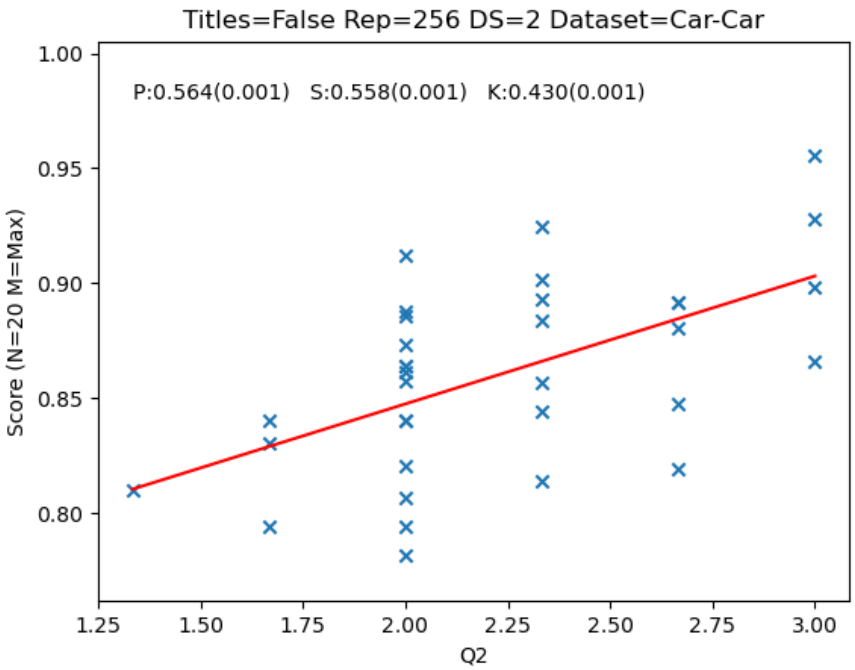


Figure 216: Correlation of the Automatic Rating System Against Expert Novelty Scores of the Car Subset When the Titles Are Not Known

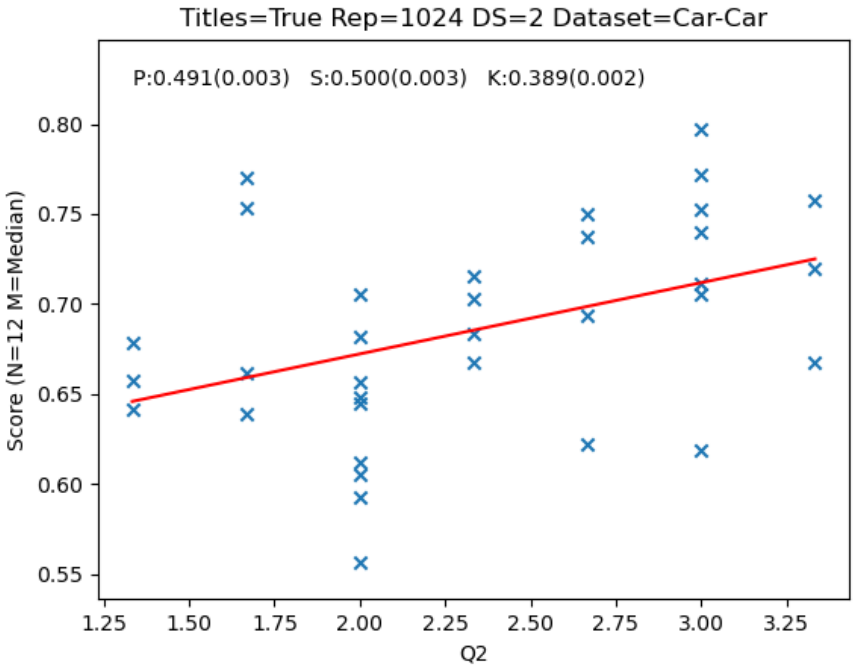


Figure 217: Correlation of the Automatic Rating System Against Expert Novelty Scores of the Car Subset When the Titles Are Known

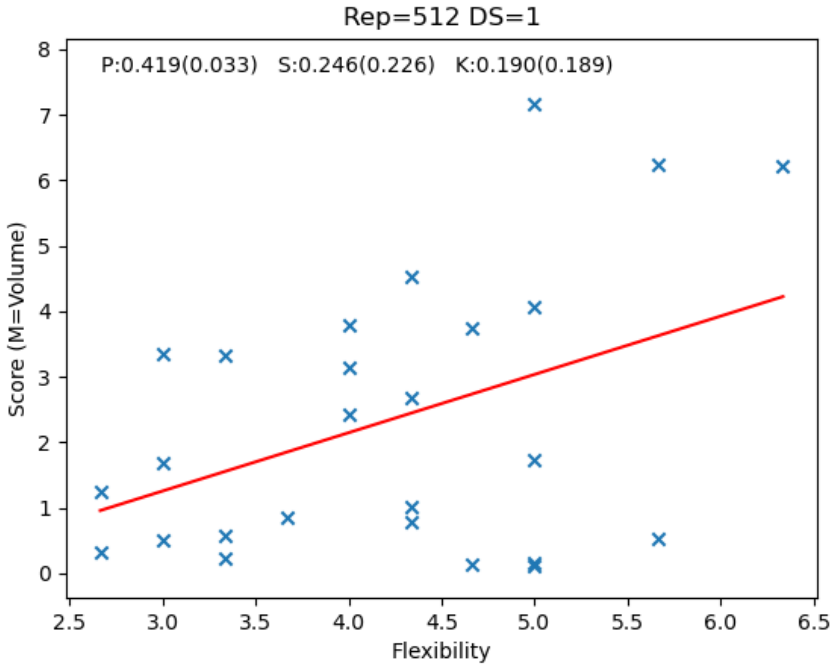


Figure 218: Correlation of the Area of the Convex Hull within the Projection Space Created by the Five Created Models to the Flexibility from the AUT

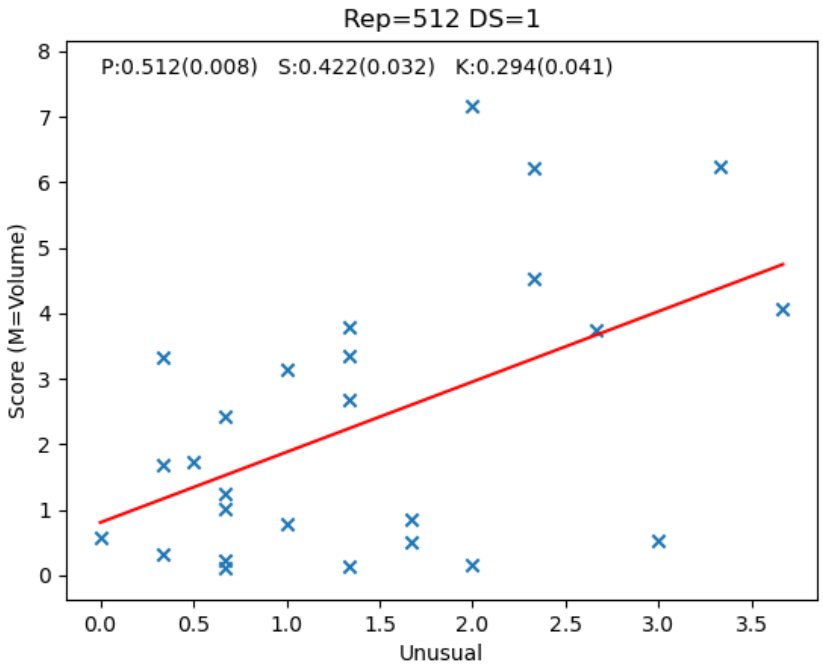


Figure 219: Correlation of the Area of the Convex Hull within the Projection Space Created by the Five Created Models to the Unusual from the AUT

8.4 Closing Remarks

In this thesis a three-step process has been proposed to allow the capturing, clustering and copying of playstyles demonstrated in an unlabelled demonstration set. In the future, it is believed that a set of playstyle agents created via this proposed workflow could be used to help improve human AI agent interactions in games and beyond, examples could include intelligent assistants, home care assistants or applications within medicine *etc.* Given that humans and AI agents each excel at different types of tasks, it is hoped that successful collaboration beyond games can lead to new and more effective solutions to complex real world problems.

Acronyms

AE Autoencoder. 37

AI Artificial Intelligence. 2, 20–24, 30, 39, 49, 189, 196, 203

AMI Adjusted Mutual Information. 125, 129, 139, 140, 142

AUT Alternative Uses Test. 13, 198, 201, 202

BC Behaviour Cloning. 16, 22, 23, 41–44, 46, 49, 153–157, 159–162, 256

BCO Behaviour Cloning from Observation. 46

BF2BC2 Battlefield 2: Bad Company 2. 34

CCL Connected Component Labelling. 168

CoM Centre of Mass. 168

DQN Deep Q-Network. 41

DTW Dynamic Time Warping. 11, 32, 36, 57, 58, 120, 125, 144–146

DTWI Dynamic Time Warping Imitation. 2, 11, 16, 26, 144, 147, 151–163, 188, 258

Esports Electronic Sports. 30

FPS First Person Shooter. 20, 33

GAIFO Generative Adversarial Imitation from Observation. 16, 47, 153–157, 159–162, 257

GAIFO-S Generative Adversarial Imitation from Observation - Single. 16, 47, 153–157, 159–162, 257

GAIL Generative Adversarial Imitation Learning. 16, 42, 45, 153–157, 159–162, 256

GAIL-T Generative Adversarial Imitation Learning - Triple. 16, 45, 153–162, 257

GANs Generative Adversarial Networks. 42, 44, 45

GP Gaussian Process. 151

GT Ground Truth. 11, 14, 15, 35, 59, 73, 120, 127, 137–140, 154–157, 159–162, 239, 240, 245, 246, 251, 252

I2L Indirect Imitation Learning. 48

IfO Imitation from Observation. 25, 26, 39, 46, 47

IL Imitation Learning. 24, 25, 39, 41–43, 49, 162

ILPO Imitating Latent Policies from Observation. 47

IRL Inverse Reinforcement Learning. 42, 44

KL Kullback-Leibler. 37, 51, 52

LB Literature Based. 7, 10, 13, 56, 62–66, 76, 102–107, 109, 215, 216

LCSS Longest Common Subsequence. 32

ML Machine Learning. 36, 39

NPC Non-Player Character. 20, 21, 24, 49, 103, 104, 106, 107, 112, 113, 116

NPCs Non-Player Characters. 20–22, 24, 28, 49, 51, 98, 99, 105, 106, 118, 195

PvE Player vs Environment. 33, 34

PvP Player vs Player. 33, 34

QA Quality Assurance. 29

RAN Random Action Noise. 7, 11, 14, 60, 62–72, 118, 125–129, 215–217, 226–234

RISE Randomized Input Sampling for Explanation. 54

RISER Randomized Input Sampling for Explanation of Representations. 7, 55, 57, 59, 101, 169, 215, 219

RISERD Randomized Input Sampling for Explanation of Representation Differences. 7, 57–59, 65, 68, 169

RL Reinforcement Learning. 7, 25, 39–42, 44, 46, 48, 99, 151, 196

SB Single Block. 7, 10, 13, 56, 63, 65–71, 102–105, 107, 108, 110–116, 217

SMCs Sensorimotor Contingencies. 23

ST-DIM Spatiotemporal DeepInfomax. 9, 14, 38, 51, 52, 60–63, 72, 75–78, 94, 95, 98, 100–105, 117, 118, 129, 131, 133–136, 139, 140, 142, 143, 223

STDIM-VAE Spatiotemporal DeepInfomax Variational Autoencoder. 2, 7, 8, 10, 11, 13, 14, 26, 51–53, 59, 61–63, 72, 75–78, 95–98, 100–105, 117–120, 127, 129, 131, 133–136, 140, 142–145, 164, 186–189, 218, 224, 225

TCN Time-Contrastive Networks. 48

TERA The Exiled Realm of Arborea. 34

UMAP Uniform Manifold Approximation and Projection. 58, 59, 187

VAE Variational Autoencoder. 9, 14, 37, 48, 51–53, 60–63, 72, 75–78, 80, 93–95, 98, 100–105, 118, 129, 131–134, 136, 139, 143, 165, 222

WOW World of Warcraft. 34

References

[1] Collins english dictionary. `https://www.collinsdictionary.com/dictionary/english/persona`. Accessed on 20.01.23.

[2] Merriam-webster dictionary. `https://www.merriam-webster.com/dictionary/two%20heads%20are%20better%20than%20one`. Accessed on 24.03.23.

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

[4] J. Agarwal, R. Nagpal, and R. Sehgal. Crime analysis using k-means clustering. *International Journal of Computer Applications*, 83(4), 2013.

[5] S. Aghabozorgi, A. S. Shirkhorshidi, and T. Y. Wah. Time-series clustering–a decade review. *Information Systems*, 53:16–38, 2015.

[6] A. Anand, E. Racah, S. Ozair, Y. Bengio, M.-A. Côté, and R. D. Hjelm. Unsupervised state representation learning in atari. *arXiv preprint arXiv:1906.08226*, 2019.

[7] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, Z. D. Guo, and C. Blundell. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*, pages 507–517. PMLR, 2020.

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

[9] Y. Bengio, Y. Lecun, and G. Hinton. Deep learning for ai. *Communications of the ACM*, 64(7):58–65, 2021.

[10] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[11] L. Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[12] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[13] K. Brantley, W. Sun, and M. Henaff. Disagreement-regularized imitation learning. In *International Conference on Learning Representations*, 2020.

[14] D. Brown, W. Goo, P. Nagarajan, and S. Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pages 783–792. PMLR, 2019.

[15] E. Brown and P. Cairns. A grounded investigation of game immersion. In *CHI'04 extended abstracts on Human factors in computing systems*, pages 1297–1300, 2004.

[16] J. Bütepage and D. Kragic. Human-robot collaboration: From psychology to social robotics. *arXiv preprint arXiv:1705.10146*, 2017.

[17] M. Campbell, A. J. Hoane Jr, and F.-h. Hsu. Deep blue. *Artificial intelligence*, 134(1-2):57–83, 2002.

[18] A. Canossa. Weaving experiences values modes styles and personas. *IO Interactive/Denmark's School of Design and Technology*, 2007.

[19] A. Canossa, M. S. El-Nasr, and A. Drachen. Benefits of game analytics: Stakeholders, contexts and domains. In *game analytics*, pages 41–52. Springer, 2013.

[20] L. Caroux, K. Isbister, L. Le Bigot, and N. Vibert. Player–video game interaction: A systematic review of current concepts. *Computers in Human Behavior*, 48:366–381, 2015.

[21] M. Carroll, R. Shah, M. K. Ho, T. L. Griffiths, S. A. Seshia, P. Abbeel, and A. Dragan. On the utility of learning about humans for human-ai coordination. *arXiv preprint arXiv:1910.05789*, 2019.

[22] X. Chen and H. Chen. A novel color edge detection algorithm in rgb color space. In *IEEE 10th International Conference On Signal Processing Proceedings*, pages 793–796. IEEE, 2010.

[23] D. Clarke and P. R. Duimering. How computer gamers experience the game situation: a behavioral study. *Computers in Entertainment (CIE)*, 4(3):6–es, 2006.

[24] M. Csíkszentmihályi. *Flow: The Psychology of Optimal Experience*. Harper and Row, 1990.

[25] R. Dadashi, L. Hussenot, M. Geist, and O. Pietquin. Primal wasserstein imitation learning. In *ICLR 2021-Ninth International Conference on Learning Representations*, 2021.

[26] S. Demediuk, P. York, A. Drachen, J. A. Walker, and F. Block. Role identification for accurate analysis in dota 2. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 15, pages 130–138, 2019.

[27] A. Drachen, J. Green, C. Gray, E. Harik, P. Lu, R. Sifa, and D. Klabjan. Guns and guardians: Comparative cluster analysis and behavioral profiling in destiny. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2016.

[28] A. Drachen, R. Sifa, C. Bauckhage, and C. Thurau. Guns, swords and data: Clustering of player behavior in computer games in the wild. In *2012 IEEE conference on Computational Intelligence and Games (CIG)*, pages 163–170. IEEE, 2012.

[29] A. Drachen, C. Thurau, R. Sifa, and C. Bauckhage. A comparison of methods for player clustering via behavioral telemetry. *arXiv preprint arXiv:1407.3950*, 2014.

[30] A. D. Edwards, H. Sahni, Y. Schroecker, and C. L. Isbell. Imitating latent policies from observation. *arXiv preprint arXiv:1805.07914*, 2018.

[31] C. Eggert, M. Herrlich, J. Smeddinck, and R. Malaka. Classification of player roles in the team-based multi-player game dota 2. In *International Conference on Entertainment Computing*, pages 112–125. Springer, 2015.

[32] A. Elgammal and B. Saleh. Quantifying creativity in art networks. In *Proceedings of the Sixth International Conference on Computational Creativity June*, page 39, 2015.

[33] X. Fang, S. Chan, J. Brzezinski, and C. Nair. Development of an instrument to measure enjoyment of computer game play. *INTL. Journal of human–computer interaction*, 26(9):868–886, 2010.

[34] M. Ferguson, C. S. Deterding, A. Lieberoth, M. Malmdorf Andersen, S. Devlin, D. Kudenko, and J. A. Walker. Automatic similarity detection in lego ducks. In *ICCC'20: Eleventh International Conference on Computational Creativity*. Association for Computational Creativity (ACC), 2020.

[35] M. Ferguson, S. Devlin, D. Kudenko, and J. A. Walker. Player style clustering without game variables. In *International Conference on the Foundations of Digital Games*, pages 1–4, 2020.

[36] M. Ferguson, S. Devlin, D. Kudenko, and J. A. Walker. Imitating playstyle with dynamic time warping imitation. In *Proceedings of the 17th International Conference on the Foundations of Digital Games*, pages 1–11, 2022.

[37] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016.

[38] G. Frasca. Rethinking agency and immersion: video games as a means of consciousness-raising. *Digital Creativity*, 12(3):167–174, 2001.

[39] T. Gangwani and J. Peng. State-only imitation with transition dynamics mismatch. *arXiv preprint arXiv:2002.11879*, 2020.

[40] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[41] C. Gordillo, J. Bergdahl, K. Tollmar, and L. Gisslén. Improving playtesting coverage via curiosity driven reinforcement learning agents. *arXiv preprint arXiv:2103.13798*, 2021.

[42] K. Grace, M. L. Maher, D. Fisher, and K. Brady. Data-intensive evaluation of design creativity using novelty, value, and surprise. *International journal of design creativity and innovation*, 3(3-4):125–147, 2015.

[43] D. Greene, A. Tsymbal, N. Bolshakova, and P. Cunningham. Ensemble clustering in medical diagnostics. In *Proceedings. 17th IEEE Symposium on Computer-Based Medical Systems*, pages 576–581. IEEE, 2004.

[44] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

[45] P. Guardini and P. Maninetti. Better game experience through game metrics: a rally videogame case study. In *Game Analytics*, pages 325–361. Springer, 2013.

[46] C. Guerrero-Romero, A. Louis, and D. Perez-Liebana. Beyond playing to win: Diversifying heuristics for gvgai. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 118–125. IEEE, 2017.

[47] C. Guerrero-Romero, S. M. Lucas, and D. Perez-Liebana. Using a team of general ai algorithms to assist game design and testing. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2018.

[48] K. Hausman, Y. Chebotar, S. Schaal, G. Sukhatme, and J. J. Lim. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. *Advances in neural information processing systems*, 30, 2017.

[49] R. Hayes and P. Beling. Unsupervised hierarchical clustering of build orders in a real-time strategy game. *The Computer Games Journal*, 7(1):5–26, 2018.

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

[51] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[52] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pages 4565–4573, 2016.

[53] C. Holmgård, M. C. Green, A. Liapis, and J. Togelius. Automated playtesting with procedural personas through mcts with evolved heuristics. *arXiv preprint arXiv:1802.06881*, 2018.

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

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

[56] C. Jennett, A. L. Cox, P. Cairns, S. Dhoparee, A. Epps, T. Tijs, and A. Walton. Measuring and defining the experience of immersion in games. *International journal of human-computer studies*, 66(9):641–661, 2008.

[57] R. Jonschkowski and O. Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39(3):407–428, 2015.

[58] A. Jordanous. Evaluating evaluation: Assessing progress and practices in computational creativity research. *Computational creativity: The philosophy and engineering of autonomously creative systems*, pages 211–236, 2019.

[59] P. Karampiperis, A. Koukourikos, and E. Koliopoulou. Towards machines for measuring creativity: The use of computational tools in storytelling activities. In *2014 IEEE 14th International Conference on Advanced Learning Technologies*, pages 508–512. IEEE, 2014.

[60] J. C. Kaufman, J. A. Plucker, and J. Baer. *Essentials of creativity assessment*. John Wiley & Sons, 2008.

[61] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, pages 341–348, Santorini, Greece, Sep 2016. IEEE. The Best Paper Award.

[62] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[63] C. Lamb, D. G. Brown, and C. L. Clarke. Evaluating computational creativity: An interdisciplinary tutorial. *ACM Computing Surveys (CSUR)*, 51(2):1–34, 2018.

[64] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[65] T. Lesort, N. Díaz-Rodríguez, J.-F. Goudou, and D. Filliat. State representation learning for control: An overview. *Neural Networks*, 108:379–392, 2018.

[66] Y. Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[67] Y. Li, J. Song, and S. Ermon. Infogail: Interpretable imitation learning from visual demonstrations. *Advances in Neural Information Processing Systems*, 30, 2017.

[68] F. Liu, Z. Ling, T. Mu, and H. Su. State alignment-based imitation learning. *arXiv preprint arXiv:1911.10947*, 2019.

[69] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[70] L. McInnes, J. Healy, N. Saul, and L. Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

[71] E. D. Mekler, J. A. Bopp, A. N. Tuch, and K. Opwis. A systematic review of quantitative studies on the enjoyment of digital entertainment games. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 927–936. ACM, 2014.

[72] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[73] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[74] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2146–2153. IEEE, 2017.

[75] A. Y. Ng, S. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[76] F. Nielsen and F. Nielsen. Hierarchical clustering. *Introduction to HPC with MPI for Data Science*, pages 195–211, 2016.

[77] R. Nugent and M. Meila. An overview of clustering applied to molecular biology. *Statistical methods in molecular biology*, pages 369–404, 2010.

[78] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in neural information processing systems*, pages 2863–2871, 2015.

[79] J. C. Osborn and M. Mateas. A game-independent play trace dissimilarity metric. In *FDG*, 2014.

[80] T. L. Paine, C. Gulcehre, B. Shahriari, M. Denil, M. Hoffman, H. Soyer, R. Tanburn, S. Kapturowski, N. Rabinowitz, D. Williams, et al. Making efficient use of demonstrations to solve hard exploration problems. *arXiv preprint arXiv:1909.01387*, 2019.

[81] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[82] V. Petsiuk, A. Das, and K. Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018.

[83] J. A. Plucker, M. C. Makel, and M. Qian. *Assessment of Creativity*, page 44–68. Cambridge Handbooks in Psychology. Cambridge University Press, 2 edition, 2019.

[84] C. Rasmussen. The infinite gaussian mixture model. *Advances in neural information processing systems*, 12, 1999.

[85] S. Reddy, A. D. Dragan, and S. Levine. Sqil: imitation learning via regularized behavioral cloning. *arXiv preprint arXiv:1905.11108*, 2019.

[86] A. Romero. What they don't tell you: 4 ways humans still vastly outperform ai. `https://towardsdatascience.com/what-they-dont-tell-you-4-ways-humans-still-vastly-outperform-ai-ba640aae0d4#:~:text=Humans%20are%20also%20great%20at,outside%20the%20limits%20of%20AI`. Accessed on 28.03.23.

[87] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[88] C. Sammut, S. Hurst, D. Kedzier, and D. Michie. Learning to fly. In *Machine Learning Proceedings 1992*, pages 385–393. Elsevier, 1992.

[89] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[90] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[91] E. Shelhamer, P. Mahmoudieh, M. Argus, and T. Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307*, 2016.

[92] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[93] B. C. Stadie, P. Abbeel, and I. Sutskever. Third-person imitation learning. *arXiv preprint arXiv:1703.01703*, 2017.

[94] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[95] F. Tencé, C. Buche, P. De Loor, and O. Marc. The challenge of believability in video games: Definitions, agents models and imitation learning. *arXiv preprint arXiv:1009.0451*, 2010.

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

[97] F. Torabi, G. Warnell, and P. Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

[98] F. Torabi, G. Warnell, and P. Stone. Imitation learning from video by leveraging proprioception. *arXiv preprint arXiv:1905.09335*, 2019.

[99] F. Torabi, G. Warnell, and P. Stone. Recent advances in imitation learning from observation. *arXiv preprint arXiv:1905.13566*, 2019.

[100] R. R. Torrado, P. Bontrager, J. Togelius, J. Liu, and D. Perez-Liebana. Deep reinforcement learning for general video game ai. In *Computational Intelligence and Games (CIG), 2018 IEEE Conference on*. IEEE, 2018.

[101] A. Tychsen and A. Canossa. Defining personas in games using metrics. In *Proceedings of the 2008 conference on future play: Research, play, share*, pages 73–80, 2008.

[102] A. Van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv e-prints*, pages arXiv–1807, 2018.

[103] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[104] M. Vlachos, D. Gunopoulos, and G. Kollios. Discovering similar multidimensional trajectories. In *icde*, page 0673. IEEE, 2002.

[105] Z. Wang, J. S. Merel, S. E. Reed, N. de Freitas, G. Wayne, and N. Heess. Robust imitation of diverse behaviors. *Advances in Neural Information Processing Systems*, 30, 2017.

[106] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[107] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[108] C. J. C. H. Watkins. Learning from delayed rewards. 1989.

[109] K. Wu, E. Otoo, and A. Shoshani. Optimizing connected component labeling algorithms. In *Medical Imaging 2005: Image Processing*, volume 5747, pages 1965–1976. International Society for Optics and Photonics, 2005.

[110] D. Xu and Y. Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015.

[111] A. Ye. You don't understand neural networks until you understand the universal approximation theorem. https://medium.com/analytics-vidhya/you-dont-understand-neural-networks-until-you-understand-the-universal-approximation-theorem-85b3e7677126. Accessed on 28.03.23.

[112] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. *ACM sigmod record*, 25(2):103–114, 1996.

[113] B. Zheng, S. Verma, J. Zhou, I. Tsang, and F. Chen. Imitation learning: Progress, taxonomies and opportunities. *arXiv preprint arXiv:2106.12177*, 2021.

[114] X. Zhu, Z. Xu, and T. Khot. How creative is your writing? In *Proceedings of the workshop on computational approaches to linguistic creativity*, pages 87–93, 2009.

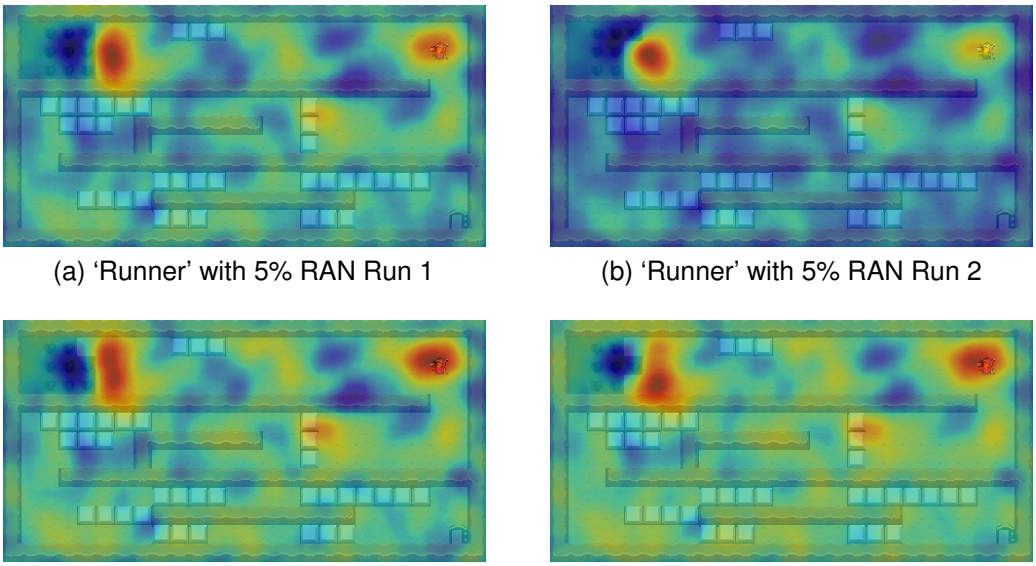
[115] Z. Zhu, K. Lin, B. Dai, and J. Zhou. Off-policy imitation learning from observations. *Advances in Neural Information Processing Systems*, 33:12402–12413, 2020.

Appendix A Representation Analysis

Below further examples are provided of the representation analysis conducted.

A.1 Black Smoke

A.1.1 RISER



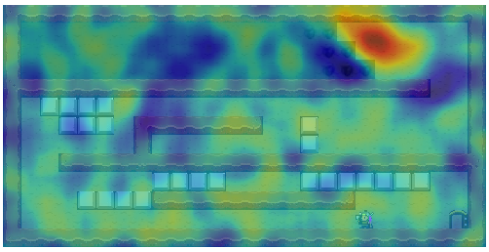
(a) ‘Runner’ with 5% RAN Run 1

(b) ‘Runner’ with 5% RAN Run 2

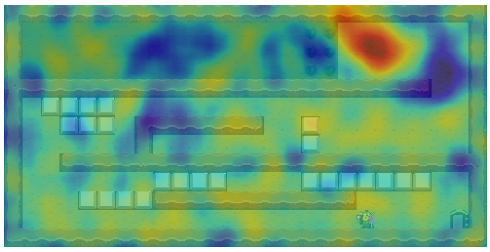
(c) ‘Collector’ with 5% RAN Run 1

(d) ‘Collector’ with 5% RAN Run 2

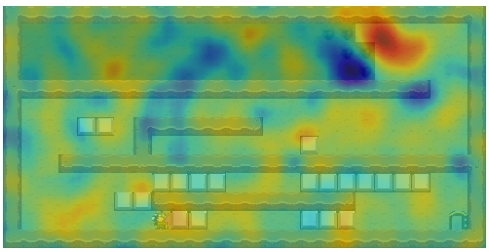
Figure 220: Frame 5 Importance Maps Using ‘LB’ Masks



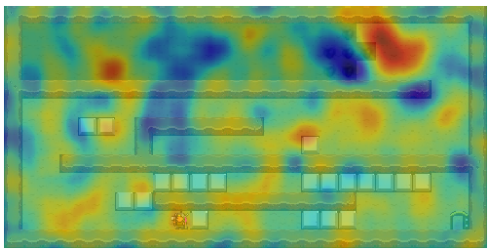
(a) ‘Runner’ with 5% RAN Run 1



(b) ‘Runner’ with 5% RAN Run 2

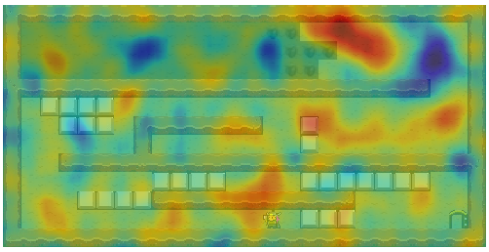


(c) ‘Collector’ with 5% RAN Run 1

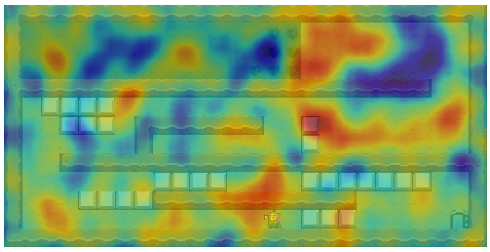


(d) ‘Collector’ with 5% RAN Run 2

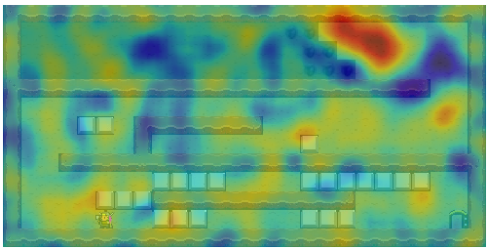
Figure 221: Frame 75 Importance Maps Using ‘LB’ Masks



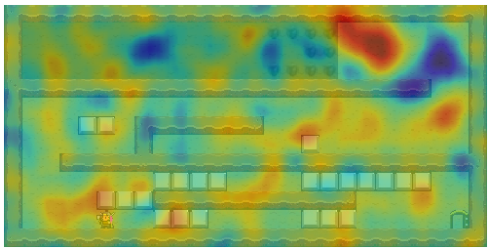
(a) ‘Runner’ with 5% RAN Run 1



(b) ‘Runner’ with 5% RAN Run 2

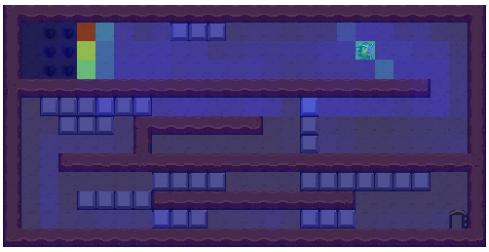


(c) ‘Collector’ with 5% RAN Run 1

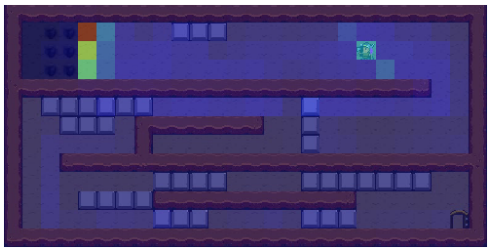


(d) ‘Collector’ with 5% RAN Run 2

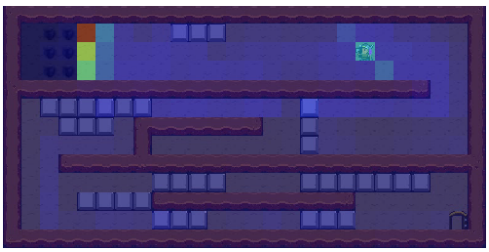
Figure 222: Frame 65 Importance Maps Using ‘LB’ Masks



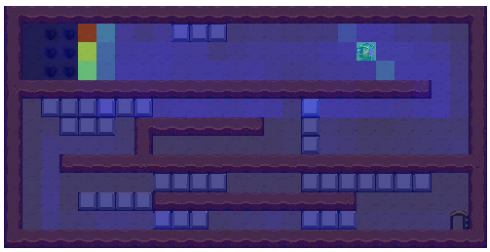
(a) ‘Runner’ with 5% RAN Run 1



(b) ‘Runner’ with 5% RAN Run 2

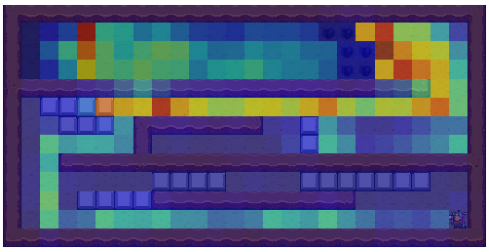


(c) ‘Collector’ with 5% RAN Run 1

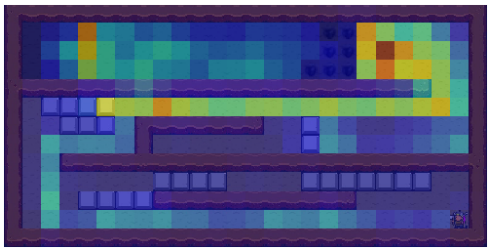


(d) ‘Collector’ with 5% RAN Run 2

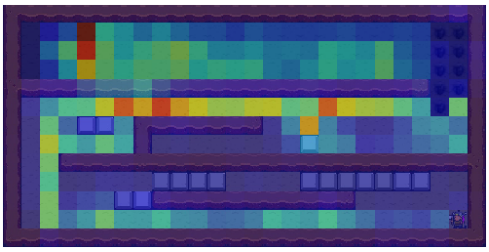
Figure 223: Frame 0 Importance Maps Using ‘SB’ Masks



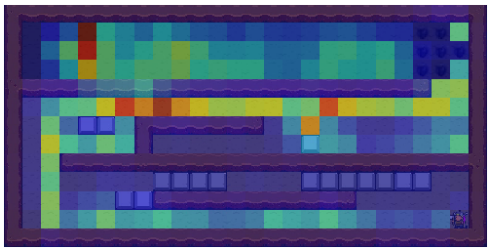
(a) ‘Runner’ with 5% RAN Run 1



(b) ‘Runner’ with 5% RAN Run 2



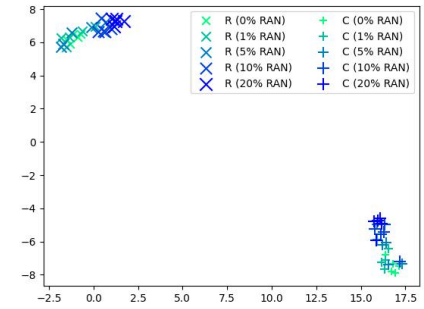
(c) ‘Collector’ with 5% RAN Run 1



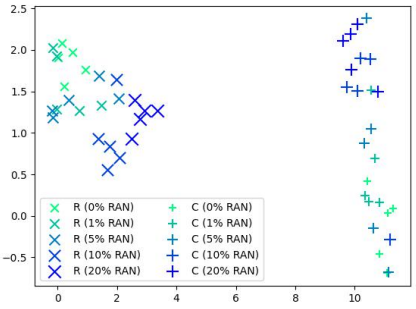
(d) ‘Collector’ with 5% RAN Run 2

Figure 224: Last Frame Importance Maps Using ‘SB’ Masks

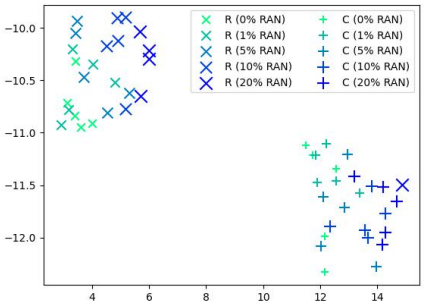
A.1.2 Projections



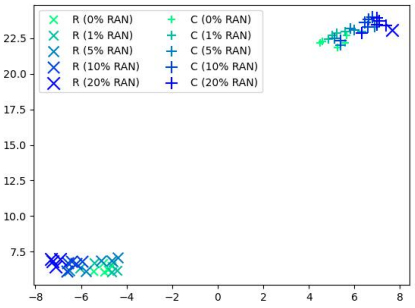
(a) Projection 1



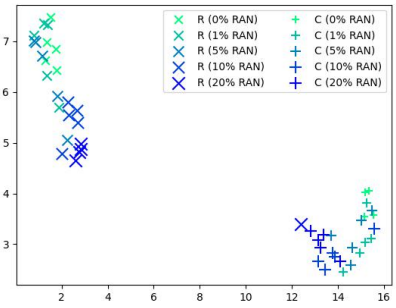
(b) Projection 2



(c) Projection 3



(d) Projection 4

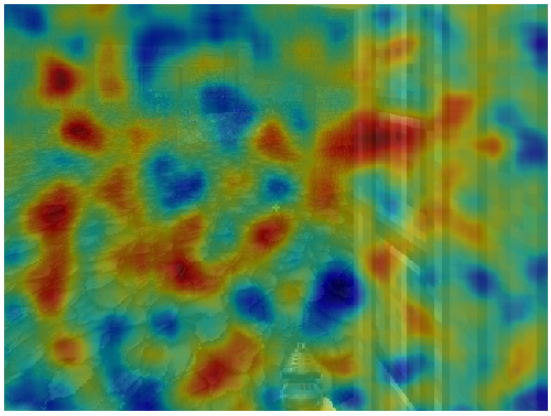


(e) Projection 5

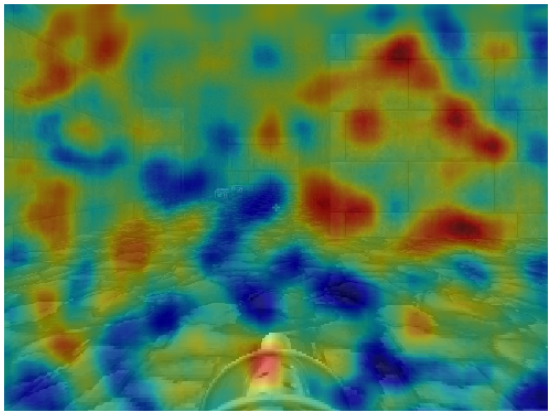
Figure 225: Projections of Blacksmoke Gameplays using the STDIM-VAE

A.2 VizDoom

A.2.1 RISER

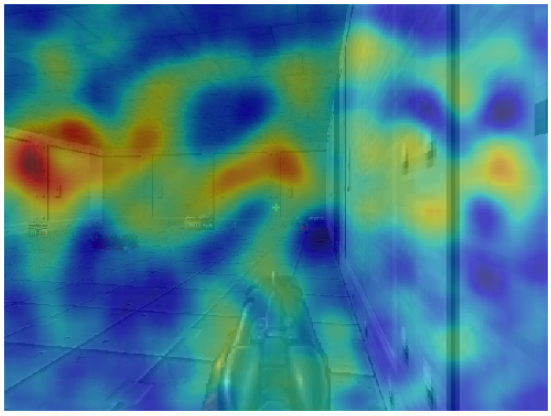


(a) Frame 18 of a Gameplay Utilising ‘Corner Camping’ with the ‘Shotgun’ (Playstyle ID 4) Run 1

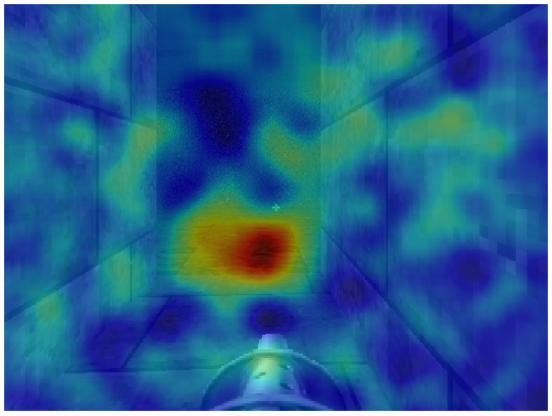


(b) Frame 117 of a Gameplay Traversing the ‘Long’ Route with the ‘Chain Gun’ (Playstyle ID 2) Run 1

Figure 226: Importance Maps that Demonstrated a Broad Frame Focus

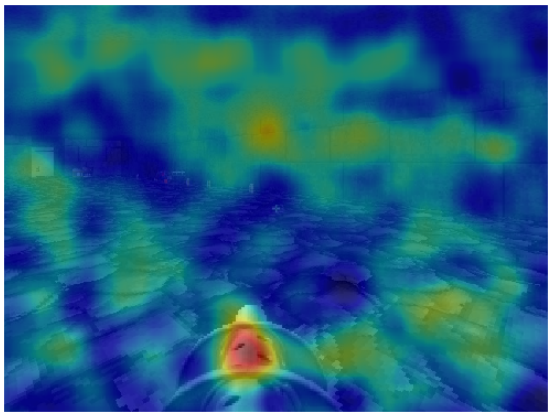


(a) Frame 188 of a Gameplay Utilising ‘Corner Camping’ with the ‘Shotgun’ (Playstyle ID 4) Run 1

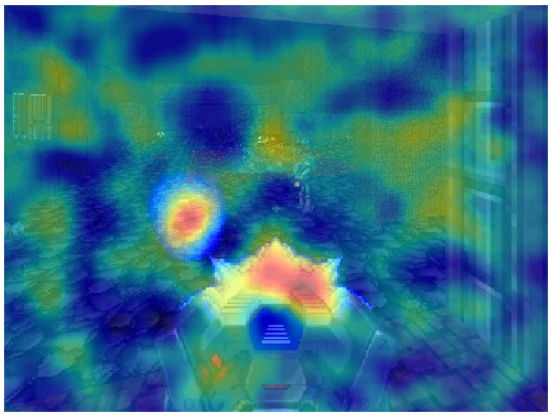


(b) Frame 104 of a Gameplay Traversing the ‘Long’ Route with the ‘Chain Gun’ (Playstyle ID 2) Run 1

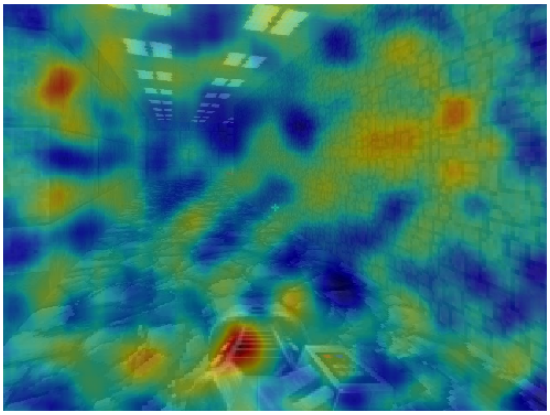
Figure 227: Importance Maps that Demonstrated a Focus on the Player’s Surroundings



(a) Frame 63 of a Gameplay Traversing the ‘Long’ Route with the ‘Chain Gun’ (Playstyle ID 2) Run 1

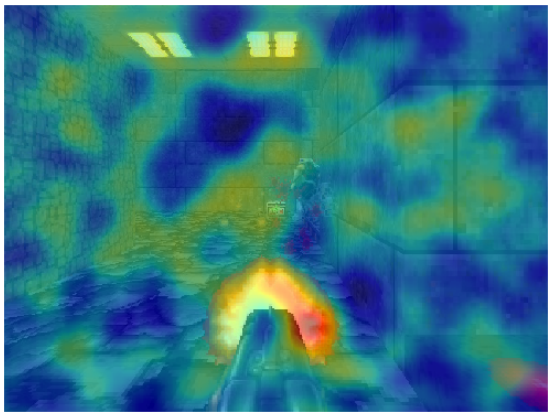


(b) Frame 299 of a Gameplay Utilising ‘Window Camping’ at ‘Window 1’ with the ‘BFG9000’ (Playstyle ID 6) Run 1

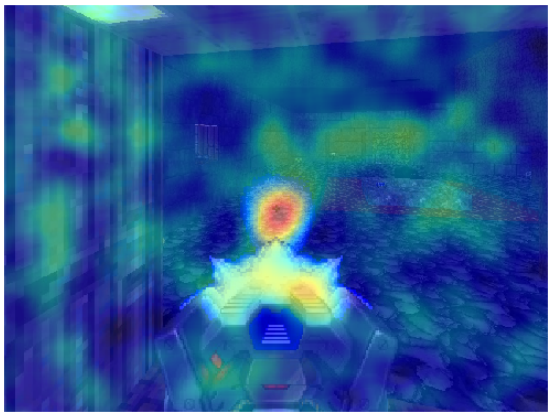


(c) Frame 78 of a Gameplay that ‘Camps’ in the ‘Main Room’ with the ‘Plasma Rifle’ (Playstyle ID 9) Run 1

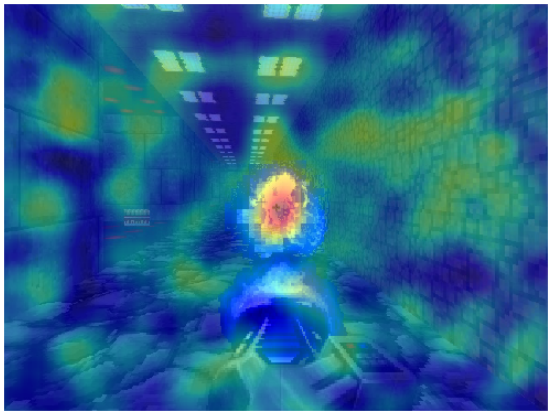
Figure 228: Importance Maps that Demonstrated a Focus on the Weapon



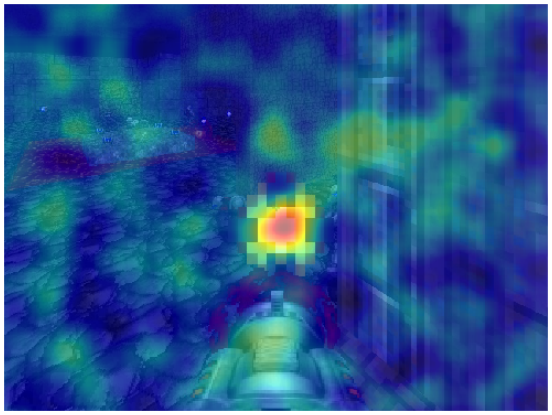
(a) Frame 43 of a Gameplay Utilising ‘Corner Camping’ with the ‘Shotgun’ (Playstyle ID 4) Run 1



(b) Frame 215 of a Gameplay Utilising ‘Window Camping’ at ‘Window 1’ with the ‘BFG9000’ (Playstyle ID 6) Run 1



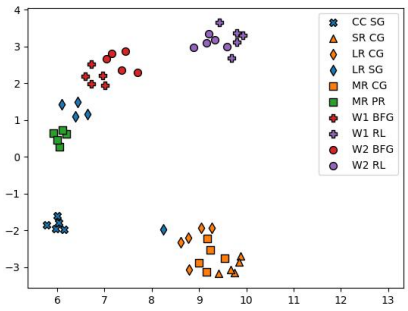
(c) Frame 336 of a Gameplay that ‘Camps’ in the ‘Main Room’ with the ‘Plasma Rifle’ (Playstyle ID 9) Run 1



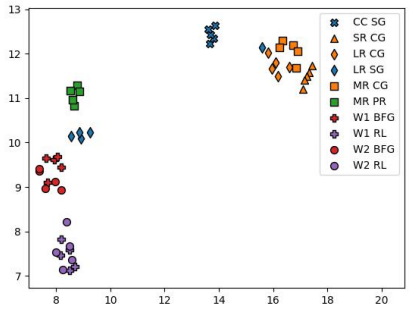
(d) Frame 141 of a Gameplay Utilising ‘Window Camping’ at ‘Window 1’ with the ‘Rocket Launcher’ (Playstyle ID 5) Run 1

Figure 229: Importance Maps that Demonstrated a Focus on Fired Projectiles

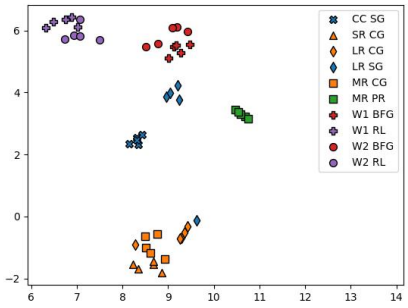
A.2.2 Projections



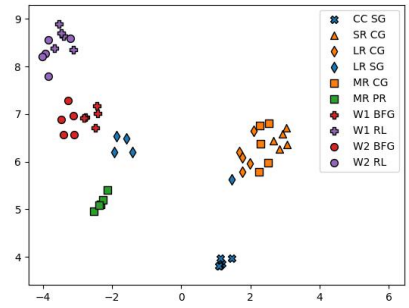
(a) Projection 1



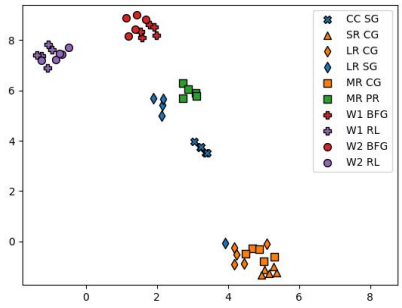
(b) Projection 2



(c) Projection 3

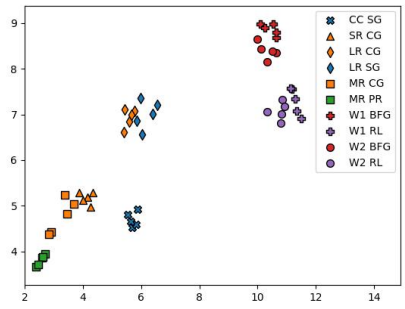


(d) Projection 4

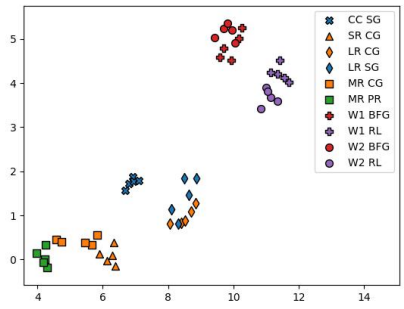


(e) Projection 5

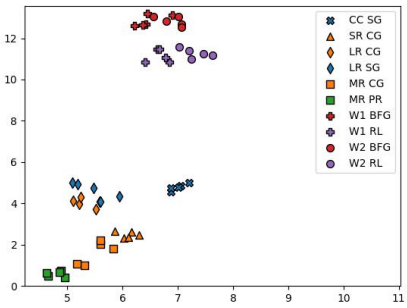
Figure 230: Projections of VizDoom Gameplays using the VAE



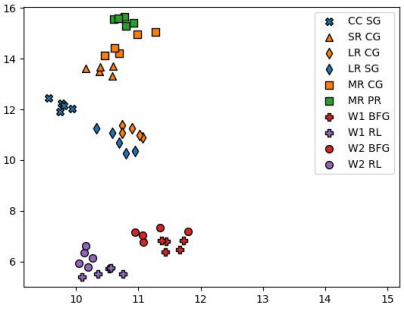
(a) Projection 1



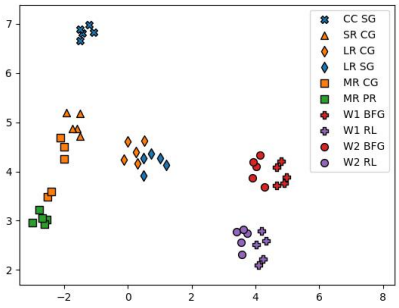
(b) Projection 2



(c) Projection 3

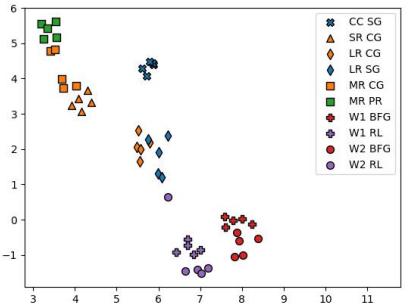


(d) Projection 4

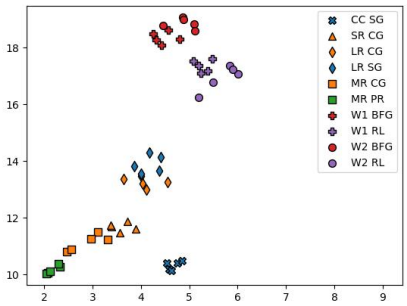


(e) Projection 5

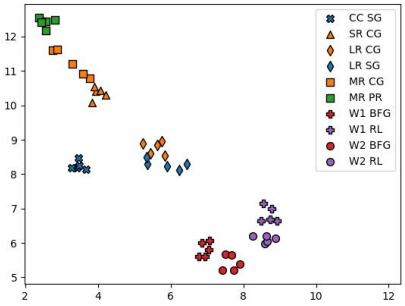
Figure 231: Projections of VizDoom Gameplays using the ST-DIM



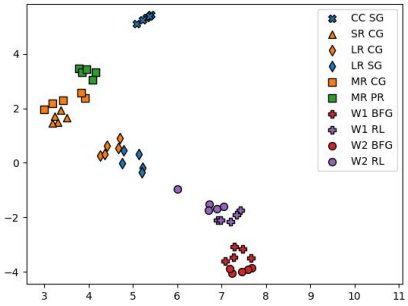
(a) Projection 1



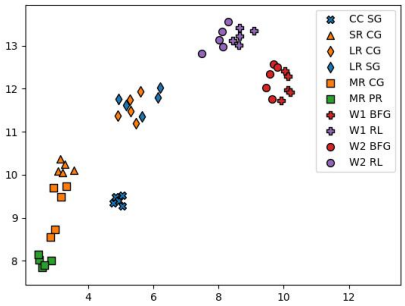
(b) Projection 2



(c) Projection 3



(d) Projection 4

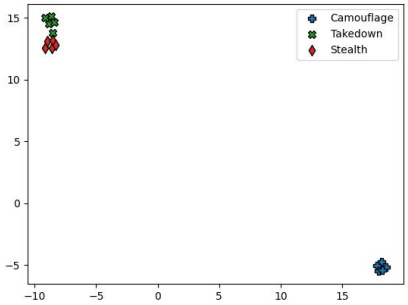


(e) Projection 5

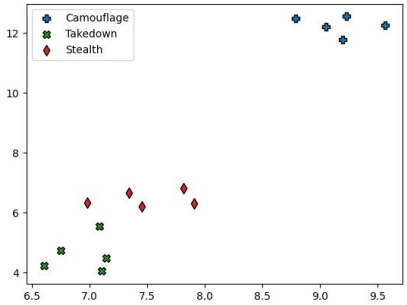
Figure 232: Projections of VizDoom Gameplays using the STDIM-VAE

A.3 Hitman

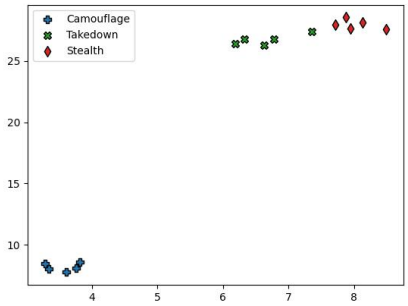
A.3.1 Projections



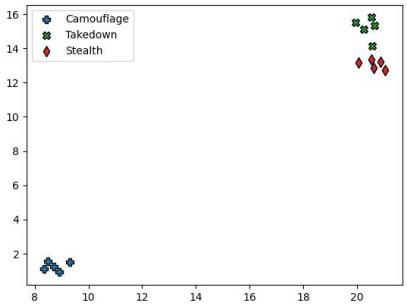
(a) Projection 1



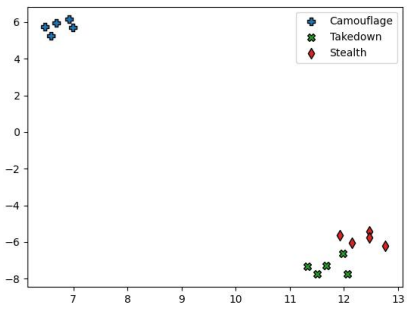
(b) Projection 2



(c) Projection 3



(d) Projection 4



(e) Projection 5

Figure 233: Projections of Hitman Gameplays using the STDIM-VAE

Appendix B Clustering Results

Below the clustering results for the full breath of setups explored can be found. However, it is noted that for some game variables a value of zero was possible, hence causing an error for the standard cosine metric. For these instances the relevant graph sections have been left blank. Further instances have also been left blank where issues with clustering have arisen from bad variable selection causing all gameplays to be represented by the same vector.

B.1 Black Smoke

B.1.1 Game Variable Series Clustering

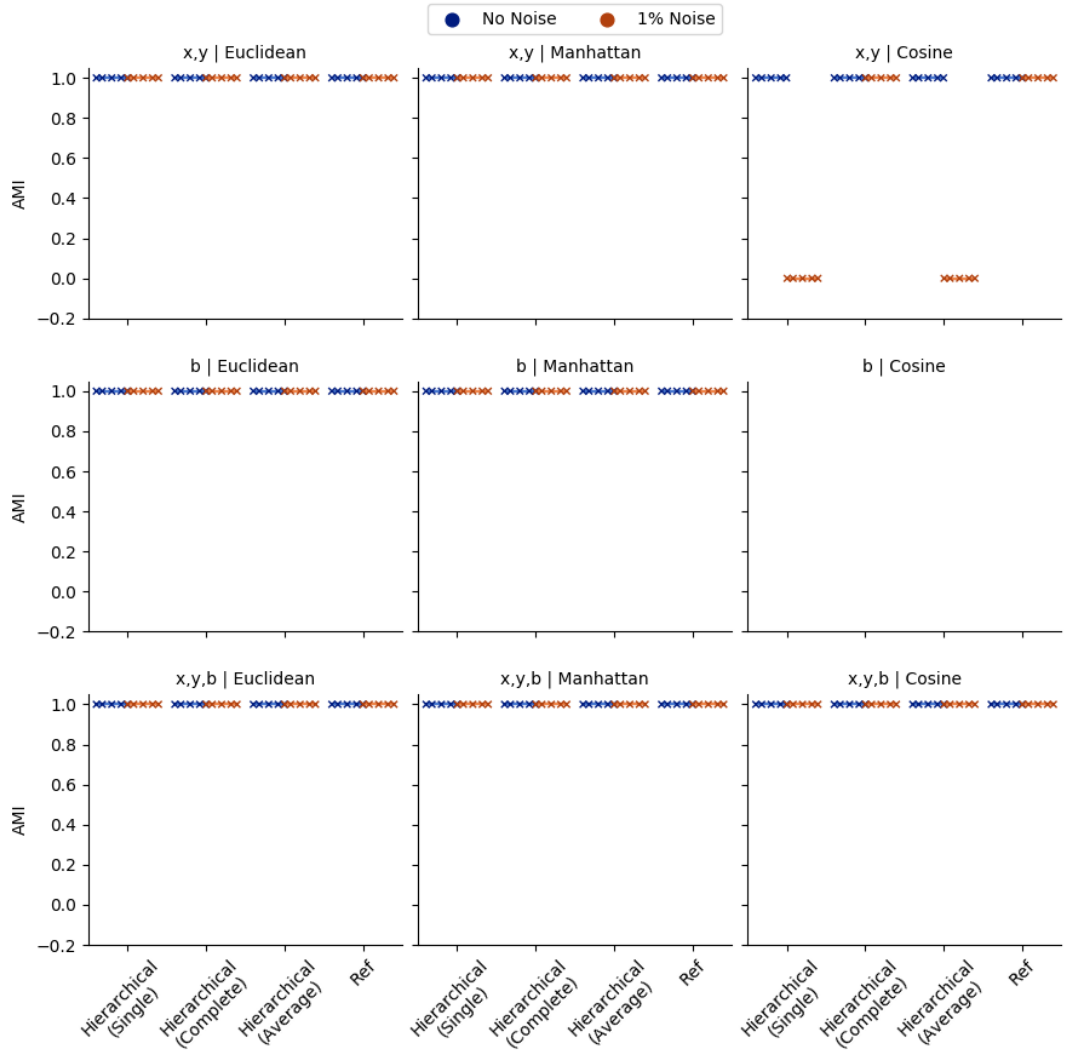


Figure 234: Series Game Variables Clustering on 0% and 1% RAN Datasets

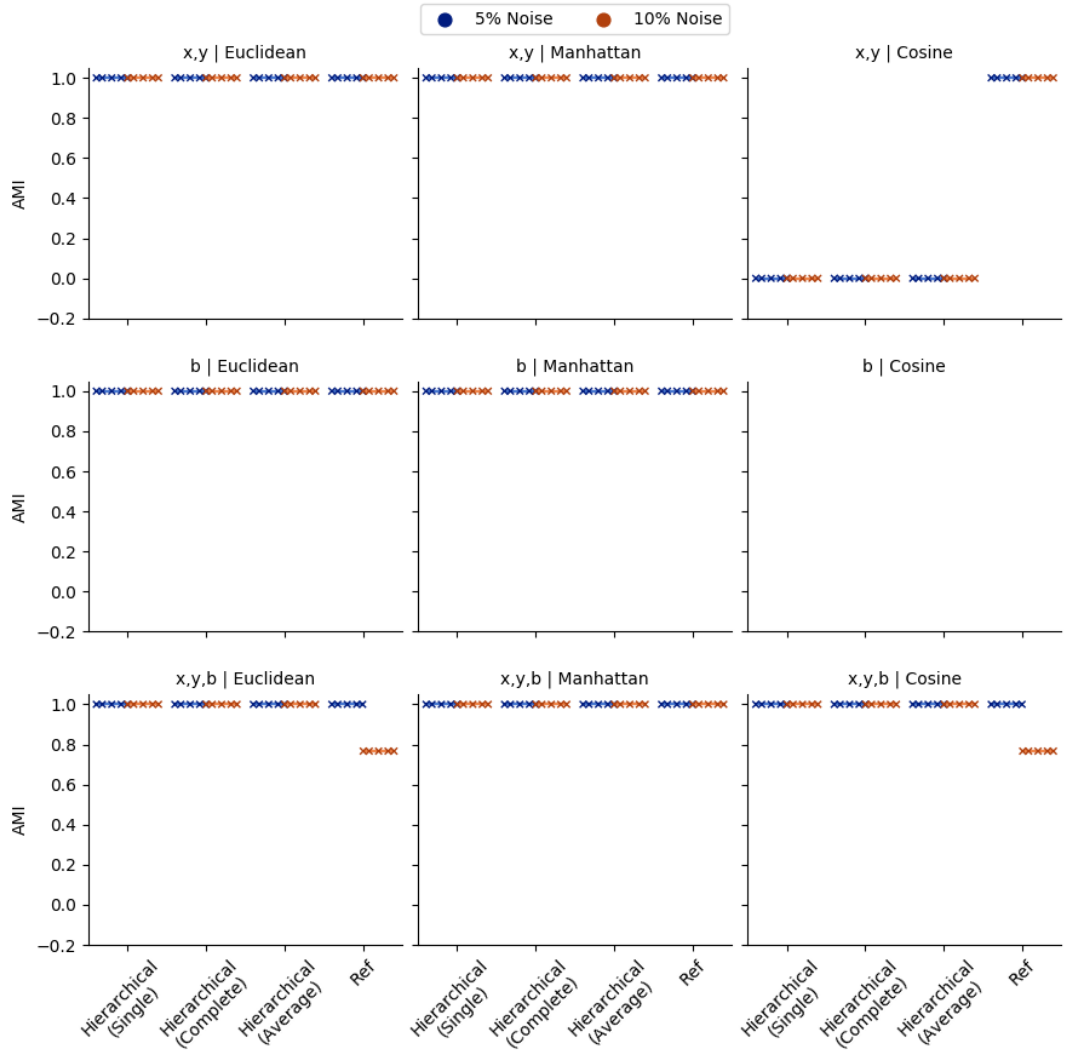


Figure 235: Series Game Variables Clustering on 5% and 10% RAN Datasets

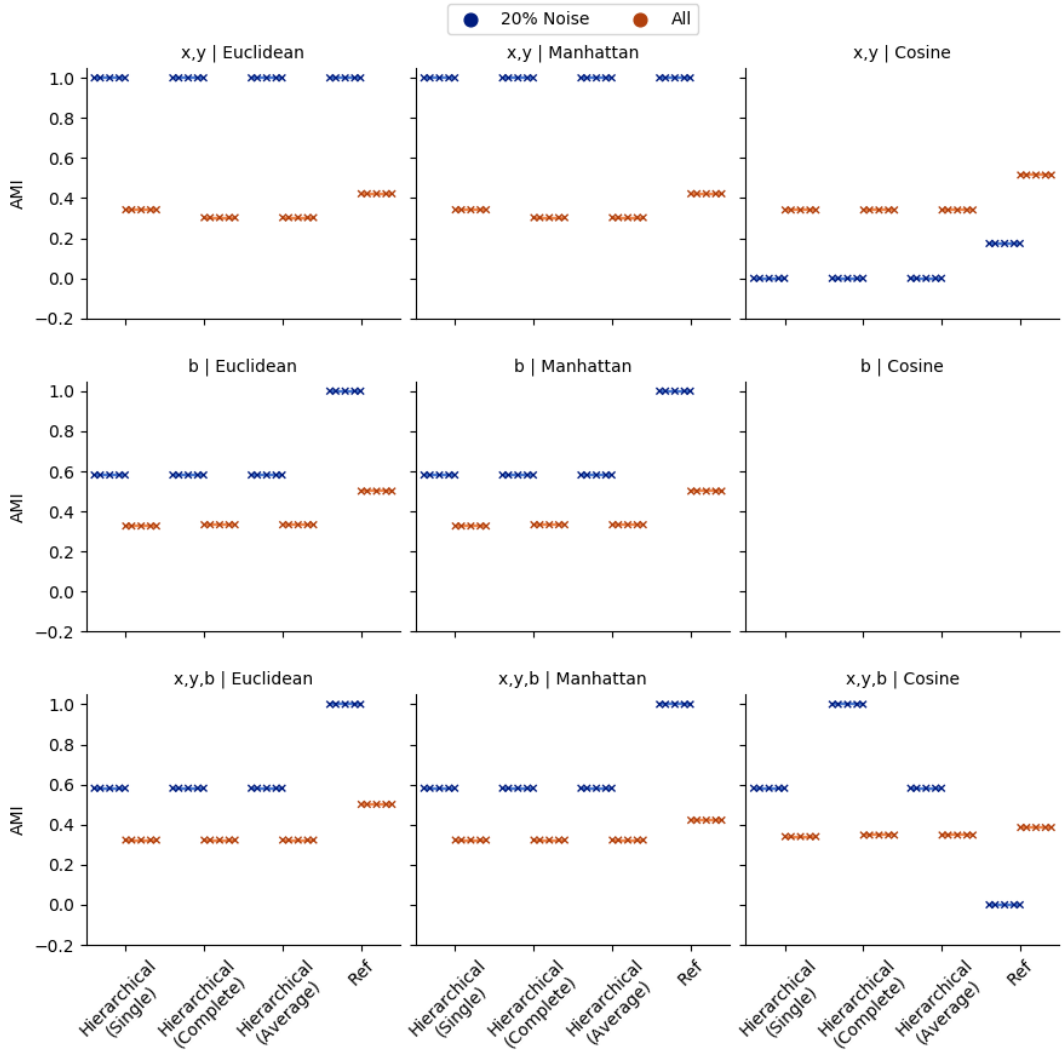


Figure 236: Series Game Variables Clustering on 20% and All RAN Datasets

B.1.2 Game Variable End Value Clustering

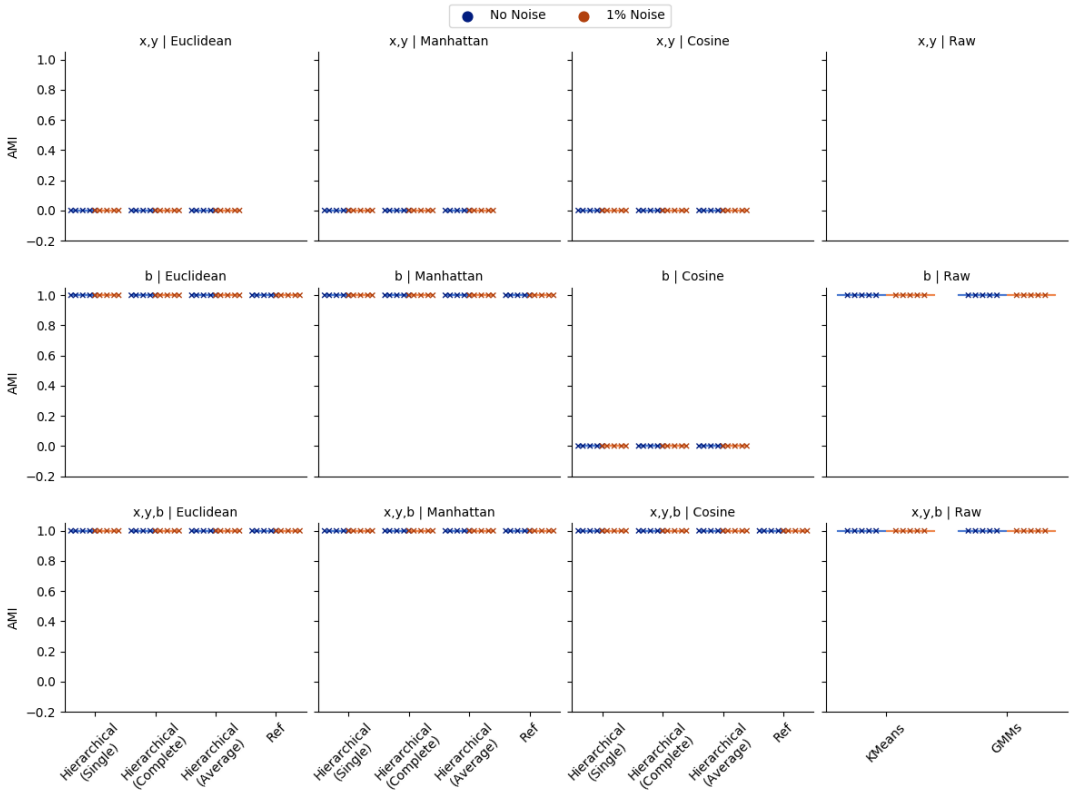


Figure 237: End Game Variables Clustering on 0% and 1% RAN Datasets

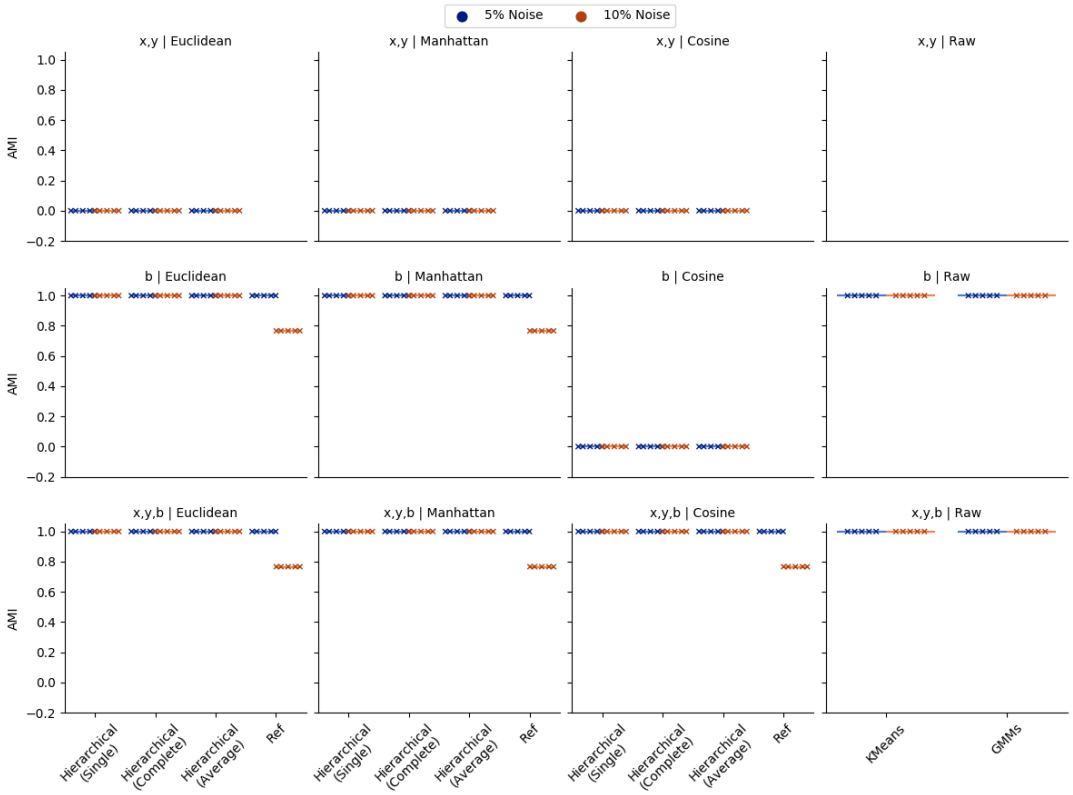


Figure 238: End Game Variables Clustering on 5% and 10% RAN Datasets

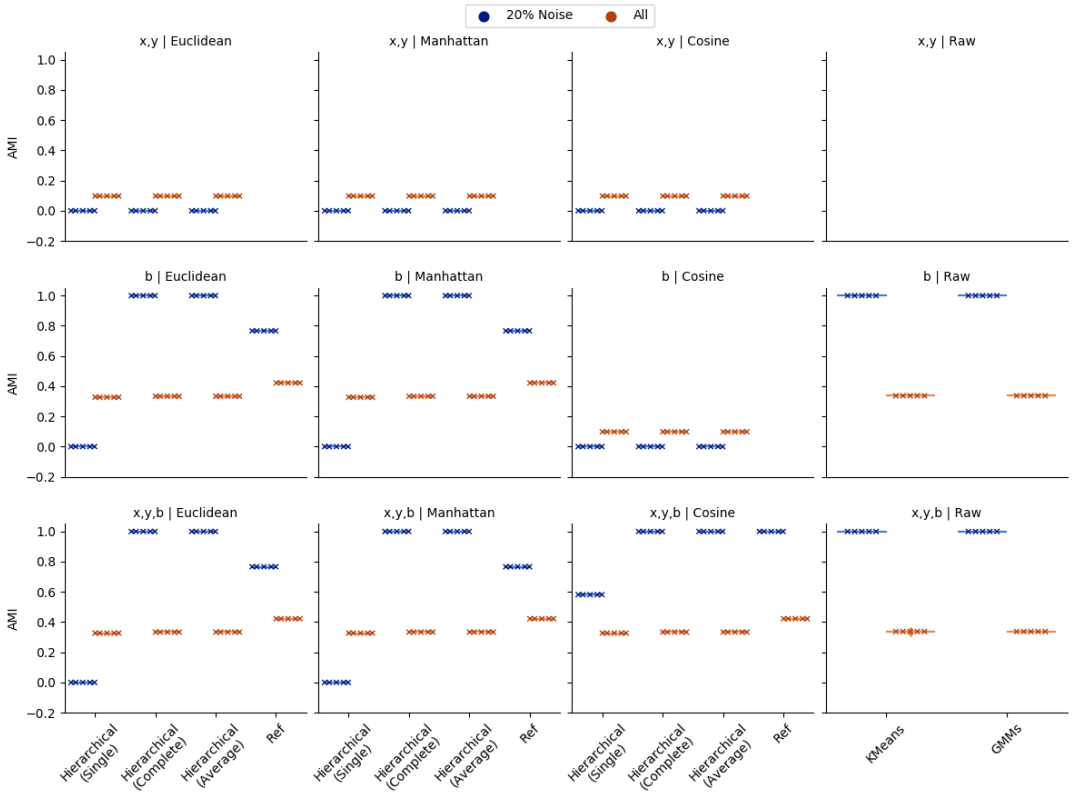


Figure 239: End Game Variables Clustering on 20% and All RAN Datasets

B.1.3 Representation Clustering

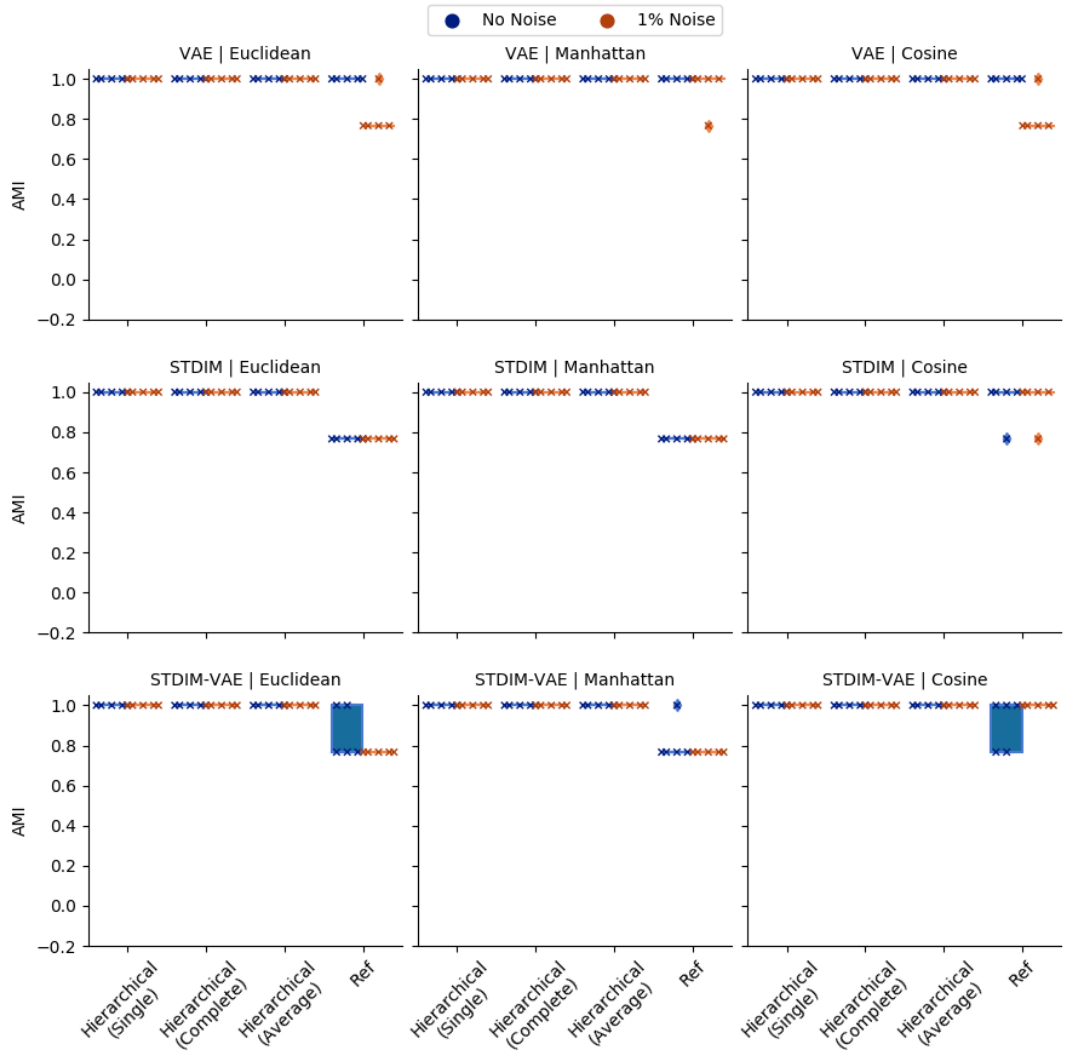


Figure 240: Representation Clustering on 0% and 1% RAN Datasets

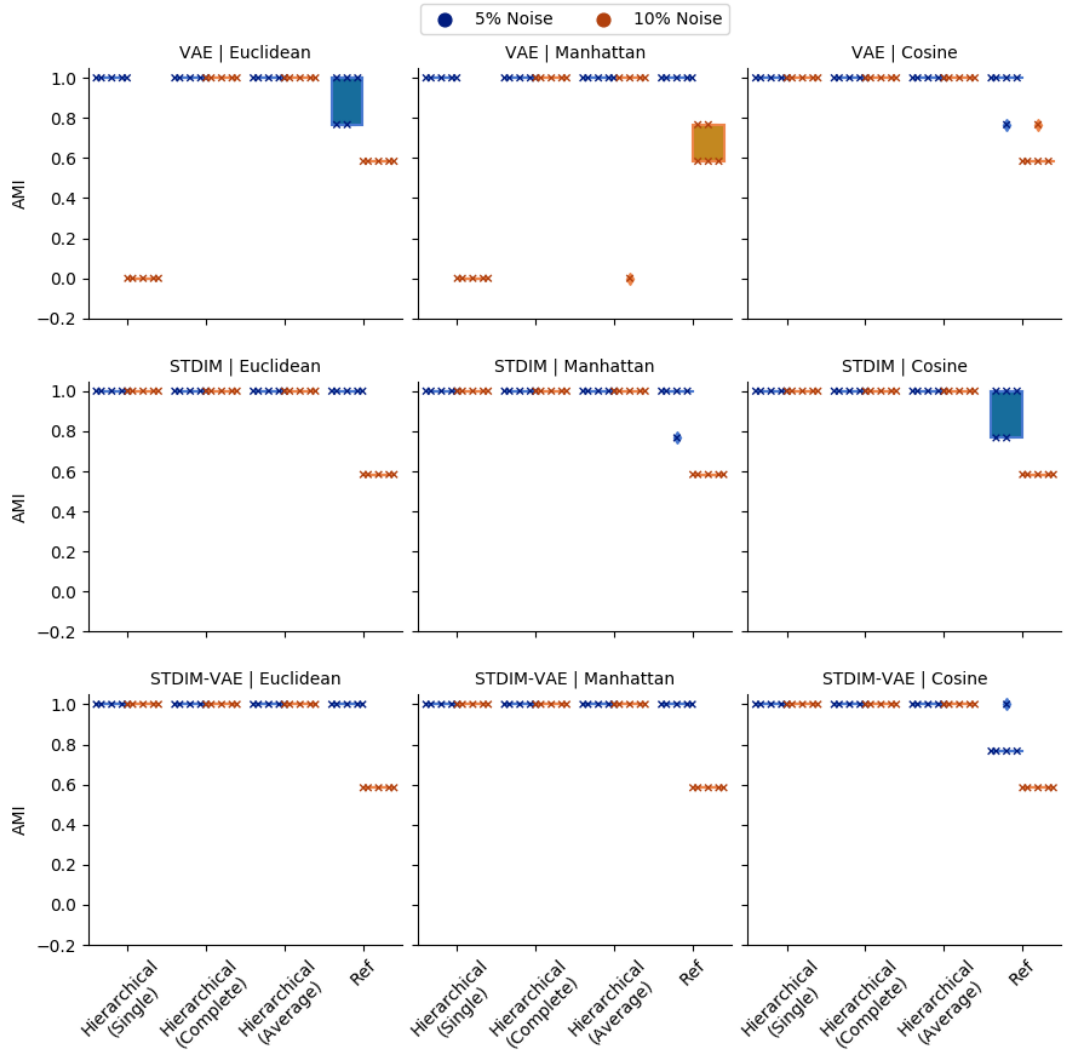


Figure 241: Representation Clustering on 5% and 10% RAN Datasets

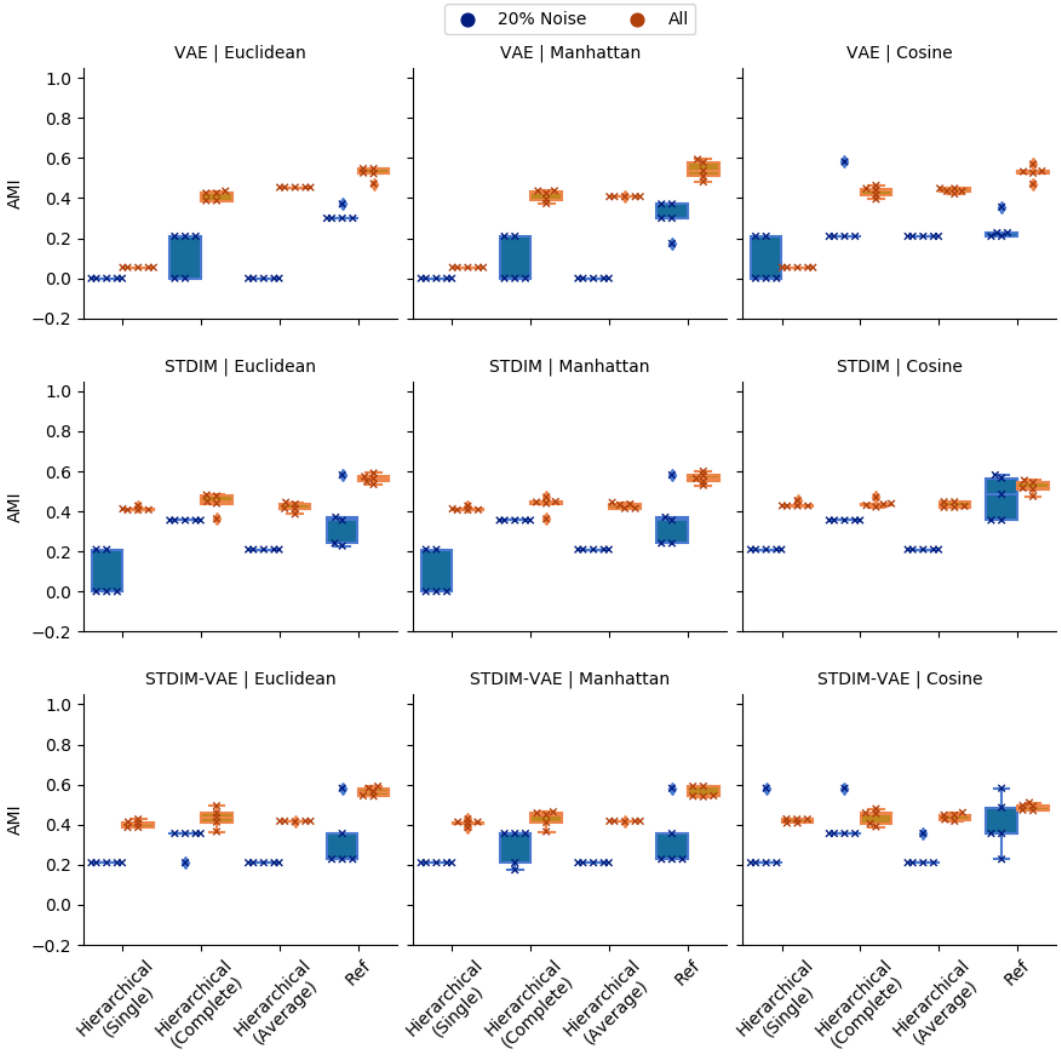


Figure 242: Representation Clustering on 20% and All RAN Datasets

B.2 VizDoom

B.2.1 Game Variable Series Clustering

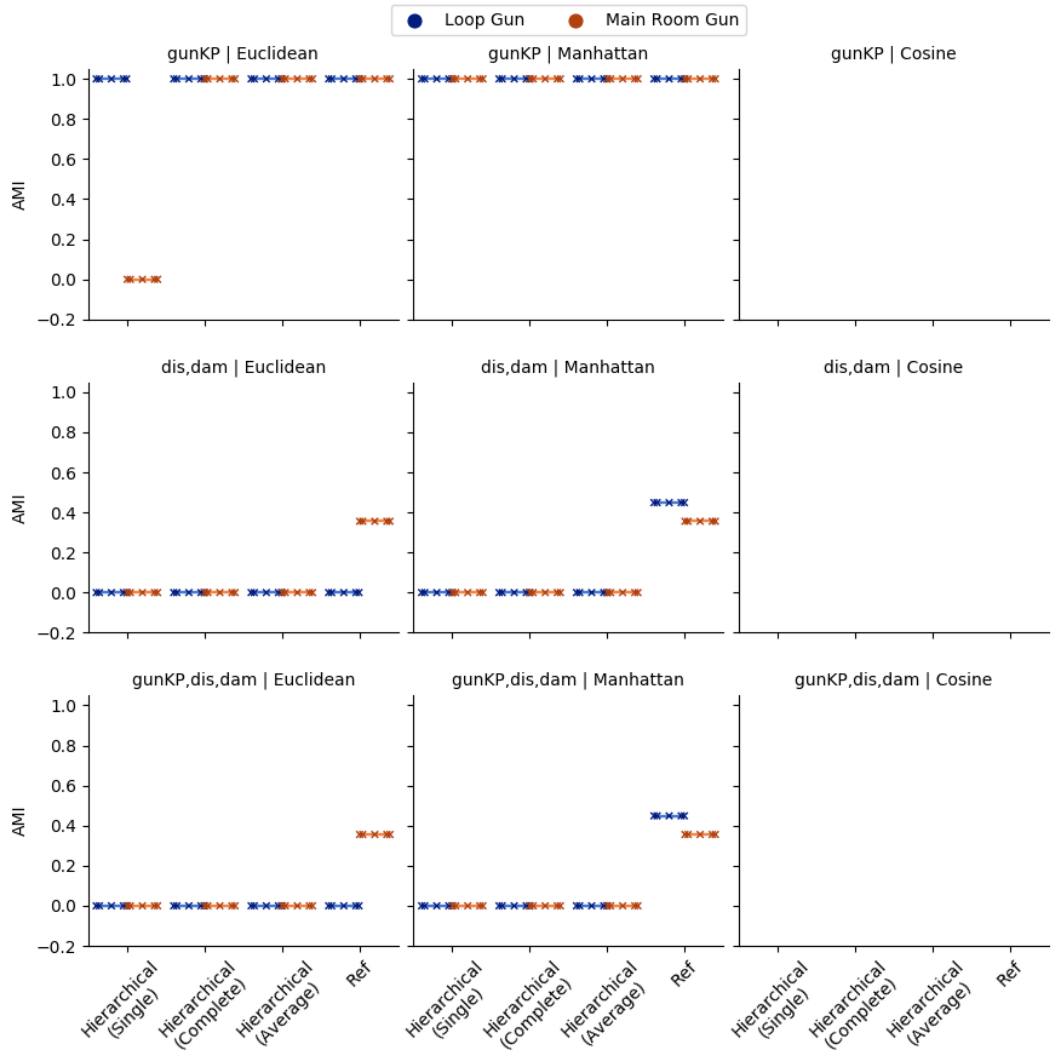


Figure 243: Series Game Variable Clustering on ‘Loop Gun’ and‘Main Room Gun’ Datasets

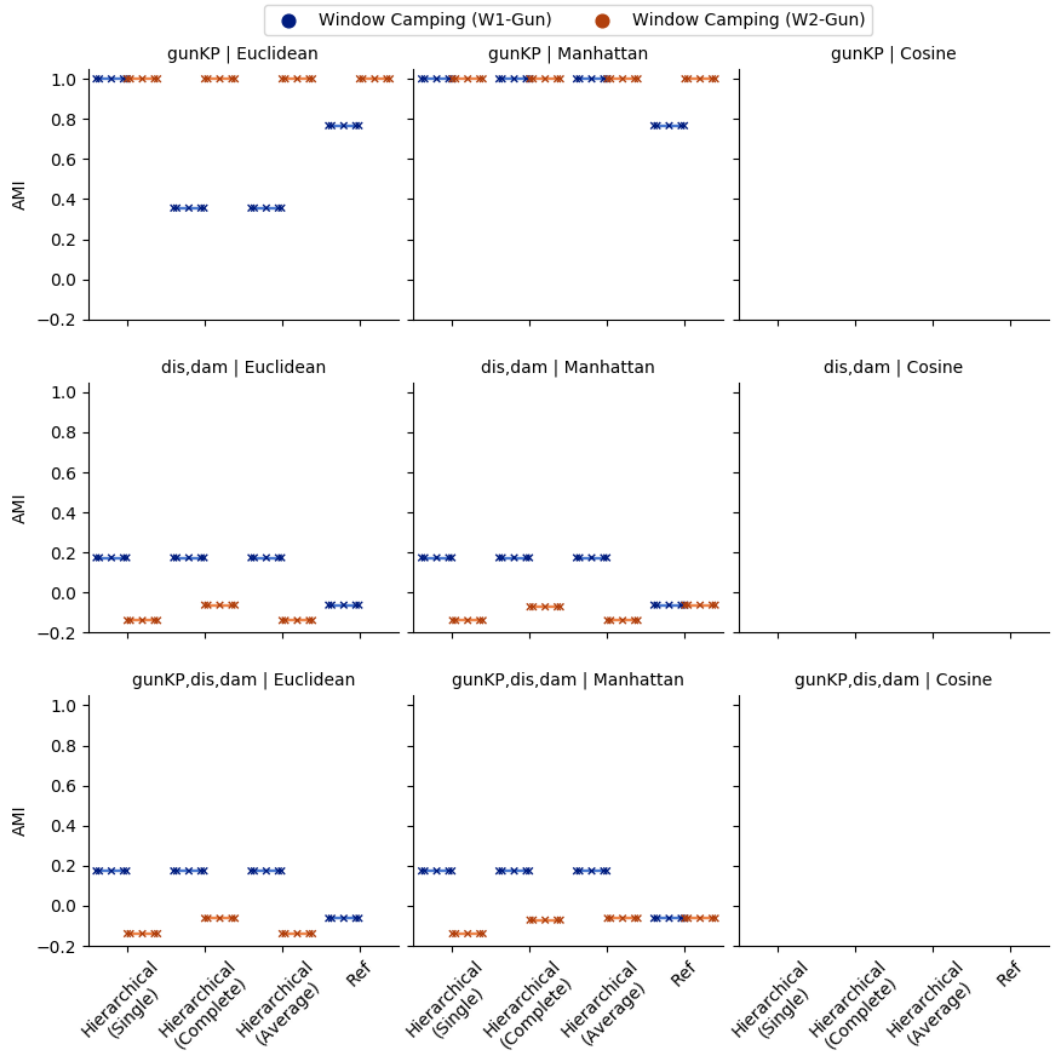


Figure 244: Series Game Variable Clustering on ‘Window 1 Gun’ and ‘Window 2 Gun’ Datasets

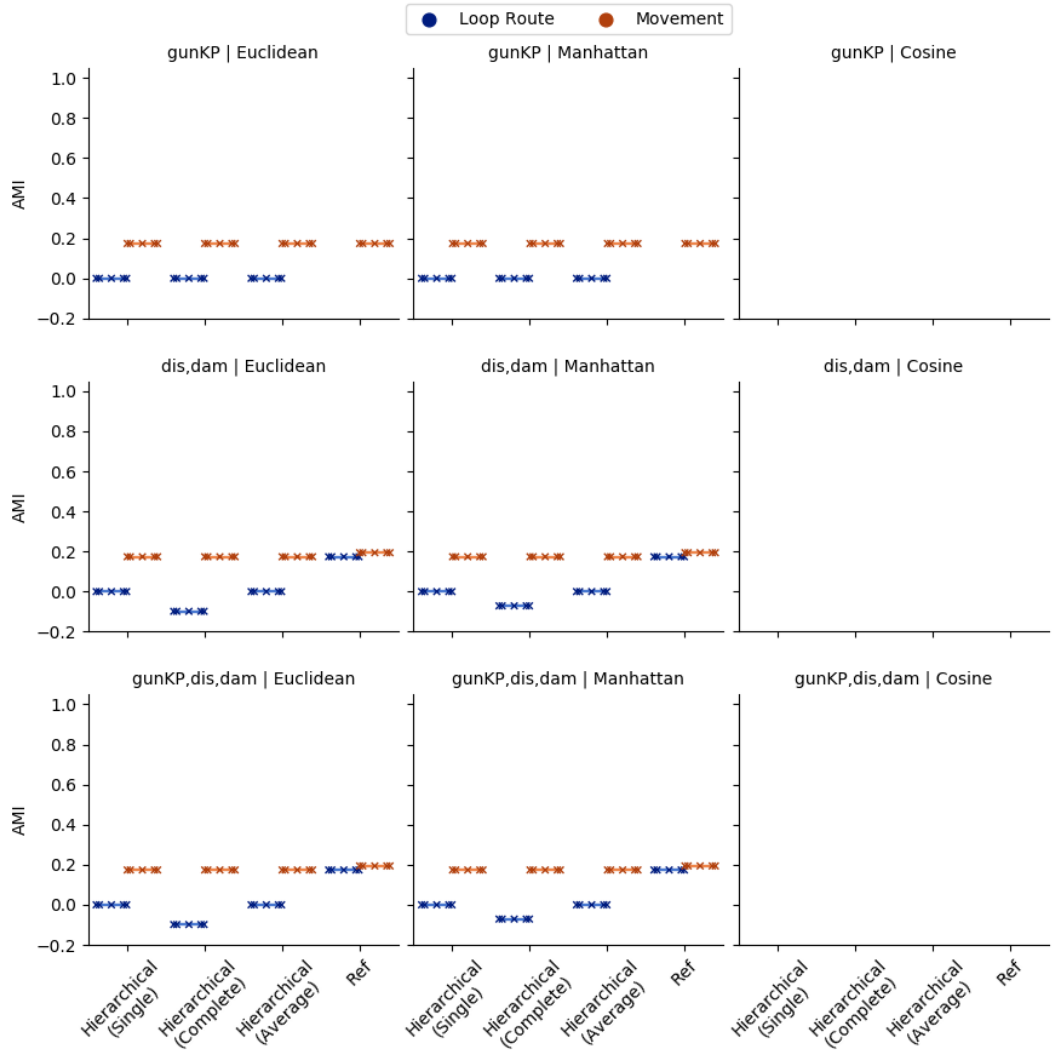


Figure 245: Series Game Variable Clustering on ‘Loop Route’ and ‘Movement’ Datasets

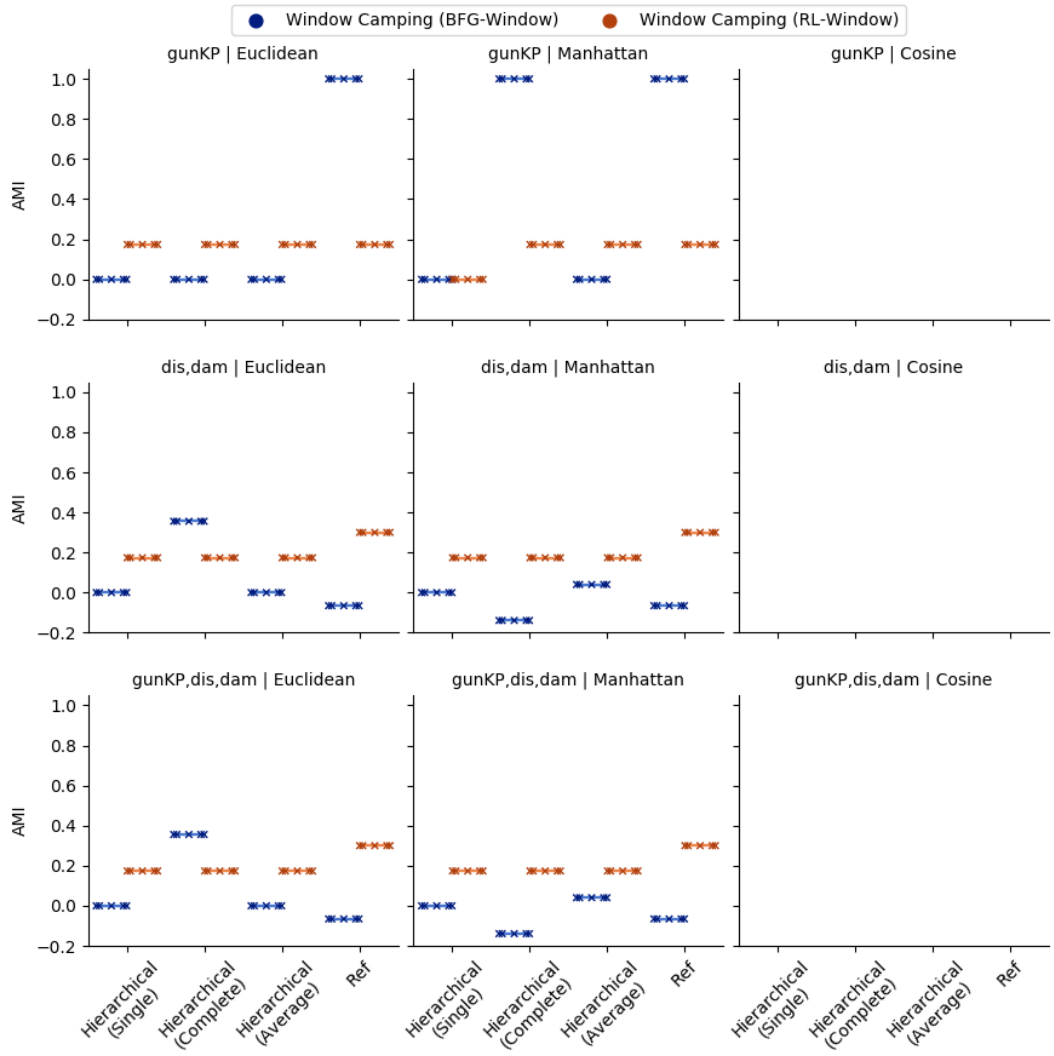


Figure 246: Series Game Variable Clustering on ‘BFG9000 Window’ and ‘Rocket Launcher Window’ Datasets

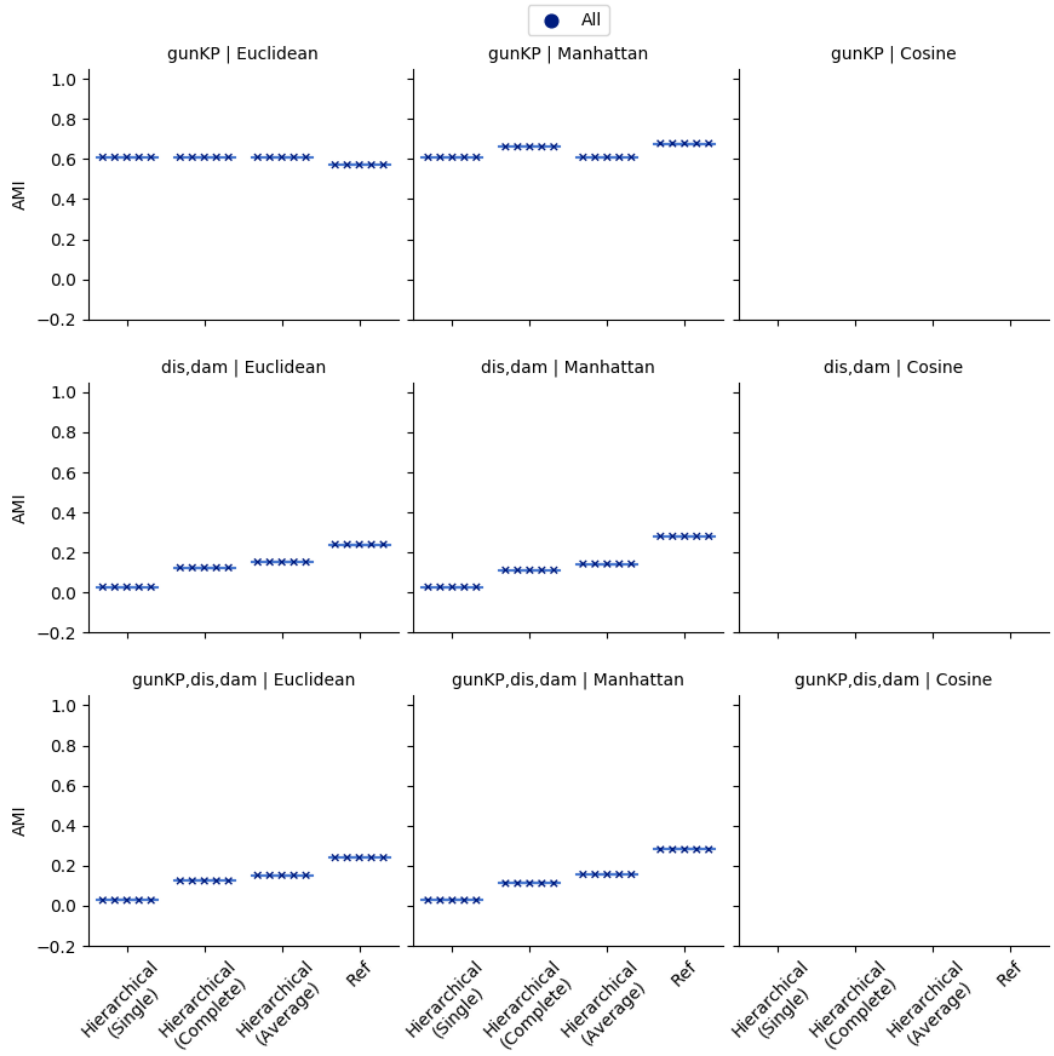


Figure 247: Series Game Variable Clustering on Full Dataset (Ground Truth = Gun and Position)

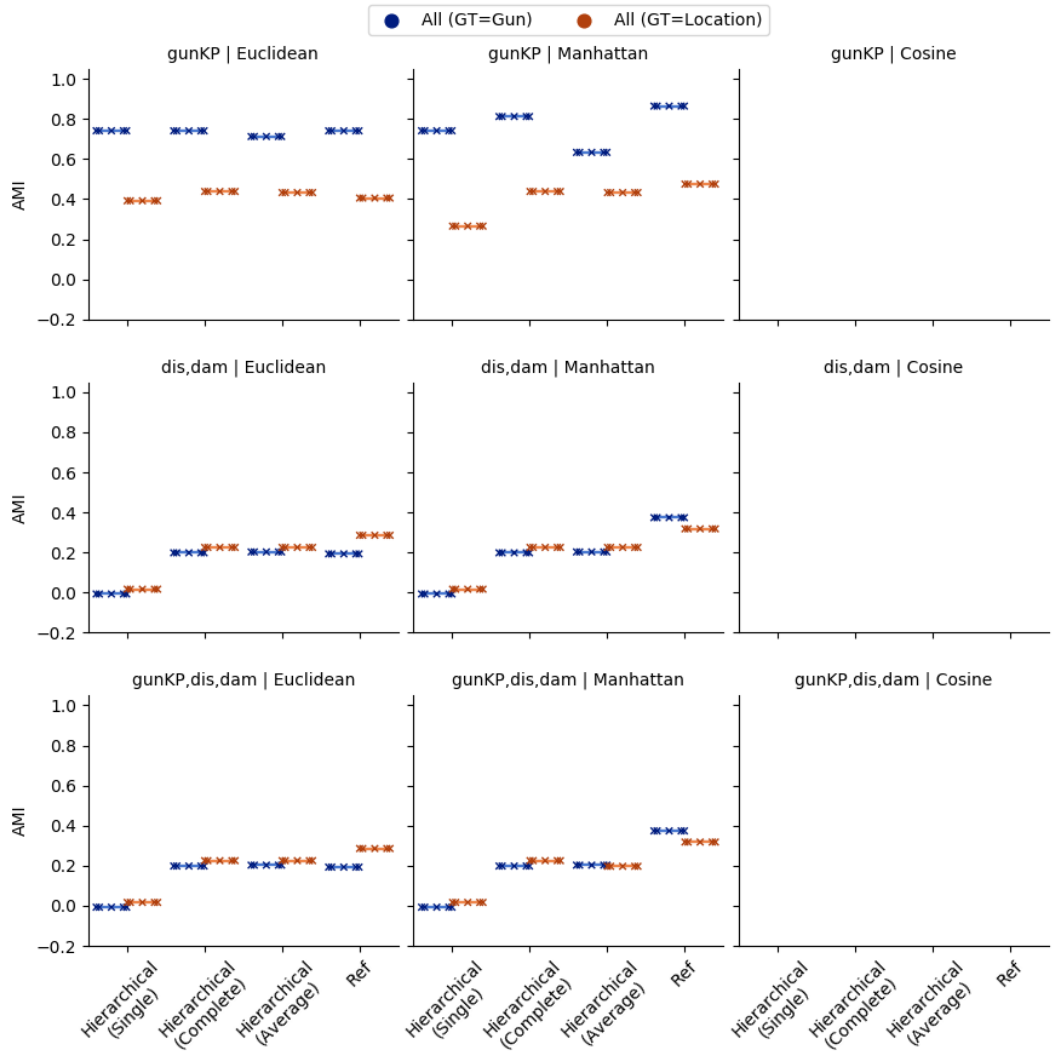


Figure 248: Series Game Variable Clustering on Full Dataset (Ground Truth = Gun or Position)

B.2.2 Game Variable End Value Clustering

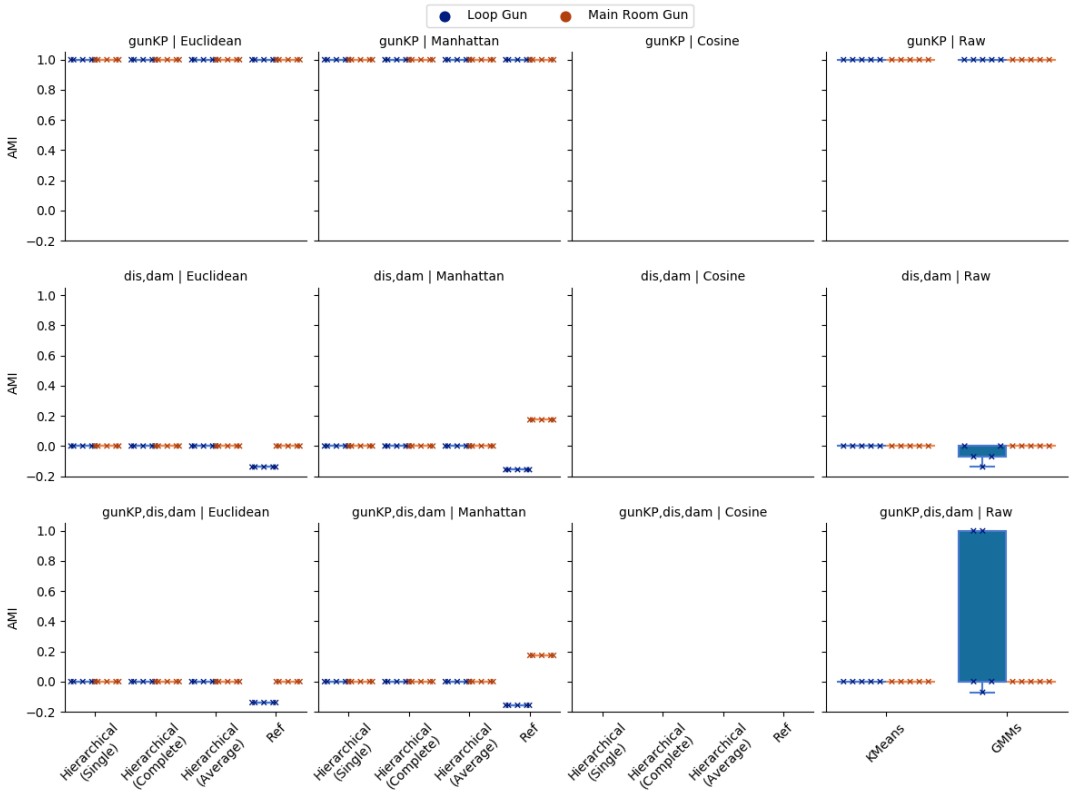


Figure 249: End Game Variable Clustering on 'Loop Gun' and 'Main Room Gun' Datasets

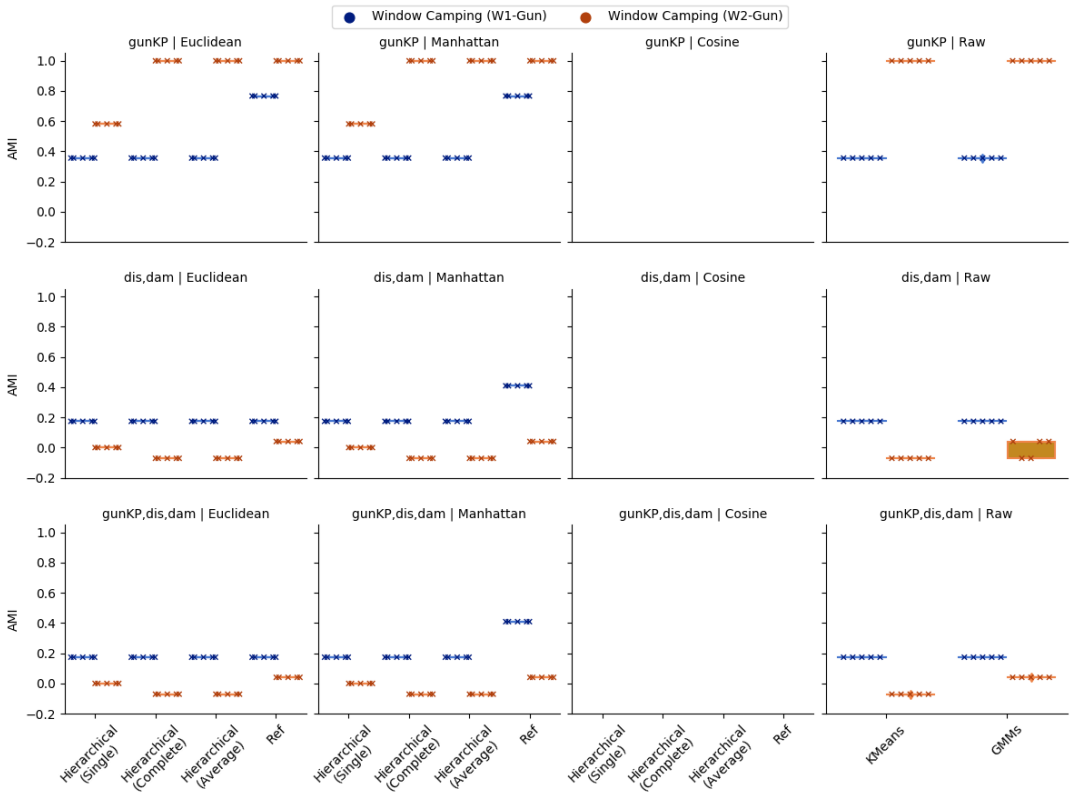


Figure 250: End Game Variable Clustering on ‘Window 1 Gun’ and ‘Window 2 Gun’ Datasets

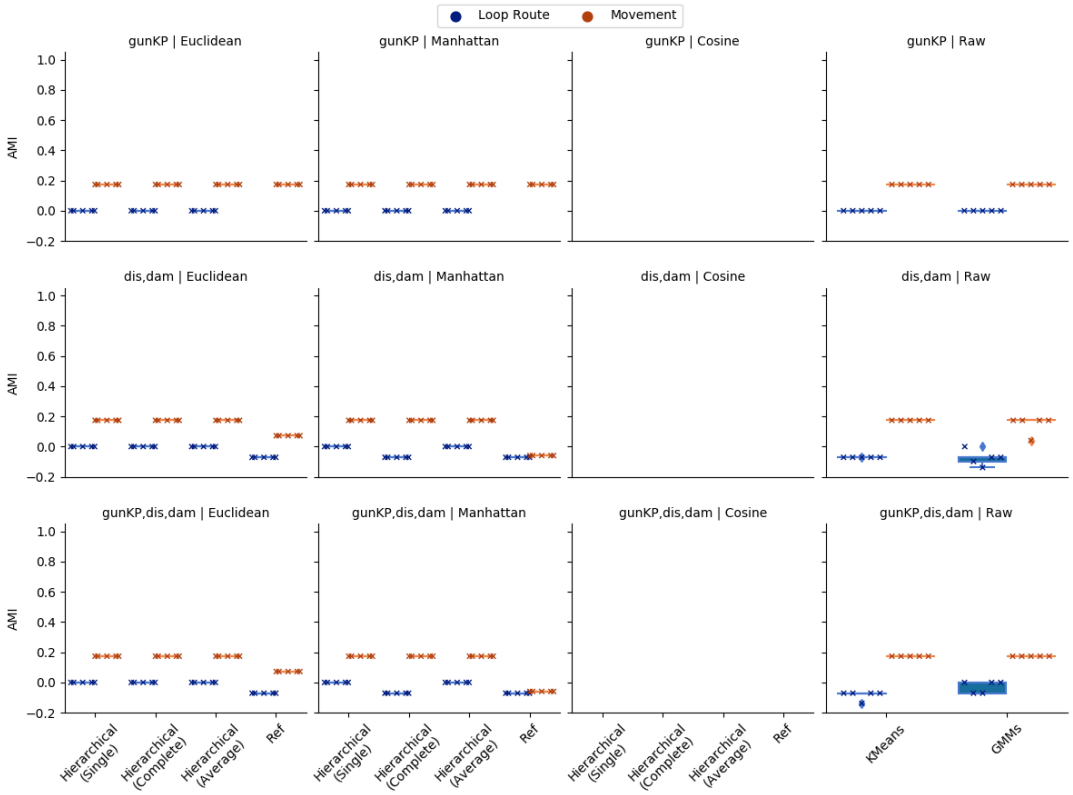


Figure 251: End Game Variable Clustering on ‘Loop Route’ and ‘Movement’ Datasets

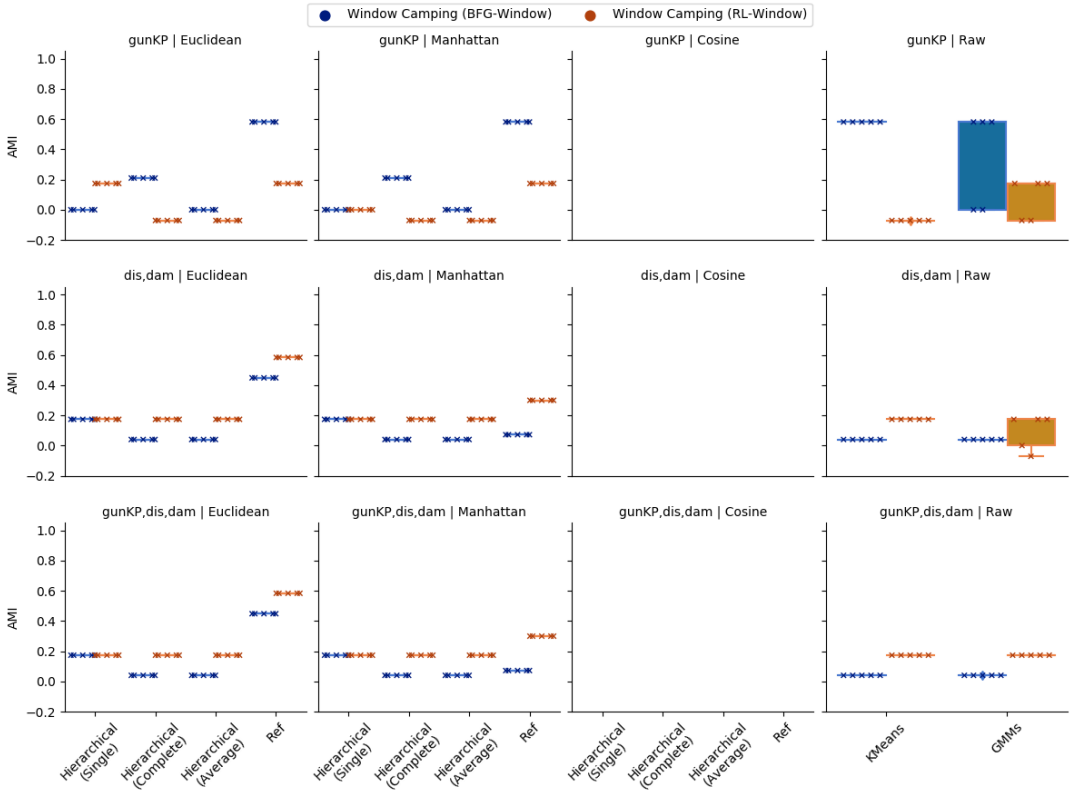


Figure 252: End Game Variable Clustering on ‘BFG9000 Window’ and ‘Rocket Launcher Window’ Datasets

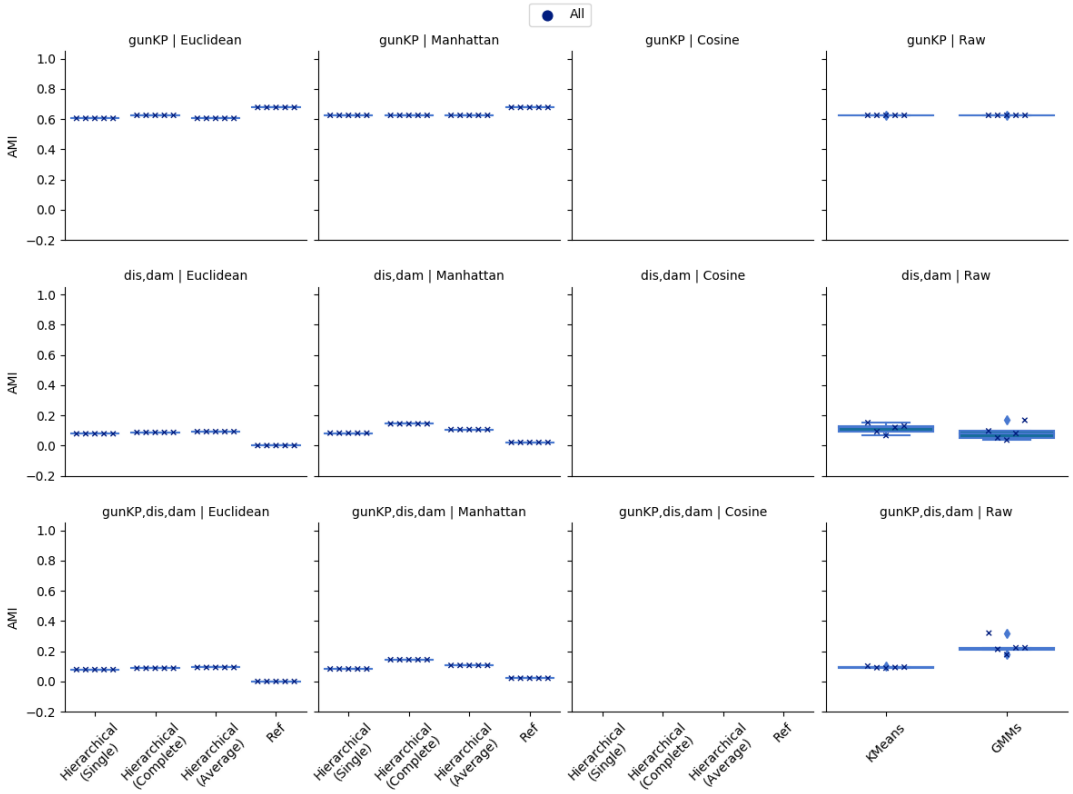


Figure 253: End Game Variable Clustering on Full Dataset (Ground Truth = Gun and Position)

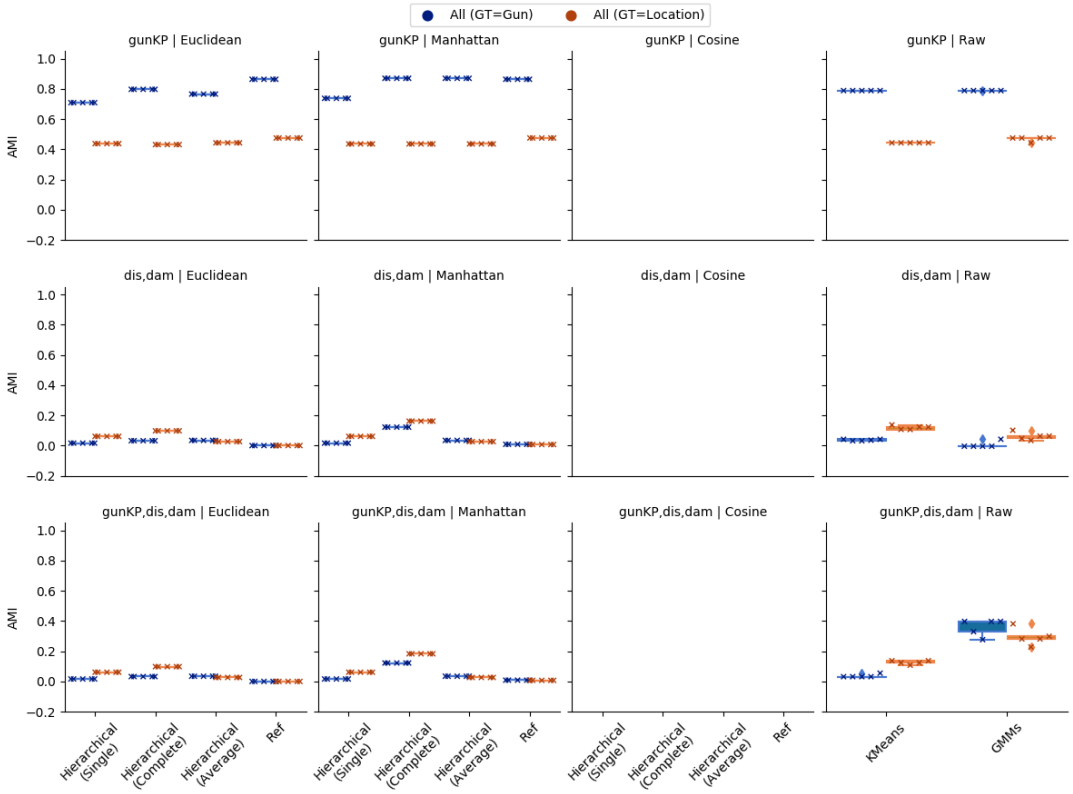


Figure 254: End Game Variable Clustering on Full Dataset (Ground Truth = Gun or Position)

B.2.3 Representation Clustering

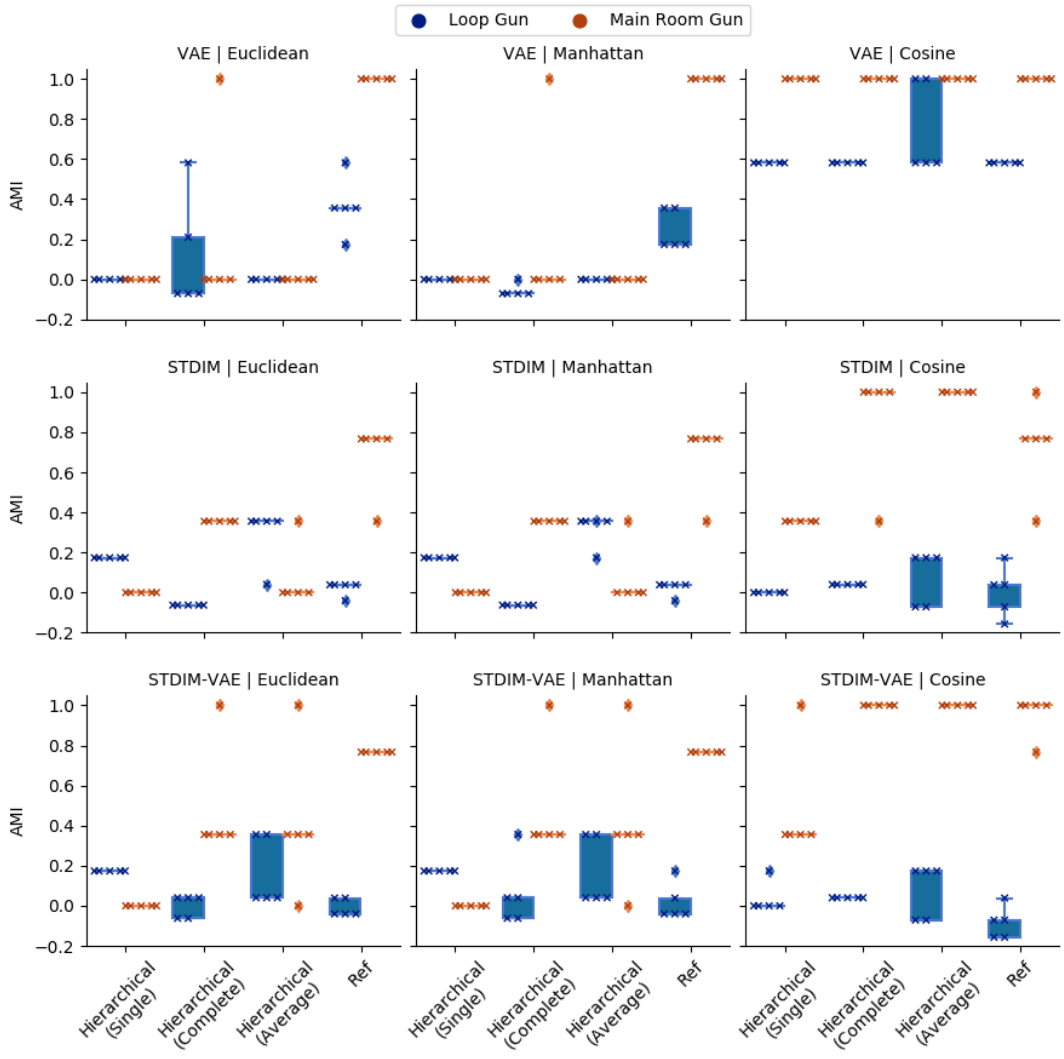


Figure 255: Representation Clustering on ‘Loop Gun’ and ‘Main Room Gun’ Datasets

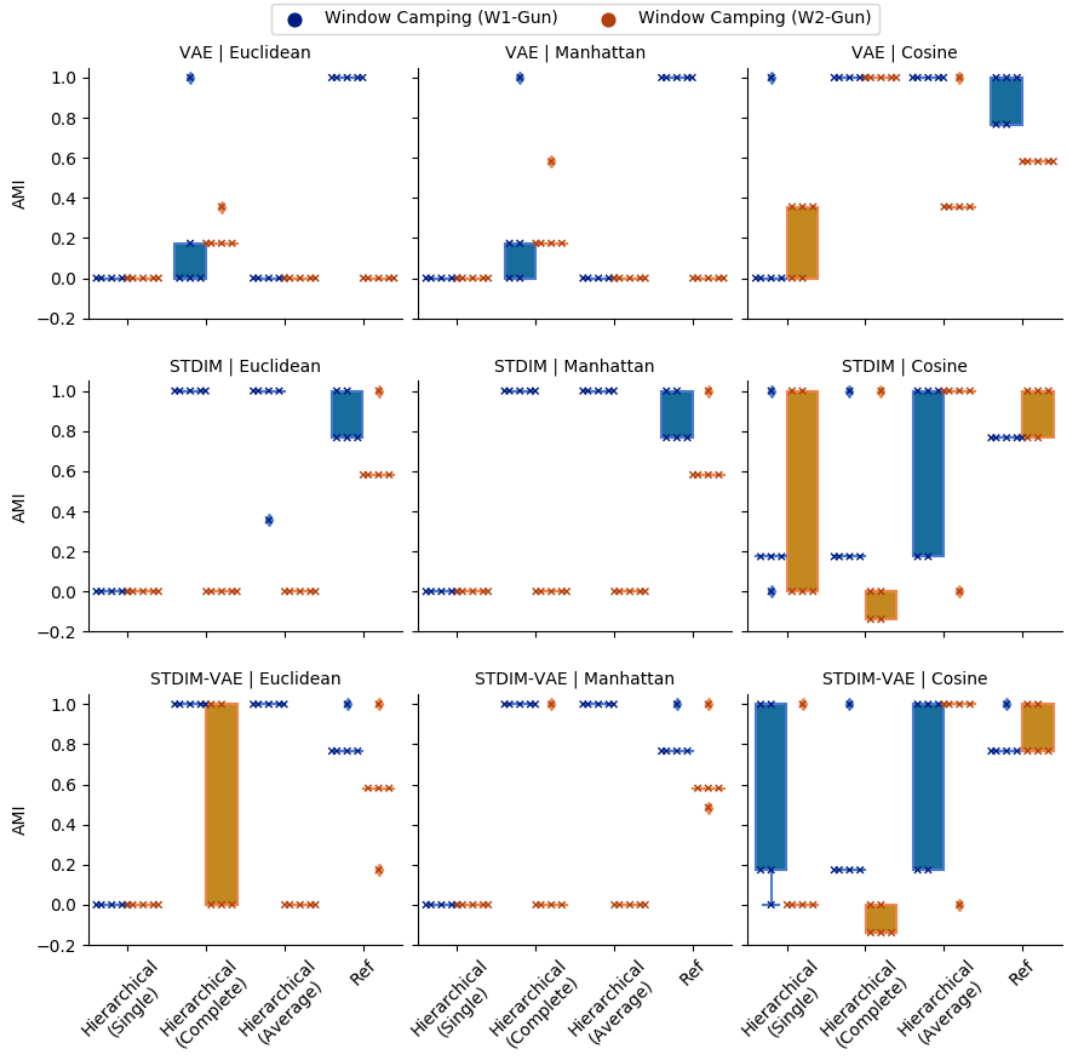


Figure 256: Representation Clustering on ‘Window 1 Gun’ and ‘Window 2 Gun’ Datasets

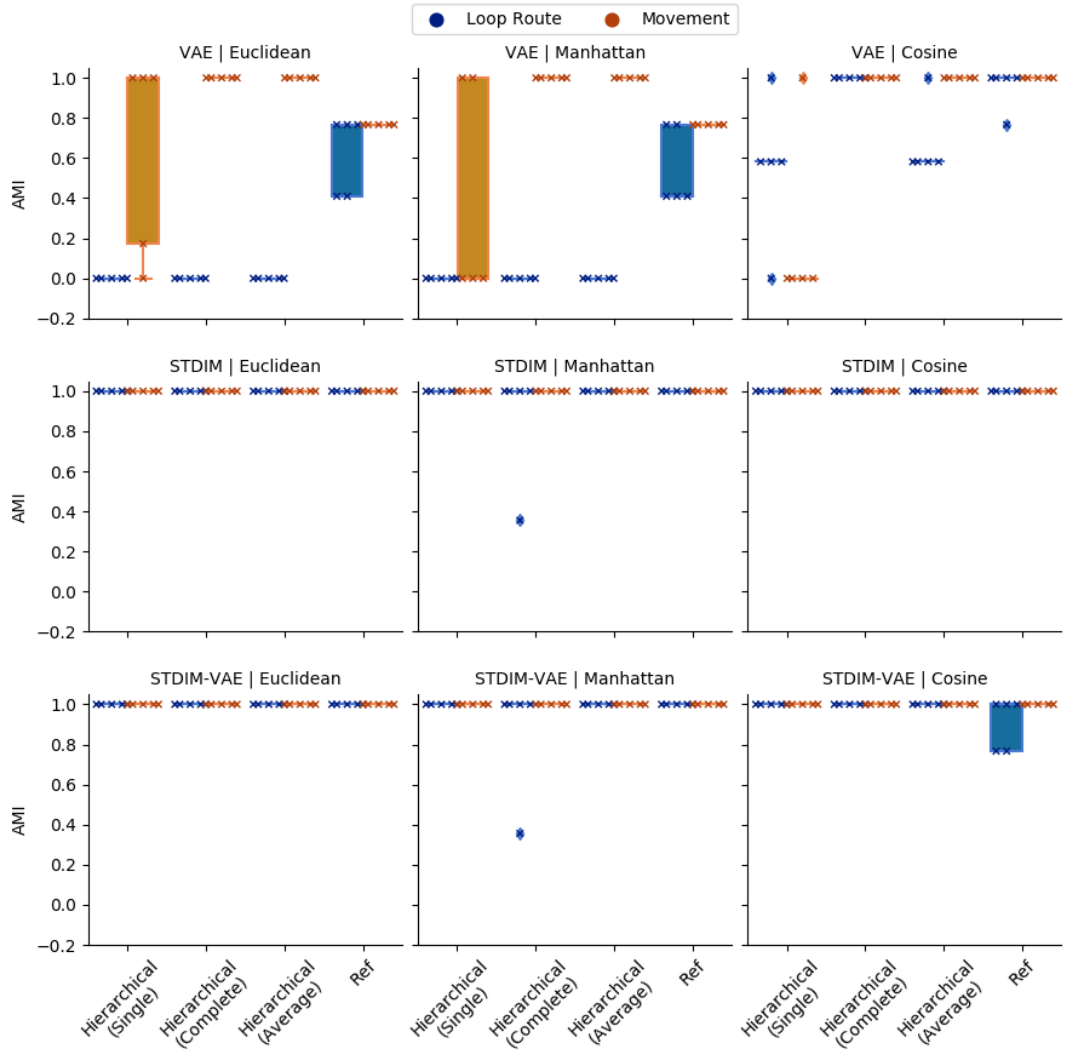


Figure 257: Representation Clustering on ‘Loop Route’ and ‘Movement’ Datasets

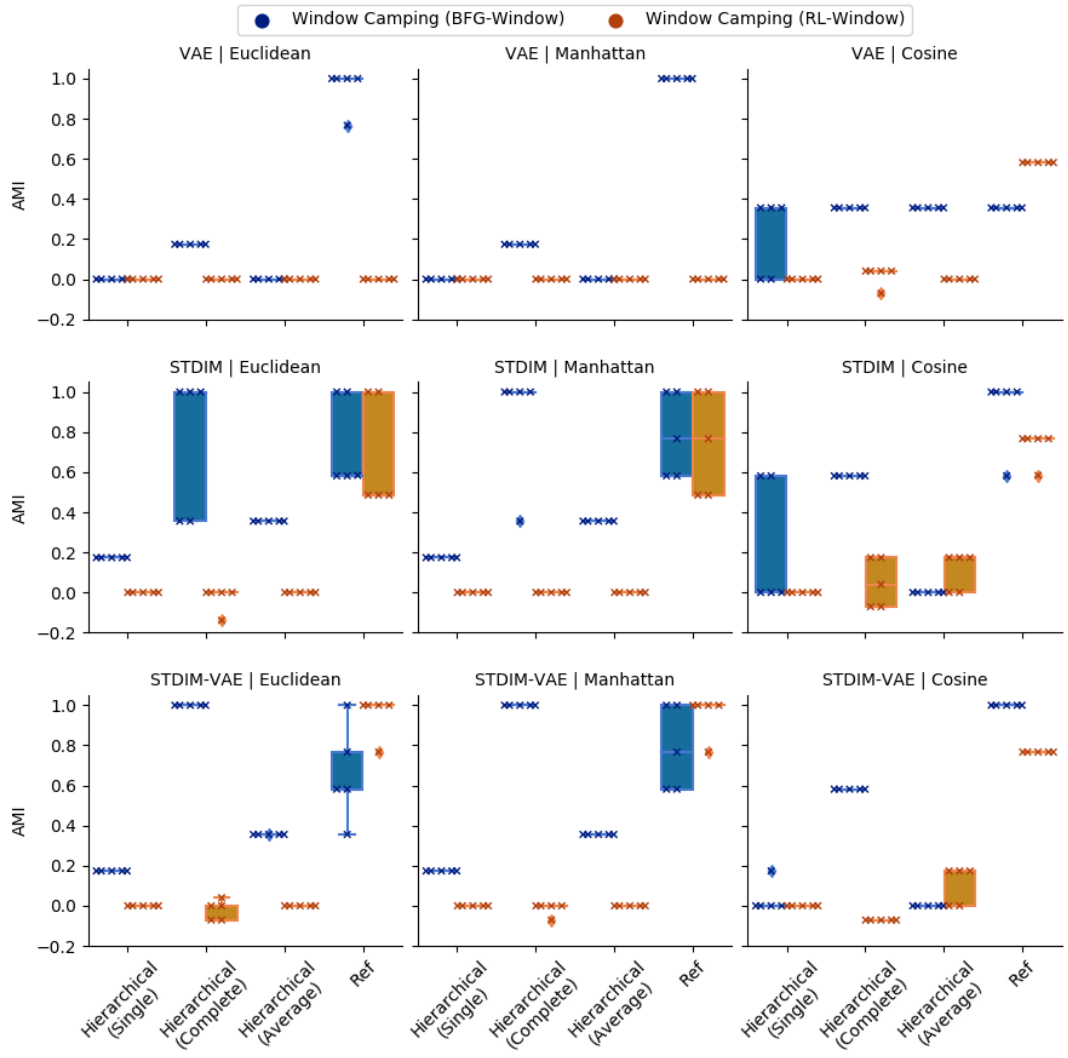


Figure 258: Representation Clustering on ‘BFG9000 Window’ and ‘Rocket Launcher’ Window Datasets

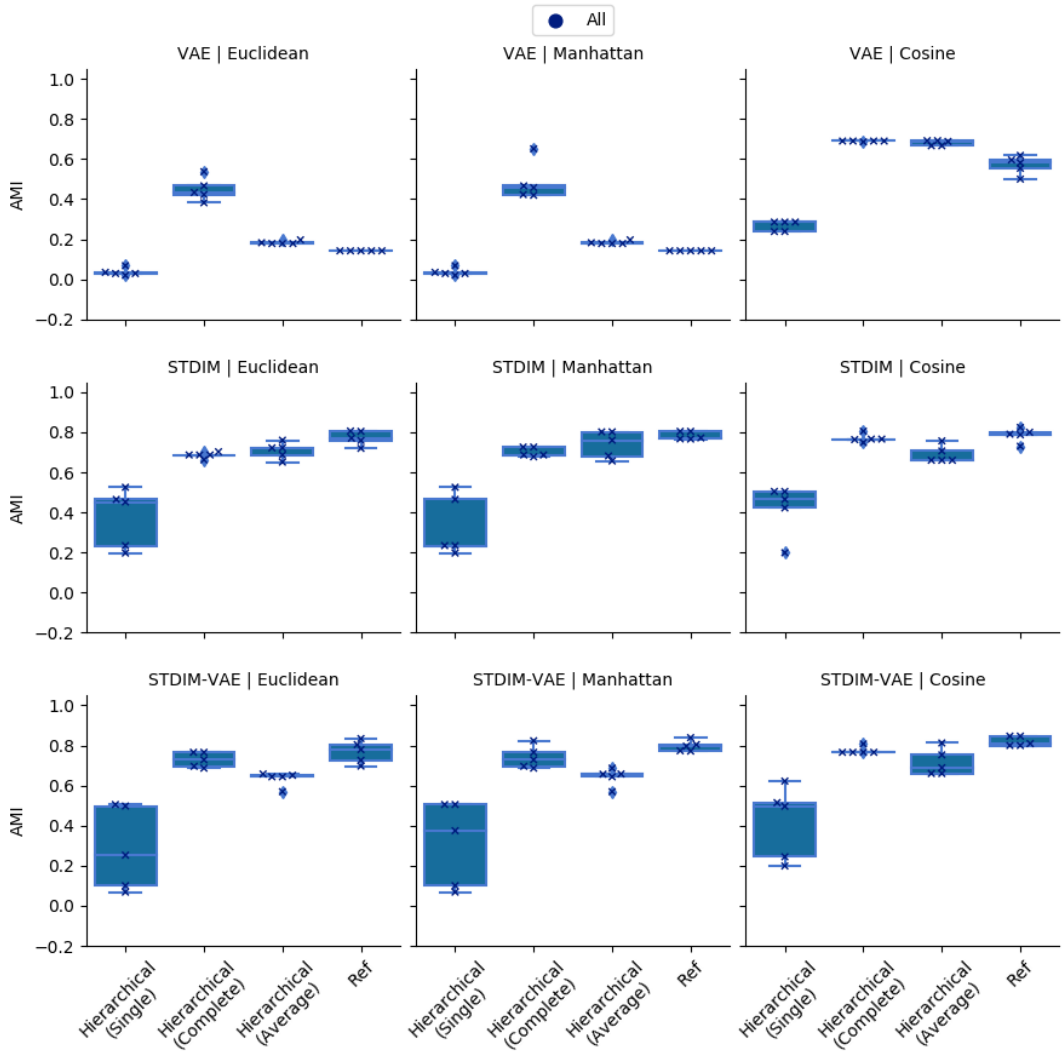


Figure 259: Representation Clustering on Full Dataset (Ground Truth = Gun and Position)

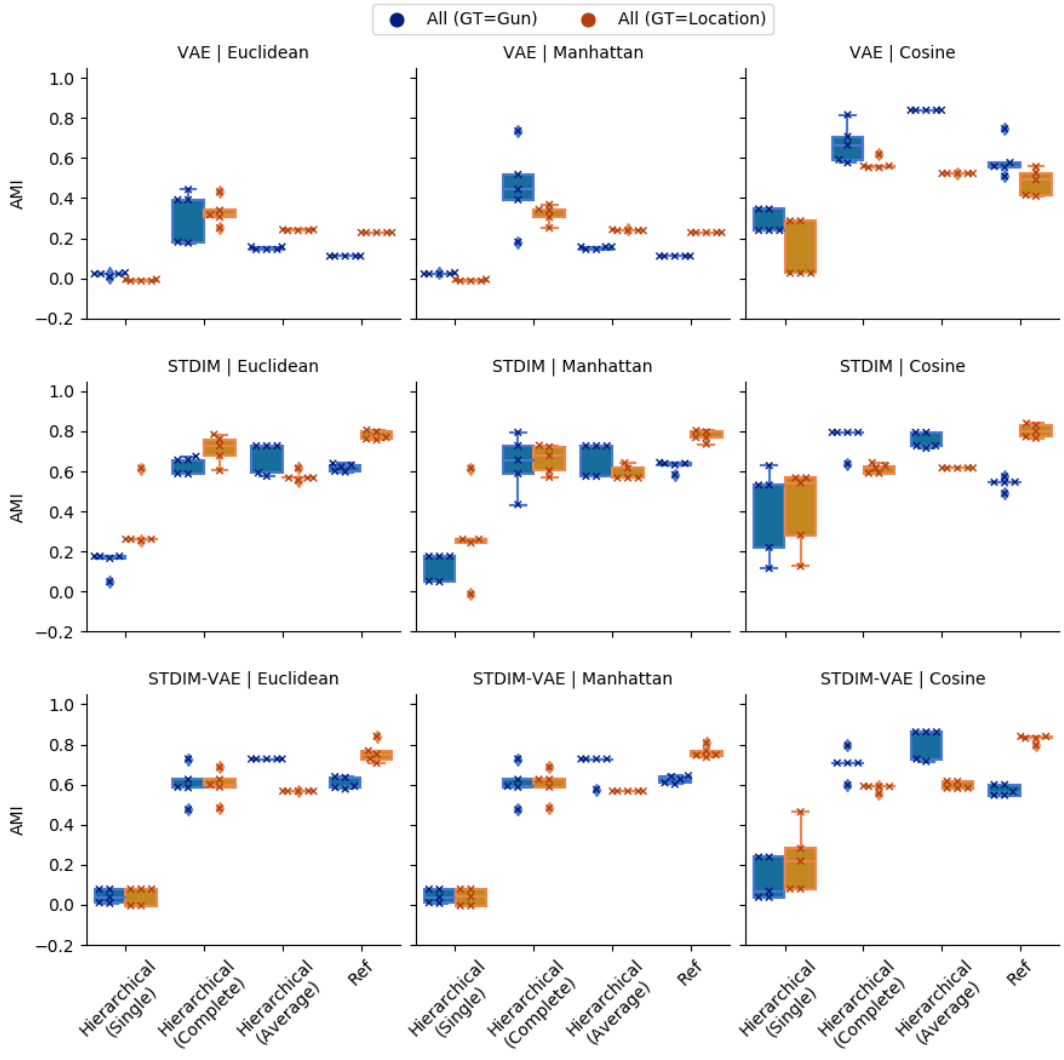


Figure 260: Representation Clustering on Full Dataset (Ground Truth = Gun or Position)

B.3 Hitman

B.3.1 Game Variable Series Clustering

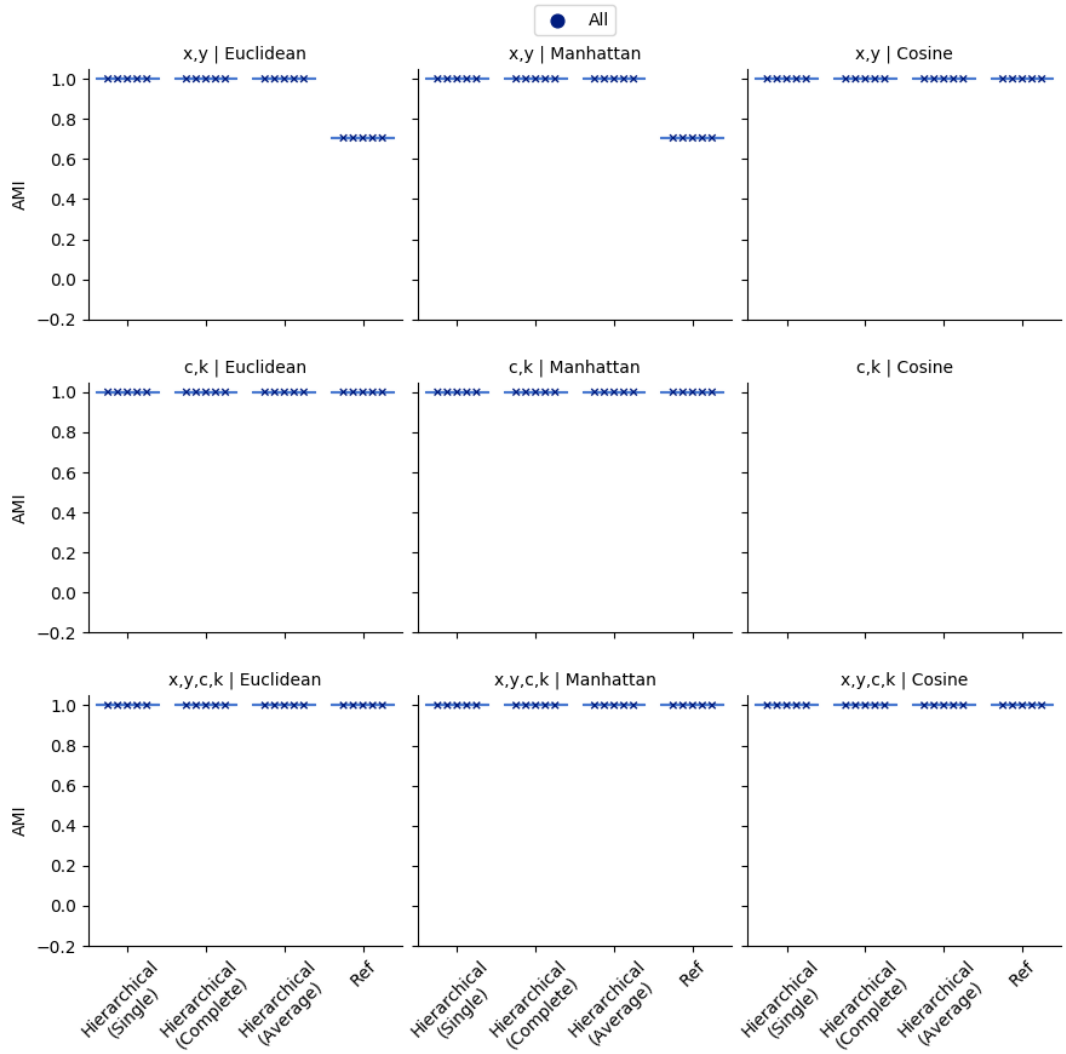


Figure 261: Series Game Variable Clustering on Full Dataset

B.3.2 Game Variable End Value Clustering

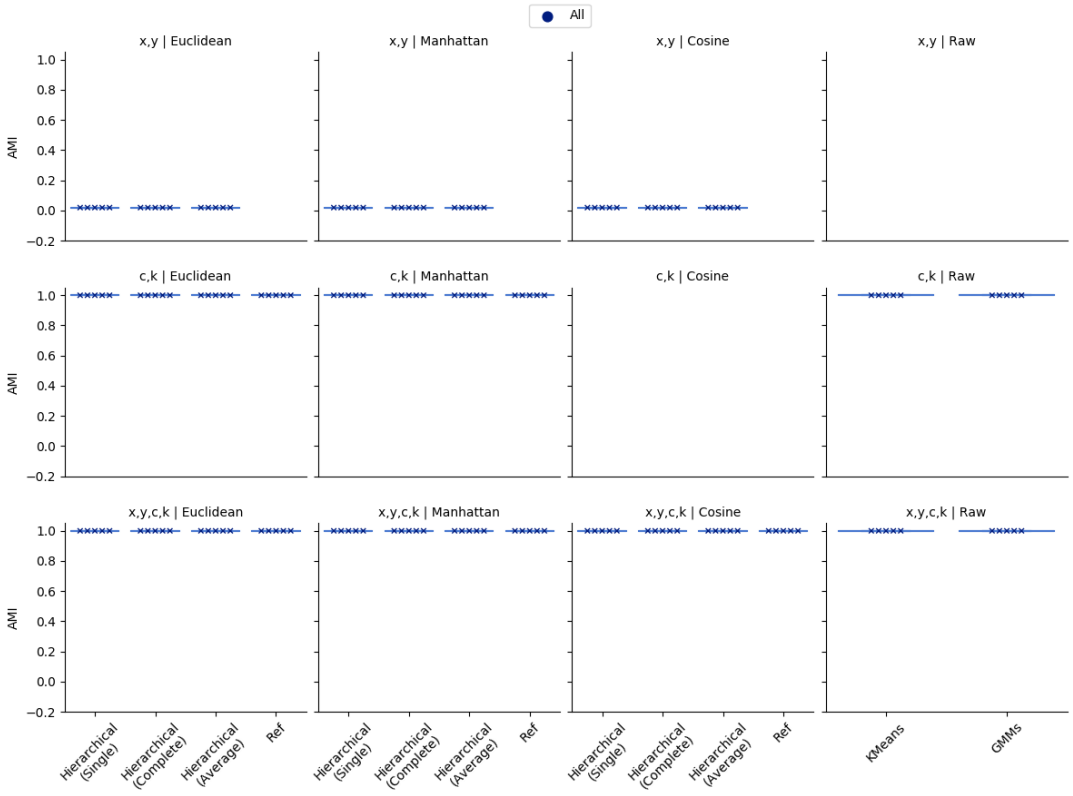


Figure 262: End Game Variable Clustering on Full Dataset

B.3.3 Representation Clustering

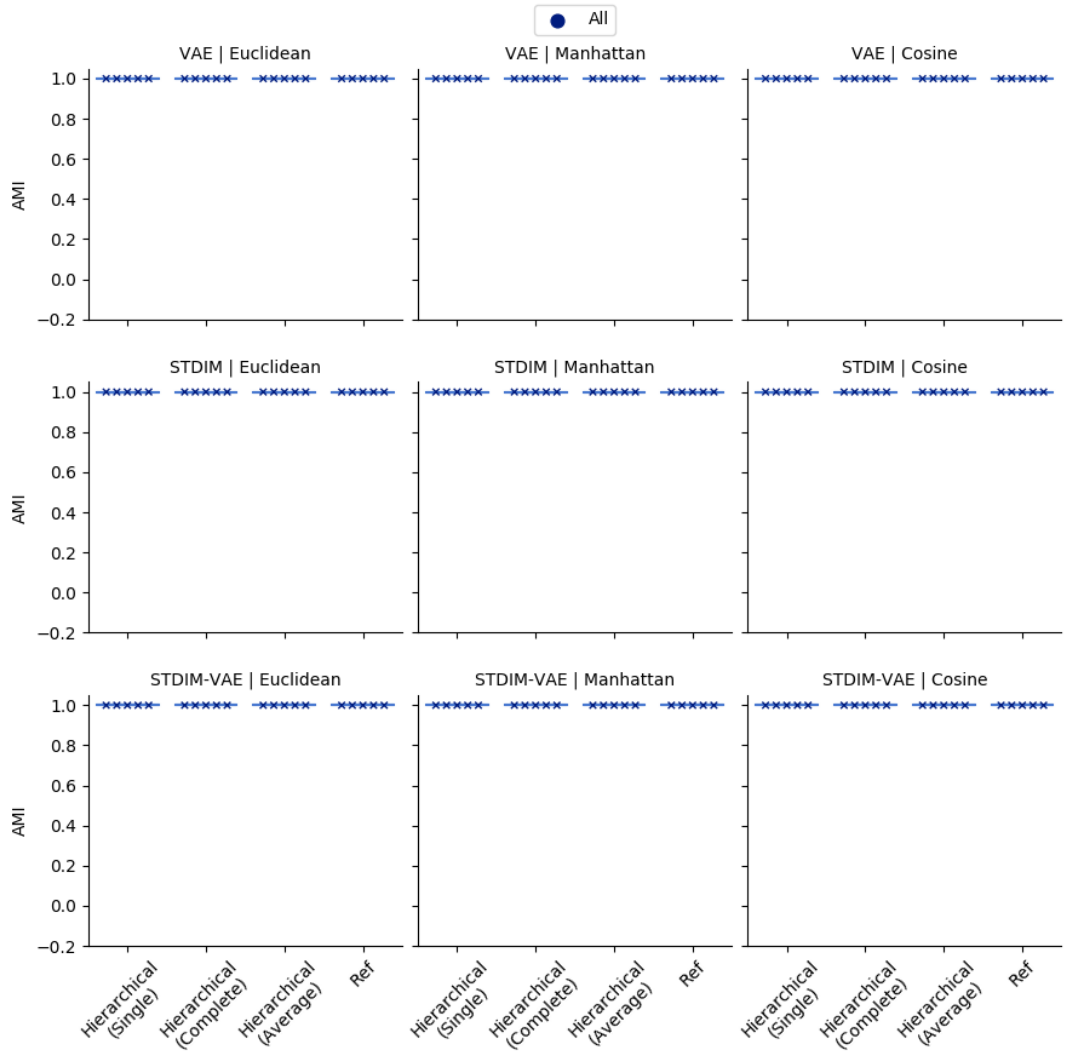


Figure 263: Representation Clustering on Full Dataset

Appendix C Imitation Parameter Sets

Below are all the tuned parameter sets for each method evaluated.

Table 28: Tuned Parameter Set for BC

Parameter	Value
Batch Size	32
Learning Rate	0.0001298

Table 29: Tuned Parameter Set for GAIL

Parameter	Value
Adversary Entropy Coefficient	0.007580
Adversary Gradient Coefficient	15
Adversary Learning Rate	0.0008885
Dampening Factor	0.02543
Entropy Coefficient	0.006351
Gamma	0.9694
CG Iterations	15
KL Threshold	0.01706
GAE Factor	0.9651
Learning Rate	0.0008076
Timesteps per Batch	512
VF Iterations	5

Table 30: Tuned Parameter Set for GAIL-T

Parameter	Value
Adversary Entropy Coefficient	0.001100
Adversary Gradient Coefficient	15
Adversary Learning Rate	0.0008045
Dampening Factor	0.01459
Entropy Coefficient	0.009801
Gamma	0.9184
CG Iterations	20
KL Threshold	0.02914
GAE Factor	0.9727
Learning Rate	0.0002322
Timesteps per Batch	1024
VF Iterations	5

Table 31: Tuned Parameter Set for GAIFO

Parameter	Value
Adversary Entropy Coefficient	0.004170
Adversary Gradient Coefficient	20
Adversary Learning Rate	0.0002984
Dampening Factor	0.006979
Entropy Coefficient	0.003230
Gamma	0.9643
CG Iterations	20
KL Threshold	0.002455
GAE Factor	0.9720
Learning Rate	0.0008781
Timesteps per Batch	2048
VF Iterations	5

Table 32: Tuned Parameter Set for GAIFO-S

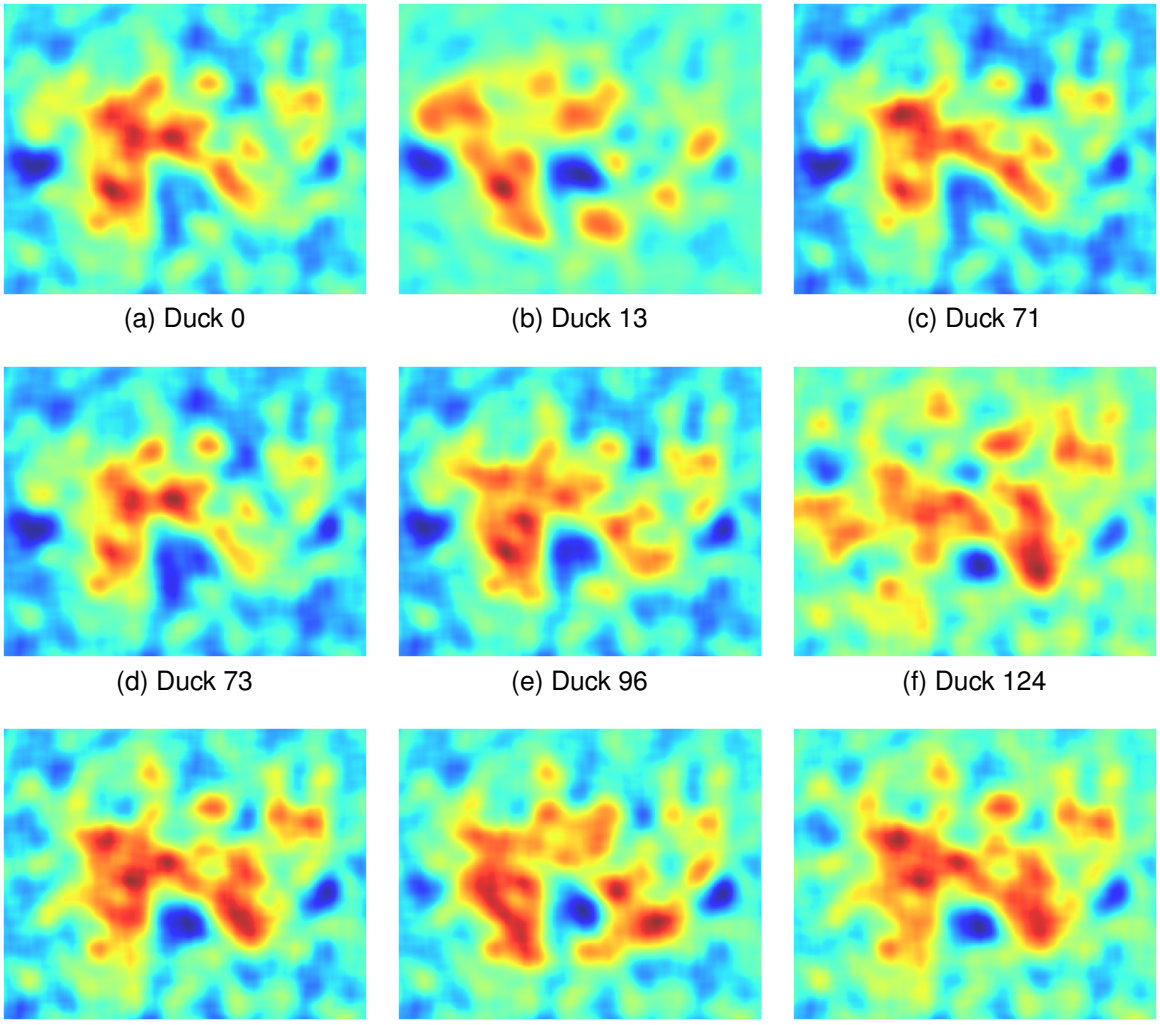
Parameter	Value
Adversary Entropy Coefficient	0.006498
Adversary Gradient Coefficient	15
Adversary Learning Rate	0.0005939
Dampening Factor	0.01289
Entropy Coefficient	0.006248
Gamma	0.9635
CG Iterations	20
KL Threshold	0.03103
GAE Factor	0.9529
Learning Rate	0.0007154
Timesteps per Batch	2048
VF Iterations	5

Table 33: Tuned Parameter Set for DTWI

Parameter	Value
Batch Accumulator	Mean
Double	True
Dueling	False
End Epsilon	0.1
Epsilon Check Step	5
Epsilon Max Step Ratio	0.9
Epsilon Min Steps	10
Gamma	0.9256
Head Freeze	2
Learning Rate	0.0004751
Prioritized	False
Replay Start Size	2000
Target Update	5000

Appendix D Common Duck Model ‘Fingerprints’

Below the full set of focus ‘fingerprints’ can be found for each of the common duck designs.

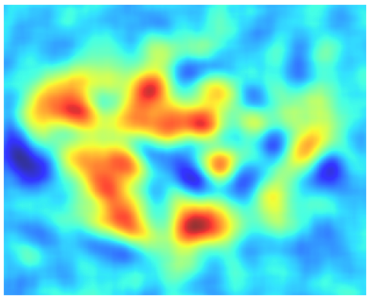


(a) Duck 0 (b) Duck 13 (c) Duck 71

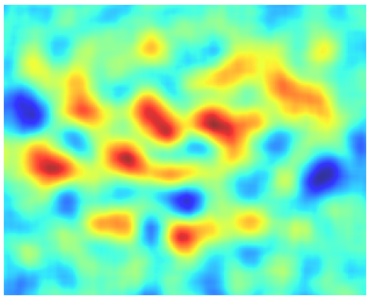
(d) Duck 73 (e) Duck 96 (f) Duck 124

(g) Duck 125 (h) Duck 133 (i) Duck 158

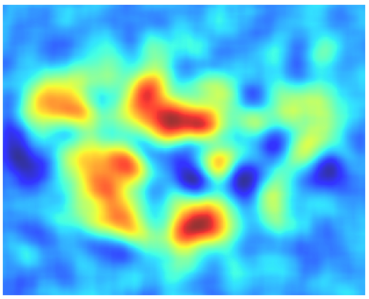
Figure 264: Focus ‘Fingerprints’ for ‘Standing Ducks’



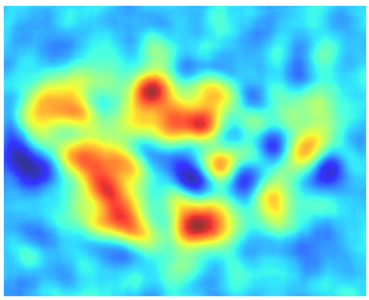
(a) Duck 3



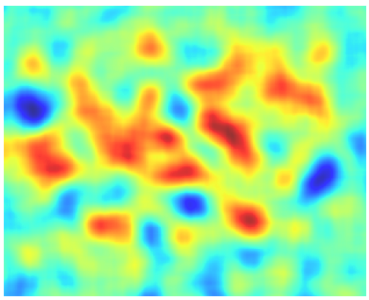
(b) Duck 25



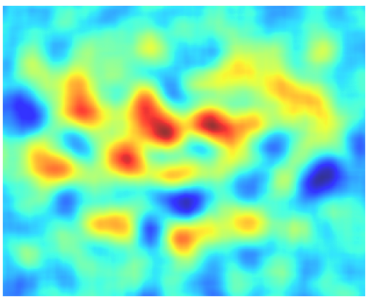
(c) Duck 33



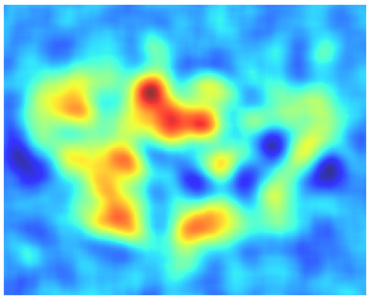
(d) Duck 34



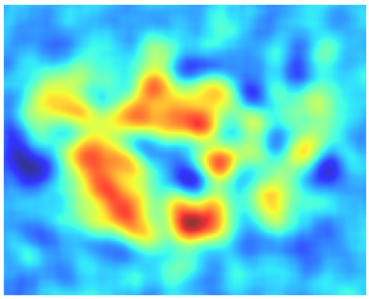
(e) Duck 44



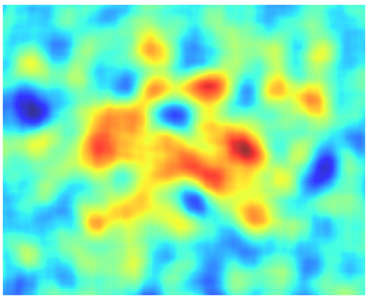
(f) Duck 53



(g) Duck 54



(h) Duck 101



(i) Duck 132

Figure 265: Focus ‘Fingerprints’ for ‘Sitting Ducks’