UTRECHT UNIVERSITY
Department of Information and Computing Science

Game and Media Technology master thesis

The added value of pupillometry features to predict the experienced difficulty

First examiner:
Sander Bakkes

Candidate:
Ciska Vriezenga

Second examiner:
Christof van Nimwegen

October 28, 2024

Abstract

In the context of video games, dynamic difficulty adjustment (DDA) offers a dynamic solution to match the in-game difficulty to the player's needs. In this study, we aimed to assess the added value of pupillometry features in the context of DDA and player experience modeling to predict experienced difficulty. A user study was conducted, during which participants played nine rounds of Pac-Man at different difficulties and for which gameplay-, game context- and pupillometry data were gathered together with participants' responses regarding game experience. Multiple random forest classifiers were trained on different feature subsets, with and without pupillometry features, to predict experienced difficulty. We found that the addition of pupillometry features did not lead to a performance improvement of the classifiers. This finding was supported by the results of our data analysis; only for the distributions of two of the four pupillometry features significant differences were found for the lowest and some of the other levels of self-reported experienced challenge. No convincing inverted u-curve was found to describe the relation between pupil size and experienced difficulty. A cautious inverted u-curve was found for pupil size with respect to the in-game difficulty. Based on our results we question whether pupillometry, as objective measure, is informative in the prediction of the subjective perception of difficulty. We propose future directions with respect to pupillometry features, the prediction of experienced difficulty and the prediction of the optimal load, together with a follow up study for which the constructed dataset can be reused.

Contents

1 Introduction **3**

2 Related work **6**
2.1 Dynamic Difficulty Adjustment (DDA) 6
2.2 Mental workload 15
2.3 Pupillometry 23
2.4 Pac-Man 32

3 Design of the Pac-Man testbed game **43**
3.1 Adaptations to the original game 44
3.2 Pac-Man challenge hierarchy 46
3.3 Game features 47
3.4 Mapping adaptive game components 50
3.5 Difficulty settings 57

4 Method **59**
4.1 Pilot study 59
4.2 Experimental design 60
4.3 Materials and apparatus 67
4.4 Participants 68
4.5 Finalizing feature set 69
4.6 Data analysis 75

5 Results **77**
5.1 Participant responses 77
5.2 Random forest classifiers 83

6 Discussion **96**
6.1 Pupillometry features in respect to predicting experience difficulty . . . 96
6.2 Results reviewed in perspective to the experience of difficulty 97
6.3 Limitations 99
6.4 Future work 101

7 Conclusion **104**

References **112**

1. Introduction

The level of challenge of a video game is consistently identified as the most important aspect of good game design [1]. Games that are too easy can be experienced as boring and games that are too hard can lead to frustration [2]. Video games in which the level of challenge matches the skill of the player have a higher entertainment [3]. In order to match the level of challenge to the player's skills, many video games offer the option to choose from a static set of difficulty levels, e.g. easy, normal or hard. However, these static difficulty levels are discussed to be insufficient, amongst others because they are too coarse and broad and thereby do not suit the actual level of the player [3]–[6].

Compared to static levels of difficulty, dynamic difficulty adjustment (DDA) offers a more dynamic solution to match the level of difficulty to the skill level of the player [7]. A DDA procedure estimates the level of a player and automatically adapts game elements to provide a suitable level of challenge [8], [9]. Since a suitable level of challenge results in a higher entertainment value [3], DDA has the potential to increase a game's enjoyment and is of interest for the entertainment industry. Furthermore, a suitable level of challenge can result in a state of flow wherein a player is completely immersed in a task [10], [11]. Flow is an enjoyable state with peak productivity [11] which can be interesting in the context of health and educational games. Thus, DDA can be relevant in the context of applied games.

Two main components of DDA systems can be viewed, namely the mechanism to assess the player and the in-game difficulty adjustment mechanism to provide a suitable difficulty [12]. Previous work provides different DDA approaches to assess the level of a player as well as different adjustment methods together with different types of adjusted content to adapt the in-game difficulty level [3], [13], [14]. For example, to estimate the level of a player a technique called player modeling can be applied [15]. A player model estimates the level of a player by considering multiple types of data, amongst others gameplay data, game context data and psychosensory measurements [16], [17]. Gameplay data refers to all elements derived from the direct interaction between the player and the game [17]. Game context data captures momentaneous state of the game during play, excluding gameplay data [17]. Examples of adaptations of game elements to offer a suitable difficulty, are the modification of the behavior of non-player characters (NPC's), the alteration of the game world and adaptations to the avatars supplies like ammo and health [3], [4], [18].

In this research we focus on the player assessment component of DDA, more precisely we focus on the use of pupil size measurements as input data in the context of DDA. Recent head mounted displays (HMDs) with eye tracking technology offer great potential to facilitate the use of eye-tracking metrics in the implementation of DDA [19]. Pupil dilation has been previously researched in the context of video games and game difficulty (e.g. [20], [21]). As an indicator of load, measurements of pupil dilation can inform a DDA system to help to identify an ideal level of challenge [22]. In the context of a math educational video game, pupillary response data was found to improve random forest classifier accuracy for the difficulty of levels [23]. Different pupillometry measures can be used as input features to train a model to predict the experienced game difficulty. Commonly used pupillometry measures are *peak pupil dilation* (maximal dilation in measurement interval), *peak pupil latency* (elapsed time between the stimulus and *peak pupil dilation*) and the *mean pupil dilation* [24], [25]. In the current research we aimed to provide further insights regarding the added value of the commonly used pupillometry features in the context of DDA. Partly, the current research can be viewed as a repetition of the work of Mitre-Hernandez et al. [23]. However, instead of predicting two levels of difficulty we focused on the prediction of experienced difficulty with multiclass classifiers. Additionally, we used the arcade game Pac-Man, which we believe offers a more dynamic gameplay compared to the math educational video game used in the study of Mitre-Hernandez et al. [23].

We aimed to answer the following main research question:

> "*To what extent does the addition of pupillometry features to an input feature set improve the accuracy of a random forest classifier in predicting experienced difficulty?*" (RQ1)

Furthermore, to gain a more detailed insight with respect to commonly used pupillometry features in the context of DDA, we aimed to answer the following question for each of the used pupillometry features:

> "*To what extent does this pupillometry feature statistically correlate to the self-reported experienced game difficulty?*" (RQ2)

An experiment was performed in which participants played multiple rounds of Pac-Man with different difficulties. During this experiment gameplay related features, pupillometry measurements and participant responses regarding the experienced difficulty were gathered to form a dataset. Sequentially, statistical analysis was applied to answer RQ2 for each of the constructed pupillometry features. Furthermore, multiple feature subsets were created from the main dataset and used to train multiple random forest models to predict the experienced difficulty. The performance of classifiers trained on feature sets with and without pupillometry features were compared and feature importance analysis was performed.

The main goal of this research was to map the added value of pupillometry in the context of DDA and player experience modeling. Based on our results, reviewed in the perspective of the experience of difficulty, we provided future directions and proposed a follow up study for which the constructed dataset can be reused.

2. Related work

2.1 Dynamic Difficulty Adjustment (DDA)

In this section, after a short introduction to DDA, we discuss multiple aspects related to DDA. Firstly, to gain a better understanding of difficulty in games, we share some background on difficulty and challenges from a game design perspective. Secondly, we provide an overview of different types of game components that can be adapted to alter the difficulty of a game (*adaptive game components*). Additionally, we view different types of DDA classifications and types of DDA approaches, together with examples of previous work. Furthermore, we provide an overview of different types of input for DDA systems in order to assess the player. We end this section with a short summary of the insights these subsections offer in the context of the current work.

2.1.1 DDA Introduction

Providing a good level of challenge for the player can be viewed as a key feature of successful video games and as the most important aspect of good game design [1], [6]. According to the flow channel model as proposed by Csikszentmihalyi [11], the offered level of challenge an activity offers directly relates to the performance perception, see the diagram in Figure 2.1. Csikszentmihalyi [11] explains the flow channel using a game of tennis to illustrate. In short, for a person new to tennis, simply hitting the ball over the net provides a suitable level of challenge. This challenge meets the player's expertise and can lead to the state of flow. The state of flow refers to an enjoyable state with peak productivity wherein a person is completely immersed in a task or activity [11]. However, when the player's skills improve the player can grow bored just batting the ball over the net. On the other hand, when being new to tennis and competing with a more practiced opponent can provide a too high challenge that might result in the experience of anxiety. Noteworthy, according to Csikszentmihalyi [11] to get in the flow channel not only the actual skills are of influence, but also the skills we think we have. Similarly, "it is not only the 'real' challenges presented by the situation that count, but those that the person is aware of" [11].

Koster [2] incorporated the concept of flow in the context of video games. Koster [2] stated that to keep the player in a state of flow when the player gradually improves her skill level, the game difficulty level should be gradually heightened as well. Chen [26] states that an interactive experience designed for a broader audience "must offer

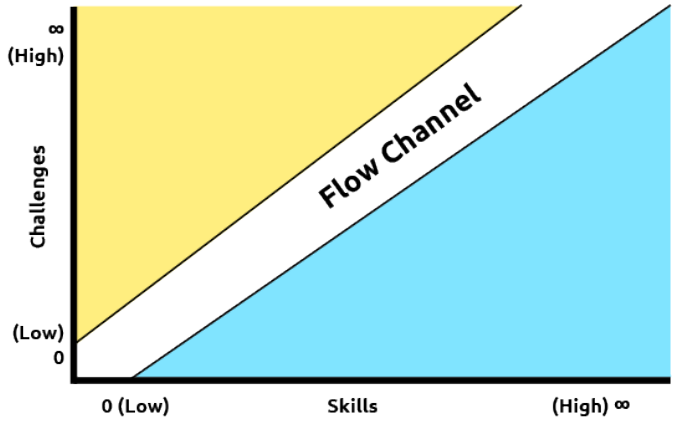

Figure 2.1: The flow channel model originally proposed by Csikszentmihalyi [11]. A level of challenge that meets the player's skill can lead to the state of flow, a too low level of challenge can lead to the experience of boredom and a too high level of challenge can lead to the experience of frustration [11].

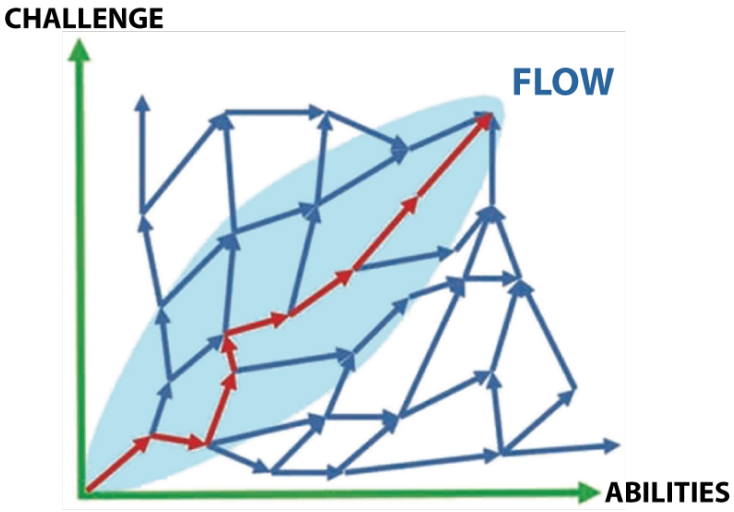

Figure 2.2: "Designers adapt players' flow experience through the choices they deliberately build into the experience." [26].

many choices, adapting to different users' personal flow Zones" (flow channels), see Figure 2.2. To avoid too many choices, game designers can "embed the player's choices into the core activities of the interactive experience"[26].

In many video games a player can choose between different static levels of difficulty (e.g. easy, normal and hard) to select an appropriate level of challenge. However in previous work it is argued these static levels fail to meet the actual level of the player, amongst others by being too coarse and broad [3]–[6]. Dynamic difficulty adjustment (DDA) offers a dynamic solution to provide a level of difficulty that matches the level of a player. DDA is a technique in video games to automatically and in real-time adjust scenarios, parameters and behaviors to follow the player's level of skill and thereby preventing boredom or frustration [14]. DDA can thus aim to keep the player in the flow channel.

The application of DDA in a game is not straightforward, it entails more than providing more health when the player performance drops [14]. Successful DDA systems should maintain the internal balance and feedback mechanisms [18]. Andrade et al. [6] proposed three basic requirements for DDA. Firstly DDA should be able to quickly identify and adapt to a player's initial level [6]. Secondly, DDA should fastly and closely track and adapt to the player's improvement or falling level [6]. Thirdly, the adaptation process should be unperceivable for the player and the game must remain believable. When the adaptation process of DDA is perceivable for the player, players are known to take advantage of this adaptation.

An example of an DDA implementation that does not meet this third requirement is the so-called rubber-banding mechanism frequently applied in racing games. This mechanism refers to the application of negative feedback of the player's position relative to the NPC opponents to prevent the gap between the player and computer opponents to grow too large, as if the player is attached to the other cars by a rubber band ([27]). Rubber-banding can be implemented by simply slowing down opponents that are ahead of the player or speeding up opponents that are behind the player [28]. Additionally to the speed increment and decrement of opponents, in MarioKart the awarded random power a trailing player receives after picking up a power-up has a higher chance to be more powerful then when the player is in the lead ([27]). Players can take advantage of the Rubber banding adaptation by lacking behind purposefully and speeding up just before the finish in the final lap.

2.1.2 Game difficulty and challenges

Different perspectives on difficulty are presented in previous work [14]. In this subsection we share Adams' [12] perspective on difficulty and challenges.

Adams [12] discusses three types of difficulty, namely *absolute difficulty*, *relative difficulty* and *perceived difficulty*. *Absolute difficulty* can be determined by *intrinsic skill* and *stress* [12]. *Intrinsic skill* refers to the level of skill needed to overcome a game challenge leaving out any element of time pressure [12]. When an unlimited amount of time is available, the skill level required of a player can be reviewed with respect to the challenge and the type of tasks within that challenge [12]. For example, the skill level needed to hit a target is amongst others dependent on the distance to the target and the aiming skill of the shooter.

When a challenge includes time pressure, the factor *stress* is of influence to the *absolute difficulty* [12]. Respectively, a shorter or longer time limit results in a more or less *stress*ful situation respectively [12]. Adams [12] provides an example of the game Tetris. The challenges in Tetris performed without a time limit would not require a high level of skill, thus Tetris requires a low level of *intrinsic skill*. It is the time pressure

which makes Tetris *stress*ful and which mainly contributes to Tetris's *absolute difficulty*.

To review the *absolute difficulty* of a challenge, the required amounts of *intrinsic skill* and imposed *stress* can be compared to a trivial challenge of the same type [12]. For example, when designing an enemy, it can be compared to a trivial enemy that stands still, can be killed with one hit and does not harm the avatar [12].

Relative difficulty refers to "the difficulty of a challenge relative to the player's power to meet that challenge". *Relative difficulty* is dependent on the *absolute difficulty* and the *power provided* [12]. *Power provided* refers to the power the game provides to the player [12]. For example, the challenge of defeating an enemy will be easier when the game provides the player a sword with high hit points compared to a sword with low hit points.

The *perceived difficulty* is defined as "the *relative difficulty* minus the player's experience at meeting such challenges [(in-game experience)]" [12]. In other words, the *perceived difficulty* of a challenge is the difficulty that a player senses, this difficulty type considers the player's experience. Adams [12] combines the different factors into one equation, see Equation (2.1).

$$\text{perceived difficulty} = \text{absolute difficulty} - (\text{power provided} + \text{in-game experience}) \tag{2.1}$$

Furthermore, with respect to challenges, in games a player faces several challenges at a time and these challenges can be organized in a hierarchy of challenges (*challenge hierarchy*) [12]. The *topmost-level* in the *challenge hierarchy* includes the game's victory condition and victory conditions for separate levels. The *lowest-level* consists of atomic challenges, these are the smallest chunks of challenge that are indivisible and which a player needs to deal with separately [12]. Examples of *lowest-level* challenges are fighting an enemy that causes an instant threat and opening a locked door [12]. Both the challenges of the *topmost-* and *lowest-level* are usually shared with the player explicitly [12]. The challenges of the *intermediate-level* are usually not communicated to the player [12]. By leaving these *intermediate-level* challenges implicit, the player is left to figure out what she is supposed to do, also referred to as 'the fun part' of a game [12]. *Intermediate-level* challenges can require players to develop a certain strategy [12]. Players usually face multiple challenges from different levels in the *challenge hierarchy* at the same time [12]. In short, Adams' [12] hierarchy of challenges allows one to review the challenges of a game at different levels; at a detail level (*lowest-level* challenges), in a more in an overarching way (*topmost-level* challenges) and on a strategic level (*intermediate-level* challenges).

Adams' [12] perspective on difficulty and challenges was utilized in the current research, to review aspects of difficulty and challenges in Pac-Man, in the context of player assessment and in-game difficulty adjustment. We mapped a *challenge hierarchy* for Pac-Man and reviewed each challenge to identify *game features* that can serve as input features for our random forest models. We reviewed the *intrinsic skill*, *stress* and *power provided* in Pac-Man to identify game components that can be altered in order to adapt the difficulty (*adaptive game components*).

2.1.3 Adaptive Game Components

Different views on *adaptive game components* are shared in review papers on DDA [14], [29]–[31]. Zohaïb's [14] identifies *parameters*, *behavior* and *scenario* as *adaptive game components*, but these three categories are not reviewed to a further extent. In Sepulveda et al.'s [31] overview of recent work in the context of DDA, *adaptive game components* are referred to as factors and are categorized as *attributes*, *behaviors* and *events*. Here, *attributes* refer to values of game objects and mechanics, like traits of the character, number of enemies, time limits and the pace of a game [31]. With *behaviors* Sepulveda et al. [31] refer to the behavior of NPCs and behavior of game objects, like the pattern of watch towers in a stealth game. The category *events* refer to "predefined occurrences that arise under certain circumstances"[31]. In Bontchev's [30] literature review of "adaptation in affective video games" three options for *adaptive game components* are shared, namely 'adjusting level content' similar to Sepulveda et al.s [31] *attributes*, *modification of artificial intelligence* similar to Sepulveda et al.s [31] *behavior* category and *automatic level generation*. An example of automatic level generation is the adaptation fo 2D-platform levels, where amongst others the placement, size and spatial diversity of gaps are adaptable [32]–[34]. In a survey of "adaptive challenges in games and simulations" [29], five categories are presented; *game worlds and its objects*, *gameplay mechanics*, *NPC and AI*, *game narrative* and *game scenarios and quests*. This categorization differs from the others by the explicitly distinguished last two categories, where *game narrative* adaptations can be viewed as the alteration of a sequence of events serving the story telling and *game scenarios and quests* adaptation as an alteration of the flow of events and actions [29].

A more holistic view on *adaptive game components* is provided by Hunicke [18], by viewing adaptations as regulation of the game's underlying economy. Hunicke [18] differentiates between *supply* and *demand adaptations*. *Supply adaptations* affect the player's inventory by placing more or less items in the field (e.g. ammunition and health packs) or by modifying properties of the player or items (e.g. player strength, a weapon's accuracy). *Demand adaptations* affect the impact of enemies, like manipulation of the hit points and accuracy of enemies.

In the context of the current research, we propose a categorization of *adaptive game components* of four categories, namely *attributes*, *behavior*, *game level/-world layout* and *events*. Here, we view *attributes* as values of game objects including the player's avatar. With *behavior* we refer to NPC behavior. The category *game level/world layout* concerns the topology and obstacle placement in a level or game world. With *events* we refer to adaptations that alter the order of events. We excluded categories regarding the game scenario and narrative, like the categories *scenario* [14], *game scenarios and quests* and *game narrative* [29], since these are not relevant in the testbed Pac-Man. Also the category *gameplay mechanics* [29] is left out, since alterations that fall in this category would be perceivable for the player and thereby would not meet the third DDA requirement.

After mapping challenges of Pac-Man in a *challenge hierarchy* (see 2.1.2), in order to identify *adaptive game components* of Pac-Man we reviewed these mapped challenges through the lens of each of the four *adaptive game components* categories (*attributes*, *behavior*, *game level/-world layout*; and *events*). This systematic approach enforced us to consider all types of *adaptive game components* that can be of influence to a challenge and in turn the in-game difficulty. Therefore, we expected this approach to enable us to review a broad set of *adaptive game components* in order to find *adaptive game components* of interest in the current research.

2.1.4 DDA approaches

Different classifications of various DDA approaches are presented in review papers about DDA [14], [29]–[31], [35], [36]. For example, Lopes and Bidarra [29] identify two types of DDA, namely online adaptivity, referring to the ability of a game to adjust to its players in real time and offline adaptivity, referring to adaptations prior to initiating the gameplay. Zohaïb [14] provides eight categories to classify DDA approaches, amongst others *probabilistic methods*, *dynamic scripting*, *reinforcement learning* and *affective modeling using EEG*[1]. In the rest of this section we discuss three straightforward implementation methods presented by Sepulveda et al. [31] and Paraschos and Koulouritios's [35]classification, accompanied by examples of previous work to provide more insight in the large range of different DDA approaches.

Sepulveda et al. [31] discuss three implementation methods that are according to them more straightforward methods to model the difficulty faced by the player, namely, *metrics*, *probabilistic methods* and *dynamic scripting*. The *metrics* method consists of the application of a multiplier to variables that change the difficulty by adapting game content. Additionally, weights can be added to balance the relationship between the altered variables affected by the multiplier and the variables used to evaluate the player per-

[1]For a full overview of all categories we refer the reader to Zohaïb's [14] review paper.

formance. *Probabilistic methods* calculate an expected future value to predict states and events, enabling the DDA system to prevent for example a too low health before the predicted time of death [14], [31], [37]. Within the *dynamic scripting* approach scripts for NPCs are built from manually designed rulebases. The rules in these rulebases are assigned a weight used as the probability the rule gets selected for a NPC script. The weights are updated depending on success or failure of rules in NPC scripts during gameplay [31], [38], [39].

Paraschos and Koulouriotis [35] proposed a classification for approaches that combine DDA and procedural content generation (PCG), representing four different main approaches with respect to the assessment of a player's performance, emotions or preferences and the way the DDA system is constructed. The four categories are *Player performance*, *Player affective state*, *Agents* and *Experience modeling* [35]. The first category *player performance* covers the approaches that assess player performance to balance difficulty to the player's skill level [35]. Approaches within this category use performance and behavior metrics to evaluate player performance [35]. Some examples of these metrics are "kill ratio", "attained game score" and "number of deaths". An example from this category is the work of Li et al. [40] in which DDA was applied to a mobile parkour game where the player needs to pass through obstacles in the environment and collect as many fortunes as possible along the way. In this work, the adaptation mechanism was designed based on the number of obstacles passed (metric) [40].

The second category *Player affective state* covers DDA approaches that measure the affective state of a player during play. Approaches presented previous work that fall within this category often used biofeedback devices and questionnaires for the affective assessment. Other types of measurements have been previously presented, amongst others player facial expressions. An example is the work of Blom et al. [41] where facial expression recognition was utilized to guide an online game personalization process in the video game Infinite Mario Bros. In this research the emotions "neutrality", "happiness" and "anger" were tracked with facial expression recognition [41]. A Gradient Ascent Optimisation (GAO) technique was employed to optimize the challenge levels to yield happiness and minimize neutrality and anger of the game based on the [41].

The third category *agents* concern adaptation approaches that utilize decision-making agents for tailoring the game content [35]. DDA approaches within this category employ a learning algorithm with player behavior and game state information as input, enabling agents to learn from the past player actions [35]. The previously discussed dynamic scripting approach introduced by Spronck et al. [38], [39] can be placed within this category. Another example of an approach that falls within this category is the work of Tan et al. [42], in which ideas from reinforcement learning and evolutionary

computation were applied in the context of a racing game. In this approach a chromosome containing probabilities for behavioral rule selection was updated after each round and used to select a subset of behavior components for the NPC opponent [42].

The fourth category *experience modeling* describes approaches that utilize models trained with machine learning algorithms to estimate player characteristics, such as skills [35]. Based on the estimations of player characteristics the game environment is adapted to the player. In turn, feedback from the game environment is used as input of the model's evaluation function. Paraschos and Koulouriotis [35] identify three sub categories within the fourth category *experience modeling*, namely *behavior and preference models*, *experience models* and *alternative and combined models*.

An example of the subcategory *behavior and preference models* is the work of Andrade et al. [6], in which reinforcement learning was applied to dynamically model the players' reactions to different difficulty conditions. This research applied DDA to a rehabilitation game, where the player used a robot attached to the player's wrist as a control device to control a squirrel collecting nuts falling down from trees. The reinforcement learning agent learned at runtime how the player responded to changes in the game difficulty level [6]. The environment of the agent was defined by a simple 2D matrix defined by two difficulty parameters, namely "nut drop rate" and "distance to nut" [6]. The agent actions consisted of navigation through this 2D environment, moving towards the direction of the most difficult game level, and thereby the agent actions changed the game difficulty). After a predefined period (2 released nuts) a performance value was taken as an estimate of the player behavior in response to the new game difficulty condition [6].

With respect to the subcategory, *experience models*, different player aspects have been modeled to personalize game experience in previous work [35]. For example, Shaker et al. [43] applied neuroevolutionary preference learning to train engagement, frustration and challenge experience models with behavioral data from gameplay and player's visual behavior as input. Another example is the work of Yannakakis and Hallam [44] who demonstrated a methodology for optimizing player satisfaction during a game session.

The subcategory *Alternative and combined models* concern the approaches in which player models were combined with other model types, like affective models [35]. For example Chanel and Lopes [45] constructed deep data-driven affective models to determine anxiety and boredom from electrodermal activity measurements (EDA) measured with skin resistance sensors. These sensors can measure the sweat gland activity, where an increased activity indicates the experience of arousing emotions [45]. A user evaluation indicated the deep neural-network model (constructed through ensemble learning by combining 20 networks) to be capable of effectively adjusting the overall difficulty

during a game of Tetris. Another example within the subcategory *alternative and combined models* is the work of Blom et al. [46], where facial expression data was used to model player behavior and predict the player's affective state.

When discussing different types of DDA approaches, we follow Paraschos and Koulouriotis' [35] classification. However, due to ambiguity of the Paraschos and Koulouriotis' [35] fourth category *experience modeling* and its second subcategory *player experience models* we refer to the fourth category simply as *player modeling*. This research focused on *player modeling*. We gathered data and trained multiple random forest classifiers to estimate the player experience with respect to experienced difficulty.

2.1.5 Input data

Yannakakis and Togelius [17] identify three main types of input data of a player model, namely *gameplay data*, *objective data*, *game context data*. Although Yannakakis and Togelius [17] describe these three types in the context of player modeling, we believe these three types can be of interest in any DDA approach, also those that do not utilize player models. In this subsection we shortly discuss each of these three types of input data.

Firstly, *gameplay data* refers to all elements derived from the direct interaction between the player and the game, also called *player metrics* [17]. In other words, these interpretable measures of gameplay can concern anything that a player is doing in a game environment [17]. These measures are either general measures like performance and time spent on a task or game-specific measures like spatial locations of a player and selected weapons [17]. An example of utilizing gameplay input is the research of Kazmi and Palmer [47]. In this work an action recognition algorithm is developed to correlate particular player actions to the expertise level of a player and to adapt the game accordingly [47].

Objective data consists of measurements of responses to game stimuli, like physiology changes, speech and body movements [16], [17]. Psychosensory measurements allow to monitor the player's body alterations reflecting the player's emotional responses to a game [17]. Various psychosensory measurements were previously applied in DDA research, amongst others brain computer interfaces [7], [10], [48], heart-rate and galvanic skin response [49]. Another example of physiological input is pupil size. In previous work in the area of pupillometry and game difficulty, a correlation between pupil size and game difficulty and appraisal was found [22]. The results of this research indicated the potential of the appliance of pupillometry measurements in the context of user-adaptive interfaces in general [22].

Game context data can be viewed as a form of game metrics, capturing momentaneous state of the game during play, excluding gameplay data [17]. The game context can be argued to be necessary input to detect cognitive and affective responses of a player in a reliable manner [17]. *Game context data* can be argued to be necessary to interpret *gameplay data* and *objective data* [17].

Apart from these three main types of input data, Yannakakis and Togelius [17] also discuss static *player profile* information and *linked data*, which both could enhance the capacity of a player model. Here, a player profile includes all information about the player and is not directly linked to gameplay. For example, information on player personality, general demographics and player expertise level [17]. *Linked data* concerns data retrieved from web services, like social media posts, game reviews written and relevant semantic information extracted from diverse Web content [17].

In the context of the current research the three main types (gameplay, objective and game context data) were considered as input data. Additionally, player expertise (*player profile data*) was expected to be relevant, questions regarding game experience were included in the demographics questionnaire. We expected players with a high player expertise to experience a lower mental workload for the same difficulty setting compared to players with a low player expertise.

2.2 Mental workload

The concept of mental workload (MWL) has been widely discussed in the field of human factors and ergonomics [50]–[52] and was established during the 1980s [52]. The assessment and prediction of MWL can assist in the design of complex systems and automation [53] and has been extensively researched in the context of transport-related applications, like aviation, air traffic control and driving [52]. WML is a multi-dimensional construct [54], [55], however in a (too) simple manner, MWL can be viewed as the amount of resources invested in a task. In this section we first discuss the MWL related concepts, like *mental resources*, *task demand*, *task performance* before viewing the concept of MWL to further extend.

2.2.1 Mental workload related concepts

The amount of resources invested in a task are limited by the finite capacity of human *mental resources* [53]. When reviewing single *task performance*, the invested *mental resources* to a single task can be reviewed relatively to the operator's capacity [56]. The deployment of resources is under voluntary control, thus the relative amount of resources invested does not only depend on the operator's capacity, but also on the amount of resources an operator is willing to allocate [56].

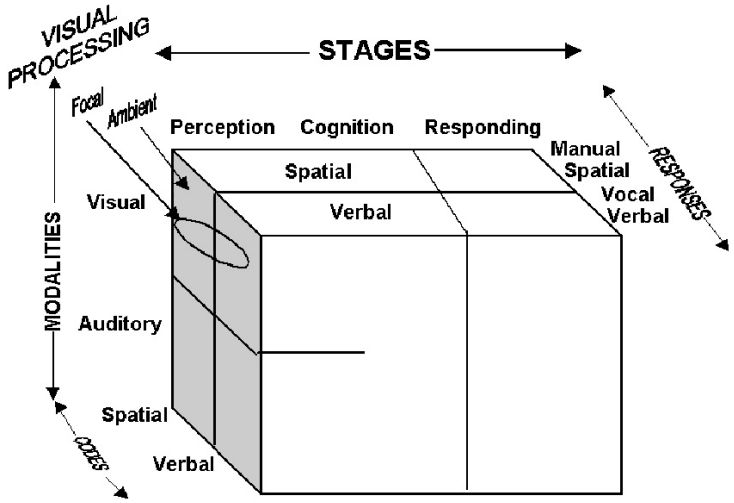

Figure 2.3: The 4-D multiple resource model [60].

In previous work different models are proposed to describe the limited amount of human *mental resources* [56], [57]. Kahneman's [58] *unitary resource model* assumes a single pool of *mental resources* that can be invested in different processing activities. The multiple resource model as proposed by [53], [59], [60] presents a modular view on *mental resources*, it defines separate resources that can be applied to different processing activities [60], see Figure 2.3. These separate resources can be defined in terms of a set of four dichotomies of information processing, namely stages of processing, codes of processing, perception modalities and visual channels [60]. The multiple resource model can assist in predicting resource *overload* in *multitask performance* as a result of multitask resource requirements viewed in terms of the four dimensions of the model [60]. A higher interference between simultaneously performed tasks is predicted if these tasks demand the same resources, for example when both tasks demand verbal processing (codes) [60].

The *performance-resource function* (RPF) relates the level of *task performance* to the invested *mental resources* on a task [53], [61]. Figure 2.4 displays the RPF of two tasks, whereas task B can be viewed as either less difficult (demanding less resources) or has received more practice compared to task A [53]. The same amount of resources spent results in a higher performance for task B compared to task A. At the right of point 3 on curve B in Figure 2.4, a higher amount of spent *mental resources* does not lead to a higher performance, here the task can be viewed to be *data-limited*, which means it is limited by the quality of data instead of the invested resources [53], [61]. Thus, after reaching a tasks data limit an increase in the amount of invested *mental resources* does not lead to a higher performance, thereby performance does not always correlate with the relatively invested *mental resources* [53], [61]. When changes in the amount of invested resources do influence the resulting performance a task is said to be *resource-limited*, see the region on curve B left to point 3 Figure 2.4 [53], [61]. *Mental resource* capacity differs per

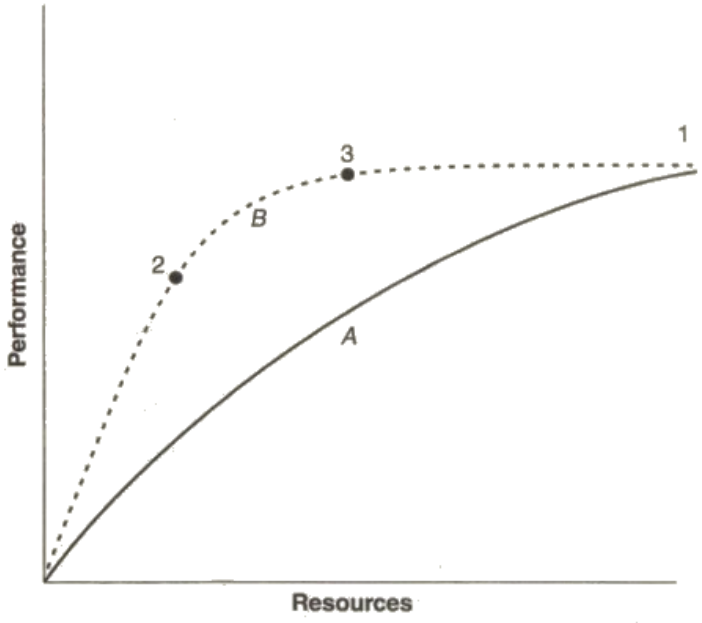

Figure 2.4: "The performance resource function and practice. Task B is practiced or easy; task A is unpracticed or difficult" [53].

individual, the same level of performance reached by two individuals does not imply an equal allocated amount of resources, one person may have plenty of resources left whereas the other does not [62]. Thus, when a task is *resource-limited* for one individual it is not necessarily *resource-limited* for another individual as well.

De Waard [56] explicitly differentiates between *task demand*, *task complexity* and *task difficulty*. *Task demand* is determined by the (subjective) goal set for *task performance* [56]. *Task complexity* depends on the number of required stages of processing [56]. Thus, *task demand* and *task complexity* can be viewed in isolation, without consideration of the individual operating a task, although a goal for a task can be subjectively set [56]. *Task difficulty* is dependent on the processing effort required by the individual for *task performance*, amongst others it is dependent upon context and capacity [56].

2.2.2 Defining mental workload

There are many varying definitions of MWL [52], [54], [63], [64], that focus on different aspects related to MWL, like task(s) demands, the effort spent to meet task(s) demands, the level of performance of a task(s) and the experienced expended effort [57]. Presenting a thorough overview of the different definitions of MWL is out of the scope of this research, for a comprehensive overview of previously provided definitions the reader is referred to overviews as presented in earlier work [52], [54], [63].

To clarify the concept of MWL, Van Acker et al [63], performed a concept analysis of MWL with the Walker and Avant [65] method. The analysis identified four elementary dimensions of MWL, namely "cognitive work demands, interacting with the human cognitive architecture, inducing cognitive physiological processing and a cognitive subjective experience" [63]. Van Acker et al [63] combined these four dimensions in the following conceptual definition of MWL: "Mental workload is a subjectively experienced physiological processing state, revealing the interplay between one's limited and multidimensional cognitive resources and the cognitive work demands being exposed to".

The Walker and Avant [65] concept analysis method applied to MWL allows to distinguish its building blocks and map these to the core (*defining attributes*), the preceding elements (*antecedents*) and outcome elements (consequences) of the concept of MWL [63]. The concept analysis identified Spending cognitive resources and subjective experience triggered as *defining attributes* , constituting the core of the definition of MWL. In other words, MWL "implies the cognitive resources that directly attend to cognitive work demands and the cognitive experience directly triggered by those work demands" [63]. Cognitive work demands and cognitive architecture were identified as the *antecedents* of MWL. Within concept analysis,*antecedents* are the elements that precede a concept, the components or situations that make the occurrence of the concept possible [65]. Thus, cognitive work demands and cognitive architecture can be viewed as components that proceed the subjectively experienced processing state. In previous work in the area of WML, performance is often considered as being an element of WML, however, the concept analysis identified performance as one of the*consequences* of MWL [63]. Hence performance can be viewed as an outcome of MWL.

2.2.3 Mental workload regions

In previous work, different views have been shared on the relationship between performance and *task demand*. In early work, the relationship between performance and *task demand* has been theoretically described as a non-linear relation where a higher *task demand* (eventually) results in a lower level of performance (i.e. [66]–[68], see Figure 2.5). Within this view, three regions can be identified ([66], [68], see regions A, B and C in Figure 2.5. Region A describes low demands with high performance and can be viewed as a low operator workload [56], [68]. Here an increase in *task demand* does not lead to performance decrements because the operator has a reserve capacity and can thus compensate [68]. In region B the demand exceeds the operators' capability to compensate for the higher demand, a degradation in performance follows the increased workload [56], [68]. Region C describes an extremely high demand together with performance at a minimum level [68].

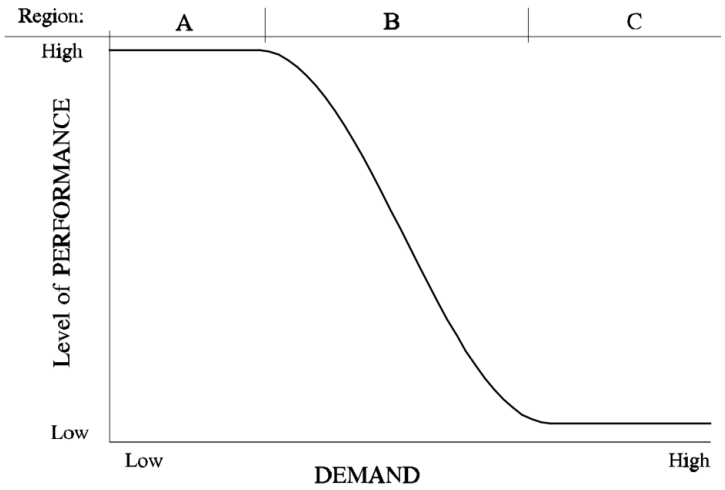

Figure 2.5: "Hypothetical relationship between demand and performance"[56].

Underload and *overload* are two sub-optimal workloads which can lead amongst others to performance degradation, attentional lapses, errors, frustration, fatigue, quality loss and time loss [69] as cited in [52], [63]. Different views exist on the concept of *underload* and *overload* [52]. For example, Wickens [53] considers the reserve capacity of resources. Wickens [53] refers to *underload* when an operator has a reserve capacity left and *overload* when all resources are supplied [53]. In other work effects on performance are viewed to indirectly identify the *underload* and *overload* effect [52].

In case of too little stimulation *underload* can occur; resources are allocated elsewhere or otherwise shrink through underuse (cf. [70], as cited by [52]). Low demand tasks can lead to difficulties in maintaining attention, an increased reaction time, an affected state like boredom which results in a reduction in total capacity and impedes the allocation of resources [54], [56].

The region of *overload* is captured within the theoretical relationship between demand and performance as displayed within Figure 2.5 (region B and C) [53]. To also capture the region of *underload*, De Waard [56] proposed an extended version of the demand performance relationship, which would result in addition of a deactivation or D-region at the left of Figure 2.5. The addition of the D-region results in an inverted-U relationship between *task demand* and performance as visible in Figure 2.6. Noteworthy, this inverted-U relationship shows similarities to the Yerkes-Dodson [71] as cited in [56] relationship which describes the relation between stimulus strength and learning [56] and the arousal theory accounting for the effects of arousal level on performance [56], [72].

In between the regions of *underload* and *overload* there is an *optimum range* of mental workload which is associated with best performance, see Figure 2.6 [52]. In previous work this *optimum range* has been viewed as an optimal state that is dependent on the

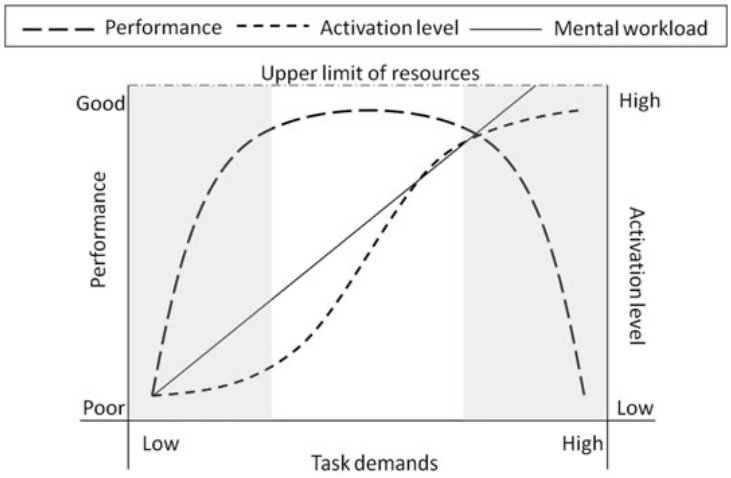

Figure 2.6: "The relationship between activation level, workload (*task demand*) and performance (adapted from de Waard 1996)" [52].

balance between demands and capabilities [11]. Maintaining a state of optimal performance does not only depend on *task demand*. In case demands do not meet the capabilities of an operator, an operator can compensate to some extent by the investment of additional resources [52]. The effort required to compensate in case of an *underload* or *overload* to keep performance at a high level can be viewed as state-related effort in case of *underload* and task-related effort in case of *overload* [56].

2.2.4 Mental workload predictions

Predictions of mental workload can help to assess the mental workload a task imposes on an operator, which is of value in the context of operator and system performance [53], [54]. Predictive models can predict performance breakdowns and reveal properties of task demand that lead to workload-*overload*, also referred to as crossing the redline of *overload* [52], [73]. Because predictive models can be applied before complex systems are built they can be applied to prevent workload-*overload* accidents and safe costs [73]. Workload predictions can inform design choices with respect to demand reduction [52], [73], [74]. Wickens [53] distinguishes between relative- and absolute predictions, a workload and performance comparison between two or more configurations and a workload estimation in an absolute manner respectively.

In previous work different predictive models have been proposed that consider different aspects of tasks, offer different methods of computation and are of different complexity [59]. An example of a more complex method is the computational multiple resource model which "computes the amount of interference predicted between two tasks as a function of competition of those tasks for shared resources" [59]. A more simplistic method is the timeline analysis and prediction (TLAP) which reviews the time required to perform a task and time available within a task sequence [59], [74].

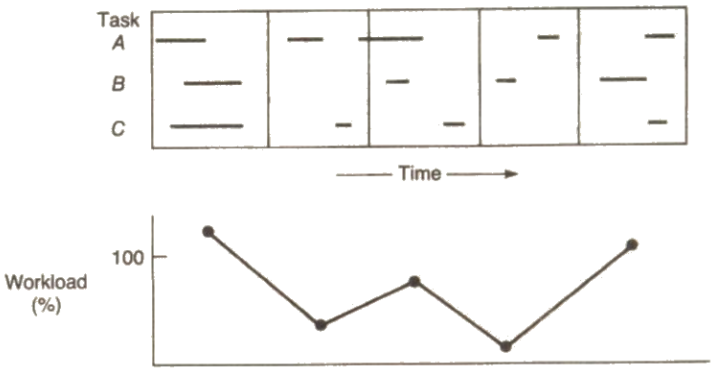

Figure 2.7: "Time-line analysis. The percentage of workload at each point is computed as the average number of tasks per unit time within each time window" [53].

TLAP considers time as a quantifiable variable and can be applied to predict workload for both single-and multi-task scenarios [52], [59], see the example in Figure 2.7. Time-line analysis and prediction (TLAP) reviews the time required to perform a task and time available within a task sequence [74]. The TLAP is a method to map operations and key events of a scenario on a timeline and thereby define when tasks have to occur and when related sequencing must start in order to complete on time [74]. Multiple task characteristics can be considered during the TLAP, for example, "tasks that must be performed at a critical time", "tasks that can be slipped for earlier or later performance", "tasks that must be performed in a given sequence" [74]. After mapping tasks on a timeline, workload estimates can be produced by dividing the time required by time available [74]. According to Parks and Boucek [74] resulting values above 80% can be considered as *overload*, here human operators can be expected to drop some auxiliary tasks.

In some task scenarios quantifiable variables can be identified in the context of (multi-) tasks performance and workload predictions, for example "the number of aircraft "handed" by an air traffic controller at any one time, the arrival rate of customers to a store clerk" [53]. In case of a relative prediction, a higher 'count' of a quantifiable variable often results in a lower time-sharing performance and a higher mental workload [52], [53]. In the context of absolute predictions of MWL, quantifiable variables can be considered to predict which 'count' lead to *overload* [53], for example the redline for working memory has been noted around at seven chunks of information [52].

In the context of the current research, the predictive model computational multiple resource model was viewed as too complex and expected not to add value compared to the more simplistic TLAP method (see also [59]). We applied TLAP in a free manner, by mapping the expected workload for our testbed game Pac-Man, whereby we considered the number of ghosts that are hunting Pac-Man and the different game states (scatter, chase and frightened).

2.2.5 Mental workload measurements

MWL measurements enable the assessment of "the mental cost of performing tasks in order to predict operator and system performance" [54]. Many different MWL metrics have been previously presented and these reflect the multidimensional nature of MWL [52]. Different views exist on MWL measurement categories [55]. In most areas three main categories are distinguished, namely *performance-*, *subjective-* and *physiological measures* [52].

Performance measures consider performance as an indicator of workload, within this category *primary task measures* and *secondary task measures* can be viewed as sub-categories [55]. *Primary task measures* assess workload by measuring operator performance on the task of interest, for example measures of error rates, performance speed, reaction time, accuracy [54], [68]. While it is advisable to always first examine the performance on the *primary task* [53], [55], *primary task performance* measures may be insufficient to estimate the workload on a *primary task*. Wickens [53] and De Waard [56] present multiple limitations of primary *task performance*. Amongst others, changes in primary *task demand* are not always reflected in changes in performance, for example due to reserve capacity of the operator, see region A in Figure 2.5 [53], [56]. In the *overload* region where performance degrades, *primary task measures* can be viewed as important assessment techniques [68], see region B in Figure 2.5. While performance measurements may not reflect the operator's workload throughout the different workload regions, in the current research *primary task measurements* can be considered to complement the pupil diameter measurements. With respect to the current research, when viewing *primary task measures* in the context of DDA we see similarities to the gameplay measures that consider the player's performance. In the current research we gathered gameplay performance data, however we did not use these as indicators of workload, but to predict the experienced difficulty.

Secondary task measures require concurrent performance of two tasks by an operator [68]. An estimate of workload of the task of interest (*primary task*) can be derived from the performance on an additional task (*secondary task*). The *secondary task performance* is associated with the spare capacity unused by the *primary task* [52], or in other words, it is "assumed to be inversely proportional to the *primary task* resource demands" [53]. A *secondary task* needs to be designed to occupy the same resources of the *primary task* but also not intrude upon or increase the MWL on the *primary task* [52]. Multiple limits of *secondary task measures* are discussed in previous work (e.g. see [54], [56], [68]). Due to the intrusive nature of the *secondary task measurements*, this type of measurements is not considered as a valid option for the current research.

Subjective measures, also referred to as self-assessment measures, include subjective rating scales and self-report measures (e.g. [52], [55]). A well-known subjective MWL assessment technique is the NASA Task Load Index [75]. Some examples of limitations of *subjective measures* are that these measures may be subject to biases, are not well suited to continuous monitoring of workload and might not capture those facets of workload that are inaccessible to consciousness [76]. In the context of games and DDA we believe that *subjective measures* utilized as input of a model are impractical, because these interrupt the continuous flow of the game. Although these measures can serve as output labels to train a predicting model in the context of DDA, we prefer player experience labels. Therefore, this measurement category was not applied in the current research.

Physiological measures rely on the assumption that bodily responses correlate with MWL, these measures derive workload from physiological activity of the operator [55]. *Physiological measures* of different bodily functions exist, like measures of the brain function, measures of the eye function, measures of the cardiac function and measures of the muscle function [68]. These measures can be classified in three categories, namely Autonomic Nervous system(ANS)-measures like pupil diameter and hearth rate measures, Central Nervous System(CNS)-measures like brain activity measures and peripheral responses like spontaneous muscle activity and eye movement [56]. *Physiological measures* are "capable of discriminating levels of capacity expenditure in nonoverload situations" [68]. An advantage of physiological measurement procedures is that these do not introduce extraneous signals into the operator tasks and thereby can be viewed as relatively unobtrusive [77]. Other advantages of *physiological measures* are that these do not require an overt response by the operator and can be collected continuously [56]. A disadvantage of *physiological measures* is that these measures mostly require specialized equipment [77]. Overviews of *physiological measures* of workload together with their disadvantages and advantages are presented by O'Donnel and Eggemeier [68] and Kramer [77]. In the current work we researched the added value of pupil size measures (*physiological measures*) in the context of DDA.

2.3 Pupillometry

The pupil size metric is a non eye-movement metric that can be used as an indicator of workload [19]. Pupil dilation increases with increasing *task demand* [78], a higher load is associated with larger pupils [19]. Because the pupil size can be affected by external factors such as lighting conditions, this metric is not completely reliable as an indicator of workload [19]. In this section we review the different types of pupil responses, aspects to consider with respect to measurements like filtering and taking a baseline and relevant previous work.

2.3.1 Pupil responses

Pupil responses can be categorized according to three distinct kinds of stimuli. Firstly, the pupil light response (PLR) is the pupil constriction in response to brightness and the dilation of the pupil in response to darkness [79]. Secondly, the pupil near response (PNR) is the constriction of the pupil in response to looking at a nearby object and the dilation of the pupil in response to looking at a far-away object [79]. Thirdly, the psychosensory pupil response (PPR) is a pupil dilation that occurs after an arousing stimulus, thought or emotion, also referred to as reflex dilation, effort-related dilation and arousal-related dilation [79].

When changing from light to dark environments the PLR can result in a pupil diameter increase around 3 to 4 mm [24]. Compared to the PLR pupil size changes, task-evoked pupil size changes are much smaller, around 0.1 to 0.5 mm [24] Noteworthy, differences in pupil size at an inter-individual level exist, amongst others related to differences in ethnicity and age [78]. Also, tiredness can indirectly affect pupil size; "spontaneous fluctuations in pupil ... seem to reflect fluctuations in level of arousal, and are especially pronounced when someone is tired" [80].

In the context of the current research we focused on the PPR and therefore, we discuss this response in more depth. Within the PPR category, a distinction can be made between two types, namely an orienting response and a slower arousal-or mental effort related response. This orienting response consists of a brief pupil dilation is elicited rapidly after something has captured attention [79]. The orienting response "is characterized by a fast pupil dilation that peaks between 0.5 and 1 s after stimulus onset" [79]. The slower arousal- or mental effort-related responses are linked to high-level cognition [79]. This type of response is endogenous, its size and profile reflect how mental effort and arousal evolve over time [79].

Hess and Polt [81] claimed first that pupil dilations indicate mental effort; they observed a correspondence between participant's pupil size when solving arithmetic problems and the problem difficulty. This correspondence between mental effort and pupil size was later confirmed in many contexts, in a variety of studies an increase of *task demand* or difficulty led to an increase of pupil size [58]. This pupillary dilation as a function of the mental load required to perform a cognitive task is also referred to as task evoked pupillary reflex (TERP) [82]. TERP has been obtained for a variety of cognitive processes, like short-term memory, language processing, reasoning, perception, sustained attention and selective attention [82]. In an interactive context, within an experiment the pupil size of participants was reviewed while performing an interactive task consisting of multiple subtasks and also here pupil size was found to correlate with varying mental workloads of these subtasks [83].

Although pupil size was found to increase "when task processing resources demands do not exceed available resources", conflicting results were found under conditions of *overload* [84], as cited by [85], [86]. In many previous works the TERP was researched to examine capacity-limited processing and pupil diameter was found anticorrelated with the depletion of a limited pool of resources by cognitive processes [25]. Granholm et al. [85] also researched pupil dilation for an excessive load in the context of a digit span recall task, aside from a low load and moderate load. Granholm et al.'s [85] "findings suggest that pupillary responses increase systematically with increased processing demands that are below resource limits, change little during active processing at or near resource limits, and begin to decline when processing demands exceed available resources".

Apart from pupil dilation in response to an increase of mental load, pupil dilation also occurs in reaction to emotionally engaging stimuli, regardless of the hedonic valence of these stimuli [87]. Pupil response is thus insensitive to the quality of the affect [88]. The pupillary reflex has been researched in different contexts with respect to emotional arousal [88]. Xu et al. [89] researched the influence of background luminance, emotional arousal and *task difficulty* level on pupil dilation. Luminance conditions were found to take priority over emotional arousal and cognitive demands in pupil size changes [89]. However, by using the difference between the average pupil size of the first half and second half of a task interval as a feature, a significant difference for pupil size with respect to *task difficulty* was found [89]. In previous work results indicated that "cognitive demands take priority over arousal factors in affecting the pupillary response" [90]. In this work, Stanners et al. [90] research suggests that "pupil response will show an arousal effect only when the cognitive demands of the situation are minimal". In context of the current work we kept in mind a possible influence of emotional arousal on pupil size with respect to pupil size metrics gathered for difficulty settings experienced by the player as too easy. Therefore, to allow for a better understanding of this possible confounding factor we additionally reviewed which elements and game events in our testbed game can lead to the inducement of arousal.

To sum up, in the current research we focused on pupil dilation as an indicator of mental workload. We stress the importance in the context of our current work to take the influence of surrounding brightness, screen luminance and emotional arousal into account with respect to our experimental setup and the selection of pupillometry features.

2.3.2 Pupil-size data preprocessing

Mathôt et al. [80] provide an overview of preprocessing steps pupil size measurements for pupillometry research. Firstly, tackling the issue of missing data, which occurs

when an eye tracker fails to extract the pupil from the camera image, often resulting in a pupil size of 0 [80]. Missing pupil size entries can either be replaced with interpolated values or can be simply ignored [80]. Trials with too much missing data are sometimes excluded from analysis [80]. Winn et al. [24] advice to drop trials when data consists of a larger percentage of blinks (> [15%, 25%]). In the current research we replaced missing pupil size entries with interpolated values and excluded trails with too much missing data.

Secondly, handling incorrectly reported pupil size values [80]. Often this occurs in case of eye blinks which "are characterized by a rapid decrease in pupil size, followed by a period of missing data, followed by a rapid increase in pupil size" [80]. To deal with eye blinks, cubic-spline interpolation can be applied to replace the missing and distorted data with a smooth line [80]. Winn et al. [24] recommend to apply interpolation from 50 ms before up to 150 ms after a blink "to avoid task-uncorrelated high-frequency changes in pupil size". In the current research we applied interpolation to replace data entries recorded during eye blinks and follow Winn et al. [24] recommendations with respect to the interpolation window.

Thirdly, pupil size changes due to eye movements, which some trackers can distinguish from real pupil-size changes [80]. When a participant looks straight at the lens of a tracker, the pupil is recorded as a near-perfect circle, when a participant for example moves her eyes to the left this results in a lower horizontal diameter of the pupil viewed from the perspective of the tracker. Mathôt et al. [80] offer three main approaches to deal with these eye movement artifacts. Firstly, by comparing conditions that are matched in terms of eye position [80]. Secondly, Mathôt et al. [80] describes data-driven correction, which uses linear regression to remove position artifacts from pupil-size data. However, this is an unadvisable procedure according to Mathôt et al. [80], because "not all effects of eye position on pupil size are artifactual" and therefore "any corrective technique that assumes this is problematic" [91] (we refer the reader to Mathôt et al. [91] for more details). Thirdly, "model-driven correction, which uses knowledge about the relative positions of the camera, eyes, and eye tracker" [80]. In the current research we used the pupil size from the computed pupil diameter in millimeters the GP3 tracker offers which is assumed to include model-driven correction.

Additionally to the preprocessing steps advised by Mathôt et al. [80], before interpolation the appliance of filters can be considered. For example, Winn et al. [24] applied an n-point smoothing average filter. Klinger et al. (2011) found changes faster than 10 Hz to be uncorrelated across the eyes, which justifies low-pass filtering at 10 Hz. Mannaru et al. [92] applied a Hampel filter to remove outliers, followed by a low-pass Butterworth filter at 4 Hz which was justified "since most of the pupillary activity falls in the frequency range of 0 - 4 Hz". Additionally, pupil size measurements of left and right

eyes can be averaged to reduce measurement noise, because "the left and right eyes exhibit matching pupillary responses" [93]. In previous work no significant difference was found for pupil sizes between the right and left eyes at different light amplitudes [94]. In the current work we applied filters to our data in order to remove outliers and remove noise and we averaged the left and right eye pupil sizes.

2.3.3 Pupil-size baseline correction

To reduce the impact of random pupil-size fluctuations between trials, baseline correction can be applied [80]. Baseline correction refers to the comparison of recordings in the experimental state against recordings taken in a subsequent resting state; the baseline state [95]. A baseline is commonly gathered for each trial rather than for a whole session and is based on measurements recorded immediately before the stimulus [24]. Regarding the length of the baseline period, durations range from 100 ms to 2 seconds [24], but also longer periods were used (e.g. 10 seconds [96] and 5 minutes [97]). Mathôt et al. [80] indicate that longer periods up to 1,000 ms "have the disadvantage of being susceptible to pupil-size fluctuations during the baseline period". Short baseline periods, like 10 ms, "have the disadvantage of being susceptible to recording noise". Winn et al. [24] demonstrate that "variation in the absolute baseline duration should play no substantial role in reporting pupil dilation" (100 to 3000 ms). According to Winn et al. [24] a baseline period of one second is the common practice. In the current research we applied baseline correction by comparing gameplay runs (trial) to baseline data recorded subsequent to each run. As for baseline period length, we used the commonly used baseline duration of one second.

Different approaches for the retrieval of a baseline pupil size are presented in previous work. Baseline pupil sizes can be obtained in preceding periods of rest (e.g. [87]) or while performing the same activity as during the experiment intervals but with a low *task demand* (e.g. [98]). In user studies where participants view a screen, often baseline pupil sizes are collected by letting the participants fixate on a blank screen (e.g., [96]). Instead of displaying a blank screen, a scrambled image can be displayed, created by scrambling all pixels in an image that represents the view displayed during the trial [23]. A scrambled image is meaningless and thereby suited for baseline retrieval [23]. Furthermore, because a representative scrambled image has the same mean intensity as the view displayed during a trial, screen luminance remains stable during baseline and trial period [23]. Mitre and Hernandez [23] introduced a grid scramble filter to generate images for baseline estimation "to better estimate baseline pupil size and to reduce the screen luminescence effect". To generate a grid scramble image, first a representative image is splitted into a n x m (n columns and m rows) and sequentially scrambling pixels per region in this grid. Grid scrambled images were found to result in better baseline estimates compared to black, white and scrambled images [23]. With

respect to different grid sizes no significant differences were found (86, 1010, 2020) [23]. However, Mitre and Hernandez [23] argued a grid size of 86 "better obscures the meaning of the in-test image". Based on the findings of Mitre and Hernandez [23], in the current research we displayed a grid scrambled image with a grid size of 86 (or in our case a 68 for a portrait gameview) during the baseline period.

With respect to the appliance of baseline correction, two main approaches exist, namely *subtractive*, in which pupil size is converted to an absolute difference from baseline pupil size and *divisive*, the conversion to a proportional difference from baseline pupil size [80]. These approaches are also referred to as absolute subtraction and proportional transformation, where the latter "can be considered [as] an additional follow-up step following baseline subtraction" [24]. The proportional transformation can be viewed as unnecessary considering absolute dilation is independent of baseline size, however, in previous work it was found relevant in the context of differences in pupil reactions related to differences in age [24]. In the current work we therefore worked with both *subtractive* and *divisive* baseline corrections separately. Noteworthy, *divisive* baseline correctio is more susceptible for distortion due to incorrect unusually small baseline measurements compared to *subtractive* baseline correction [80]. Therefore, Mathôt et al. [80] recommend using *subtractive* baseline corrections instead of *divisive* baseline correction. In case proportional change is a relevant measure, pupil size is preferably divided "by the grand mean pupil size during the baseline period averaged across all trials" [80].

Four other recommendations are provided by Mathôt et al. [80] which are relevant for the current research and were taken into account. Firstly, it is advisable to perform data preprocessing prior to baseline correction, but also, to not assume this leads to clean data [80]. Thirdly, a visual comparison between the uncorrected data and baseline-corrected data is advisable, to view if the correction reduced variability and did not qualitatively change the overall results. Fourthly, baseline artifacts occur within 220 ms after a baseline period, these artifacts are thus distinguishable from real effects by considering their timing [80]. Fifthly, it is advisable to remove trials in which baseline pupil sizes appear as unrealistically small, because these baseline artifacts can catastrophically affect the overall results [80]. Unrealistic small pupil size measurements can occur due to multiple reasons, amongst others because of noisy data, partly closed eyes or eye blinks [80]. Because the baseline pupil size distribution varies, a fixed minimum threshold might not catch all problematic trials [80]. Instead of using a fixed threshold, Mathôt et al. [80] advise to plot a histogram of baseline pupil sizes which allows to visually determine a minimum baseline pupil size.

2.3.4 Pupillometry measures

Commonly applied pupillometry measures are *peak pupil dilation* (maximal dilation in measurement interval), *peak pupil latency* (elapsed time between the stimulus and *peak pupil dilation*) and the *mean pupil dilation* [24], [25]. Other examples of pupillometry measures are the area under the curve and the slope of the curve rise and fall periods [25]. A single large analysis window can be used to detect main effects, multiple separate analysis windows can be applied to separate different phases of a trial [24]. Also, pupil size analysis can be locked to stimulus events [24].

In the current work we focused on two common measures, *peak pupil dilation* and *mean pupil dilation*. Our trials consisted of gameplay runs and can be viewed as considerably long compared to the more commonly short trials in previous work in the area of pupillometry and mental load. We expected that multiple phases can be identified within a gameplay run, depending on the game state and events of the testbed game. Therefore, we aimed to apply multiple analysis windows allowing us to separate the different phases.

Different normalization methods can be applied to compensate for individual differences in pupil size between participants. Examples of previously applied normalization methods are expressing deviation from the baseline as "percent or proportional change from baseline ...[or as a] percent change of average experimental trial value versus average control trial values", "expressing change in pupil size as proportion of the full dynamic range elicit by the pupillary light reflex (light-evoked range)" and the appliance of a z-score transformation [24]. Different normalization methods have different advantages, amongst others some methods address inter differences in variability in dilation, while other methods correct for average differences [24]. In the current research z-standardization was applied to normalize the pupil features.

2.3.5 Pupillometry in game context

Pupil size is researched in multiple previous studies in the context of games. Köles et al. [20] gathered pupil size measurements of participants playing the game Tetris in multiple difficulties, where the difficulty level was altered by increasing fall speed of the blocks. Pupil size measurements were normalized per participant through division with the mean of the baselines recorded at the beginning and end of the procedure [20]. A significant difference in normalized pupil diameter was found between the very easy difficulty level and the more difficult levels, but no significant difference was found when comparing the more difficult levels [20]. Based on their findings Köles et al. [20] suggest that individual analysis of signals could be more informative compared to the average values which they used during their data analysis [20]. We wonder if the findings regarding significant differences in pupil diameters indicate that

the participants did not reach the *overload* region. In our testbed game we strived to implement a large difficulty range, including 'too difficult' levels to try to inflict the state of *overload*.

The game Tetris was also used as a testbed game in another study researching pupil diameter for three game difficulties altered similarly to Köles et al. [20] by varying block fall speed [21]. Pupil size metrics were normalized per participant by subtracting the mean of the measurements across the three difficulties played [21]. Significant differences between the three levels were found, pupil size was found to positively correlate with workload [21]. Noteworthy, participants were also asked to perform a *secondary task* during the experiment to allow assessment of the demands of the primary Tetris task, but no significant differences were found in *secondary task performance* [21]. Where we viewed *secondary measurements* as too intrusive with respect to the gameplay, the findings of Malleck et al. [21] form an additional reason to exclude *secondary task performance* from the current research.

Strauch et al. [22] researched the pupil size of participants while playing a game of Pong. Strauch et al. expected to find an inverted u-curve, concurrent with findings in previous work were an inverted u-shape was found for heart rate variability ([99] as cited by [22]) and for self-reported valence ([100] as cited by [22]). Also, the expected inverted u-shape can be viewed to be concurrent with the non-linear relation between pupil diameter and *task difficulty* as found by Granholm et al. [85]. Strauch et al. [22] reason that the linear relation between pupil size and game difficulty found by Köles et al [20] might have been the result of "suboptimally chosen levels of difficulty", the *overload* region might not have been reached and thereby the results did not follow an inverted u-curve [22]. The results of Strauch et al. [22] do demonstrate an inverted u-shape for the relation between pupil size and increasing difficulty [22]. Furthermore, the results indicate that the pupil diameter can serve as an indicator of the *under- optimal-* and *overload* of a game for the player, which is strongly associated with subjective gaming appraisal [22]. Noteworthy, for the most difficult levels the results suggested a small to moderate correlation between a larger pupil size and the participant's level of engagement [22]. In the current research we built upon the previous work of Strauch by implementing difficulty levels with which we aimed to cover the entire range from *underload* to *overload*. Our research differed with respect to the testbed game; we deliberately chose to research pupillometry in the context of the more complex video game Pac-Man compared to the "simple two dimensional video game" Pong [22].

In the context of an educational video game, Mitre-Hernandez et al. [23] trained and compared multiple Random Forest binary classifiers, predicting two difficulties (easy, hard), using different sets of game and pupillometry features. Four different pupil fea-

tures were gathered per measurement interval. Firstly, the *mean pupil diameter change* (MPDC), which was the average pupil diameter subtracted by the accompanying baseline average (*subtractive* baseline correction) [23]. Secondly, *peak dilation* (PD) which "is defined as the maximal dilation obtained in the measurement interval time of the level" [23]. Thirdly, *latency to peak* (LP), "the amount of time elapsed between the beginning of the measurement interval and emergence of peak dilation" [23]. Fourthly, *average percentage change in pupil size* (APCPS), which is the average of all pupil size measurements subtracted with and subsequently divided by the baseline pupil size (*divisive* baseline correction) [23].

Statistical differences were observed for the *subtractive* size-related pupillometry features MPDC and PD [23]. No significant differences were found for the time-related and *divisive* pupillometry features, LP and APCPS respectively. Multiple random Forest classifiers were trained to predict the game difficulty [23]. The highest accuracy (87,5%) was found for the model trained with the *game features* 'total errors' and 'time to complete a stage together with the pupillometry feature PD [23]. The MPDC (*subtractive* baseline correction) and APCPS (*divisive* baseline correction) results are in line with Mathôt et al.'s [80] recommendation to use *subtractive* baseline correction instead of *divisive* baseline correction.

In a similar fashion to the work of Mitre-Hernandez et al. [23], we trained and compared Random Forest classifiers with different feature subsets to review the added value of pupillometry features in the context of DDA. Thus, to a certain extent the current work can be viewed as a repetition of the work of Mitre-Hernandez et al. [23]. We identify three main differences. Firstly, as a testbed game, we used the arcade game Pac-Man, which offers more dynamic gameplay compared to the educational video game in Mitre-Hernandez et al. [23]. Furthermore, Mitre-Hernandez [23] trained binary classifiers predicting two difficulty levels, while we trained multi-class classifiers to predict the experienced difficulty.

Where the above studies in game context researched pupillometry in relation to difficulty, Gutjahr et al. [88] researched pupillometry in the context of games in relation to the affect arousal. In this study participants played a digital exercise game and pupil size measurements were gathered and reviewed as indication of arousal [88]. The goal of the study was "to show that active and successful interaction with a dynamic exergame triggers positive emotions to previously neutral stimuli and that measuring pupil reactions to rewarding game events is suited to detect these emotional reactions ..." [88]. In this study a between-subject design was employed for three conditions, namely the experimental condition where participants actually played the exercise game, a yoked control condition with the same stimuli but without actual interaction and a control group without the game [88]. Within the experimental condition, par-

ticipants could control the avatar's height by varying the pedal rate of an ergometer in order to gather flying letters [88]. Successfully catching a letter and thereby gathering points was viewed as a rewarding game event for which pupil size measurements were gathered [88]. Statistically significant pupil dilations in response to these rewarding game events were found [88]. Noteworthy, higher reported motivation was found to be correlated with larger pupil dilations in response to rewarding game events.

When viewing the findings of the Gutjahr et al. [88] in context of the current work, we expected rewarding game events that induce arousal to influence the pupillometry features. We expected this influence to occur when a player experiences the game difficulty as (too) easy, because "pupil response will show an arousal effect only when the cognitive demands of the situation are minimal" [90] (also see 2.3.1). In the game Pac-Man, examples of rewarding game events are eating pellets and ghosts. We chose not to exclude these rewarding game events from the testbed game, because our interest lies in researching the contribution of pupillometry features in a realistic gameplay scenario. However, in the original Pac-Man game, when Pac-Man eats a pellet or ghost, the score is incremented, emphasizing these events. In our testbed game this visual feedback, the score, could be easily left out, aiming to lower arousal and the associated pupil response.

2.4 Pac-Man

In the current research Pac-Man was chosen as a testbed because we believe it offers more complex gameplay compared to the testbed games in previous work in the area of pupillometry and game difficulty (Tetris, Pong and Refraction) [20]–[23]. We argue this allowed us to research pupil features in a more dynamic gameplay setting and thereby hope to contribute to the findings in previous work. Within this section we provide a short description of Pac-Man. Furthermore we discuss previous work that utilized Pac-Man as a testbed and review the aspects of Pac-Man that are relevant in the context of the current research.

2.4.1 Pac-Man

The goal for each level in the arcade game Pac-Man is to eat all pellets in a maze while avoiding ghosts. The player enters the next level when the maze is cleared of all pellets and a new round with a reset maze begins. The ghosts can capture Pac-Man, which results in the loss of a life for Pac-Man. When Pac-Man is caught and has no lives left the game is over. Pittman [101] provides a very thorough overview of the game Pac-Man. Below, we review those aspects that are relevant in the context of the current research, namely the ghost states, ghost chase strategies, the maze and Pac-Man's preturns.

In Pac-Man three main game states can be identified, corresponding to the three behavioral ghost modes, namely chase, scatter and frightened. In the chase state the ghosts hunt down Pac-Man. Each level starts in the scatter state during which each ghost heads to its own corner. The chase and scatter states alternate at predetermined intervals. The frightened state is activated when Pac-Man eats one of the four super pellets that are located in each corner of the maze. During this state the ghosts randomly wander during the maze for a few seconds and the roles of hunter and hunted are swapped, Pac-Man can capture ghosts.

Each of the four ghosts have their own chase strategy. In short, the red ghost Blinky targets Pac-Man's current tile which results in behavior that comes across aggressive and as if Blinky is tailing Pac-Man [101]. The pink ghost Pinky targets the tile that is located four tiles in front of Pac-Man. Pinky comes across as if it is able to get ahead of Pac-Man and to cut him off [101]. Clyde, the orange ghost, alters between two chase strategies which results in seemingly unpredictable behavior at times [101]. When Clyde is closer than eight tiles away, its scatter mode target tile located below the left bottom of the maze is targeted. When Clyde is more than eight tiles away, it targets Pac-Man's tile [101]. Inky, the blue ghost, uses a more complex targeting scheme compared to the other ghosts [101]. Inky's target tile can be found by drawing a line between Blinky's current location and the tile that is two tiles in front of Pac-Man. Next, double the length of this line and then when keeping one end of the line at Blinky's position, the other line end serves as Inky's target tile [101]. We refer the reader to Pittman [101] for a more thorough overview of the chase strategies. In subsection 2.4.2 we reflect on ghost chase strategies with respect to difficulty.

At the beginning of a level and after the loss of a life, Pac-Man and the ghosts start at their starting position. Pac-Man starts at the center of the bottom half. The starting positions of Pinky, Inky and Clyde are located in the ghost house, the rectangular area in the center of the maze. Blinky's starting position is located directly above the ghost house. Blinky immediately heads towards its scatter corner at the top right of the maze and Pinky immediately follows heading for the left upper corner.

At the start of a level, the two ghosts Inky and Clyde wait in the ghost house until Pac-Man collects a specific number of pellets (dot limit, 30 and 60 respectively in the first level). First Inky's dot counter is activated and when the dot counter reaches the dot limit the ghost enters the maze, similarly then Clyde's dot counter is activated. In case Pac-Man does not eat any pellets till a time limit is reached the ghost next in line to enter the maze, exists the ghost house [101] (the ghosts dot limit is not reset).

When a life is lost, a global dot counter is used instead of the personal ghost dot counters. Blinky directly exits the ghost house while Pinky, Inky and Clyde wait till Pac-Man collects the number of pellets that matches the ghost's personal global dot counter

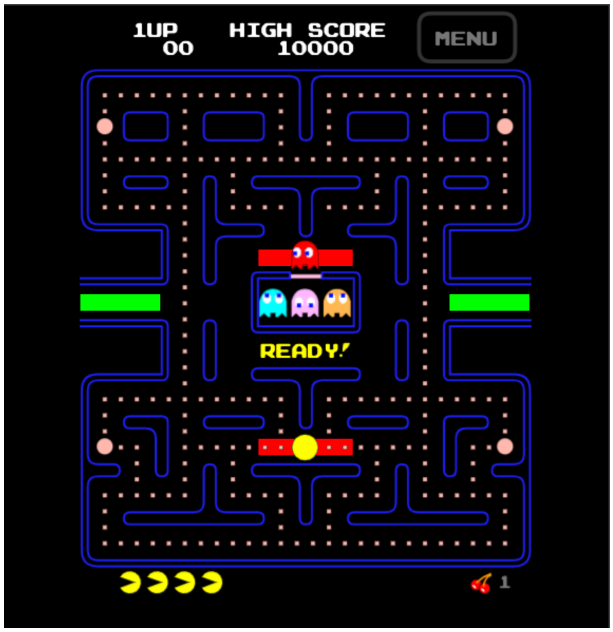

Figure 2.8: A screenshot from a browser replica of the original Pac-Man game[3]. The screenshot displays the structure of the maze and the location of the pellets and super pellets at the start of a level. Furthermore, the location of Pac-Man and the ghosts correspond to their location at the beginning of a level or after the loss of a life. The green marked zones indicate the tunnels where a speed penalty is applied to the ghosts. The red marked zones indicate the areas where the ghosts are forbidden to turn upwards.

limits (7, 17 and 32 respectively). A captured ghost is relocated to the ghost house in the center of the maze and exists when either the personal dot counter or global dot counter limits are reached.

The maze in Pac-Man is static, its structure stays the same throughout the whole game[2], see Figure 2.8. At the start of a level 244 pellets and four super pellets are placed in the maze at fixed locations. The maze contains two connecting tunnels between the right and left edges at center height, see the red areas in Figure 2.8. These tunnels function as teleports, teleporting Pac-Man and the ghosts from the left to the right and the other way around. When moving through these tunnels the movement speed of the ghosts is affected with a speed penalty, resulting in nearly half speed [101]. The areas marked in green in Figure 2.8 are zones where the ghosts are forbidden to make upward turns. Thus, in these zones the ghosts can only move from left-to-right and right-to-left [101].

Pac-Man can turn before reaching the center (pre-turn) or after reaching the center (post-turn). During a pre-turn Pac-Man moves in a 45° angle towards the next tile which results in a temporarily doubled speed [101]. Because ghosts are only able to turn or change direction at the middle of a tile, a player can use pre-turns to her ad-

[2]Level 256 does differ due to a bug, which we disregard for the remainder of this work for the sake of clarity. We refer the reader to Pittman [101] for in depth information regarding this bug.
[3]`https://www.pacman1.net/`

vantage to put more distance between Pac-Man and ghosts, this technique is known as cornering [101]. In the context of the current research, we expected that the occurrence of cornering can indicate a skilled player. Additionally, we could imagine that a higher occurrence of post-turns occurs when a player experiences a too high mental load. The occurrences of both were captured as an input feature for our Random Forest classifiers.

Noteworthy, Pac-Man is discussed to a further extent in chapter 3. Amongst others, the three ghost states are reviewed in the context of workload, we review the game difficulty aspects *intrinsic skill*, *stress* and *power provided* in Pac-Man and the mapped *challenge hierarchy* of Pac-Man is presented.

2.4.2 Difficulty in the original Pac-Man

As the player progresses through the game, the difficulty increases, up to level 21 after which each level offers the same difficulty [101]. In higher levels both pac-man and the ghosts move at a higher speed, which results in a higher *overall speed* [101]. Also, the relative movement speed of the ghosts compared to Pac-Man is increased at the higher levels [101]. Furthermore, as the levels progress the durations of the scatter and frightened states are shortened. Also, the dot limit at the start of a level and after the loss of a life and the timer limit, which both control the waiting time of the ghosts in the ghost house, are lowered [101].

The difficulty also increases within a level itself. Namely, when Pac-Man almost has cleared the pellets in the maze, Blinky becomes Elroy. Elroy moves with an increased movement speed compared to Blinky and also targets Pac-Man within the scatter state [101]. In the first level, when only 20 pellets remain Blinky turns into Elroy, moving at least as fast as Pac-Man and when only 10 pellets remain Elroy speeds up again moving faster than Pac-Man [101]. After the loss of a life, Elroy becomes Blinky again, but when all ghosts have left the ghost house, blinky changes back into Elroy [101]. As the levels progress, the difficulty is also influenced by an increment in the number of pellets left that triggers the turn of Blinky into Elroy. Noteworthy, there is no designed ending in Pac-Man, the developers "assumed the game's increasing difficulty was sufficient to prevent anyone from playing indefinitely" [101]. Beyond the 21st level the game every level is identical (with exception of the 256th level which is an incorrectly displayed level due to an overflow bug) [101].

2.4.3 Game difficulty of and game AI in Pac-Man

In this section we review previous work in the area of video game difficulty and game AI in which Pac-Man was used as a testbed game in a chronological manner. These previous studies offer insights with respect to game elements of Pac-Man that are of

influence to the in-game difficulty. Additionally, performance measures and fitness functions presented in some of these studies, provided properties that were used as input features for our Random Forest models.

2.4.3.1 Ghost behaviors and measure of interest

Yanakakis and Hallam [102], [103] applied neuro-evolution mechanisms to a test-bed version of Pac-Man. In [102], an off-line evolutionary learning mechanism and on-line learning approach was applied to train ghosts against three different types of simulated Pac-Man players. Yannakakis and Hallam [102] believe that the "interest generated by the opponents" to be directly related to the interest of a computer game. In [102] a measure of interest of any predator / prey game dependent on three criteria is introduced (interest measure). The first criterion is that a game is "neither too hard nor too easy", and depends on the average number of steps needed to catch Pac-Man and the maximum number of steps in N games [102]. The second criterion is defined as "when there is diversity in Ghosts' behavior over the game" and depends on the number of steps needed by Pac-Man to win a game and the minimum number of steps needed to catch Pac-Man [102]. The third criterion "when Ghosts' behavior is aggressive rather than static" considers the extent to which ghosts cover the maze, this criteria is dependent on the entropy of the ghosts' cell visits [102]. For a full overview of the interest measure and three criteria we refer the reader to Yannakakis and Hallam [102]. In the context of the current research, the elements used within the definitions of the interest measure criteria served as input features (see 3.3).

Furthermore, in [102], three emerging ghosts' behaviors of the off-line trained ghosts together with three fixed behaviors were tested against the three types of simulated Pac-Man players. With respect to performance, the three fixed behaviors, *random*, *followers* and *near-optimal*, cover a distinctly different range for the three Pac-Man types. Here, performance is a normalized measure dependent on the number of Pac-Man kills and eaten pellets [102]. Random ghosts "randomly decide their next available movement" and scored a performance value in the range of [0.400, 0.446] against the three Pac-Man types [102]. *Followers* were "designed to follow PacMan constantly", reducing the greatest of the relative horizontal or vertical distance from Pac-Man.*Followers* scored a performance value in the range of [0.776, 0.839] against the three Pac-Man types [102]. The *near-optimal* ghost strategy was "designed to produce attractive forces between ghosts and Pac-Man as well as repulsive forces among the ghosts" [102]. *Near-optimal* ghosts scored a performance in the range of [0.96, 1.0] against the three Pac-Man types [102]. We argue that the performance ranges of these three fixed ghosts' behaviors to be directly related to Pac-Man's difficulty. Therefore, in the current research these three fixed ghosts behaviors as defined and described by Yannakakis and Hallam [102] were interesting in the context of difficulty adaptation of our testbed game.

In [103] the on-line learning mechanism from [102] was tested over more complex versions of the stage (maze) "to explore the relation between the interest measure and the topology of the stage". In this work Yannakakis and Hallam [103] defined a complexity measure that expresses complexity as inversely proportional to the average corridor length of the stage. We considered alterations to the maze, but ultimately decided to use the original Pac-Man maze.

2.4.3.2 Measure of flow

Beume et al. [104] propose a *measure of flow* that views the game from the player's point of view, instead of viewing the game from the game predators' perspective [102]. This flow measure depends on "the time-fraction in which the player is confronted with interesting situations" [104]. The flow measure distinguishes these "situations by the number of ghosts that are near Pac-Man and weight the situations by this number to reflect the difficulty" [104]. Beume et al. [104] argue that Yannakakis and Hallam's [102] third criterion which depends on ghosts' entropy over the maze may be misleading. They reason that a maximal entropy can occur while the ghosts completely ignore Pac-Man, which results in a high interest value but does not actually contribute to the game's interest [104]. Beume et al. [104] performed a user study to gather data from participants regarding experienced fun and difficulty and compared this with the interest measure and flow measure [104]. Both measures were found to describe the experienced level of fun, but did not capture the perceived difficulty [104]. Noteworthy, the positive correlation between the perceived game fun and the interest measure was found to be caused byYannakakis and Hallam's [102] second criterion "when there is diversity in Ghosts' behavior over the game". Furthermore, Beume et al. [104] found game fun to be "much more predictable for inexperienced players than for frequently playing probands".

In the context of the current research, Beume et al.'s [104] flow measure was utilized as input feature, complementing the three criterion of Yannakakis and Hallam's [102] interest measure as input features. Furthermore, the findings of Beume et al. [104] regarding the difference between inexperienced and experienced players informed us to gather relevant related aspects per participant in a questionnaire, like a participant's gaming frequency and gaming skill.

2.4.3.3 Trees to control ghosts

Monte-Carlo for Trees can be trained to control the ghosts in Pac-Man, a longer simulation duration results in a higher ghost win rate [105]. Thereby, the simulation duration can be used as property to adjust the difficulty of the game [105]. However, because of the computational intensiveness of Monte-Carlo method, it is not satisfactory to apply this method to the ghosts in Pac-Man [105]. Instead, Liu et al. [105] trained multi-

ple less computational intensive artificial neural networks (ANNs) with data created by the Monte-Carlo Trees for different multiple simulation times. The data was optimized by only using data from games won by the ghosts [105] For the resulting ANNs a fixed win rate was measured [105].

Similarly to the Monte-Carlo for Trees in [105], Upper Confidence bound for Trees (UCTs) can be trained to control ghosts in Pac-Man [106]. For UCTs a longer simulation duration also results in a higher ghost win rate [106]. Like the Monte-Carlo for Trees in [105], UCTs are computational intensive and therefore not suitable in the context of DDA [106]. Where Liu et al. [106] used optimized data (data from games won by ghosts only), in [106] data from all games was used to train ANNs. Li et al. [106] reason that the performance of an ANN is dependent on the quality of the training data, "without enough occurrences for a certain path, the ANN cannot be trained well". And vice versa, as "simulation time grows, the UCT collected data becomes more and more precise, which results in better trained ANN" [106]. The results in [106] indeed indicate that by using UCTs created data with different UCT simulation times, multiple ANNs with different levels of performance can be created. By loading ANNs with different levels of performance for the control of the ghosts in Pac-Man, different levels of game difficulty can be offered [106].

The approach of Li et al. [106] is very interesting in the context of the current research, as it would allow us to create a set of multiple ANNs for the ghost control that result in different levels of difficulty. We believe this approach to be more time consuming compared to the three fixed ghosts' behaviors introduced by Yannakakis and Hallam [102] which also offer different levels of difficulty. Due to the limited scope of the current research our preference therefore went to implementing Yannakakis and Hallam's [102] three fixed ghosts' behaviors to control the difficulty. Still, in future work, the approach of Li et al. [106] could allow for a more precise difficulty adaption, since the UCT simulation duration can be tuned as needed to in turn tune the ANNs performance.

2.4.3.4 Game factor analysis

Fraser et al.'s [107] introduced a methodology for evaluating the impact of game factors on a set of player experience response variables[4] and applied this to the game Pac-Man. The set of factors in Pac-Man consisted of generic agent and fruit factors, together with more specific factors for both two Pac-Man algorithms and two ghost algorithms. Nine metrics of performance goals were selected, namely *score*, *number of steps*, *number of close calls*, *number of repeated steps*, *number of fruits collected*, *number of tokens collected*, *number of power-pellets collected*, *number of ghosts eaten* and *number of levels completed*.

[4]For a full overview of this methodology we refer the reader to [107].

These nine response variables offer insights at a more granular level compared to for example Yannakaki s and Hallam's [102] *interest measure* and Beume et al. [104] *measure of flow*. Therefore, these nine response variables were interesting as input features in the context of the current research.

Regarding the results of Fraser et al. [107] we here review the results of a subset of response variables and factors which we consider most relevant in the context of the current research. We reason that a more difficult game results in a lower *score*, in a lower *number of tokens collected* and lower *number of levels completed*. We also review a subset of factors below, namely only the factors that can be applied regardless of the implemented agent algorithms. These factors are the set of bonus factors and generic ghost factors. The bonus factors are "the number of steps the bonus item is available" (*fruit time*), "the perceived value of the bonus item" (*perceived value of fruit*) and "the frequency in steps at which the bonus will be generated" (*fruit frequency*). The generic ghost factors are "Manhattan distance of the vision range" (*ghost vision*), "number of steps in the flee and death state time" (*flee time*, *death time* respectively).

In Fraser et al. [107] the influence of the factors on the response variables was measured for the paired combinations of the two Pac-Man algorithms and two ghost algorithms (four pairs in total). Per pair, the results are shared for those factors for which a statistically significant effect on a response variable was found, indicated as either positive or negative influence [107]. For the *fruit frequency* factor no statistically significant influences were found on all response variables [107]. Only in one of the four Pac-Man and ghost pairs a few response variables were significantly affected by the *fruit time* and *perceived value of the fruit*, however the response variables in our subset of focus (*score*, *number of tokens collected* and *number of levels completed*) were significantly unaffected [107]. Informed by these results, in the context of the current research we believed that *bonus fruit* factors were not suitable as difficulty parameters.

The *ghost vision* factor had a significant negative influence on all response variables in our subset of focus, the results for this factor "suggest a vast improvement in the efficiency and performance of the ghosts" [107]. In the current research, the implemented *ghost behavior* did not depend on the *ghost vision* factor and was found irrelevant.

For all Pac-Man and ghost pairs in Fraser et al.'s [107] experiment, the ghost factors *flee time* and *death time* showed a significant positive influence on all response variables in our subset of focus, except for one pair where no significant influence on the *number of levels completed* was found for the factor *flee time*. In other words, these two ghost factors can be viewed to decrease the level of difficulty. Noteworthy, these factors can only affect the difficulty during the ghost frightened game state and not during the chase and scatter game states (see 2.4.1). We expected many players to experience the frightened game state as a minor break from the chase state. We therefore also expected

that when a participant is asked to either rank or rate Pac-Man gameplay runs the focus is placed on the difficulty of the chase states. Thus, although these two factors could serve as a difficulty parameter, in the context of the current research our preference went to difficulty parameters that either affect difficulty in an overall manner or during the chase state.

2.4.3.5 Player feedback to estimate player's skill

Ebrahimi and Akbarzadeh [108] proposed "a self-organizing system (SOS) to adjust the difficulty level of games" and applied this to Pac-Man. ANN together with Interactive Evolutionary Algorithms were used to evolve NPCs online. In the context of the current research, the off-line and on-line training approaches are of interest. However, due to the limited scope of this research we are not planning to implement a neuro-evolutionary controller for ghosts. Therefore, here we do not fully review the learning approaches of the SOS as proposed by Ebrahimi and Akbarzadeh [108]. Nonetheless, the fitness function of the online training in [108] is relevant to review with respect to input features in the current research. During the online training the fitness function is replaced by human evaluation; the player's skill level is estimated by using the player's feedback. Noteworthy, in the performed experiment no human participants were involved, during the online training phase three Pac-Man types were simulated representing players with different skills [108]. The fitness is calculated per time frame of ten steps and is a combination of three types of player feedback, namely number of keys pressed, number of times to switch between keys and the number of times that Pac-Man hits the walls [108]. For each of these three feedback types the measured value is compared against an ideal value given Pac-Man start position and end position in the specific frame. To illustrate, for the number of pressed keys, the delta number of steps between the position of Pac-Man at the beginning and end of a time frame is compared to the measured number of keys.

In the context of the current research, we believe player feedback types can serve as input features. The three player feedback types as presented in [108] can be complemented with other types of player feedback. For example, the timing of Pac-Man's turning by measuring the occurrence of pre-turns and post-turns in relation to the total number of turns within a frame (see 2.4.1).

2.4.3.6 Facial expressions

In one of the experiments in [46], perceived in-game difficulty was predicted by analyzing players' facial expressions in the game Pac-Man. In this experiment the participants played three versions of Pac-Man in which the difficulty was altered by changing Pac-Man's speed. In these versions, Pac-Man's speed compared to the ghosts' speed was slower (version a), higher (version c) and similar as in the original Pac-Man (version

b). In the context of the current research the findings of Blom et al. [46] regarding the experienced difficulty are of interest. Version c was expected to be experienced as less difficult, since Pac-Man's speed was higher than the ghosts' speed. However, Blom. et al. [46] observed that a higher level of game-playing skill was required to keep up with this fast version C and only experienced players rated it at a lower difficulty. Based on these findings, we expected that the effect of alterations of speed on the experienced difficulty differs per person and depends on the player's experience. Therefore, in our experiment we included player experience input features, like estimated skill level and player experience in the context of 2-d maze and prey/predator games. Noteworthy, in [46], "the player was granted an unlimited number of lives in order to avoid unnecessary frustration for some levels". Frustration is a negative high arousal and the induction of either positive or negative high arousal can lead to a pupil response [79], [87]. In order to minimize pupil responses other than task-evoked pupil responses, we excluded the visual feedback of the number of lives in our experiment.

2.4.3.7 Player frustration prediction

In [109] the game Pac-Man was used as a testbed in the context of the prediction of player frustration. Wolterink and Bakkes [109] performed a user study in which participants played five levels of a modified Pac-Man game. While playing, after intervals of 30 seconds the participant was asked to rate her overall experienced level of frustration for the preceding gameplay interval. Also, participants were asked about the absence or presence of five components of frustration. Recorded *game features* and user behavior together with the participant responses formed a dataset which was used to train multiple random forest classifiers to predict player frustration [109]. Feature importance analysis was applied to map the importance of the input features [109]. Four of the five components of frustration scored relatively high in feature importance, with "repeatedFailures" scoring highest by far [109]. Random forest classifiers that were trained to predict frustration component responses yielded a relatively mediocre accuracy [109]. As a possible explanation, Wolterink and Bakkes [109] mention that the set of gameplay features might not accurately reflect when the frustration components are triggered. Additionally, regarding the "repeatedFailures" component, Wolterink and Bakkes [109] suggested another possible explanation for the mediocre accuracy; frustration might not be caused by actually repeatedly failing, but by the perception of repeatedly failing.

The findings in Wolterink and Bakkes [109] were interesting to review in the context of the current research to minimize the experience of frustration to prevent workload unrelated pupil responses. We reviewed which elements can lead to the perception of failing, for example, the feedback of the score in Pac-Man and Pac-Man's lives. However, we believed it was not feasible to fully exclude or alter elements that lead to the

perception of failing. To elaborate, we expected the perception of failing can also be dependent on personal goals set by the player and by repeatedly failing these, for example “first clear the right quartile of pellets”. We could not track such personal goals. Furthermore, although the set of gameplay features implemented by Wolterink and Bakkes [109] was created in the context of the prediction of frustration, it offered a good starting set for our experiment.

3. Design of the Pac-Man testbed game

In the current work we were interested in task-evoked pupillary response in the context of the complex and dynamic gameplay Pac-Man offers. An available off-the-shelf Pac-Man testbed implementation did not meet our requirements. The codebase of this implementation was built around Unity's 2d movement functionality, which does not entirely suit the arcade grid based movement of Pac-Man thereby resulting in unsmooth control. In the context of our research we wanted to avoid unsmooth control since this can be expected to lead to the experience of frustration (negative high arousal) and thereby lead to a pupil response unrelated to workload [79], [87]. We expected a custom build from scratch resulted in less time spent compared to a full refactor of the available codebase, therefore we started from scratch with a new implementation in Unity.

Four main parts can be distinguished in our testbed implementation. Firstly, the implementation of the game itself. Secondly, the implementation of gameplay and game context data output that serve as data to calculate the input features for our random forest models (*game features*). Thirdly, the implementation of adaptable game components in order to alter the game difficulty (*adaptive game components*). The last part consisted of an implementation that arranged the connection to the server of the used eye tracker on a separate thread in order to retrieve pupil data.

The actual implementation of the first three parts were preceded with consciously made design choices, which were informed by systematic reviews of multiple aspects. These design choices and their substantiation are described within this section to a further extent. In short, with respect to the implementation of the game part, we applied minor adaptations to the original Pac-Man game aiming to minimize workload unrelated pupil responses. With respect to the implementation of the data output, we first gained insight into the possible relevant *game features* which the output data serves. Finally, with respect to the implemented *adaptive game components* we identified which game components can be adapted in order to adjust the difficulty and prioritized these before selecting our set.

3.1 Adaptations to the original game

3.1.1 Minimize non workload related pupil responses

Our research focused on workload related pupil responses, therefore we wanted to prevent workload unrelated pupil responses. We aimed to minimize the pupil light response by implementing our testbed game in grayscale colors only. In the original game the ghosts switch between their personal color and dark blue when switching from chase state to frightened state. Within a grayscale version the differences between the chase state and frightened state are minimized which in turn minimizes the impact of this switch to the pupil light response.

Furthermore, we wanted to prevent pupil responses in reaction to emotionally engaging stimuli that induce the experience of arousal and frustration (see 2.3.1). We therefore made three additional adaptations to the original Pac-Man game. None of the adaptations described below alter the essential gameplay. Based on findings in previous work, we can expect rewarding game events like gathering points to induce arousal to influence the pupillometry features (see 2.3.5). Within the video game Pac-Man we view the feedback of Pac-Man's score, normally visualized on the screen, as the central element with respect to the communication of the gathered points. By excluding this from our customized game we intend to decrease the inducement of arousal and the corresponding pupil response. Since the gathering of bonus fruit is solely related to scoring points and becomes redundant when excluding the feedback of the score, we also refrained from implementing the bonus fruit mechanic.

Additionally, we can expect the experience of failure to induce frustration (see 2.4.3.7) and to affect the pupillometry features. Within the video game Pac-Man we view Pac-Man being captured by a ghost as the main failure event of the game. Since our interest lies in researching the contribution of pupillometry features in a realistic gameplay scenario, we will keep this mechanic included. However, we did choose to exclude the feedback of Pac-Man lives which would be visualized on the screen when following the original Pac-Man game. Pac-Man's death is emphasized by this feedback of Pac-Man's lives. We hope that by excluding the feedback of Pac-Man lives, the sense of failure when Pac-Man dies is minimized, specifically when otherwise the number of lives would have run out.

3.1.2 Chase-, scatter- and frightened state

During both the chase- and scatter states the ghosts take on the role of predator and can capture Pac-Man. These roles swap during the frightened state. During the chase- and scatter states avoiding ghosts is necessary to stay alive, while during the frightened

state Pac-Man can move freely and can even capture ghosts. Therefore, the chase- and scatter states can be viewed as being more difficult compared to the frightened state. Furthermore the level of difficulty of the chase state compared to the scatter state can be viewed higher, because during the chase state the ghosts purposefully chase Pac-Man, while in the scatter state they chase their own scatter tile in one of four corners of the maze. In other words, the frightened state can be viewed as a minor break from being a prey and the scatter state can be viewed as a minor break from being chased. Thereby, we expect players will answer questions regarding the experienced difficulty mainly with the chase state in mind. With respect to predicting the experienced difficulty, our main interest lies in game- and pupillometry data collected during the chase state.

Rounds of Pac-Man can progress in different ways over time, for example different orders of states can occur because eating a super pellet triggers the frightened state. Figure 3.1 displays an estimation of the relative workload during a round of pacman, inspired by time-line analysis (see 2.2.4). We expect that the experienced workload highly depends on the number of ghosts that chase Pac-Man. At the beginning of a run, only Blinky is outside the ghost house in scatter mode, moving to the upper right corner. The player can mainly focus on the collection of pellets, therefore we expect a low workload. Pinky immediately leaves the house and moves towards the upper left corner; an expected increase in workload. Sequentially, Inky and Clyde leave the ghost home after Pac-Man has eaten 30 and 90 pellets respectively. Around the same time that Inky leaves the house, the ghosts enter chase mode, resulting in an increase in expected workload. In one round multiple occurrences of similar curves are to be expected, since after dying the level restarts and Pac-Man will be chased again. The same applies after a temporary break of being chased (frightened- and scatter state and after using a teleport).

Since we are mainly interested in the chase state, it is interesting to heighten the duration of the chase in order to gather more data regarding this state. We heightened the relative duration of the chase state by excluding the scatter state. Although the exclusion of the scattered state does alter the gameplay of Pac-Man, we believe it does not affect the essential gameplay of the prey predator mechanism. Excluding the scatter state simply simplifies the behavior of the ghosts as predators; the ghosts always chase down Pac-Man.

3.1.3 Game round duration limitation

Because of the exclusion of Pac-Man lives, it was necessary to implement another way to limit the duration of a round of Pac-Man. To prevent boredom and fatigue during the experiment, we aimed for a round's duration between 30 to 90 seconds per round. However, to ensure data collection each round had to be long enough that even when

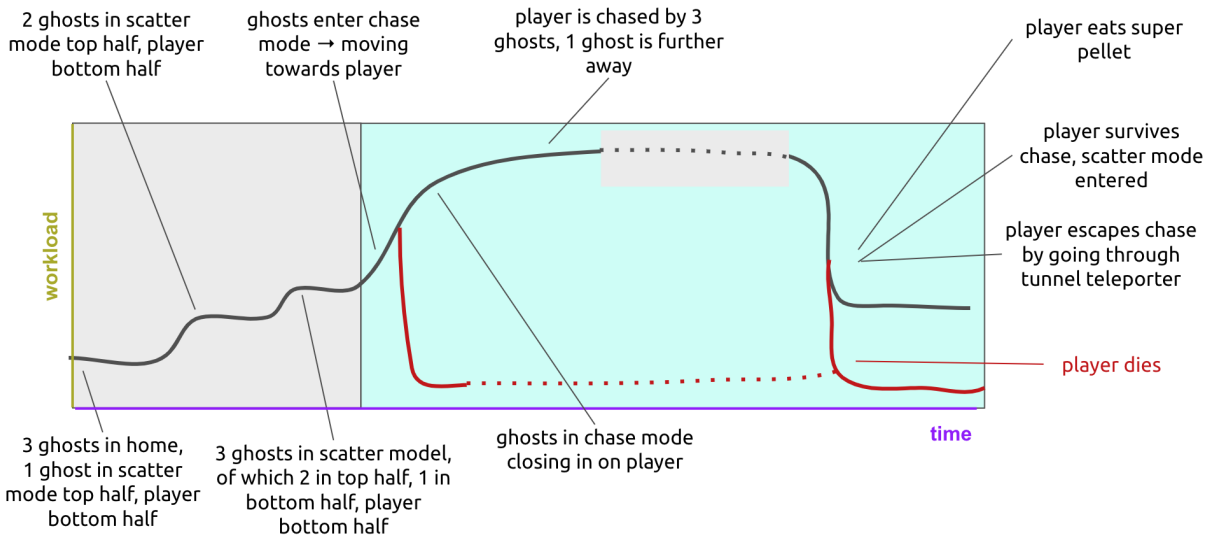

Figure 3.1: Our prediction of the workload during a run in the original Pac-Man. When the number of ghosts that are actively chasing Pac-Man in the maze increases, the predicted workload increases as well. Then, after varying amounts of time different scenarios can occur. Namely, Pac-Man can eat a super pellet, triggering the frightened state, Pac-Man survives the chase state and then the game switches to scatter state, Pac-Man uses a teleport and thereby shakes off the ghosts on his tail or a player can simply die. Then the cycle repeats again.

a player repeatedly quickly dies still enough chase states could be utilized for data collection. When the duration of the round exceeds one minute a round is stopped after dying 5 times.

3.2 Pac-Man challenge hierarchy

Since challenges are directly related to game difficulty, we believe that the challenges in Pac-Man serve as a good starting point for mapping a set of *game features* that can be used to predict the experienced game difficulty and workload. In order to gain a good overview of the challenges in Pac-Man, we mapped the *challenge hierarchy* for the original Pac-Man (also see 2.1.2).

The topmost-level in a *challenge hierarchy* includes the game's victory condition and victory conditions for separate levels [12]. The goal of a level within the game Pac-Man is clear, namely, to clear the maze of dots. However Pac-Man has no designed ending (see 2.4.2) [101]. In other words Pac-Man has no game victory condition that can be viewed as the main goal. We believe this absence leaves room for a player to choose a self determined game victory condition (personal main goal). For example, a player can view *beating a (personal) high score* as winning the game. Or a player might focus on a more achievable and more short-term main goal, for example *collect as many pellets*. We grouped the gathered personal main goals in four self-defined categories, namely

Ghost survival and capture, *Progression goals*, *Collection goals*, *Highscore goals*, see Table 3.1. This mapping of personal main goals enabled us to map the *challenge hierarchy* to a further extent, compared to when we would have focussed on the designed level goal *Clear the maze of dots* only. Worthy of mention, since this mapping is based on reasoning only, we do not expect to have covered the full possible set of personal main goals that a player can adopt within the game Pac-Man. Also noteworthy, within this step we reviewed the original game of Pac-Man, thus a game including the feedback of Pac-Man's lives and the score and the bonus fruit mechanic.

Table 3.1: The gathered personal main goals a player can adopt in Pac-Man, assigned to the four self-defined categories.

Ghost survival and capture	Progression goals	Collection goals	High score goals
Survive as long as possible	Reach a high level	Collect as many pellets	Beat high score
Capture as many ghosts	Set a new personal level record	Collect as many fruits	Set a new personal high score
			Create as many ghost combos

We reviewed the gathered personal main goals a player can adopt (Table 3.1) in order to identify challenges at lower levels. For example, considering the personal main goal *Beat high score*, amongst others we identified the atomic challenges *Eat bonus fruit*, *(Consecutively) eat ghosts*. Some personal main goals can be connected to the same challenges at lower levels, for example the main goal *Capture as many ghosts* also leads to the atomic challenge *(Consecutively) eat ghosts*. Due to this shared connectivity, displaying our set of personal main goals and their connections to other challenges within the *challenge hierarchy* would lead to an unclear overview. Instead, for the sake of clarity, the *challenge hierarchy* displayed in Figure 3.2, only shows a selection of personal main goals.

3.3 Game features

To gather gameplay and game context data (*game features*) that can serve as input features for our random forest models (see 2.1.5) we performed four main steps, namely, gathering *game features* applied in previous work (Step 1), gathering *game features* for challenges within the created *challenge hierarchy* (Step 2), gathering game context *game features* (Step 3) and combining these sets of *game features* (Step 4).

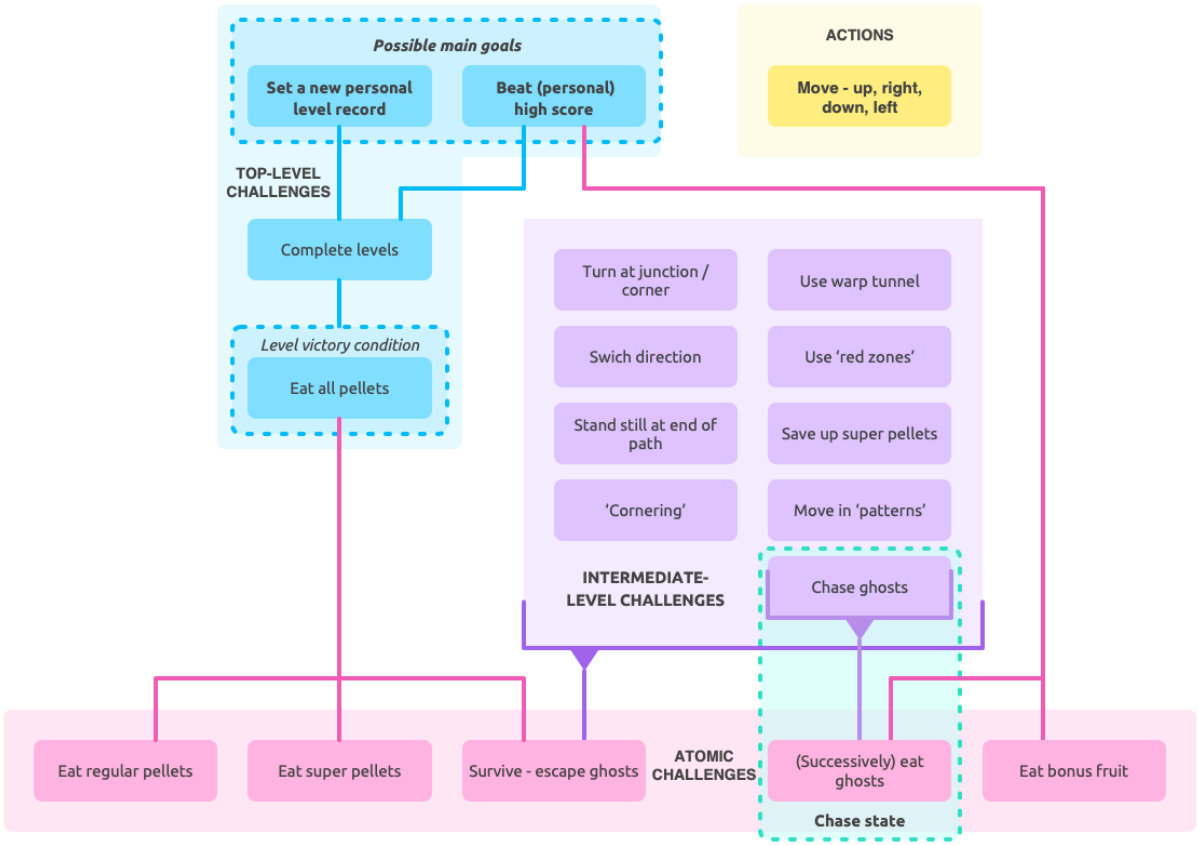

Figure 3.2: The mapped *challenge hierarchy*, displaying the top-level challenges, the intermediate-level challenges and atomic challenges together with the actions a player can perform. Because of the shared connectivity between the personal main goals a player can adopt and other challenges in the hierarchy, this figure only shows a selection of personal main goals for the sake of clarity.

3.3.1 Step 1 - Game features in previous work

In order to gather *game features* that were discussed in previous work in which Pac-Man was used as a testbed, we looked at the previous work discussed in section 2.4.3 (namely: [46], [102]–[109]). See Appendix A for a full overview of the set of gathered features.

3.3.2 Step 2 - Game features abstracted from challenges

Next, we gathered *game features* for each challenge within the created *challenge hierarchy*. Firstly, we reviewed which challenges were irrelevant in our customized game (in which the feedback of the score and Pac-Man's number of lives and the bonus fruit mechanic are excluded), see Appendix B. Sequentially, for each element deemed relevant we mapped possible *game features*. During this process, we identified two different categories, namely event and state, with which we refer to the standard game terms game event and game state. Some examples of event *game features* are # *of direction switches*, # *of teleports*. Examples of state *game features* are # *times chasing ghosts* and *percentual standing still duration*.

The distinction between event and state *game features* enabled us to review the *game features* in a more generic manner. Thereby, we gained a more structured set of gathered *game features*. This in turn enabled us to identify additional event and state *game features* that were not related to the *challenge hierarchy*. Namely, *game features* related to the events *death* and *Pac-Man visits a tile* (event), *Ghost visits a tile* (event, per ghost) and state *Pac-Man is being chased*. See Appendix D for all gathered event and state *game features*.

Furthermore, the distinction between events and states was used to design separate data set formats in order to store occurred events and states. These data sets formed the format for the csv output data. In our testbed game both the occurred events and states are stored in separate csv output files per game session. See Appendix C for an overview of the generic event and state data sets.

3.3.3 Step 3 - Game context game features

To map game context *game features*, we first reviewed which elements of the game change over time and which can differ per game session. The game attributes that change within a game are the game state (e.g. chase mode or frightened), the positions of all five entities, the number of existing regular pellets and each of the four super pellets existence. The difficulty setting can differ per game session. This data formed the essential game context data set, see Appendix C.

Sequentially we reviewed which related overarching elements depend on this essential game context data set in order to identify game context *game features*. For example, we identified the *average number of ghosts that are in chase mode* as a game feature (excluding ghosts that are in the ghost house). Although this value can be calculated based on the ghosts' positions and game state, it was easily extracted from ghosts' in-game attributes and therefore added to the game context data set, see Appendix C.

Other examples of identified game context *game features* are distance related *game features*. For example, we believe a game session in which the ghosts are mainly spread out in the maze results in a different game experience for the player compared to a game session in which the ghosts move close together. Due to this we identified the *average inner distances between all ghosts* and the *average distance between the ghosts and Pac-Man*. Another example, when viewing the ghosts as a group, the distance between the group viewed as a whole and Pac-Man could be of influence to the game experience. We identified the *average distances between the ghosts and their centroid* and the *average distance between Pac-Man and the ghosts' centroid* as *game features*. For a full overview of all game context *game features* see Appendix D.

3.3.4 Step 4 - Combining all *game features*

At this step we noticed that the resulting set of *game features* gathered in step 2 and 3, was far more extensive compared to the gathered set of *game features* applied in previous work (step 1). Only a few relevant *game features* from our overview of *game features* applied in previous work were not yet covered in the set of *game features* resulting from step 2 and 3. Namely, *level completion time* [109], *entropy of the ghosts' cell visits* (see 2.4.3.1) [102], [103] and *measure of flow* (see 2.4.3.2) [104]. Additionally, Yannakakis and Hallam [102], [103] used the sum of the delta distance between a ghost and Pac-man as input for their online learning fitness function within the training process of the ghosts' AI, in order to promote ghosts that move toward Pac-Man. We included this value as a game feature in our experiment. For a full overview of all *game features* see Appendix D.

3.4 Mapping adaptive game components

To gather game- and pupillometry data to predict the experienced difficulty, we let participants play multiple rounds of Pac-Man with different levels of game difficulty (within subject design, see 4.2.1). The difficulty was altered by adapting game components that affect the level of challenge (*adaptive game components*) (see 3.5). Some component adaptations affect the difficulty during the full period of the chase state, for example the adaptation of the *overall speed*. Some component adaptations will not always affect the difficulty during the full period, for example an addition of teleports

at the top and bottom of the maze. Such additional teleports would lower the difficulty because they offer the player another escape route when being chased by the ghosts. However, this additional escape route would only be relevant to a player when Pac-Man is near the teleport and in case a player is aware of this escape strategy. Therefore, this example illustrates that some component adaptations would not necessarily result in a different experienced difficulty.

In identifying and selecting *adaptive game components* to be implemented in our testbed game, we went through a systematic process consisting of the following seven steps, to a further extent discussed in the rest of this section. As mentioned we focused on the impact of difficulty during the chase state, our main state of interest (see 3.1.2).

We started by mapping the *adaptive game components* applied in the original Pac-Man game (step 1, also see 2.4.2). Then we mapped *adaptive game components* applied in previous work Pac-Man testbed games (step 2, also see 2.4.3). Step 3 consisted of mapping the *adaptive game components* by reviewing the game difficulty aspects intrinsic skill, stress and power provided in Pac-Man (also see 2.1.2). We mapped *adaptive game components* by looking through the lens of our custom selection of four categories of adaptive game components, namely *attributes*, *behavior*, *game level/-world layout* and *events* (step 4, also see 2.1.3). We then combined all identified and relevant *adaptive game components* from the first four steps (step 5). We evaluated and prioritized the resulting list (step 6) of *adaptive game components* based on multiple preconditions and valuations. Finally, we selected three high ranking *adaptive game components* of the resulting prioritized list (step 7).

3.4.1 Step 1 - Adaptive game components in original Pac-Man game

We first mapped *adaptive game components* in the original Pac-Man game. Due to the clear overview presented by Pittman [101], this step turned out to be quite straightforward. We thoroughly reviewed Pittman's [101] overview in order to map *adaptive game components* in the original Pac-Man game. For the resulting set of *adaptive game components*, see Appendix E.

3.4.2 Step 2 - Adaptive game components in previous work

To map *adaptive game components* applied in Pac-Man testbed games in previous work, we looked at the previous work discussed in section 2.4.3 Game difficulty of and game AI in Pac-Man (namely: [46], [102]–[109]. For each identified *adaptive game component* we reviewed its relevance in the context of this work. For example, for the identified *adaptive game component ghost behavior* we found different strategies. For some of these strategies we found clear information in previous work about the ghost performance specifically related to game difficulty. We viewed these as relevant options for the

ghost behavior adaptive game component while the others were marked as irrelevant. See Appendix F for the outcome of this step.

3.4.3 Step 3 - Review of intrinsic skill, stress and power provided

We also identified *adaptive game components* by reviewing Pac-Man from the perspective of the game difficulty aspects *intrinsic skill*, *stress* and *power provided* (also see 2.1.2). In this section we will share our reasoning with respect to each of these three game difficulty aspects.

3.4.3.1 Intrinsic Skill

As mentioned in 2.1.2, *Intrinsic skill* refers to the level of skill needed to overcome a game challenge leaving out any element of time pressure [12]. Since Pac-Man is a continuous game, meaning all entities perform actions simultaneously, viewing Pac-Man without any elements of time pressure, can only result in an imaginative turn-based version of Pac-Man. In a turn-based game entities perform actions in turns instead of performing actions continuously. Thus, in order to view the game Pac-Man without the element of time pressure, we have to imagine a turn based version of Pac-Man. Within this imaginative turn-based version of the game Pac-Man we view the outcome of a player's turn as the choice to which adjacent tile Pac-Man should move. In such a turn-based version of Pac-Man a player can take all the time necessary to decide on Pac-Man's next direction. This enables a player to carefully plan and execute a route through the maze, in order to escape / capture ghosts (in the chase / frightened state respectively), to collect pellets and to collect bonus fruit. Here, we view navigation through the maze, consisting of route planning and execution, as a central skill in the game Pac-Man. Thereby, *intrinsic skill* of the original Pac-Man game can be viewed as navigation as well. Examples of *adaptive game components* that we expect to be of influence to required level navigation skill are *maze topology*, *regular pellet locations* and *bonus fruit locations*. For a full overview of the *adaptive game components* of influence to the required navigation skill see Appendix G.

3.4.3.2 Stress

As mentioned in 2.1.2, when a challenge includes time pressure, the factor *stress* is of influence to the absolute difficulty [12]. Since we aim to identify *adaptive game components* that can possibly be relevant for our testbed game by viewing aspects related to the game difficulty, we are not interested in mapping the absolute difficulty of Pac-Man itself. Therefore, we leave absolute difficulty out of scope. However, we do review elements of Pac-Man that are related to time pressure and thereby of influence to the factor *stress*.

We believe that the time-pressure a player experiences during the chase state is mainly related to the ghosts moving continuously and trying to capture Pac-Man. Here, the speed of the ghosts matters. When comparing a higher ghost movement speed to a lower speed, a higher speed leaves the player less time to plan and execute a route through the maze. Therefore, we identify *relative movement speed of ghosts (compared to movement speed Pac-Man)* as adaptive game component. Additionally, when comparing a higher Pac-Man movement speed to a lower speed, the time window to perform a move action at a corner or junction is shorter at a higher speed. Since an alteration of Pac-Man's movement speed also alters the *relative movement speed of ghosts (compared to movement speed Pac-Man)*, instead we indicate an adaption of all movement speeds, namely *overall movement speed* as adaptive game component.

We identify four scenarios during which a player can experience the opportunity to collect pellets without being chased for a short moment of time, which can be experienced as a less stressful moment. Firstly, at the start of a game and after a loss of a life, the ghosts wait in the ghost house before entering the maze. Similarly, after a ghost is captured, it returns to the ghost house, sometimes waiting shortly before entering the maze again in chase mode. Thirdly, during the scatter game state ghosts traverse a repeating path in their own corner aiming for their personal scatter tile. Lastly, during the frightened game state, a player is granted a small window of time to chase and eat ghosts. The waiting time in the ghost house in the first two scenarios depends on the *ghost dot limit* and *ghost personal global dot counter threshold* (see 2.4.1) and which we therefore both identify as *adaptive game components*. Additionally, we identify *scatter game state duration* and *frightened game state duration* as *adaptive game components*.

When nearing the end of the frightened state, a blinking animation of the blue ghosts informs the player that the frightened state is almost over. We imagine this blinking animation can induce a heightened experience of time pressure. Therefore, we also identify *blinking ghosts animation duration* as an adaptive game component. See Appendix G for the overview of the *adaptive game components* related to *stress*.

3.4.3.3 Power provided

As discussed in 2.1.2, *power provided* refers to the power the game provides to the player [12]. We argue that the relative movement speed of Pac-Man compared to the ghosts can be viewed as a *power provided* to Pac-Man. Since, when Pac-Man moves faster than the ghosts, Pac-Man can more easily escape the ghosts during the scatter and chase state and can more easily capture ghosts during the frightened state. In turn, completing the level by collecting all pellets will be more easy. Also with respect to *power provided* we identify *relative movement speed of ghosts (compared to movement speed Pac-Man)* as an adaptive game component.

We can also view the warp tunnels in the perspective of *power provided*. Only ghosts receive a movement penalty within the warp tunnels. Therefore, when ghosts tail Pac-Man through these tunnels, the relative movement speed of Pac-Man compared to the ghost(s)s is heightened. Additionally, when ghosts are near Pac-Man at one of the two sides of the maze in the center, Pac-Man can use the warp tunnels to teleport to the other side. Ghost do not use the teleport in such a controlled manner, they teleport after entering a warp tunnel but do not use the teleport mechanism in their path planning. Therefore, we view the teleport mechanism as *power provided* to the player. Successively, we identify *the ghost speed penalty in warp tunnels*, *number of warp tunnels* and *number of teleports* as *adaptive game components*.

When Pac-Man eats a super pellet, the frightened state is activated and the roles of prey and predators are swapped, enabling Pac-Man to hunt down ghosts. Within this state Pac-Man is given the opportunity to collect pellets without being chased by ghosts. We therefore view super pellets as a *power provided* to the player and identify *the number of super pellets* as *adaptive game components*.

Finally, when Pac-Man is captured by ghosts, Pac-Man loses a life. When no lives are left, the player is game over. Pac-Man's lifes can thus be viewed as a *power provided* to the player. We identify the *number of Pac-Man lifes* as an adaptive game component. See Appendix G for an overview of the *adaptive game components* related to *power provided*.

3.4.4 Step 4 - Review of adaptive game components categories

We reviewed the mapped challenges in our mapped *challenge hierarchy* (see 3.2) through the lens of each of the four *adaptive game components* categories (*attributes*, *behavior*, *game level/-world layout*; and *events*). For example, for the atomic challenge *(Consecutively) eat ghosts* we identified the *adaptive game components number of ghosts*, *ghost speed*, *Pac-Man speed* and *Frightened state duration* within the category *attributes*. Furthermore, for this challenge we identified *ghost behavior during frightened game state* within the *behavior* category. Additionally, within the *adaptive game component* category *game level /-world layout* we identified *maze topology*, *maze complexity* and *captured ghosts relocation*. For a full overview of the within this step mapped *adaptive game components* see Appendix H.

3.4.5 Step 5 and 6 - Overview of all adaptive game components

The outcome of the first four steps was merged in a combined list with all identified relevant *adaptive game components* (step 5). Additionally, we performed three steps in order to prioritize the gathered *adaptive game components*, which are discussed to further extent below. In short, we reviewed all *adaptive game components* with respect to the DDA requirements, their impact on the difficulty during the chase state, the expected impact to the game difficulty and the expected implementation complexity.

3.4.5.1 DDA Requirements

We reviewed the *adaptive game components* with respect to the DDA requirements. We formulated these DDA requirements shared by Andrade et al. (2005) (also see 2.1.1) as follows: "1. A DDA system implementing this adaptation is able to quickly identify and adapt to a player's initial level"; "2. A DDA system implementing this adaptation is able to fastly and closely track and adapt to the player's improvement or falling level."; "3. The adaptation is unperceivable by the player and the game must remain believable.". For each *adaptive game component* we indicated if it meets the requirement, indicated by a number in the range [1, 5] referring to [*not probable*, *very probable*] respectively.

Noteworthy, during this step we noticed that for some *adaptive game components* the timing of the adaptation matters. In these cases we reviewed three options with respect to timing, namely *immediately*, *after Pac-Man dies* and *after level completion*. For example, we expect that an alteration of the number of super pellets during the gameplay is quite perceivable and thus does not meet the third requirement. However, after the completion of a level we expect it will be noticeable but a player can also assume it is part of the setting newly reached level design. Considering the timing of the *adaptive game components* led to a different ranking with respect to the DDA requirements.

Finally, we excluded the *adaptive game components* that scored either *not probable* or *somewhat improbable* for at least one of the DDA requirements. For the remaining *adaptive game components* we calculated the mean of the three assigned DDA requirement values. The results can be found in Appendix I.

3.4.5.2 Impact on difficulty

Firstly, since we expect participants to rank the experienced difficulty mainly based on their experience during the chase state (see 3.1.2), the *adaptive game components* for which we do not expect any to impact the difficulty during the chase state were excluded for further reviewing. Next, we indicated the expected impact to the game difficulty in range from [1, 5] referring to a [*low*, *high*] impact respectively. Noteworthy, for some *adaptive game components* reasoning about the possible impact on the difficulty did not lead to a clear estimation. We marked these as "unpredictable". An example is the *adaptive game component* "dynamic starting location: in quartile with the most pellets vs. in quartile with the least pellets". We reasoned as follows.

> "*In case all the pellets from some of the quartiles are collected, positioning Pac-Man's start location in the quartile with the most pellets saves Pac-Man a trip to this quartile. However, in our experience it is often the case that near the end of a level multiple quartiles still contain a few pellets. In this case, when starting in the center of the quartile with the*

most pellets, Pacman still needs to navigate to the other quartiles with only a few pellets left in order to complete the level. Therefore, we can not estimate if it actually matters if Pacman starts at the quartile with the most pellets or in a quartile with less pellets. Instead of aiming to make the game more easy by positioning Pac-Man within the quartile with the most pellets, we can also imagine the opposite, namely making the game more easy by positioning Pac-Man within the quartile with the least number of pellets. Because, locating a player inside the quartile with the least number of pellets allows her to focus on finishing this quartile first enabling her to afterwards focus on a smaller part of the maze, which in turn could heighten her focus and make the game slightly easier."

We excluded all *adaptive game components* that scored *low* and those which we indicated as unpredictable regarding their impact to the difficulty. The results can be found in Appendix I.

3.4.5.3 Expected implementation complexity

In order to map our expectations with respect to the implementation complexity of the *adaptive game components*, we assigned a number in the range [1, 7] corresponding with [*very low*, *very high*] complexity respectively. We used the range [1, 7] instead of [1, 5] simply because we noticed a higher granularity was beneficial for our mapping. Due to the limited scope of the project, we excluded *adaptive game components* that scored 5 or higher, thus for which we expected a relatively high implementation complexity. This resulted in only one exclusion, namely the exclusion of "switching between multiple neural networks to control ghosts, with different performance, dependent on the UCT simulation time of which the data is used to train the neural networks", an identified *adaptive game component* applied in the research of Li et al. [106]. The other remaining *adaptive game components* were thus deemed doable within the scope of this research. The results can be found in Appendix I.

3.4.6 Step 7 - Selection adaptive game components

Appendix J displays the resulting list of *adaptive game components*. This list is sorted based on to which extent the *adaptive game components* were indicated to meet the DDA requirements and the estimated impact on difficulty from high to low. We selected the top 4 *adaptive game components* for our testbed game, whereby the first two *Ghosts - movement speed* and *Pac-Man - movement speed* are combined into one, namely *relative movement speed of the ghosts compared to Pac-Man*. This resulted in a set of three *adaptive game components*. Namely, *relative movement speed of the ghosts compared to Pac-Man, increase vs. decrease speed* (*relative speed*), *game speed - overall movement speed, increase vs. decrease speed*(*overall speed*), *Ghost - chase behavior*, switching between the three fixed ghost chase strategies as described in Yannakakis and Hallam [102], [103], namely *random*, *followers* and *near-optimal* (*ghost behavior*).

3.5 Difficulty settings

The three selected *adaptive game components* were implemented in our testbed game. For each of the three components we created an *easy*, *medium* and *hard* difficulty setting. These three settings of the *ghost behavior adaptive game component* correspond to the three fixed ghost chase strategies *random*, *followers* and *near-optimal* respectively [103]. The three settings of the *relative speed* and of the *overall speed* were set and tuned in an iterative manner. The settings were tested multiple times by both an inexperienced gamer and an experienced gamer who plays quickly paced games weekly. Noteworthy, these two components' settings were further tweaked during the pilot study (see 4.1).

The resulting difficulty space is displayed in Figure 3.3 and consists of 27 different combinations of difficulty settings. Since an experiment with 27 rounds of Pac-Man would be too lengthy duration wise, we selected a smaller set of combinations. First, we excluded the combinations with opposite settings per axis, thus sets of settings with both *hard* and *easy*, for example *ghost behavior*: *hard*, *overall speed*: *hard* and *relative speed*: *easy*. This resulted in a diminished set of 15 combinations. Next, since we still deemed 15 rounds as too much, we further downscaled our set. Because we can imagine players getting frustrated when ghosts move quicker compared to Pac-Man, we found this setting of least interest. Therefore, we viewed the central set of combinations to consist of different settings regarding the *ghost behavior* and *overall speed* and the *relative speed* as a way to further intensify the most difficult settings. In other words, within 7 of the 9 combinations the *relative speed* was set to *medium* and only in two combinations too *easy* and *hard* to further intensify the combination of *ghost behavior*: *easy*, *overall speed*: *easy* and *ghost behavior*: *hard*, *overall speed*: *hard* respectively. The numerical order as displayed in Figure 3.3 expresses our expectations regarding the chronological increase in terms of the difficulty level.

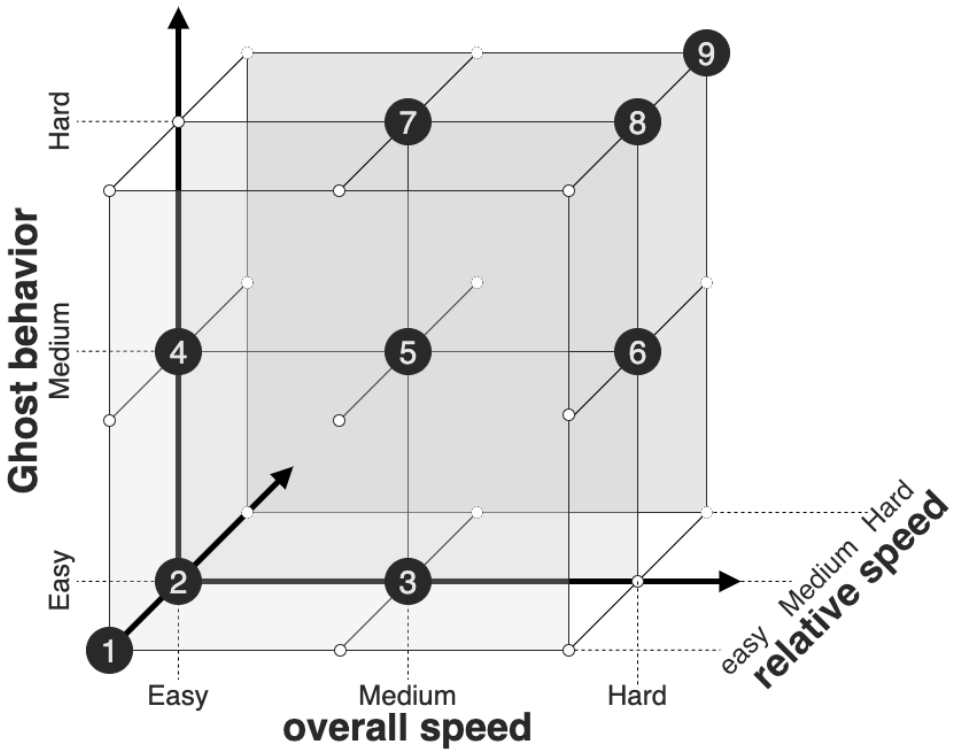

Figure 3.3: The difficulty space as implemented in the testbed game, consisting of three difficulty settings *easy*, *medium* and *hard* for the three *adaptive game components ghost behavior*, *relative speed* and *overall speed*. The three settings of *ghost behavior* in chronological order correspond to the three fixed ghost chase strategies *random*, *followers* and *near-optimal* respectively [103]. The three settings of the *relative speed* and of the *overall speed* were set and tuned in an iterative manner. A set of nine combinations was selected, in this figure depicted by the nine numbers of which the numerical order corresponds to the expected chronological difficulty increase.

4. Method

A user study was performed in which participants played multiple rounds of Pac-Man with different levels of difficulty in order to gather data and create a dataset. Within the user study the participants played a practice round of Pac-Man in order to allow the participant to familiarize with the game and the controls. Sequentially the participants played nine rounds of Pac-Man at different difficulties. For each round, gameplay, game context and pupillometry data were collected together with participant responses regarding game experience in order to form a dataset. Further below we elaborate on the experimental design and the procedure of the experiment.

Following the experiment, a statistical analysis was applied to analyze the relation between the features in the dataset and participant's answers regarding game experience. Multiple random forest models were trained on subsets of the created dataset to predict the participants' game experience responses. The accuracy of these trained classifiers was analyzed and feature importance analysis was performed in order to map the added value of the used pupillometry, gameplay and game content features.

4.1 Pilot study

Before arriving at the final experimental setup as described in the following subsection, 4.2.4, we performed a small pilot study in order to tune the initial experimental setup we had in mind. Within this pilot study we performed the full procedure three times fully and one time only partly by focussing on the level of clearity questionnaire questions only. We applied the PMI thinking tool of Edward de Bono [110], which enabled us to perform a structured scan[1], see Appendix K.

Because the first round of our pilot study already provided us with some useful insights, we decided to immediately alter minor parts in order to be able to test the slightly adapted experiment again. For an overview of the outcome of the PMI and the minor changes to our experiment setup see Appendix L. Changes made to the testbed game based on the pilot study findings were the increasing of the absolute speed of

[1] De Bono's [110] PMI is an attention-direction tool with which you first focus on gathering the Plus Points (P), then toward the Minus points (M) and lastly the Interesting points (I) [110]. The first step consists of gathering these three types of points, which should be done in a disciplined manner in only two to three minutes [110]. Sequentially, within the second step of the PMI thinking tool one can observe and react to the outcome of the first step [110].

the game and the *relative speed* of the ghost with respect to Pac-Man. Furthermore, in the game sessions with the highest speed settings the controls did not always react fast enough, we therefore heightened the frame rate from 30 fps to 60 fps which solved this issue.

Other adaptations mainly concerned survey questions and information shared with the participants. For example, some changes regarded sentence constructions to improve clarity of questions. Also, we switched from a 5-point likert scale to a 7-point likert scale. Our initial choice for a 5-point likert scale was based on findings that a 5-point likert scale can be experienced as more quick and easy to use compared to a 7-point likert scale [111]. During the pilot study we noticed "For the two questions about the experienced difficulty and effort cost, most given answers were located around the middle of the likert scale, slightly more answers at the left side". We therefore choose to switch to the 7-point likert scale in order to offer a more granular scale, which has also been found to be more accurate [112].

4.2 Experimental design

4.2.1 Experimental conditions

We applied a within subject design in which each participant played nine rounds of Pac-Man with different difficulty settings, excluding the practice round. In order to minimize the influence of a possible learning curve, we applied a simplified version of the latin square. Due to the nine different difficulty levels a full application of the latin square would have required a multiple of nine participants. We expected a sample size of 20 and chose a suitable strategy mimicking the rotation of conditions of a Latin Square.

Based on our own experience with the selected nine combinations of difficulty settings (see 3.5), we expected that the settings of the *overall speed* would be most consciously noticed by the participants. We therefore constructed the nine difficulty combinations from this perspective. See Figure 4.1, we can identify three groups with respect to the *overall speed* setting, namely (1, 2, 4), (3, 5, 7) and (6, 8, 9). When starting in the middle of the nine sets and then continuing to a next set of difficulty settings by adding 4 (wrapping values > 9), we get the order 5, 9, 4, 8, 3, 7, 2, 6, 1. Within this order we can distinguish three groups of three combinations, each group starting with a different *overall speed* setting ((5, 9, 4), (8, 3, 7) and (2, 6, 1)). Furthermore, the sequence contains a consistent difference between the sequential combinations, the difference is always 4 or 5 steps. There is one occurrence in this sequence where a combination follows another that uses the same difficulty setting for *overall speed*, namely 3 and 7. However, since these two combinations use *easy* and *hard* for *ghost behavior*, we expect the difficulty to

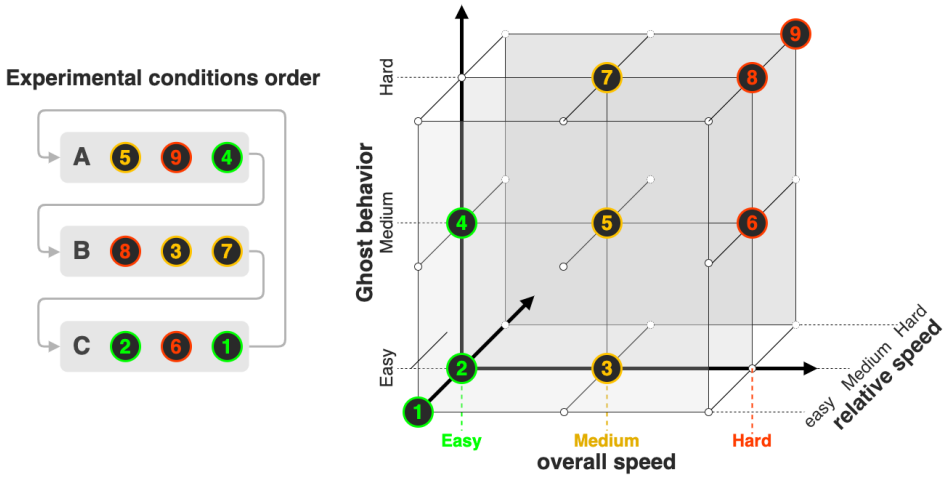

Figure 4.1: On the left, the order of the experimental conditions spread amongst the three participant groups A, B and C. With this we aimed to mimic the rotation of a Latin Square. To illustrate, the order of group B is 8, 3, 7, 2, 6, 1, 5, 9, 4. Here, the condition numbers refer to the difficulty settings that consist of a combination of three settings for *ghost behavior*, *relative speed* and *overall speed*, displayed in the difficulty space on the right (also see Figure 3.3).

be sufficiently divergent.

The participants were divided in three groups (A, B and C), whereby each group started at a different subset of three combinations. Thus, all participants played nine rounds with the nine difficulty combinations, starting at difficulty combination 5, 8 or 2, see the experimental condition order in Figure 4.1. For example, a participant assigned to group B played nine rounds with the difficulty sequence 8, 3, 7, 2, 6, 1, 5, 9, 4. Noteworthy, the practice round was always played at level 5, in which all three *adaptive game components* were set to *medium*. Thereby, the difference between the practice level and either one of the three starting levels was kept lowest.

4.2.2 Questionnaire

Three questionnaires were composed, namely a demographics questionnaire, a short questionnaire asked after each round of Pac-Man which consisted of three questions regarding the experienced difficulty and a questionnaire at the end with questions related to the overall game experience.

4.2.2.1 Demographics questionnaire

The demographics questionnaire consisted of questions regarding the participants' gender and age. Since tiredness can affect pupil size indirectly (see 2.3.1), the questionnaire also included the shortened fatigue questionnaire (Verkorte Vermoeidheid Vragenlijst: VVV) [113] in order to capture the degree of fatigue of participants. Additionally, we wanted insight into the participants' skill level. In order to sort partici-

pants into gamer skill categories, often "hours per week" is used as an indicator [114]. However, self-chosen gaming skill questions appear to be a stronger metric compared to gaming frequency [114]. We therefore implemented questions regarding both gaming frequency to capture previous familiarity with video games and questions regarding gaming skill in order to capture expert/non-expert demographics. Both types of questions were asked with respect to gaming in general, arcade games and Pac-Man. See Appendix M for the demographics questionnaire.

4.2.2.2 Pac-Man experience questionnaire

Three questions were asked after each round of Pac-Man asked the participants to estimate the experienced challenge, the felt competence and the invested workload. The first question, regarding the experienced challenge, was based on the MiniPXI Challenge construct question "The game was not too easy and not too hard to play" with a 7-point likert scale ranging from *strongly disagree* to strongly agree [115]. Since we were interested in the experienced challenge instead of if the experienced challenge suited the participants, we adapted this statement to "The last game was very hard to play." (CHAL).

The second question reviewed the experienced competence. The MiniPXI Mastery construct bears resemblance to related constructs of Competence in Self-Determination Theory [115]. The statement "I felt I was good at playing this game" was adapted to "I felt I was very good at playing the last game." (COMP) in order to emphasize the question referred to the last round of Pac-Man. Noteworthy, since the MiniPXI Mastery construct was recognized as a multidimensional construct, the single-item measure may serve to generally screen for Mastery, but not accurately reflect its multi-dimensionality [115].

The third question reviewed the experienced workload, since in the current work we focus on workload related pupil responses. To the authors knowledge, no standardized question exists to capture an estimation of the invested workload in the context of games. Instead we based our question on a statement from the in-game version of the Game Experience Questionnaire [116], namely "I had to put a lot of effort into it" of the component Challenge. We adapted this statement to "I had to put a lot of effort into the last game." (WRKL), again in order to emphasize the question referred to the last round of Pac-Man.

Noteworthy, we are aware that comparing rating-based responses across participants neglects the existence of interpersonal differences regarding the rating scale. In comparison, pair-wise ranking minimizes the effects of self-reporting subjectivity biases [17] and could therefore be viewed as more suitable. However, we chose a rating-based experience questionnaire for two reasons. Firstly, we consider the duration of the game rounds of about one minute to be too long for pair-wise ranking. Secondly, we view the implementation and corresponding processing of pair-wise ranking is out of the scope of this project.

4.2.2.3 Overall game experience questionnaire

At the end of the experiment the participants were asked to fill in one last questionnaire. This questionnaire consisted of statements about the participants' experience and gaming behavior in all games of the entire experiment. These questions were constructed based on the relevant challenge hierarchies top-level- and intermediate-level challenges in order to gain insight in the participants' possible self determined game victory conditions and applied strategies respectively. For example, the intermediate-level challenge *Save up super pellets* was reflected in the question "I consciously saved up super pellets so that I could use them later at a convenient time.". Another example is the intermediate-level challenge *Stand still at end of path* which was covered in the question "I consciously tried to shake off ghosts by standing still at the end of paths.".

Additionally, we added questions that concerned the speed of the game and the effect of the ghosts, both related to the implemented *adaptive game components*. The two questions regarding game speed were "I found the slow paced levels more easy compared to the quick levels." and its mirrored version "I found the quick paced levels more difficult compared to the slower levels.". The questions were mirrored to keep participants engaged. The questions regarding ghosts and the experienced *stress* were "I found the game stressful when the ghosts were near me." and "I found the game stressful when the ghosts were enclosing me.". We expected that the *measure of flow*, which percentually represents the number of ghosts nearby Pac-Man [104], to turn out to be an important game feature in predicting challenge and workload. Therefore, we were interested in the participants' experiences regarding ghosts nearing and enclosing Pac-Man. Noteworthy, we gathered answers to these questions since we expect that these can offer valuable insights, amongst others in perspective to the *challenge hierarchy*. However, due to the limited scope of the current work we postpone detailed analysis of the answers to future work.

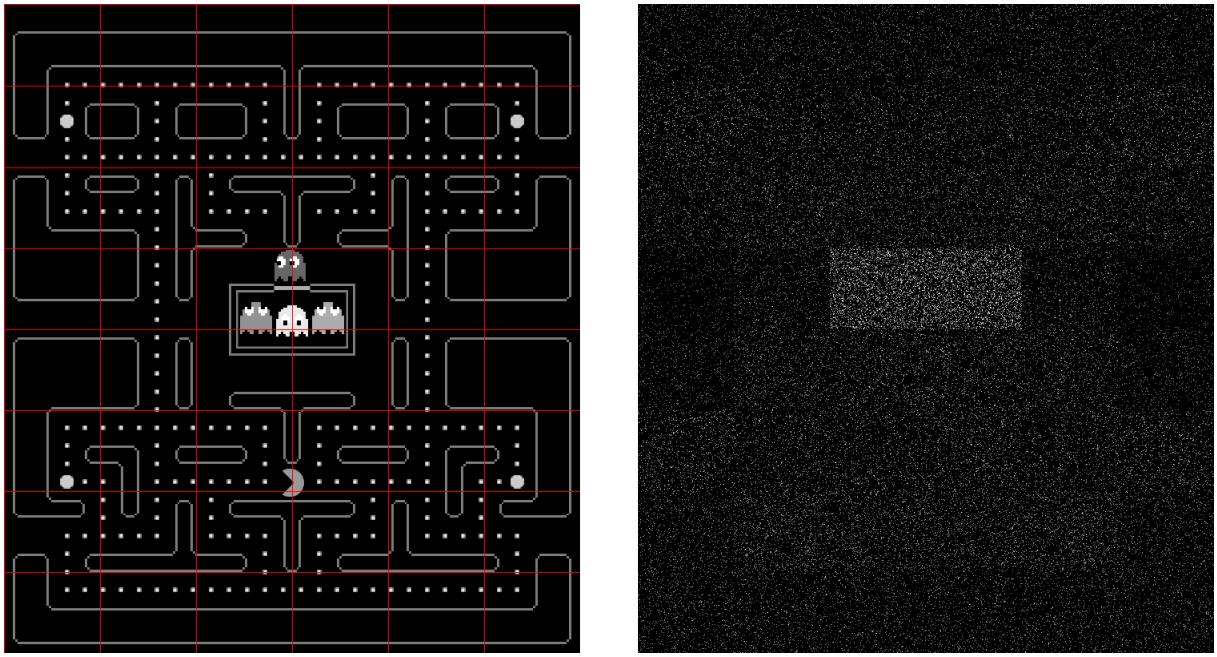

Figure 4.2: On the left, a screenshot of the maze taken at the start of Pac-Man round. The added red lines on top indicate the 6x8 grid used to create the grid scrambled image that was used as baseline image, displayed on the right.

4.2.3 Data collection

4.2.3.1 Baseline recordings

Each round of Pac-Man was preceded with a baseline period of 8 seconds during which the participants watched a grid scrambled image (also see 2.3.3). Custom written methods in python were used to scramble an image of the beginning of a level of our Pac-Man testbed[2]. We applied a grid of 6 columns and 8 rows, see the left of Figure 4.2 with red lines indicating the grid and the resulting grid scrambled image on the right.

4.2.3.2 Pac-Man data collection

During all rounds of Pac-Man, including the practice rounds, we collected gameplay-, game content- and pupillometry data (see Appendix C). While our focus with respect to data analysis lay on the chase state, we also recorded data during the frightened states for possible later reviewing. In order to differentiate between the states, the game data also includes the game states. To synchronize the pupillometry data coming from the eye tracker server and the gameplay- and game content data coming from the game engine, we included clock time ticks. This synchronization was verified during the final development phase.

[2]To generate a scrambled image, the image data was loaded and stored with the python image library Pillow, also referred to as PIL.

4.2.4 Experimental procedure

The experimental supervisor (*supervisor*) used a checklist to ensure each step of the procedure was executed, see Appendix N. Upon arrival at the experiment location, the participants were asked to take a seat behind the desk in the darkened room. The *supervisor* explained that the room was darkened because recordings would be made of the pupil size and she shared that no video was recorded, only data would be stored.

Next, the participants read written information on screen regarding the experiment and were asked for their consent. Sequentially, they filled in the demographics questionnaire (see Appendix M). After this the *supervisor* switched between the Unity application and eye tracker software in order to calibrate. The calibration process itself was discussed on forehand and the participant was asked "Are you comfortable? Try to maintain this position in terms of distance from the camera while playing.". The *supervisor* further explained that some movement was alright. This also became visible by looking at the footage from the eye tracker camera, which is displayed in the window of the eye tracker software, participants could see their eyes move offscreen when they moved approximately more than 15 cm forwards and 15 cm backwards.

After the calibration the *supervisor* provided a recap of the essential mechanics in Pac-Man (all participants were at least slightly familiar with Pac-Man), also see the items in Appendix N below "Practice". Furthermore, the *supervisor* explained the purpose of the practice round, namely to get used to the Pac-Man gameplay and the controls. Sequentially, the participants read written information on screen about the scrambled image (also shown on screen) that would be displayed before each round. The participants were asked in this text to simply relax and only look at the grid scrambled image. After the practice round the *supervisor* went through the three survey questions with the participants and asked the participants to take enough time after each round to (re)read and answer the questions. Furthermore, the participants were explicitly asked to keep the order of the answers in mind, with the negative answer at the left and the positive at the right of the answer scale.

Next, another calibration was performed after which the participant continued with the nine rounds of Pac-Man. Each round was preceded by a progress bar showing the progress of the nine rounds and was preceded by baseline recording of 8 seconds displaying the grid scrambled image. Each round of Pac-Man was followed up with the three questions regarding the game experience within that round (see Appendix O). See Figure 4.3 for these four sequential screens. At last, a final questionnaire was presented to the participants, containing game experience questions regarding all rounds of Pac-Man together (see Appendix P).

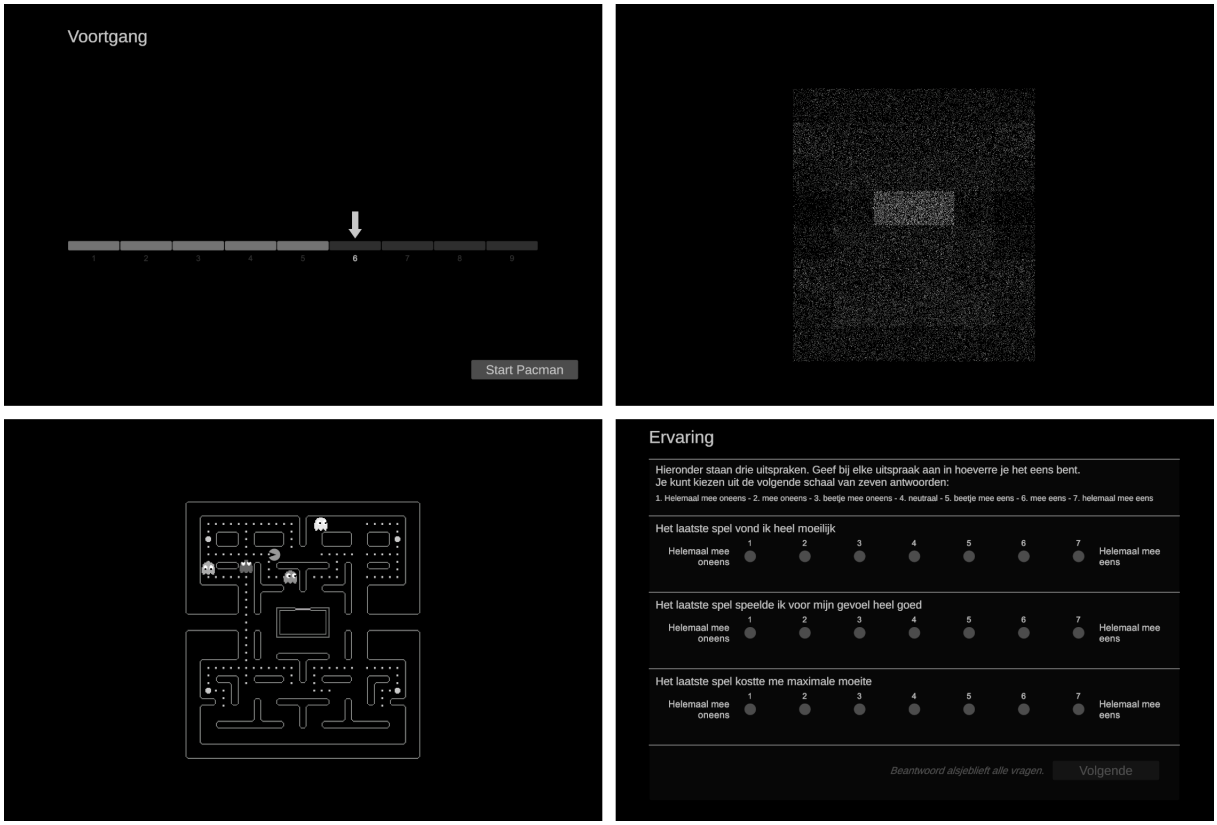

Figure 4.3: The four screens that were repeated sequentially for the nine rounds of Pac-Man central to the experiment. slightly zoomed in for clarity. Starting with a progress bar (top left), continued by the grid scrambled image (top right), the Pac-Man game (bottom left) and the three game experience questions (bottom right).

4.3 Materials and apparatus

The relatively low-cost eye-tracker Gazepoint GP3-HD was used with a 60 Hz sampling frequency. We chose this eye-tracker hardware because of its low cost and the accompanying open standard API. The testbed game was implemented in Unity and was extended with a custom build module that connected with the server of the Gazepoint GP3-HD eye-tracker in order to retrieve and record pupil data. Data retrieval was turned on and off before and after each baseline recording and game session. A 27 inch screen was used, with a resolution of 2560 x 1440 on a Windows desktop computer. The experimental setup is displayed in Figure 4.4.

To avoid pupil dilation caused by fluctuations in naturally present daylight, we set up our hardware in a completely darkened room. A RGB light bulb was set to a low intensity of red light to enable vision in the room. Since high intensity blue light can result in a sustained constriction for many minutes, due to the post-illumination pupil response [80], we chose a red color which does not contain blue light.

Apart from the calibration of the eye tracker, the full experiment was implemented in Unity, consisting of shared written information, multiple questionnaires, baseline recordings and the testbed game. The calibration of the eye tracker was thus the only part that required a switch between the Unity build and the software of the Gazepoint GP3-HD eye-tracker. The thoughts behind the design decision to implement the rest of the experiment in one Unity project were that by minimizing the need to switch between different applications, the chance to make procedural mistakes was minimized. Additionally, implementing all in one unity project enabled full control regarding the visual style of the application used throughout the experiment. For example, we could use a questionnaire in a full screen mode with a black background, white font and without distracting menu's of other applications, which we deemed relevant to minimize pupil light responses. Furthermore, to minimize the pupil light responses during the switch to the Gazepoint GP3 software for calibration, the windows environment was turned to dark mode.

Both the game data and pupillometry data were processed and transformed into features in the programming language R[3]. Since in our experience the Scikit Learn offers more options and a more convenient interface with respect to tuning random forests hyperparameters compared to R packages like tuneRanger and RandomForest, the Scikit Learn package was used to perform the actual training and statistical analysis [117].

[3] `https://www.r-project.org/` R is a language and environment for statistical computing and graphics. R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible.

Figure 4.4: Photo to showcase the experimental setup, with the eye tracker placed beneath the screen slightly tilted upwards to capture the eyes and the red light. Noteworthy, for creating the picture extra daylight was let into the room.

4.4 Participants

The experiment was carried out by 32 participants. The data recorded for three participants was excluded based on observations. Two of these three participants did not succeed in finishishing the practice round. For both of these participants, controlling Pac-Man smoothly through the maze, even without considering the ghosts, already seemed to be quite a challenge resulting in distinctly different gameplay from the other participants. The third one accidentally mirrored the answer scale a few times of the questions after each round, this was verbally verified. Furthermore, due to a high number of invalid samples in the pupillometry recordings, data recorded for 10 other participants was excluded as well (also see 4.5.2.1 and 4.5.2.2).

The final feature set was constructed with data recorded from 19 participants, of which 16 regarded themselves as male and 3 as female. Table 4.1 shows the distribution with respect to age. When considering the middle of each age group as the group's mean value, the mean age of the participants was approximately 26. With respect to games most participants played video games on a weekly (7) or daily basis (5), while quickly paced 2D games are played less often. All participants played Pac-Man at least once, most participants (11) played it more than 10 times. None of the participants viewed themselves as an expert in Pac-Man, most participants viewed themselves as *slightly beginner* (6) or intermediate (5). For an overview of the answers to the demographics questions, see Appendix Q.

No participants had to be excluded when considering the fatigue score of the participants (see 4.2.2.1). Since most participants were students we interpreted the scores for the category students and although 8 had an above average score, none had a high

fatigue score.

Table 4.1: The age distribution of the participants.

Age Group	Participant count
16 t/m 18	1
19 t/m 21	2
22 t/m 24	5
25 t/m 27	6
28 t/m 35	4
36 t/m 43	0
44 t/m 51	1

4.5 Finalizing feature set

4.5.1 Game features

Before finalizing our feature set (consisting of both *game features* and pupillometry features), we processed insights gained during our experimental procedure. For example, as expected we noticed some participants tended to save up super pellets; we saw them collecting the pellets around a super pellet but leaving the super pellet itself. Some of the participants did this in a secure manner by collecting all pellets surrounding the super pellets, others simply did not take the paths of the super pellets. Instead of considering the junctions around super pellets as initially planned, we believed the percentage of super pellets divided by the percentage of all pellets provide a better representation and therefore added this feature.

Furthermore, we reviewed the generic type of each feature. We recognised features representing a frequency, a percentage and an average. For features representing an average, like the average distance between the ghost centroid and Pac-Man, we added another feature representing the standard deviation.

Furthermore, we excluded some of the initial features. Some of these were mirrored features, for example the percentage of successful chases correlates with the percentage of unsuccessful chases ($succesPercentages = 1 - unsuccesPercentage$). Others required some extra manual processing to calculate and were more specific versions of other features. For example, when reviewing it turned out that the death event was not correctly stored in the game code. With additional functions this was easily retrievable from the data, however the more specific features combining death with the number of nearby ghosts were excluded due to the limited scope of the project. Noteworthy, we

also decided to exclude the difficulty settings and participant id features. We wanted to prevent the classifiers to distinguish participants and to focus on settings.

Finally, for some of the features the *overall speed* difficulty setting was of influence. For example, to compare the feature *frequency of the number of times Pac-Man was chased* at a lower *overall speed* to the same feature at a higher speed, a correction is necessary. Therefore, all features that were impacted by the *overall speed* setting were corrected. The final set of features together with the excluded features can be found in Appendix R.

4.5.2 Pupillometry features

4.5.2.1 Pupillometry data preprocessing

The GP3 eye tracker data contains the left and right eye pupil diameter in millimeters together with two flags indicating if the left and right pupil diameter data is valid. Also, a blink ID is available, with a value 0 for samples wherefore no blink was detected and a unique value indicating blinks. This data required preprocessing before we could extract pupillometry features. We applied six preprocessing steps in the programming language R.

The first step consisted of indicating invalid pupil size samples based on samples indicated as blinks and marked as invalid in the raw data. The pupil data recorded with the GP3 eye tracker required a different approach compared to our initial plan to follow the Winn et al.'s [24] recommendations to apply interpolation from 50 ms before up to 150 ms after a blink. First of all, when reviewing the data we noticed that during a blink ID the valid flags for both the left and right eye were also set to invalid. Furthermore, we found the pupil size data to start a decrease of value 4 to 3 samples before the occurrence of a blink, corresponding at 60 hz to roughly 67 ms and 50 ms respectively. The same decrease in value was also found for pupil samples marked as invalid apart from blinks. Therefore, instead of excluding 3 samples (50 ms) before blinks only, we excluded 4 samples before invalid indicated samples (thus including blinks). Furthermore, after reviewing the data of multiple recordings, we found excluding 150 ms after blinks to be a too long period. Possibly task-uncorrelated high-frequency changes in pupil size after blinks occur during the periods indicated as blinks in the GP3 data. Instead of 150 ms, an exclusion of 67 ms corresponding to 4 samples was also enough to remove peaks and dips after blinks and invalid samples. To sum up, we set the samples indicated as invalid together with the 4 preceding and tailing samples to the value`NA`[4]. Additionally, as a second step, we excluded all rounds of Pac-Man record-

[4]In the language R, `NA` is a value that can be assigned to numeric data and is referring to "not a number".

ings (trials) which contained 25% or a higher amount of NA samples (on average 24% of the samples were invalid).

The third and fourth steps were based on the data preprocessing as described in Mannaru et al. [92], (see 2.3.2), namely the appliance of a Hampel filter and a lowpass Butterworth filter. Before we could apply these filters, the NA values in the pupil size data were temporarily replaced by applying linear interpolation. The Hampel filter was applied to remove outliers, using the default window size of 13 (K = 6, thus 6 samples before and samples after the reviewed sample) and a sigma value of 1.5 since the default of 3 did not remove outliers thoroughly enough. In the appliance of the second-order lowpass Butterworth filter we used a cutoff frequency of 4, "since most of the pupillary activity falls in the frequency range of 0 - 4 Hz" [92].

Fifthly, after restoring the NA values we applied cubic spline interpolation in order to interpolate the missing pupil size values indicated as NA (in the R na.spline function the method was set to monoH.FC to prevent oscillating). In our final step, we averaged the pupil size measurements of left and right eyes to reduce measurement noise.

4.5.2.2 Baseline correction

Before calculating the baseline, the pupillometry data recorded during the baseline period was preprocessed as described in 4.5.2.1. Pupillometry data preprocessing. Sequentially, in R we calculated the baseline by taking the average pupil size of one second of contiguous samples from the end of the data. Where we planned to apply a threshold of 25% with respect to the allowed percentage of invalid samples (regular applied thresholds are in the range of [15%, 25%], [24]). However, after reviewing baseline recordings that did not meet the 25% threshold, we found recordings with approximately 35% or less invalid samples in the last part of the recording still to be valid. Therefore, we increased our threshold for the baseline to 35%.

Furthermore, in some recordings the requirement of the threshold was not met due to multiple short repetitions of invalid samples (e.g. a blink) at the end of the recording, an example is provided in the plot at the bottom of Figure 4.5. After reviewing, we believed these recordings to be valid as well. We implemented a function that allowed the window to move sample by sample back in time (thus away from the end) with a maximum offset of one second, until the window contained sufficient valid data. We chose one second as a maximum offset from the end in order to consider the last 2 seconds of the data only. We believed that allowing the window to move even further away from the end could lead to an unrepresentable baseline, because of a possible present pupil light response due to the change in light intensity between the baseline- and preceding screen. Additionally, at the beginning of the baseline period, the participants often showed a small reaction (moment of shock, sitting up straight, staring

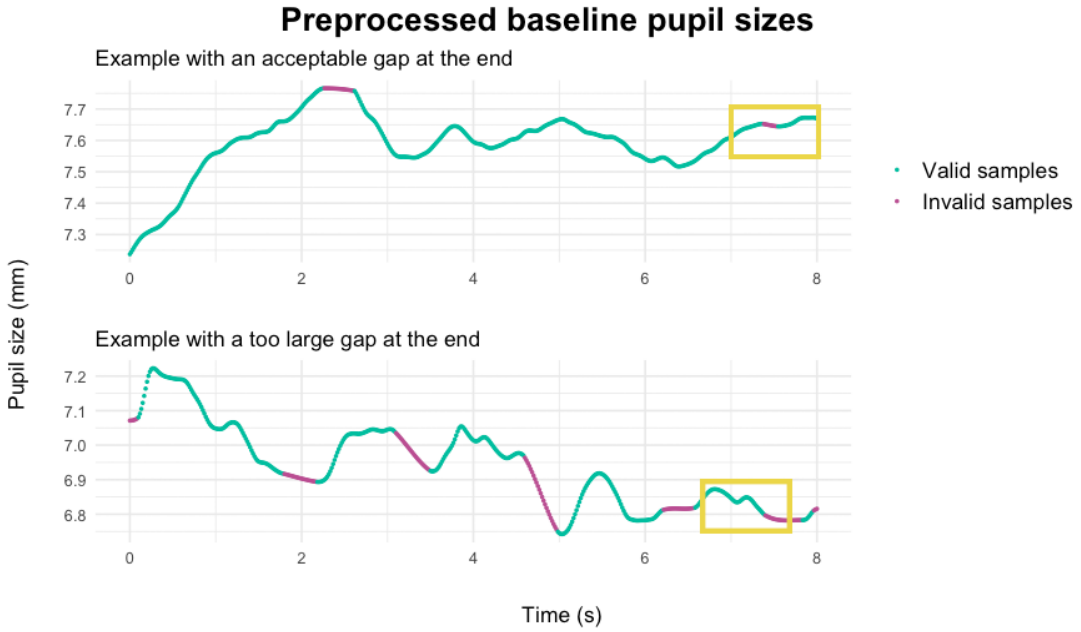

Figure 4.5: Two plots of preprocessed absolute pupil sizes recorded during the baseline period preceding rounds of Pac-Man. In the plot at the top the number of invalid samples in the last second is beneath the threshold of 35%. Here, the last second was used to calculate the baseline size. In the plot at the bottom the number of invalid samples in the last second is above the threshold of 35%. Here, the one second window used to calculate the baseline size, is moved forward in time in order to base the baseline on enough valid samples.

unnaturally) and sometimes looked down while placing their fingers on the arrow keys, which both also may have caused fluctuating pupil sizes mainly in the beginning of the baseline recordings.

Noteworthy, instead of heightening the threshold, we could have chosen to consider a smaller window, but we decided against this option. In some baseline recordings the pupil sizes fluctuated at the end of the recordings and therefore we preferred taking the average of one second instead of a shorter period, to decrease the impact of recording noise.

Furthermore, as advised by Mathôt et al. [80] we reviewed the final baselines to prevent inclusion of unrealistically small baselines. Herefore, we reviewed the two lowest found baselines and compared these with the sequential pupil size measurements corresponding rounds of Pac-Man. We did not indicate these baselines as unrealistic small.

We applied *subtractive* and *divisive* baseline corrections separately, to reduce the impact of random pupil-size fluctuations between trials (see 2.3.3). The *subtractive* baseline correction was calculated by subtracting the calculated baseline of the corresponding round of Pac-Man. For the *divisive* correction, we used the average baseline of

all rounds, as advised by Mathôt et al. [80]. However, we diverted from the advice of Mathôt et al. [80] to take the average of all samples recorded during the baseline periods per participant. Instead, we took the average of the calculated baselines, thus considering one second of the end of the baseline recordings only. Reason for this was the noticeable pupil size fluctuations at the beginning of many of the baseline recordings. The *divisive* baseline correctio was applied by dividing the preprocessed pupil size data with the averaged baselines of the nine rounds of Pac-Man per participant.

4.5.2.3 Initial pupillometry features

We planned to consider the *peak pupil dilation* and *mean pupil dilation* (see 2.3.4). We wanted to review the chase and frightened state separately. We wanted to calculate the *mean pupil dilation* by taking the average of all pupil sizes recorded during chase states. Furthermore, we imagined that the moment in the game with the most 'heat' is of influence to the player when answering the questions regarding the game experience of that trial. Therefore, we also planned to utilize the *mean pupil dilation* for the chase state with the highest mean pupil size as a separate pupillometry feature.

4.5.2.4 Pupillometry data synchronization issue

In order to share our final pupillometry features we have to share a very impactful issue that arose after the experiment. Prior to the execution of our experiment we verified that the timeticks of the clock available within the unity engine used for the game data corresponded to the timeticks of the clock used in the eye tracker software. However, after the execution of our experiment we noticed a discrepancy between the timeticks in the game data and pupillometry data. We thoroughly reviewed all data with respect to this discrepancy and found that the data was valid, since amongst others the discrepancy growed and corresponded to the time between the experiments and the time between the data of different rounds of Pac-Man corresponded to in both the game data and pupillometry data. However, the synchronization of the data was a lot more complicated than if we could have simply synchronized the data based on timeticks values.

Regarding possible solutions of the 'timetick-issue', we did consider other options in order to synchronize the data. We figured we could not simply align the start of both types of data, because a delay occurs when starting sending the server a request to start collecting. Also, the pupillometry data is recorded on a separate thread which in between recordings sleeps with intervals of 1 second, thus an additional fluctuating delay occurs at the start of a round of Pac-Man. However, we do think it would be possible to synchronize the data by working backwards from the end of the both types of data, by amongst others considering multiple aspects regarding the timing of the end. But in the limited scope of this research, we found this such a time consuming

step and therefore decided against it.

4.5.2.5 Final set of pupillometry features

After noticing the synchronization issue we redesigned our set of pupillometry features with respect to the *mean pupil dilation* features. Since, due to the synchronization issue, we could not only consider the *mean pupil dilation* of the chase states. In search of a different strategy, we reviewed plots of the pupil size data to gain a better view of the measurements and noticed regions with lower pupil size values. After reviewing the corresponding gameplay data we found that the start of these parts approximately coincided with Pac-Man's death, see the example provided in Figure 4.6. This simultaneous occurrence is no coincidence. After Pac-Man is captured by the ghosts, a fade animation of one second and successively a new countdown of 3 seconds is triggered, resulting in a minor break from the active gameplay. The lower pupil measurements coinciding with these minor breaks indicate a lower workload as could be expected. We viewed these lower measurements to be unrepresentable with respect to the invested workload during the active gameplay and decided to only consider the active gameplay.

In order to only consider the active gameplay, we were interested in a percentage of the highest pupil sizes, since the lower ones could indicate inactive gameplay. To pinpoint a percentage threshold for the highest pupil sizes, we first reviewed the percentages of active gameplay for each trial. Per trial we calculated the percentage of active gameplay by subtracting the sum of animation durations (consisting of the animation at the start and after each death) from the round's duration and then sequentially dividing the result by the round's duration. The lowest found percentage of active gameplay was 34%. We rounded this percentage down 30% and used this to filter the 30% highest pupil sizes per trial. Next, we took the average of these highest pupil sizes, resulting in the measure *highest pupil dilations*. Additionally we still considered the *peak pupil dilation* as a feature as well.

4.5.2.6 Pupillometry feature normalization

In order to allow for comparison between participants, we normalized the pupillometry features. We applied z-standardized normalization within participants across the nine rounds of Pac-Man. Herefore, we could only consider trials of participants of which all trials were indicated as valid. The other trials were excluded. To sum up, the final features of the pupillometry dataset consisted of the z-standardized *highest pupil dilations* and *peak pupil dilation* for both the *subtractive* and *divisive* corrected pupil sizes (*sub_highest*, *sub_peak*, *div_highest*, *div_peak*).

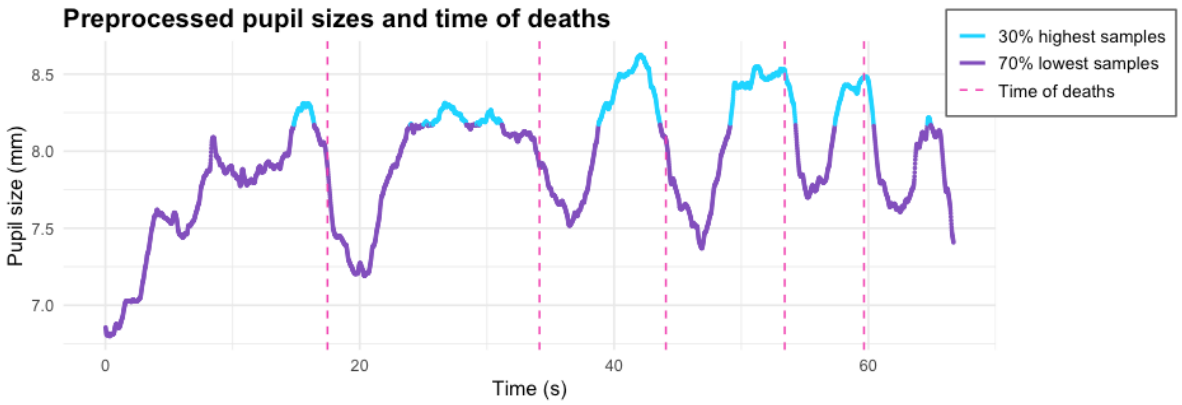

Figure 4.6: The time of Pac-Man deaths and preprocessed absolute pupil sizes recorded during a round of Pac-Man.

4.6 Data analysis

In order to answer our main research question "*Does the addition of pupillometry features to an input feature set improve the accuracy of a random forest classifier in predicting experienced difficulty?*" (RQ1), we trained multiple random forests with feature sets with and without pupillometry features and compared their performance. Random forests (RF) were introduced by Breiman [118]; "Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest". In other words, a random forest is a collection of decision trees constructed from different parts of the data. The prediction of a forest is the average of the predictions of all its trees. We used a classification RF because the targets of our dataset are constructed from likert scale answers, where the intervals between the answers are not necessarily equal, thus our targets are ordinal.

There are three combined reasons why we choose RF as a learning method. First of all, it suits our dataset, for which we expect a non-linear relationship between the input features and predicted classes, either experienced challenge, competence and invested workload. Secondly, a RF is also robust when training on high dimensional datasets, since our dataset consists of 190 samples and 134 features, we can consider our dataset to be quite high dimensional. Thirdly, we can review variable importance of a trained RF model [119], [120]. Although we want to compare models trained on different feature sets, with and without our pupillometry features, variable importance provides further insights in the possibly added value of the use of pupillometry features as input features to predict the experienced challenge.

Also, we applied statistical analysis in order to analyze the relation between the features and the participant's answers regarding the experienced game difficulty. Hereby, amongst others we review the correlation between pupillometry features and the reported experience of the game difficulty in order to answer our second research question *"To what extent does this pupillometry feature statistically correlate to the self-reported experienced game difficulty?"* (RQ2). Both the results and the process of the statistical analysis and of the trained RFs are described in section 5 Results.

5. Results

5.1 Participant responses

5.1.1 Response distribution

The distribution of the level of agreement responses for the experienced challenge statement "The last game was very hard to play." (CHAL) are displayed in Figure 5.1. This distribution can be viewed as slightly negatively skewed. These results correspond with the observations made during the experimental procedure, where we noticed people tended to use only part of the scale and often the "strongly agree" answers were seemingly ignored for CHAL. We took this imbalance into account while training our RFs, by applying a stratified split, stratified K-fold cross validation and by considering the F1 weighted average to review the classifier's performance.

5.1.2 Demographics and responses correlation

To gain more insight into possible associations between demographics and the level of agreement CHAL responses we performed a chi-square test. We only found significant differences for the levels of experience with Pac-Man ("How often did you play Pac-Man? (estimation)"), p-values 0.042 respectively. Noteworthy, no significant difference was found for the responses to the experience statements across the self-estimated skill level for Pac-Man (p-value 0.249). All other p-values can be found in the table "Chi-

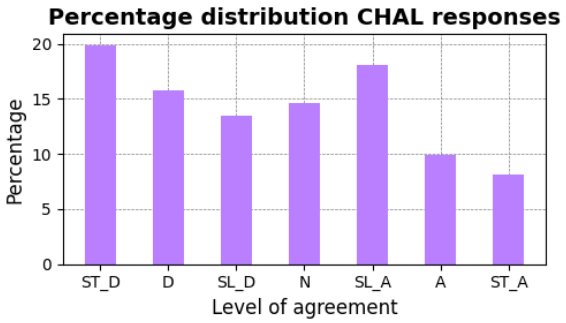

Figure 5.1: The percentage distribution of the level of agreement responses for the CHAL game experience statement ("The last game was very hard to play") for all participants. The level of agreement x-axis labels refer to "strongly disagree" (ST_D), "disagree" (D), "slightly disagree" (SL_D), "neutral" (N), "slightly agree" (SL_A), "agree" (A), "strongly agree" (ST_A).

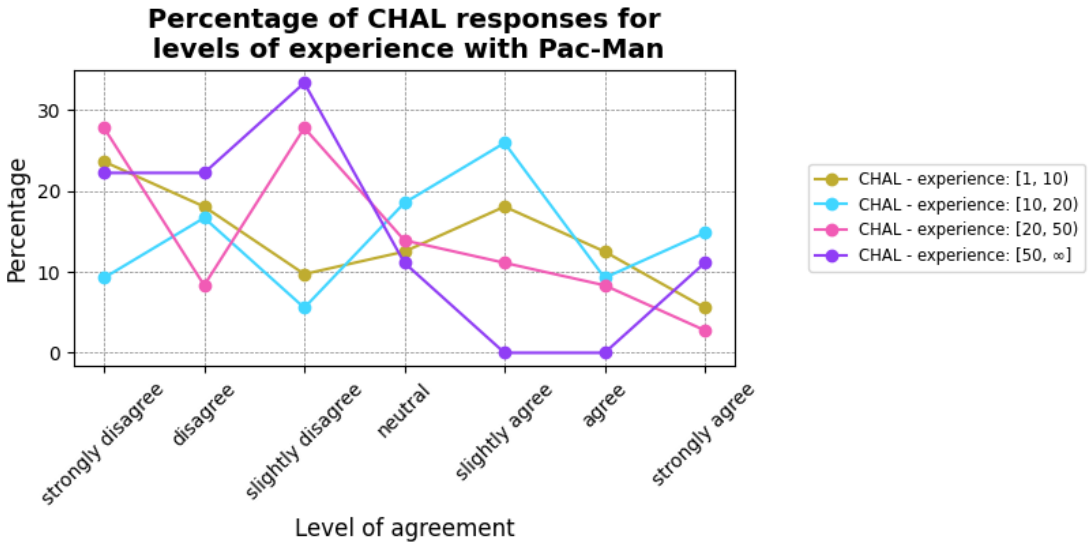

Figure 5.2: The percentage distribution of the level of agreement responses provided for the CHAL game experience statement ("The last game was very hard to play"), shown for four groups of participants with different self estimated levels of experience with Pac-Man.

square p-values for demographics and experience classifiers" in Appendix S.

The distributions of CHAL responses plotted per self estimated level of experience with Pac-Man are displayed in Figure 5.2. Roughly viewed, the participants with a higher experience with Pac-Man provided lower level of agreement responses for CHAL. We further reviewed the self estimated level of experience with Pac-Man by comparing it to the feature *death_freq*, which represents the frequency of Pac-Man deaths in a game round which we believe can be viewed as a simplistic performance measure. A Kruskal-Wallis test did not show a significant difference in *death_freq* values across the groups defined by the provided experience with Pac-Man answers (p value = 0.72).

5.1.3 Difficulty levels

For Pac-Man difficulty levels 7 and 8, on average lower CHAL responses were received in comparison to the responses for difficulty level 5 and 6, see Figure 5.3. The *relative speed* difficulty setting for these four levels was constant, set to *medium*, while the other two difficulty settings differed. The *overall speed* difficulty setting for difficulty level 5 and 7 was *medium* and *hard* for difficulty level 6 and 8. The *ghost behavior* difficulty setting was set to *hard* for difficulty level 7 and 8 and set to *medium* for difficulty level 5 and 6. The *ghost behavior hard* corresponds to the *near-optimal ghost behavior* and *medium* to *followers*. In other words, for the levels with the *near-optimal* ghosts (level 7 and 8) on average lower CHAL were received in comparison to the levels with the *followers* (level 5 and 6), at *relative speed* set to *medium*. A chi-square test was performed to examine

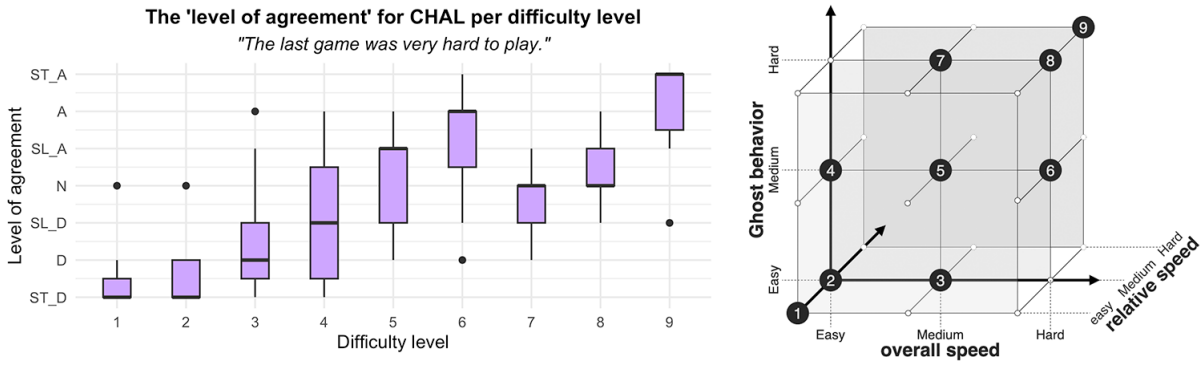

Figure 5.3: On the left, boxplots of the level of agreement responses provided for the CHAL game experience statement ("The last game was very hard to play") per difficulty level. The level of agreement x-axis labels refer to "strongly disagree" (ST_D), "disagree" (D), "slightly disagree" (SL_D), "neutral" (N), "slightly agree" (SL_A), "agree" (A), "strongly agree" (ST_A). On the right, the difficulty space as implemented in the testbed game, with the nine selected combinations. Their numerical order corresponds to the expected chronological difficulty increase.

the relationship between the difficulty level and CHAL responses and a significant association was found (p-value 1.51e-22).

5.1.4 Participant groups

Because more than a third of the data was excluded (see 4.5.2.1 and 4.5.2.2) the data of the participant groups A, B and C, for which a different order of level difficulty was implemented, are not represented equally. The features of group A are constructed with data of four participants, group B is constructed with data of seven participants and group C is constructed with data of eight participants. To review if the order had an impact on the responses for CHAL, we plotted in chronological order of appearance the differences of level of agreement between each group and the other two groups for the same difficulty level, see Figure 5.4. Here, we also plotted the responses for COMP and WRKL, in order to gain a wider overview which could assist in identifying a possibly present influence of the experimental order. Noteworthy, the level of agreement responses are ordinal for which the distances between labels are not necessarily equal and can thus not be treated as numeric values. However, in Figure 5.4 these distances are considered to be equal, only in order to review a possible impact of the three different experiment condition orders on the responses.

The average differences per group for CHAL and WRKL slightly differ, whereby group A differences are mostly negative, group B is mostly positive, and group C displays a lower range of differences around zero. The differences for COMP are seemingly inverted, higher average difference values for group A and lower values for group B. A more detailed view of the level of agreement responses per group per difficulty level is provided by the boxplots in Appendix S. The plotted lines do not follow a horizontal

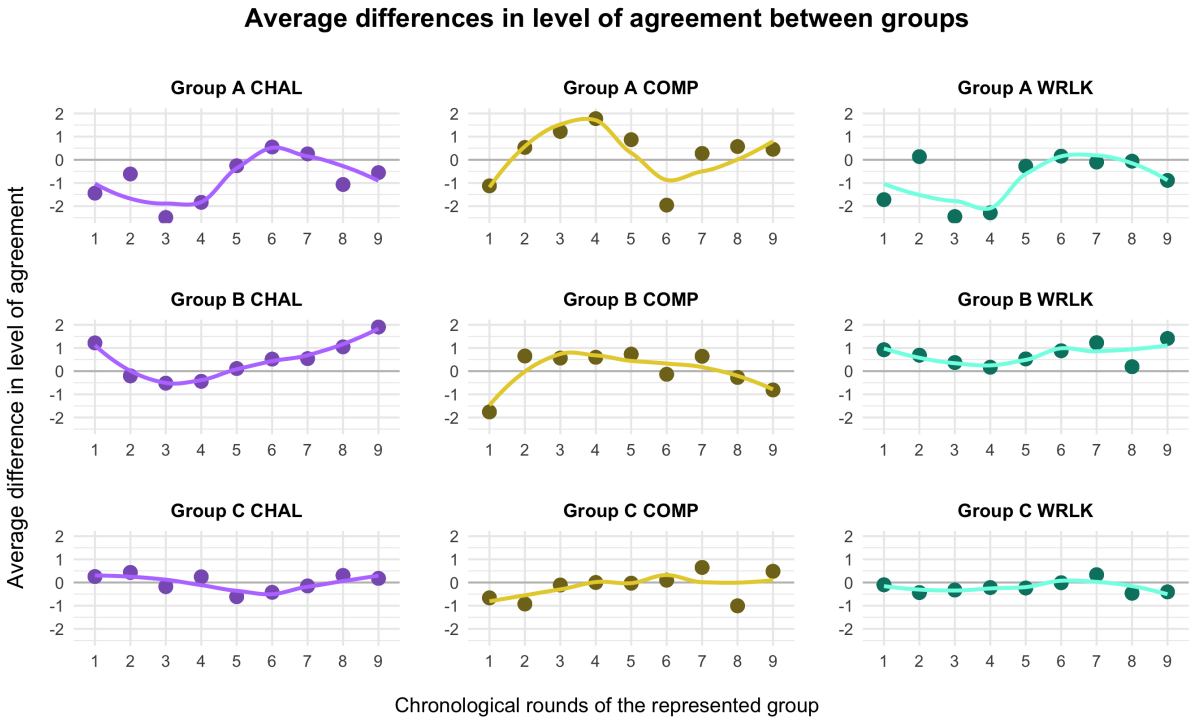

Figure 5.4: For each of the three participants groups which followed a different order of difficulty levels, the difference between the group's average level of agreement and that of the other two groups for the same level of difficulty is displayed. These differences are displayed in the chronological order of the rounds of Pac-Man per group. Whereas the difficulty order for group A was 5, 9, 4, 8, 3, 7, 2, 6, 1. The difficulty order for group B was 8, 3, 7, 2, 6, 1, 5, 9, 4. The difficulty order for group C was 2, 6, 1, 5, 9, 4, 8, 3, 7. The three plots per group represent the level of agreement for CHAL ("The last game was very hard to play."), COMP ("I felt I was very good at playing the last game.") and WRKL ("I had to put a lot of effort into the last game.").

tendency. Thus, no clear consistent pattern can be seen across all three groups.

To further review a possible presence of influence of the experiment condition order on the responses to the game experience statements, a chi-square test was performed. No significant p-values were found for the comparison of responses across the three participant groups per game experience statement (p-value experienced challenge 0.278, p-value estimated competence 0.660, p-value estimated workload 0.388).

5.1.5 Pupillometry features

As discussed in 4.5.2.5, the final features of the pupillometry dataset consisted of the z-standardized *highest pupil dilations* and *peak pupil dilation* for both the *subtractive* and *divisive* corrected pupil sizes (*sub_highest*, *sub_peak*, *div_highest*, *div_peak* respectively). The boxplots in Figure 5.5 capture the pupillometry features against the level of agreement responses to the three game experience statements. A cautious inverted u-curve can be seen, most visible for *div_highest*. However, when viewing plots of the *div_high-*

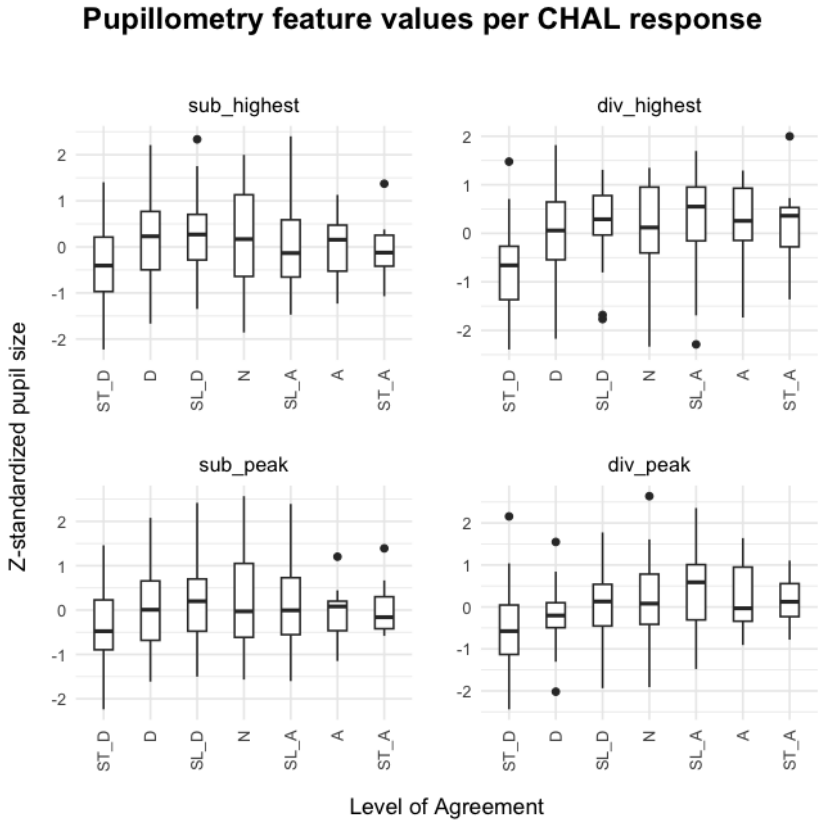

Figure 5.5: The z-standardized *subtractive* and *divisive* highest and peak pupil size features are captured in boxplots per level of agreement responses for the CHAL game experience statement ("The last game was very hard to play."). The level of agreement responses consist of "strongly disagree" (ST_D), "disagree" (D), "slightly disagree" (SL_D), "neutral" (N), "slightly agree" (SL_A), "agree" (A), "strongly agree" (ST_A).

est value across the level of agreement responses for CHAL per participant, for many participants no inverted u-curve is visible, see the left plot in Figure 5.6. Additionally, for some participants the data can come across as random. When aggregating both outer two and the middle three response labels into three groups, we can distinguish an inverted u-curve for six of the participants, see the right plot in Figure 5.6. For three participants, not all aggregated response labels are present in the data.

When viewing the pupil feature *div_highest* against the difficulty levels, an inverted u-curve can be seen for 17 out of 19 participants, although for some only cautiously, see the right plot in Figure5.7. For these 17 participants *div_highest* was lower for difficulty level nine in comparison to its value for either or both level five and six. For some of these participants the decline measured for difficulty level nine is cautious in comparison to the maximum *div_highest* at either difficulty level five or six.

A Shapiro-Wilk test indicated that not all pupillometry features met the normality data requirement of the ANOVA test. Instead, the Kruskal-Wallis test was performed on the four pupillometry features across the level of agreement CHAL responses. For both

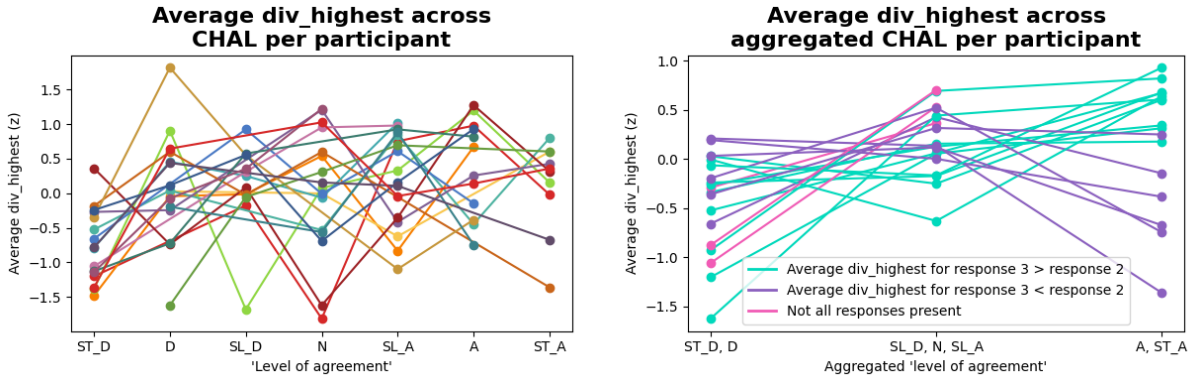

Figure 5.6: The average *div_highest* values (z-score) per participant plotted across the level of agreement responses for CHAL. In the plot on the left, all level of agreement responses are displayed separately and each color represents a participant. In the plot on the right, the level of agreement responses are aggregated, whereby both two outer responses and the three middle responses are aggregated into three groups. The level of agreement x-axis labels refer to “strongly disagree” (ST_D), “disagree” (D), “slightly disagree” (SL_D), “neutral” (N), “slightly agree” (SL_A), “agree” (A), “strongly agree” (ST_A).

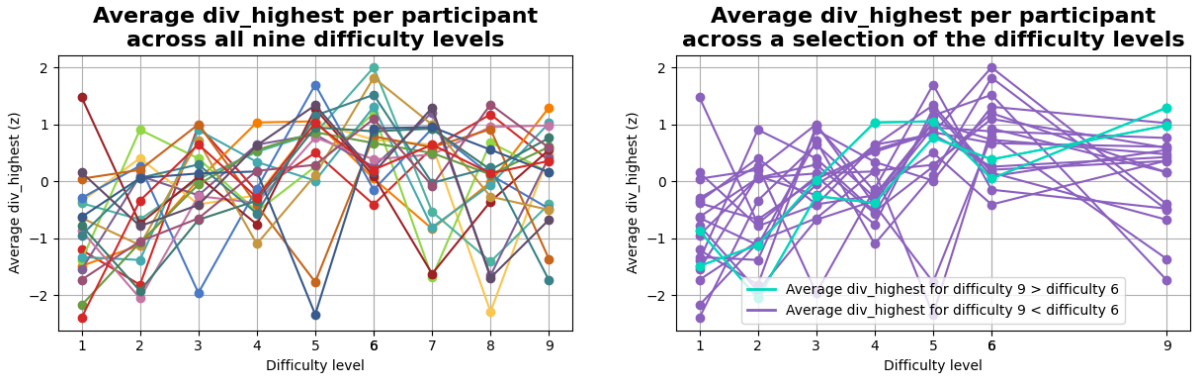

Figure 5.7: The *div_highest* values (z-score) per participant plotted across the difficulty levels. In the plot on the left, all difficulty levels are displayed and each color represents a participant. In the plot on the right, the difficulty level 7 and 8 are excluded due to the unexpected lower CHAL responses in comparison to level 5 and 6.

the *div_highest* (p-value 0.00021) and *div_peak* (p-value 0.0031) significant differences in medians were found, but no significant differences were found for *sub_highest* (p-value 0.22) and *sub_peak* (p-value 0.37). The sequentially performed post hoc Dunn's test on *div_highest* and *div_peak* indicated that only some of the distributions were significantly different (< 0.05) (see Appendix T for the results). Namely, for *div_highest* the "strongly disagree" distribution was found to significantly differ from the other distributions with exception of "disagree" and "strongly agree". For *div_peak*, the "strongly disagree" was found to significantly differ from "slightly agree".

5.2 Random forest classifiers

In this section, not only the results of our final RF classifiers are described, but also the process of selecting and tuning RF hyperparameters and feature selection. The provided description of the process was created with reproducibility in mind. The hyperparameter selection and tuning process was performed multiple times. Firstly, the process was performed to train a RF to predict the 7 original categories (the participants' responses ranging from "strongly disagree" to "strongly agree"). But also, the process was performed to train RF classifiers to predict aggregated labels, this process was performed multiple times to predict different subsets of features. The first subsection describing the hyperparameter selection and tuning process is written more elaborately, compared to its subsequent sections. Noteworthy, we trained all RFs on part of our data; we splitted our data set into a training set (75%) and test set (25%), by applying a shuffled stratified split due to our imbalanced dataset. For the final results that concern the stated research questions RQ1 and RQ2, we advise the reader to skip to 5.2.4

5.2.1 Random forest classifiers original categories

5.2.1.1 Benchline prediction

In order to compare the RFs to a benchline, we created a benchaline prediction, whereby we assigned the highest occurring answer as prediction for all samples and calculated the corresponding accuracies (micro, macro and weighted). In our RF results we viewed the F1 weighted average (*F1w_avg*) instead of accuracy to take the imbalanced response distribution into account (see 5.1.1). The *F1w_avg* of the benchline prediction for CHAL is 0.1988 (see Appendix U for the micro and macro accuracies).

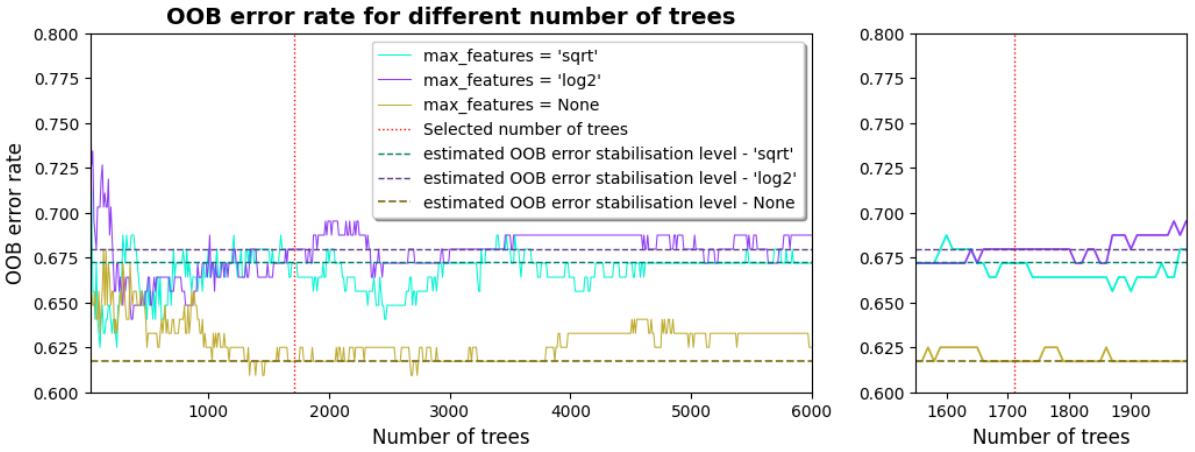

Figure 5.8: The OOB error plotted against the number of trees for RF from the Scikit Learn package with the parameter *max_features*, the maximum number of features to consider for a split, set to the values *sqrt*, *og2* and *None*. With 1710 trees the three OOB error plots were estimated to approximate the corresponding OOB error rate stabilization levels.

5.2.1.2 RF with default hyperparameters

First, we trained a RF using the Scikit-learn RF classifier with the default settings which correspond to the advise of Hastie et al. [119] to use $sqrt(number\ of\ features)$ for the number of features to consider when splitting nodes (*max_features*) and 1 for the minimum terminal node size (*min_samples_leaf*) for a classification task [117]. Furthermore, we set the Scikit-learn RF *class_weight* parameter to *balanced* to take the imbalance distribution into account, this resulted in a *F1w_avg* of 1.00 for the train set and 0.41 for the test set.

5.2.1.3 RF number of trees

According to Probst and Boulesteix [121] the number of trees (*n_estimators*) should be sufficiently high, more trees are better and for stable feature importance generally more trees are required. In order to find a sufficient number of trees, a plot of the OOB error rate against the number of trees (*OOB_plot*) can provide insight at which *n_estimators* the OOB error stabilizes [119]. To illustrate, Figure 5.8 displays the OOB error rate curves for the Scikit-learn *max_features* settings *sqrt*, *log2* and *None*. A non-monotonous pattern can be seen for *max_features sqrt* and *log2*, with lower regions of error rates before stabilizing at a higher error rate. This can occur in case of classification, but also then a large value for *n_estimators* for which the OOB error is stabilized is preferred compared to tuning *n_estimators* [121]. We estimated the settle down error for each of the three plotted error curves, see Figure 5.8 and selected 1710 for *n_estimators* in our RFs. A RF with default settings, *n_estimators* set to 1710 and *class_weight* parameter to *balanced* resulted in the *F1w_avg* of 1.00 for the training set and 0.42 for the test set.

5.2.1.4 RF randomized search

Aiming to prevent overfitting the training set, we performed multiple random searches. All conducted searches were performed with a repeated stratified K-fold cross validation, with 5 repetitions and 3 folds. Noteworthy, we used 3 folds instead of the generally used 5 or 10 folds, due to our small sample set. The hyperparameter search spaces were constructed based on insights shared by Probst et al. [122] regarding hyperparameter tuning. For tuning we selected by Probst et al. [122] discussed hyperparameters *max_features*, *min_samples_leaf*, the "minimum number of samples required to split an internal node" (*min_samples_split*) together with "the number of samples to draw from X to train each base estimator" (*max_samples*) [117]. Noteworthy, the maximum depth of the tree (*tree_depth*) is tuned in some other previous work (for example [123], as cited by [109], [119]). However, since the hyperparameters *min_samples_leaf* and *min_samples_split* already are of influence to the tree depth we did not separately include *tree_depth* as tuning parameter.

In total, we performed five random searches and one grid search to find suiting hyperparameters in order to prevent overfitting the training set. For each of these searches, the search distributions together with the hyperparameters and the weighted accuracies for the best found RFs are displayed in Table 5.1 to illustrate these steps (the regular and macro average accuracies can be found in Appendix V).

The first four searches focused on different hyperparameter search distributions, based on insights regarding hyperparameter tuning shared in previous work. Firstly, we performed a randomized search with a broad search distribution for the selected hyper parameters (*1. Broad search*). Additionally, we performed a randomized search with a range for *max_features* limited to lower values than the default $sqrt(number\ of\ features)$ (*2. Low max_features*), to allow less influential variables to be chosen for splits instead of only the strongest influential variables [124], as cited by [122]. We also performed a randomized search with the *max_features* range limited to high values (*3. High max_features*), because Goldstein et al. [25], as cited by [122] observed lower error rates for higher *max_features* values for high dimensional data (for both classification and regression) similar to findings of Segal [123] regarding data with noise features. Additionally, we performed a search with the *min_samples_split* range limited to higher values compared to the default (*4. High min_samples_split*), based on findings of Segal [123]; higher node sizes can result in performance gains for data with noise variables.

The search distributions for the last two searches were based on insights gained from the results of the first four searches. Firstly, because the *2. Low max_features* search led to a higher *F1w_avg*, we performed a search with low values for *max_features*, but then not only in combination with a range for *max_samples*, but also with a search range for both *min_samples_leaf* and *min_samples_split* (*5. low max_features*). Finally, we

performed a grid search (*6. final grid search*) with a small range of hyperparameters covering a range around hyperparameters of the best found models scoring highest on the test set (*4. High min_samples_split* and *5. low max_features*), resulting in a model with a *F1w_avg* of 0.85 on the train set and 0.44 on the test set.

Table 5.1: The hyperparameter search distributions together with the hyperparameters and weighted accuracies for the best found RF models for each performed search. Here, # f and # s refer to the number of features and number of samples in the train set respectively. For all searches, the parameter *n_estimators* was set to 1710.

	max_features	max_samples	min_ samples_leaf	min_ samples_split	Train set F1 weighted average	Test set F1 weighted average
Benchline prediction						0.20
Default parameters results *n_estimators = 1710*	'sqrt'	all samples	1	2	1.0	0.42
1. Broad search *random search range*	[3, # f]	[½ #s, #s]	[1, 10]	[2, 10]		
results	69	66	5	4	0.83	0.37
2. Low max_features *random search range*	[3, 7]	[½ # s, # s]	1	2		
results	5	54			0.99	0.44
3. High max_features *random search range*	[⅓ # f, # f]	[½ # s, # s]	1	2		
results	96	54			1.00	0.35
4. High min_samples_split *random search range*	'sqrt'	[½ # s, # s]	1	[5, 15]		
results		80		13	0.85	0.44
5. Low max_features 2 *random search range*	[3, 10]	[60, 81]	[1, 5]	[2, 15]		
results	8	60	2	13	0.82	0.46
6. Final grid search *grid search parameters*	[4, 5, 6, 7, 8, 9, 10, 11, 12]	[53, 54, 55, 59, 60, 61, 79, 80, 81]	[2, 3, 12, 13, 14]	[1, 2]		
results	11	80	1	13	0.85	0.44

Table 5.2 and Figure 5.9 display the classification report and confusion matrix respectively of the best performing model found in search 6. Final grid search (*7_categories_-model*). The F1 score is lowest for the three middle classes (representing the answers "Slightly disagree", "Neutral" and "Slightly agree"), whereas the F1 score of the middle class is zero ("Neutral"). When reviewing the confusion matrix, roughly three clusters can be distinguished, namely the first two classes, the middle three classes and the last two classes.

5.2.2 Random forest classifiers aggregated categories

Because of our final model's relatively low weighted accuracy for the test set and the varying model performance for the different classes, we aggregated the seven categories into three categories. The three clusters distinguishable in the confusion matrix displayed in Figure 5.9, informed this aggregation. We combined "strongly disagree" and "disagree" into the new class "disagree". The classes "slightly disagree", "neutral" and "slightly agree" were combined into "neutral" and "agree" and "strongly agree" into "agree".

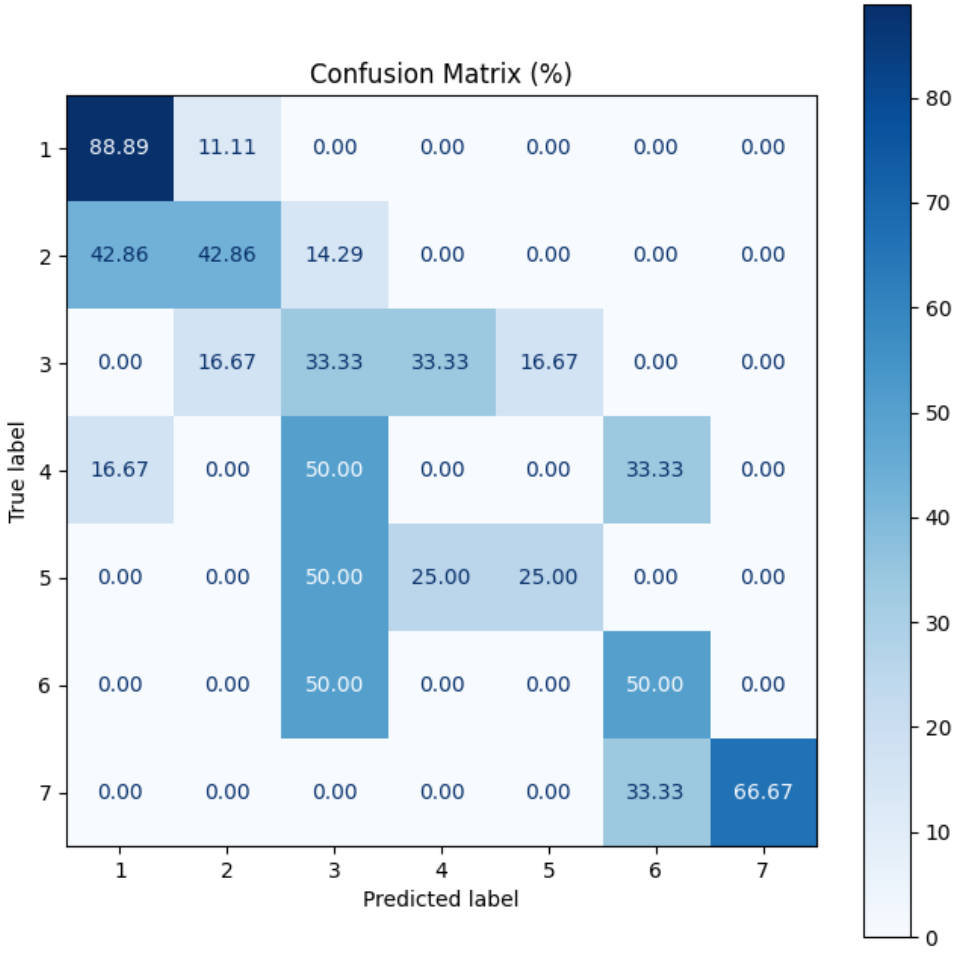

Figure 5.9: The confusion matrix of the *7_categories_model* applied to the test set.

Table 5.2: The classification report of the *7_categories_model* applied to the test set.

Classification report

	precision	recall	f1.score	support
1	0.67	0.89	0.76	9
2	0.60	0.43	0.50	7
3	0.17	0.33	0.22	6
4	0.00	0.00	0.00	6
5	0.67	0.25	0.36	8
6	0.40	0.50	0.44	4
7	1.00	0.67	0.80	3
weighted avg	0.49	0.44	0.44	43

The same steps as described in the previous subsection were applied. Firstly, we mapped the benchline predictions for challenge, competence and workload, resulting in the *F1w_avg* of 0.46, 0.45 and 0.48 respectively (see Appendix U). Sequentially, a RF with default parameters and *n_estimators* based on an *OOB_plot* (2780), resulted in a weighted accuracy of 1.00 for the train set and 0.76 for the test set. Nextly, we performed the same four searches as described in 5.2.1.3 RF randomized search, namely *1. Broad search*, *2. Low max_features*, *3. High max_features*, *4. High min_sample_split*. Again sequentially, the search distributions for the last two searches were based on insights gained from the results of the first four searches (see Appendix V for the search distributions, hyperparameters and accuracies).

As to be expected, the aggregation resulted in models with a significant performance increase compared to the models found for the original seven categories described in the previous subsection. For the best performing model found in search 6. Final grid search (*aggregated_model*), a weighted accuracy of 0.92 was found for the train set and 0.83 for the test set. Table 5.3 and Figure 5.10 display the classification report and confusion matrix respectively of our *aggregated_model*. The f1 scores of class one and two are 0.90 and 0.84 respectively. However, the f1 score of the third class is lower, namely 0.67, with a precision of 1.00, but half of the samples in the test set are assigned class 2, resulting in a recall of 0.50.

5.2.3 Feature selection

5.2.3.1 Feature selection based on Spearman rank-order correlations

In order to find the importance of features we performed RF feature importance analysis with 30 repeats on our *aggregated_model*. Only accuracy decreases were found for the

Table 5.3: The classification report of the *aggregated_model* applied to the test set.

Classification report

	precision	recall	f1.score	support
2	0.93	0.87	0.90	15
3	0.76	0.95	0.84	20
4	1.00	0.50	0.67	8
weighted avg	0.86	0.84	0.83	43

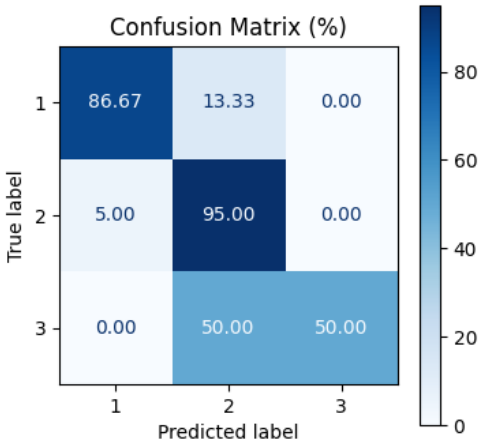

Figure 5.10: The confusion matrix of the *aggregated_model* applied to the test set.

four *features died_at_end*, *pellet_eaten_count*, *PM_is_chased_unsucces_perc*, and *death_freq*. Noteworthy, only part of their permutations resulted in an accuracy decrease, sometimes permutations actually led to a small accuracy increase, see Figure 5.11. These results can be explained by the many collinear features in our dataset, because the permutation of a collinear feature does not highly affect performance while its correlated feature offers the same information [125].

In order to apply permutation feature importance analysis in a meaningful manner, we applied feature selection. We performed hierarchical clustering on the Spearman rank-order correlations. The dendrogram which visually represents the found hierarchical clustering results, is displayed in Figure 5.12. Manual selection of one feature per cluster using a threshold of 0.35 resulted in a set of roughly 50% of the original features. We also created feature sets based on the dendrogram with the thresholds 0.15, 0.25 and 0.45. While selecting the features for the set with threshold 0.45 we noticed it became quite difficult to choose between features in a cluster. Due to this, we decided to continue with the feature set resulting from the 0.35 threshold (for the 62 selected

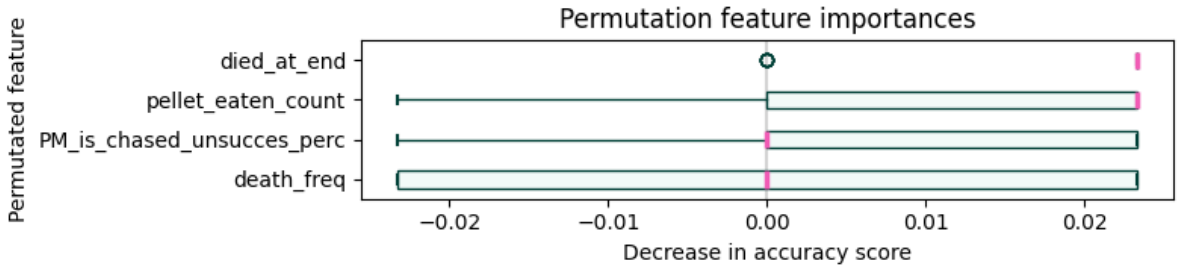

Figure 5.11: A plot of the permutation feature importance analysis results of the four features that resulted in the decreases in accuracy, with 30 permutations per feature.

features see Appendix W).

5.2.3.2 Feature selection based on feature importance

We performed the same steps to train another RF with aggregated labels and with the subset of 62 features (*subset_aggregated_model*) as the steps that were performed for the training of the *7_categories_model* and *aggregated_model* (see 5.2.1 and see Appendix V for the search distributions, the found hyperparameters and all accuracies). The performance of the new *subset_aggregated_model* was found to be equal to our *aggregated_model*, with a weighted accuracy of 0.92 for the train set and 0.83 for the test set and also the same classification report and confusion matrix, see Table 5.3 and Figure 5.10

Because the Gini feature importance analysis has been shown "to be strongly biased" [126] as cited by [122], here we focus on the results of the performed permutation feature importance, see Figure 5.13. We created two small subsets. Firstly, a subset with features for which the permutation analysis of our *subset_aggregated_model* resulted in decrease in accuracy above 0 (PI_PART), consisting of all features displayed in Figure 5.13. Secondly, a subset with features for which the median's decrease in accuracy in the permutation analysis of our *subset_aggregated_model* was above 0.005 (PI_MED), consisting of the eight features highlighted in pink Figure 5.13.

For both subsets we trained models by performing multiple searches. We followed the same training process as described previously, but with less searches by applying insights gained in previous searches (see Appendix V for the search distributions, the found hyperparameters and all accuracies). The model for the PI_MED subset (*pi_med_model*) outperformed the model for the PI_PART subset on the test set, with *F1w_avg*s of 0.83 and 0.80 respectively. Because the performance on the set did not decrease compared to the *subset_aggregated_model* trained on 62 features, we selected the PI_MED as our final subset used to perform a final analysis of the possible contribution of the pupillometry features.

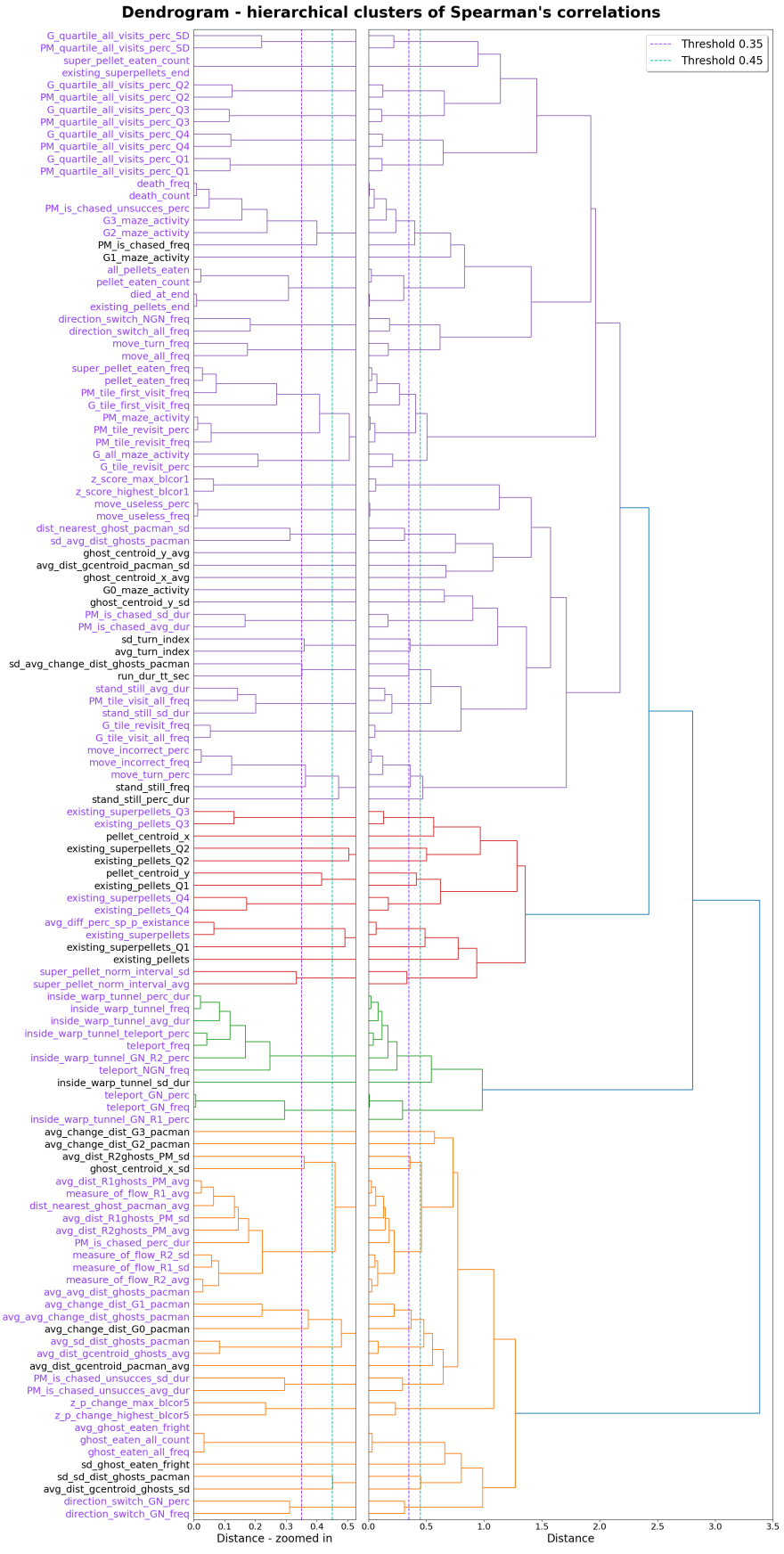

Figure 5.12: The dendrogram visually representing the results of the hierarchical clustering performed on the Spearman rank-order correlations.

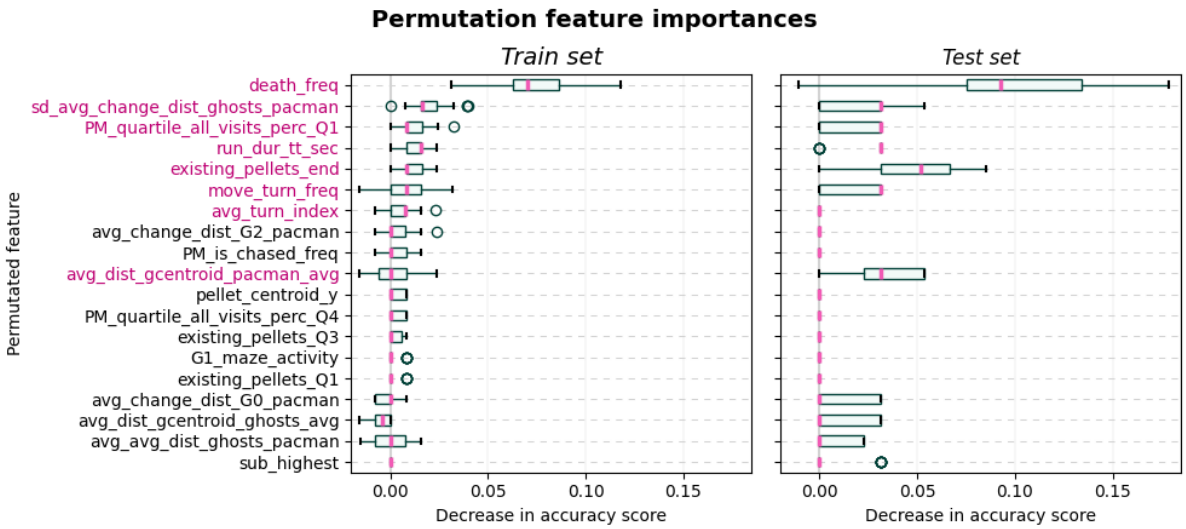

Figure 5.13: The results of the permutation feature importance analysis on the train- and test set, with 30 permutations per feature. The plots only display the features for which at least one permutation resulted in a decrease in accuracy. The features highlighted in pink font indicate the features for which the median's decrease in accuracy score is above 0.005.

5.2.4 Feature importance of pupillometry features

As a final step, we extended the PI_MED subset twice, once with the *sub_highest* and *div_highest* features (PI_MED_PUP_HIGHEST) and once with *sub_peak* and *div_peak* (PI_MED_PUP_PEAK). We did not combine the highest and peak features in one set because of the high correlation between these.

Again, performed multiple searches, following the same training process as described previously (also see Appendix V). The *F1w_avg* for the final models found for the subsets PI_MED_PUP_HIGHEST (*pi_med_pup_highest_model*) and PI_MED_PUP_PEAK (*pi_med_pup_peak_model*) together with the *pi_med_model* found for the unextended subset PI_MED are displayed in Table 5.4. The *F1w_avg*s of the performance of these three classifiers on the test set are 0.83. We found the same *F1w_avg*s for both the *aggregated_model* and *subset_aggregated_model* for the performance on the test set.

Table 5.4: The hyperparameters of the best found models for the *pi_med_model*, *pi_med_pup_highest_model* and *pi_med_pup_peak_model* together with the F1 weighted average on the train and test set.

	max_features	max_samples	min_ samples_leaf	min_ samples_split	Train set F1 weighted average	Test set F1 weighted average
pi_med_model	2	54	2	9	0.87	0.83
pi_med_pup_highest_model	4	50	2	5	0.91	0.83
pi_med_pup_peak_model	2	60	2	6	0.91	0.83

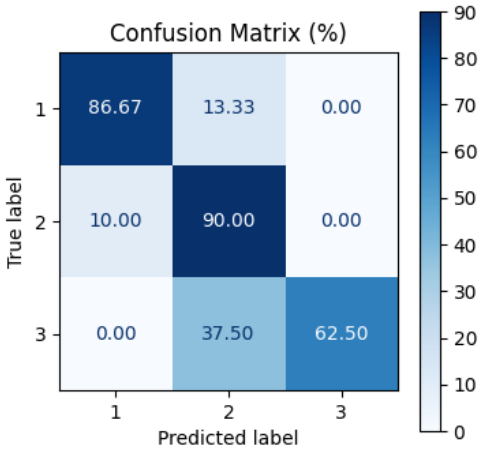

Figure 5.14: The confusion matrix of the *pi_med_model*, *pi_med_pup_highest_model* and *pi_-med_pup_peak_model* applied to the test set.

Not only the found *F1w_avg*s were equal, also the classification reports and confusion matrices for the performance of *pi_med_model*, *pi_med_pup_highest_model* and *pi_med_-pup_peak_model* on the test set were identical, see Table 5.5 and Figure 5.14. Noteworthy, differences in predictions on the test set can be found when comparing the classification report and confusion matrix with those of the *aggregated_model* and *subset_aggregated_model* displayed in Table 5.3 and Figure 5.10. The final three models slightly perform less well in predicting label 2 (the "neutral" category), correctly assigning 90% to 2 instead of 95% and incorrectly assigning 10% to label 1 instead of 5%. However, the final three models perform better in predicting label 3, namely 62.5% percent compared to 50% is assigned the correct label, with 37.5% compared to 50% incorrect assignments to label 2. To sum up, the *F1w_avg*s for the performance on the test set is equal for the final three models, *aggregated_model* and *subset_aggregated_model*. But correct predictions on the test set are slightly more evenly distributed amongst the three labels for the final three models compared to *aggregated_model* and *subset_aggregated_model*.

Table 5.5: The classification report of the *pi_med_model*, *pi_med_pup_highest_model* and *pi_-med_pup_peak_model* applied to the test set.

Classification report

	precision	recall	f1.score	support
2	0.87	0.87	0.87	15
3	0.78	0.90	0.84	20
4	1.00	0.62	0.77	8
weighted avg	0.85	0.84	0.83	43

We performed a permutation feature importance analysis for all three final models (*pi_med_model*, *pi_med_pup_highest_model*, *pi_med_pup_peak_model*), the results are shown in Figure 5.15. For each feature a description is provided in Table 5.6.

The most noticeable is the *death_freq* feature, for which the decrease in accuracy scores (*DA_scores*) are highest in all plots. In other words, the permutation of *death_freq* has the highest impact on the accuracy of all models, thus it can be viewed as the most important feature. Furthermore, the *DA_scores* found for *avg_dist_pacman_gcentroid_avg* and *sd_avg_change_dist_ghosts_pacman* are positive as well in all plots, apart from some zero values and can be viewed to form the second and third most important features. Remarkably, *avg_turn_index*'s *DA_scores* are positive for the train set, but negative for the test set for all models. For the other four non-pupillometry features, *move_turn_freq*, *existing_pellets_end*, *run_dur_sec* and *PM_perc_visits_quartile_Q1* the found *DA_scores* are roughly viewed all near zero for the train set. Noticeably, their *DA_scores* ranges are more diverse on the test set, whereby *existing_pellets_end* and *run_dur_sec* *DA_scores* are higher for all models.

Finally, reviewing the pupil features. The found *DA_scores* of the pupil features *sub_highest*, *div_highest* and *sub_peak* are zero or positive applied on the train set and for *div_peak* also negative *DA_scores* were found. For the test set, only for the *div_highest* pupil feature, all *DA_scores* except for one outlier are zero or higher. Thus in comparison to the other pupillometry features, *div_highest* results in the highest *DA_scores*. The *DA_scores* of *div_peak* in comparison to *sub_peak* on the test set are higher as well.

Table 5.6: The description of the features that are part of the final three feature sets PI_MED, PI_MED_PUP_HIGHEST and PI_MED_PUP_PEAK.

Features	Feature description
death_freq	The frequency of the occurrence of Pac-Man's deaths.
avg_dist_pacman_gcentroid	The average of the distance between Pac-Man and the centroid of the ghosts' locations.
sd_avg_change_dist_ghosts_pacman	The standard deviation of the differences between the average distances between the four ghosts and Pac-Man for each snapshot and the previous snapshot. The displacement after dying is excluded.
avg_turn_index	The average of the turn index. The turn index corresponds to values in the range [0, 7], indicating the timing of a turn; values < 3 correspond to pre-turns and values > 4 to post-turns.
move_turn_freq	The frequency of the occurrence of correct moves, referring to either turns or direction switches. Incorrect user input is excluded.
existing_pellets_end	The number of pellets at the end of the game round.
run_dur_sec	The duration of the active gameplay in a round in seconds. The countdown and death animations are excluded.
PM_perc_visits_quartile_Q1	The percentage of the number of times Pac-Man visits a tile in the top-left quartile compared to all tile visits of Pac-Man.
sub_highest	The average of the z-standardized, *subtractive* corrected, highest pupil dilations.
div_highest	The average of the z-standardized, *divisive* corrected, highest pupil dilations.
sub_peak	The z-standardized, *subtractive* corrected, *peak pupil dilation*.
div_peak	The z-standardized, *divisive* corrected, *peak pupil dilation*.

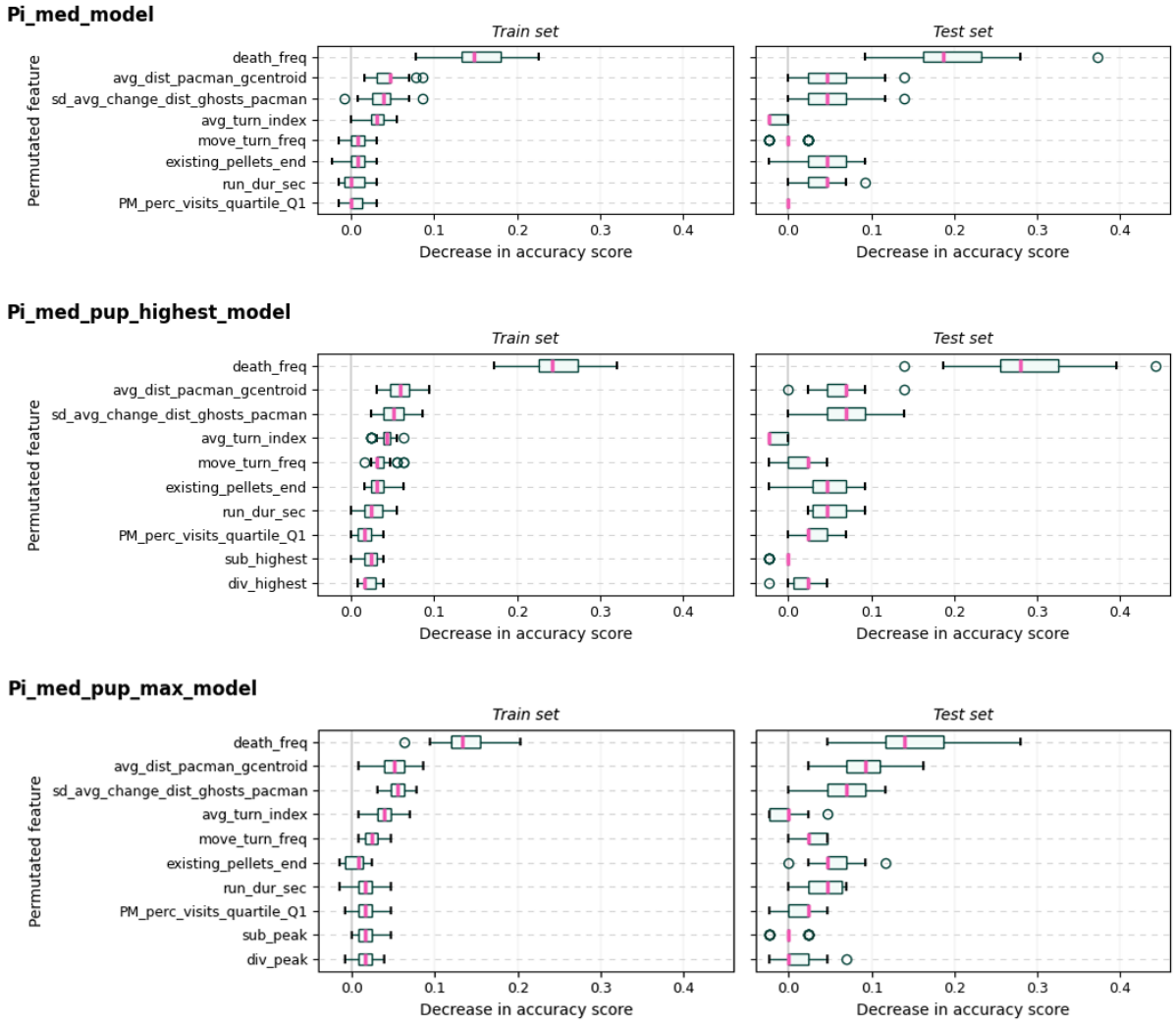

Figure 5.15: The results of the permutation feature importance analysis on the train- and test set, with 30 permutations per feature, performed on the three final models, *pi_med_-model*, *pi_med_pup_highest_model* and *pi_med_pup_peak_model*.

6. Discussion

6.1 Pupillometry features in respect to predicting experience difficulty

6.1.1 Added value of pupillometry features in predicting the experienced difficulty (RQ1)

To ensure a relevant base set of features in order to review the added value of pupillometry features, we first applied feature selection. We validated the resulting set of eight features by comparing the *pi_med_model* trained on this set, to the two models trained on the larger feature sets (*aggregated_model*, *subset_aggregated_model*). The F1 weighted average was equal, thus, no decrease in performance was found. Noteworthy, the *pi_med_model* did result in more evenly distributed correct predictions, which can be explained by the reduction of noisy and irrelevant features. We then used this feature set to review the added value of pupillometry features. For the final models trained with and without pupillometry features, both the F1 weighted average and the confusion matrices were all equal. Thus, no difference in performance was found. This equal performance indicates that the pupillometry features do not improve the accuracy of a RF in predicting the experienced difficulty, thereby answering RQ1[1].

6.1.2 Correlation between pupillometry features and experienced difficulty (RQ2)

To answer RQ2[2] for the pupillometry features *sub_highest* and *sub_peak*, no significant correlation was found. For the pupillometry feature *div_highest* the "strongly disagree" response distribution was found to significantly differ from the distributions of the responses ranging from "slightly disagree" to "agree". For the pupillometry feature *div_peak* the "strongly disagree" response distribution was found to significantly differ to "slightly agree" only.

[1] *"To what extent does the addition of pupillometry features to an input feature set improve the accuracy of a random forest classifier in predicting experienced difficulty?"*

[2] *"To what extent does this pupillometry feature statistically correlate to the self-reported experienced game difficulty?"*

In the context of predicting the experienced difficulty with a RF classifier, the results of our statistical analysis indicate that only the *div_highest* and *div_peak* are informative with respect to identifying the "strongly disagree" to some extent only. The aggregation of the RF target classes, whereby the "strongly disagree" and "disagree" response categories were aggregated, which lowered the informative value of *div_highest* and *div_peak*. This lowered informative value together with the nonsignificant differences between the other distributions for *div_highest* and *div_peak* can explain why the addition of pupillometry features did not result in an accuracy improvement for our RFs.

6.2 Results reviewed in perspective to the experience of difficulty

In this section we review three of our results in perspective to the experience of difficulty. Firstly, to further gain insights with respect to which gameplay experiences are of influence to the level of experienced difficulty, we reviewed the three features that can be viewed as most informative with respect to predicting the experienced difficulty. The permutations feature importance analysis performed on the final three models indicated *death_freq*, *avg_dist_pacman_gcentroid_avg* and *sd_avg_change_dist_ghosts_pacman* as the top three most informative features. The feature *death_freq* represents the frequency of the occurrence of Pac-Man's deaths. This feature was selected in the first step of our feature selection, based on the used distance threshold when reviewing the dendrogram of the hierarchical clusters of the Spearman's correlations it was found to be correlated with four other features. We believe that *death_freq*, together with the two correlated features number of deaths (*death_count*) and the percentage of unsuccessful chases of Pac-Man by ghosts (*PM_is_chased_unsucces_perc*) can be related to the 'experience of failure' (*PM_is_chased_unsucces_perc* is the inverse of successful chases). The two other features correlated to *death_freq*, were the measures of the second and third ghosts' activity in the maze (*G2_maze_activity* and *G3_maze_activity*[3]). To explain this correlation, a death of Pac-Man leads to repositioning ghosts two, three and four in the ghost house where they again wait before restarting the chase on Pac-Man. Thereby, a death of Pac-Man can be expected to be of influence to the ghosts' maze activity.

With respect to the other two most informative features, the second informative feature *avg_dist_pacman_gcentroid_avg* captured the average of the distance between pacman and the centroid of the ghosts locations. The third informative feature *sd_avg_change_dist_ghosts_pacman* refers to the standard deviation of the differences between the average distances between the four ghosts and Pac-Man for each snapshot and its

[3]With maze activity we refer to the entropy of cell visits in a game [102]

previous snapshot[4] (the displacement after dying was excluded). Both features were not found to be correlated with other features, based on the used distance threshold when reviewing the dendrogram of the hierarchical clusters of the Spearman's correlations. We believe both of these features can be related to the 'experience of being chased'.

Secondly, the *followers* resulted in a more difficult experienced level in comparison to the ghost strategy *near-optimal* This is against our expectations which were based on the findings of Yannakakis and Hallam [103], where the *near-optimal* ghosts performed better in experiments against different Pac-Man AI's. We believe that the *near-optimal* ghosts actually impose a higher *task demand*, because these ghosts provide a higher navigational challenge in comparison to the *followers* ghosts. Since, when the *followers* ghosts are tailing Pac-Man, Pac-Man can simply continue eating pellets while being followed by a tail of ghosts, while the *near-optimal* ghosts tend to not allow Pac-Man to escape. Based on our observations, we would describe the behavior of the *followers* ghosts in comparison to the *near-optimal* ghosts to be more directly aiming for Pac-Man. We wonder if this seemingly higher aggressiveness of the *followers* in comparison to the *near-optimal* ghosts led to a higher experience of being chased and thereby led to a higher level of experienced difficulty.

At the start of this research, we expected the experienced challenge responses to reflect the in-game difficulty relative to the participant skills. These expectations were in line with Adam's (2014) view on perceived difficulty; "the relative difficulty minus the player's experience at meeting such challenges". However, after reviewing the results in the perspective of experience, we wonder if the challenges that are most clearly noticeable are of extra influence on the experience of difficulty. This is reminiscent of Csikszentmihalyi [11] view on flow; "it is not only the 'real' challenges presented by the situation that count, but those that the person is aware of".

Another result we believe is relevant to review with respect to the experience of difficulty, is that of the found significant differences of CHAL distributions across the levels of experience with Pac-Man ("How often did you play Pac-Man? (estimation)"). A small side note, we did not find a significant difference for the estimated skill level in Pac-Man, we assume participants were being modest. Participants that estimated a higher experience with Pac-Man provided lower level of agreement CHAL responses in comparison to participants that estimated their experience with Pac-Man lower. Also, we did not find a significant difference in performance when reviewing the frequency of Pac-Man's deaths, thereby no indication of an actual higher skill level was found. Thus, it could be that participants who estimated to have played Pac-Man more

[4] A snapshot was taken each 5 frames at a framerate of 60 frames per second.

often, actually experienced the Pac-Man rounds overall lower regarding CHAL compared to participants with lower Pac-Man experience. Therefore, we wonder if not the actual skills, but the skills we think we have are of influence to the experience of difficulty. Again we are reminded of Csikszentmihalyi's (1990) view on flow, this time; to get in the flow channel not only the actual skills are of influence, but also the skills we think we have.

To sum up, reviewing some of our results in perspective of the experience of difficulty, has led us to question whether two subjective aspects were of influence to the experienced difficulty. Firstly, could it be that the experienced difficulty is not dependent on the 'real' challenges provided by the game, but those challenges the player is (mostly) aware of? Secondly, is the experience of difficulty influenced by the skill we think we have, instead of our actual skill? If this indeed the case, we would question if pupillometry features, which can provide an indicator of mental workload, can serve as an informative feature for the prediction of experienced difficulty.

Following from this, one final chain of thoughts. We imagine the pupillometry features, as indicators of mental workload, to be more informative in a RF classifier predicting in-game difficulty in comparison to predicting the experienced difficulty. Firstly, because an inverted u-curve was found more present for *div_highest* against difficulty than for *div_highest* against experienced difficulty. Secondly, when reviewing a RF classifier predicting the in-game difficulty from the perspective of MWL, a relationship can be identified between in-game difficulty, pupillometry features and *game features*. Here, in-game difficulty can be viewed as *task demand* (*antecedents* of MWL), the pupillometry features can be viewed as indicators of the core of MWL and the *game features* can be viewed as representatives of performance (consequences of MWL). When this relationship can be further substantiated in future work with RFs predicting in-game difficulty, then pupillometry features can serve a DDA system aiming to provide an *optimal load* to a player.

6.3 Limitations

The results of both the statistical analysis and the found decrease in accuracy scores for our four pupillometry features for the *pi_med_pup_highest_model* and *pi_med_pup_-peak_model* suggest that *div_highest* is the most informative pupillometry feature with respect to predicting CHAL, followed by the *div_peak* pupil feature. Thus, with respect to our data, the *divisive* baseline correction resulted in more informative pupillometry features in comparison to the *subtractive* baseline correction. For our *divisive* baseline correctio we used the average of all baselines as recommended by Mathôt et al. [80], while for the *subtractive* baseline correction we used the baseline corresponding to each trial (see 4.5.2.2). Our results could indicate that the found fluctuations in the baseline

recordings resulted in distorted *subtractive* pupillometry features, which can be viewed as a limitation with respect to these features.

When viewing the *div_highest* pupil feature per participant across the aggregated self-reported experienced difficulty, an inverted u-curve was found present for six out of 19 participants. When considering the *subtractive* measure of difficulty as representative of increasing *task demand*, this could indicate that not all participants reached a state of *overload*. However, when viewing the *div_highest* pupil feature per participant across the difficulty levels, the inverted u-curve was found present for 17 participants, even if it was only cautiously for some. For the two other participants for which no inverted u-curve was visible, we assume these participants did not reach the *overload* state. Therefore, for these participants it could be that the pupil measures do not reflect the full dynamic task evoked dynamic range (TERP). Furthermore, for some participants the decline in pupil size occurred cautiously at difficulty level nine, while for others this decline is larger and also present for difficulty level six. These differences lead to different balanced distribution of the pupil measures. In turn, the z-standardized normalization could have led to a distortion of the data, which could have lowered the informative value of our pupillometry features. With respect to future work, we believe it would be interesting to apply the cognitive evoked normalization proposed by Winn et al. [24], whereby normalization is based on the dynamic range of the task evoked response.

Another limitation could be the possible low level of immersion that pacman evoked in (some) players/participants during our experiment. We believe one can question to what extent a participant goes when playing an arcade game in a darkened room with a red light for an experiment. Since, the theory of mental workload states that the deployment of resources is under voluntary control [56], it could be possible some participants did not choose to allocate the full required amount of *mental resources* for all levels. Some observations made during the experiment substantiate this view. During the procedure, one participant shared by thinking out loud that he simply stopped spending effort on the game when they noticed that the ghosts were faster in comparison to Pac-Man (level 9). Some other participants were openly figuring out the differences between levels for some of the levels, thereby possibly spending extra *mental resources* of influence to the pupil size. This could explain seemingly partly random data and outliers in the *div_highest* plots and the less consistent inverted u-curves for some with respect to the difficulty levels. Furthermore, this could have influenced and thereby lowered the informative value of our pupillometry features. In previous work, a higher engagement and motivation was already found to correlate with a higher pupil size [22], [88]. Due to our resultings and previous findings we believe a repetition of the current research with a game with high immersion would be interesting in future work.

Participants that estimated their experience with Pac-Man ("How often did you play Pac-Man? (estimation)") higher in comparison to other participants overall provided lower level of agreement responses for the CHAL experience statements. This could not be explained by a difference in performance. With respect to limitations, for these participants, gameplay, game context and pupillometry features similar to that of other participants, could be reflected by lower CHAL responses. If so, this could have lowered the maximum achievable accuracy of classifiers to predict CHAL. Such influence of interpersonal differences in ratings is a known limitation of rating-based reporting [17]. In future work, a repetition of the current work with pairwise ranking instead of rating-based reporting could be interesting in order to minimize the influence of interpersonal differences.

The beforehand estimated order of difficulty of the nine implemented difficulties was found to differ from the experienced order of difficulty based on the CHAL responses. This could have led to a less balanced order in the three groups' experimental conditional orders. The comparison of the averaged responses per group with respect to those of the other groups, showed that group A and B provided lower and higher answers for some levels respectively. However, we could not distinguish a clear pattern across the responses per group in comparison to the other groups. Therefore, we did not identify a specific impact of the groups' condition order on the reported game experience. Because no significant differences for the responses across the three participant groups were found, we expect that the order had no high influence on the overall CHAL responses. It is however noteworthy as a limitation of this research, also because of the unbalanced spread of participants amongst the three groups due to the exclusion of data.

Due to the unforeseen synchronization issue between the clocks used by the unity engine and eye tracker server we could not focus on data of the chase state only. Instead, we based two pupillometry features on the 30% highest pupil sizes per round of Pac-Man. We believe this was a good alternative strategy. But we imagine pupillometry features extracted from the chase states to be higher correlated to *task demand* and possibly also to the experienced difficulty. We believe it is interesting in future work to reuse our data and then synchronize with to be written custom functions to retrieve pupillometry features reflecting the chase state only.

6.4 Future work

Our results indicate that pupillometry features, which were found to be informative with respect to *task demand* and mental workload in previous work, are less informative with respect to experienced difficulty. However, the shared limitations (see 6.3) could have led to less informative pupillometry features and / or experienced chal-

lenge responses, resulting in no found added value of pupillometry features in predicting experienced difficulty. Therefore, as a first step to gain further insight into the potential value of pupillometry features in the context of predicting difficulties, we believe it is interesting to perform a small follow up study reusing the current data and taking insights gained into account to minimize the influence of the shared limitations.

In this follow up, new pupillometry features can be constructed that only reflect the chase state. Similar to our application of *divisive* baseline correction, the averaged baseline can be also used for the *subtractive* baseline, to minimize a possible impact of the found fluctuations in the baseline recordings. Furthermore, cognitive evoked normalization can be applied instead of applying z-standardized normalization, by only using the data of 17 participants for which an inverted u-curve was found. Since the pupil size recordings of these participants can be expected to reflect the full TERP range per participant, these pupil size recordings can be used for cognitive evoked normalization.

Furthermore, in this proposed pilot study it would be interesting to also train RF classifiers to predict the in-game difficulty. This in-game difficulty could be based on the averaged CHAL responses for each nine difficulty settings, assuming linearity between the labels on the likert scale. This would then allow for a comparison between the pupillometry features as indicator of in-game difficulty (*task demand*) versus as indicator of the experienced difficulty. Additionally, a RF classifier predicting in-game difficulty could provide further insights with respect to the relationship between in-game difficulty (MWL *antecedents*), pupillometry features (core of MWL) and *game features* (MWL*consequences*).

Additionally, in the current work we did not construct features from other recorded eye tracker measurements like gaze position and number of blinks. This data is available in our data set and can be considered as well in the proposed follow up study. Similarly, in the current work we did not fully review the participants' COMP and WRKL responses. We imagine extended data analysis of these responses could offer additional insights with respect to experienced difficulty and the in-game difficulty.

Apart from the proposed follow up, a repetition of the current work using a game with high immersion as a testbed game with pairwise ranking would be interesting. We expect a game with higher immersion to lead to a higher engagement, leading to more representative pupillometry features. As a side note, a higher engagement could also lead to a higher arousal which is also of influence to the psychosensory pupil response, which should be accounted for. Pair-wise ranking could lead to both more representative experienced difficulty labels, since these are found to be less biased in comparison to the applied rating-based reporting in the current work.

As a final note with respect to future work, we here share a brief reflection on the applied step-by-step processes in the selection of *game features* and *adaptive game components* (See 3.3, 3.4 and 4.5.1). Where the final three selected *adaptive game components* can be viewed as pretty straightforward and would maybe not have required such a thorough selection process, we experienced the step-by-step feature selection process as extremely valuable. We believe we would not have constructed some of the features that were part of our final set without this step-by-step process. For example, the *avg_turn_index*'s feature, which represents the timing of turn and was correlated with *sd_turn_index*, was constructed by reviewing our mapped *challenge hierarchy* of Pac-Man. Another example is that of a feature based on the delta distance used in the online learning fitness function in Yannakakis and Hallam [102]. Due to our step-by-step process, we not only considered the average delta distance between the ghosts and Pac-Man (*avg_avg_change_dist_ghosts_pacman*), but also the sd, which resulted in *sd_avg_change_dist_ghosts_pacman*, part of the top three informative features. In future work, it could be interesting to further develop the applied step-by-step process into a design guide for feature selection.

7. Conclusion

In this research, we aimed to provide insights regarding the added value of the commonly used pupillometry features to predict self-reported experienced difficulty. A user study was conducted during which participants played multiple rounds of Pac-Man at different difficulties. Gameplay-, game context- and pupillometry data were collected together with participants' responses regarding game experience amongst which experienced difficulty. Based on this data a feature set was constructed, with amongst others four pupillometry features. Namely, both the *subtractive* and *divisive* corrected average of the 30% highest measured pupil sizes (*sub_highest*, *div_highest*) and both the *subtractive* and *divisive* corrected peak pupil size (*sub_peak*, *div_peak*).

Multiple random forest classifiers were trained on different feature subsets, with and without pupillometry features, to predict experienced difficulty. We found that the addition of pupillometry features did not lead to a performance improvement of the RF classifiers. These results are supported by the results of our data analysis, whereby no significant relationship was found for the *sub_highest* and *sub_peak* features across the self-reported experienced difficulty. For both *div_highest* and *div_peak*, the response "strongly disagree" was found to significantly differ from some of the other responses.

Although our results did not confirm an added value of pupillometry in predicting experienced difficulty, we do believe the research to be valuable in the context of DDA. A review of our results in perspective of the experience of difficulty has led us to question whether two subjective aspects were of influence to the experienced difficulty. Our results suggest pupillometry features to be more informative with respect to predicting in-game difficulty. Based on our findings, we can see that an important next step is to further research the difference between pupillometry features as informative features in predicting experienced difficulty versus the prediction of *task demand* viewed as in-game difficulty. We provided insights that can inform such future work and proposed a follow up study for which our data can be reused.

Pupillometry features might not turn out to be informative for a DDA system that aims to keep the player in the channel of flow. Still, we can imagine pupillometry features to be informative for a DDA system that aims to challenge the player to the maximum effort. We foresee applications of pupillometry features in a training setting for esports gamers, education- and rehabilitation games.

References

[1] P. Sweetser and P. Wyeth, "Gameflow: A model for evaluating player enjoyment in games," *Computers in Entertainment (CIE)*, vol. 3, no. 3, pp. 3–3, 2005.

[2] R. Koster, *Theory of fun for game design*. " O'Reilly Media, Inc.", 2013.

[3] E. Lach, "Dynamic difficulty adjustment for serious game using modified evolutionary algorithm," in *Artificial Intelligence and Soft Computing: 16th International Conference, ICAISC 2017, Zakopane, Poland, June 11-15, 2017, Proceedings, Part I 16*, Springer, 2017, pp. 370–379.

[4] R. Hunicke and V. Chapman, "Ai for dynamic difficulty adjustment in games. 2004," *Association for the Advancement of Artificial Intelligence (AAAI)*, pp. 2–5, 2004.

[5] A. Glassner, *Interactive storytelling: Techniques for 21st century fiction*. AK Peters/CRC Press, 2017.

[6] G. Andrade, G. Ramalho, H. Santana, and V. Corruble, "Extending reinforcement learning to provide dynamic game balancing," in *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 2005, pp. 7–12.

[7] A. Stein, Y. Yotam, R. Puzis, G. Shani, and M. Taieb-Maimon, "Eeg-triggered dynamic difficulty adjustment for multiplayer games," *Entertainment computing*, vol. 25, pp. 14–25, 2018.

[8] B. Magerko, "Adaptation in digital games," *Computer*, vol. 41, no. 6, pp. 87–89, 2008.

[9] B. Harrison and D. L. Roberts, "Analytics-driven dynamic game adaption for player retention in scrabble," in *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, IEEE, 2013, pp. 1–8.

[10] D. Afergan *et al.*, "Dynamic difficulty using brain metrics of workload," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2014, pp. 3797–3806.

[11] M. Csikszentmihalyi, *Flow: The psychology of optimal experience*. Harper & Row, New York, 1990.

[12] E. Adams, *Fundamentals of game design*. Pearson Education, 2014.

[13] G. N. Yannakakis and J. Togelius, "Experience-driven procedural content generation," *IEEE Transactions on Affective Computing*, vol. 2, no. 3, pp. 147–161, 2011.

[14] M. Zohaib, "Dynamic difficulty adjustment (dda) in computer games: A review," *Advances in Human-Computer Interaction*, vol. 2018, no. 1, p. 5 681 652, 2018.

[15] R. Houlette, "Player modeling for adaptive games," *AI game programming wisdom*, pp. 557–566, 2003.

[16] G. N. Yannakakis, P. Spronck, D. Loiacono, and E. André, "Player modeling," English, in *"Artificial and computational intelligence in games"*, ser. Dagstuhl

Follow-Ups 6, S. Lucas, M. Mateas, M. Preuss, P. Spronck, and J. Togelius, Eds., Dagstuhl Publishing, 2013, ISBN: 9783939897620.

[17] G. N. Yannakakis and J. Togelius, *Artificial intelligence and games*. Springer, 2018, vol. 2.

[18] R. Hunicke, "The case for dynamic difficulty adjustment in games," in *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, 2005, pp. 429–433.

[19] J. L. Soler-Dominguez, J. D. Camba, M. Contero, and M. Alcañiz, "A proposal for the selection of eye-tracking metrics for the implementation of adaptive gameplay in virtual reality based games," in *Virtual, Augmented and Mixed Reality: 9th International Conference, VAMR 2017, Held as Part of HCI International 2017, Vancouver, BC, Canada, July 9-14, 2017, Proceedings 9*, Springer, 2017, pp. 369–380.

[20] M. Köles, L. Szegletes, and B. Forstner, "Towards a physiology based difficulty control system for serious games," in *2015 6th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, IEEE, 2015, pp. 323–328.

[21] R. Mallick, D. Slayback, J. Touryan, A. J. Ries, and B. J. Lance, "The use of eye metrics to index cognitive workload in video games," in *2016 IEEE second workshop on eye tracking and visualization (etvis)*, IEEE, 2016, pp. 60–64.

[22] C. Strauch, M. Barthelmaes, E. Altgassen, and A. Huckauf, "Pupil dilation fulfills the requirements for dynamic difficulty adjustment in gaming on the example of pong," in *ACM Symposium on Eye Tracking Research and Applications*, 2020, pp. 1–9.

[23] H. Mitre-Hernandez, R. C. Carrillo, C. Lara-Alvarez, *et al.*, "Pupillary responses for cognitive load measurement to classify difficulty levels in an educational video game: Empirical study," *JMIR Serious Games*, vol. 9, no. 1, e21620, 2021.

[24] M. B. Winn, D. Wendt, T. Koelewijn, and S. E. Kuchinsky, "Best practices and advice for using pupillometry to measure listening effort: An introduction for those who want to get started," *Trends in hearing*, vol. 22, p. 2 331 216 518 800 869, 2018.

[25] S. D. Goldinger and M. H. Papesh, "Pupil dilation reflects the creation and retrieval of memories," *Current directions in psychological science*, vol. 21, no. 2, pp. 90–95, 2012.

[26] J. Chen, "Flow in games (and everything else)," *Communications of the ACM*, vol. 50, no. 4, pp. 31–34, 2007.

[27] E. Adams and J. Dormans, *Game mechanics: advanced game design*. New Riders, 2012.

[28] R. W. Fernandes and G. Levieux, "-logit: Dynamic difficulty adjustment using few data points," in *Joint International Conference on Entertainment Computing and Serious Games*, Springer, 2019, pp. 158–171.

[29] R. Lopes and R. Bidarra, "Adaptivity challenges in games and simulations: A survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 2, pp. 85–99, 2011.

[30] B. Bontchev, "Adaptation in affective video games: A literature review," *Cybernetics and Information Technologies*, vol. 16, no. 3, pp. 3–34, 2016.

[31] G. K. Sepulveda, F. Besoain, and N. A. Barriga, "Exploring dynamic difficulty adjustment in videogames," in *2019 IEEE CHILEAN Conference on Electrical,*

Electronics Engineering, Information and Communication Technologies (CHILECON), IEEE, 2019, pp. 1–6.

[32] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience in super mario bros," in *2009 IEEE Symposium on Computational Intelligence and Games*, IEEE, 2009, pp. 132–139.

[33] C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling player experience for content creation," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 1, pp. 54–67, 2010.

[34] N. Shaker, G. Yannakakis, and J. Togelius, "Towards automatic personalized content generation for platform games," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 6, 2010, pp. 63–68.

[35] P. D. Paraschos and D. E. Koulouriotis, "Game difficulty adaptation and experience personalization: A literature review," *International Journal of Human–Computer Interaction*, vol. 39, no. 1, pp. 1–22, 2023.

[36] J. C. Lopes and R. P. Lopes, "A review of dynamic difficulty adjustment methods for serious games," in *International Conference on Optimization, Learning Algorithms and Applications*, Springer, 2022, pp. 144–159.

[37] C. V. N. Segundo, K. Emerson, A. Calixto, and R. Gusmao, "Dynamic difficulty adjustment through parameter manipulation for space shooter game," *Proceedings of SB games*, 2016.

[38] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma, "Online adaptation of game opponent ai in simulation and in practice," in *Proceedings of the 4th International Conference on Intelligent Games and Simulation*, 2003, pp. 93–100.

[39] P. Spronck, I. Sprinkhuizen-Kuyper, and E. Postma, "Online adaptation of game opponent ai with dynamic scripting," *International Journal of Intelligent Games and Simulation*, vol. 3, no. 1, pp. 45–53, 2004.

[40] Y.-N. Li, C. Yao, D.-J. Li, and K. Zhang, "Adaptive difficulty scales for parkour games," *Journal of Visual Languages & Computing*, vol. 25, no. 6, pp. 868–878, 2014.

[41] P. M. Blom *et al.*, "Towards personalised gaming via facial expression recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 10, 2014, pp. 30–36.

[42] C. H. Tan, K. C. Tan, and A. Tay, "Dynamic game difficulty scaling using adaptive behavior-based ai," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 4, pp. 289–301, 2011.

[43] N. Shaker, S. Asteriadis, G. N. Yannakakis, and K. Karpouzis, "Fusing visual and behavioral cues for modeling user experience in games," *IEEE transactions on cybernetics*, vol. 43, no. 6, pp. 1519–1531, 2013.

[44] G. N. Yannakakis and J. Hallam, "Real-time game adaptation for optimizing player satisfaction," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 2, pp. 121–133, 2009.

[45] G. Chanel and P. Lopes, "User evaluation of affective dynamic difficulty adjustment based on physiological deep learning," in *International Conference on Human-Computer Interaction*, Springer, 2020, pp. 3–23.

[46] P. M. Blom, S. Bakkes, and P. Spronck, "Modeling and adjusting in-game difficulty based on facial expression analysis," *Entertainment Computing*, vol. 31, p. 100 307, 2019.

[47] S. Kazmi and I. J. Palmer, "Action recognition for support of adaptive gameplay: A case study of a first person shooter," *International Journal of Computer Games Technology*, vol. 2010, no. 1, p. 536 480, 2010.
[48] H. Fernandez, K. Mikami, and K. Kondo, "Adaptable game experience based on player's performance and eeg," in *2017 Nicograph International (NicoInt)*, IEEE, 2017, pp. 1–8.
[49] P. Rani, N. Sarkar, and C. Liu, "Maintaining optimal challenge in computer games through real-time physiological feedback," *Foundations of Augmented Cognition*, vol. 184, 2005.
[50] F. O. Flemisch and R. Onken, "Open a window to the cognitive work process! pointillist analysis of man–machine interaction," *Cognition, Technology & Work*, vol. 4, pp. 160–170, 2002.
[51] M. A. Vidulich and P. S. Tsang, "Mental workload and situation awareness," *Handbook of human factors and ergonomics*, pp. 243–273, 2012.
[52] M. S. Young, K. A. Brookhuis, C. D. Wickens, and P. A. Hancock, "State of science: Mental workload in ergonomics," *Ergonomics*, vol. 58, no. 1, pp. 1–17, 2015.
[53] C. D. Wickens, *Engineering psychology and human performance*. Upper Saddle River, NJ : Prentice Hall, 2000.
[54] B. Cain, "A review of the mental workload literature," *DTIC Document*, 2007.
[55] L. Longo, "A defeasible reasoning framework for human mental workload representation and assessment," *Behaviour & Information Technology*, vol. 34, no. 8, pp. 758–786, 2015.
[56] D. De Waard and V. Studiecentrum, "The measurement of drivers' mental workload. groningen university," *Traffic Research Center Netherlands*, p. 125, 1996.
[57] R. Parasuraman and D. Caggiano, "Mental workload," in *Encyclopedia of the Human Brain*, vol. 3, San Diego, Calif. : Academic Press, 2002.
[58] D. Kahneman, *Attention and effort*. prentice-Hall, 1973.
[59] C. D. Wickens, "Multiple resources and performance prediction," *Theoretical issues in ergonomics science*, vol. 3, no. 2, pp. 159–177, 2002.
[60] C. D. Wickens, "Multiple resources and mental workload," *Human factors*, vol. 50, no. 3, pp. 449–455, 2008.
[61] D. A. Norman and D. G. Bobrow, "On data-limited and resource-limited processes," *Cognitive psychology*, vol. 7, no. 1, pp. 44–64, 1975.
[62] R. Parasuraman, T. B. Sheridan, and C. D. Wickens, "Situation awareness, mental workload, and trust in automation: Viable, empirically supported cognitive engineering constructs," *Journal of cognitive engineering and decision making*, vol. 2, no. 2, pp. 140–160, 2008.
[63] B. B. Van Acker, D. D. Parmentier, P. Vlerick, and J. Saldien, "Understanding mental workload: From a clarifying concept analysis toward an implementable framework," *Cognition, technology & work*, vol. 20, pp. 351–365, 2018.
[64] R. McKendrick, B. Feest, A. Harwood, and B. Falcone, "Theories and methods for labeling cognitive workload: Classification and transfer learning," *Frontiers in human neuroscience*, vol. 13, p. 295, 2019.
[65] L. O. Walker, K. C. Avant, *et al.*, *Strategies for theory construction in nursing*. Pearson/Prentice Hall Upper Saddle River, NJ, 2005, vol. 4.

[66] R. North, S. Stackhouse, and K. Graffunder, "Performance, physiological, and oculometer evaluation of vtol landing displays," NASA, Tech. Rep., 1979.
[67] J. Tole, A. Stephens, R. Harris Sr, and A. Ephrath, "Quantification of pilot workload via instrument scan," NASA, Tech. Rep., 1982.
[68] R. D. O'DONNELL, "Workload assessment methodology," *Cognitive processes and performance*, 1986.
[69] J. R. Wilson and J. A. Rajan, "Human-machine interfaces for systems control," *Evaluation of human work: a practical ergonomics methodology*, pp. 357–405, 1995.
[70] M. S. Young and N. A. Stanton, "Malleable attentional resources theory: A new explanation for the effects of mental underload on performance," *Human factors*, vol. 44, no. 3, pp. 365–375, 2002.
[71] R. M. Yerkes, J. D. Dodson, *et al.*, "The relation of strength of stimulus to rapidity of habit-formation," *Journal of Comparative Neurology and Psychology*, pp. 459–482, 1908.
[72] D. Meister, *Behavioral foundations of system development.* John Wiley & Sons, 1976.
[73] C. D. Wickens, "Mental workload: Assessment, prediction and consequences," in *Human Mental Workload: Models and Applications: First International Symposium, H-WORKLOAD 2017, Dublin, Ireland, June 28-30, 2017, Revised Selected Papers 1*, Springer, 2017, pp. 18–29.
[74] D. L. Parks and G. P. Boucek Jr, "Workload prediction, diagnosis, and continuing challenges," in *Applications of human performance models to system design*, Springer, 1989, pp. 47–63.
[75] S. G. Hart, "Nasa-task load index (nasa-tlx); 20 years later," in *Proceedings of the human factors and ergonomics society annual meeting*, Sage publications Sage CA: Los Angeles, CA, vol. 50, 2006, pp. 904–908.
[76] G. Matthews, L. Reinerman-Jones, R. Wohleber, J. Lin, J. Mercado, and J. Abich, "Workload is multidimensional, not unitary: What now?" In *Foundations of Augmented Cognition: 9th International Conference, AC 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2–7, 2015, Proceedings 9*, Springer, 2015, pp. 44–55.
[77] A. F. Kramer, "Physiological metrics of mental workload: A review of recent progress," *Multiple task performance*, pp. 279–328, 1991.
[78] P. Van der Wel and H. Van Steenbergen, "Pupil dilation as an index of effort in cognitive control tasks: A review," *Psychonomic bulletin & review*, vol. 25, pp. 2005–2015, 2018.
[79] S. Mathôt, "Pupillometry: Psychology, physiology, and function," *Journal of cognition*, vol. 1, no. 1, 2018.
[80] S. Mathôt, J. Fabius, E. Van Heusden, and S. Van der Stigchel, "Safe and sensible preprocessing and baseline correction of pupil-size data," *Behavior research methods*, vol. 50, pp. 94–106, 2018.
[81] E. H. Hess and J. M. Polt, "Pupil size in relation to mental activity during simple problem-solving," *Science*, vol. 143, no. 3611, pp. 1190–1192, 1964.
[82] J. Beatty, "Task-evoked pupillary responses, processing load, and the structure of processing resources," *Psychological bulletin*, vol. 91, no. 2, p. 276, 1982.
[83] S. T. Iqbal, X. S. Zheng, and B. P. Bailey, "Task-evoked pupillary response to mental workload in human-computer interaction," in *CHI'04 extended abstracts on Human factors in computing systems*, 2004, pp. 1477–1480.

[84] W. S. Peavler, "Individual differences in pupil size and performance," in *Pupillary dynamics and behavior*, Springer, 1974, pp. 159–175.
[85] E. Granholm, R. F. Asarnow, A. J. Sarkin, and K. L. Dykes, "Pupillary responses index cognitive resource limitations," *Psychophysiology*, vol. 33, no. 4, pp. 457–461, 1996.
[86] G. K. Poock, "Information processing vs pupil diameter," *Perceptual and motor skills*, vol. 37, no. 3, pp. 1000–1002, 1973.
[87] M. M. Bradley, L. Miccoli, M. A. Escrig, and P. J. Lang, "The pupil as a measure of emotional arousal and autonomic activation," *Psychophysiology*, vol. 45, no. 4, pp. 602–607, 2008.
[88] M. O. Gutjahr, W. Ellermeier, S. Hardy, S. Göbel, and J. Wiemeyer, "The pupil response as an indicator of user experience in a digital exercise game," *Psychophysiology*, vol. 56, no. 10, e13418, 2019.
[89] J. Xu *et al.*, "Pupillary response based cognitive workload index under luminance and emotional changes," in *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '11, Vancouver, BC, Canada: Association for Computing Machinery, 2011, pp. 1627–1632, ISBN: 9781450302685. DOI: 10.1145/1979742.1979819. [Online]. Available: https://doi.org/10.1145/1979742.1979819.
[90] R. F. Stanners, M. Coulter, A. W. Sweet, and P. Murphy, "The pupillary response as an indicator of arousal and cognition," *Motivation and Emotion*, vol. 3, pp. 319–340, 1979.
[91] S. Mathôt, J.-B. Melmi, and E. Castet, "Intrasaccadic perception triggers pupillary constriction," *PeerJ*, vol. 3, e1150, 2015.
[92] P. Mannaru, B. Balasingam, K. Pattipati, C. Sibley, and J. T. Coyne, "Performance evaluation of the gazepoint gp3 eye tracking device based on pupil dilation," in *Augmented Cognition. Neurocognition and Machine Learning: 11th International Conference, AC 2017, Held as Part of HCI International 2017, Vancouver, BC, Canada, July 9-14, 2017, Proceedings, Part I 11*, Springer, 2017, pp. 166–175.
[93] J. Klingner, B. Tversky, and P. Hanrahan, "Effects of visual and verbal presentation on cognitive load in vigilance, memory, and arithmetic tasks," *Psychophysiology*, vol. 48, no. 3, pp. 323–332, 2011.
[94] H. H. Telek, H. Erdol, A. Turk, *et al.*, "The effects of age on pupil diameter at different light amplitudes," *Beyoglu Eye J*, vol. 3, no. 2, pp. 80–85, 2018.
[95] G. Matthews, J. De Winter, and P. A. Hancock, "What do subjective workload scales really measure? operational and representational solutions to divergence of workload measures," *Theoretical issues in ergonomics science*, vol. 21, no. 4, pp. 369–396, 2020.
[96] S. T. Iqbal, P. D. Adamczyk, X. S. Zheng, and B. P. Bailey, "Towards an index of opportunity: Understanding changes in mental workload during task execution," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2005, pp. 311–320.
[97] E. Muñoz-de-Escalona and J. J. Cañas, "Latency differences between mental workload measures in detecting workload changes," in *Human Mental Workload: Models and Applications: Second International Symposium, H-WORKLOAD 2018, Amsterdam, The Netherlands, September 20-21, 2018, Revised Selected Papers 2*, Springer, 2019, pp. 131–146.

[98] M. A. Just and P. A. Carpenter, "The intensity dimension of thought: Pupillometric indices of sentence processing," in *Reading and language processing*, Psychology Press, 2013, pp. 182–211.

[99] J. Keller, H. Bless, F. Blomann, and D. Kleinböhl, "Physiological aspects of flow experiences: Skills-demand-compatibility effects on heart rate variability and salivary cortisol," *Journal of Experimental Social Psychology*, vol. 47, no. 4, pp. 849–852, 2011.

[100] G. Chanel, C. Rebetez, M. Bétrancourt, and T. Pun, "Boredom, engagement and anxiety as indicators for adaptation to difficulty in games," in *Proceedings of the 12th international conference on Entertainment and media in the ubiquitous era*, 2008, pp. 13–17.

[101] J. Pittman. "The pac-man dossier," Accessed: Jul. 1, 2023. [Online]. Available: https://pacman.holenet.info/.

[102] G. N. Yannakakis and J. Hallam, "Evolving Opponents for Interesting Interactive Computer Games," in *From Animals to Animats 8: Proceedings of the Eighth International Conference on the Simulation of Adaptive Behavior*, The MIT Press, Jun. 2004, ISBN: 9780262291446. DOI: 10.7551/mitpress/3122.003.0062. eprint: https://direct.mit.edu/book/chapter-pdf/2310761/9780262291446_cci.pdf. [Online]. Available: https://doi.org/10.7551/mitpress/3122.003.0062.

[103] G. N. Yannakakis and J. Hallam, "A generic approach for generating interesting interactive pac-man opponents," in *IEEE Symposium on Computational Intelligence and Games (CIG05), Colchester*, Essex University, 2005, pp. 94–101.

[104] N. Beume *et al.*, "Measuring flow as concept for detecting game fun in the pac-man game," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, IEEE, 2008, pp. 3448–3455.

[105] X. Liu *et al.*, "To create intelligent adaptive game opponent by using monte-carlo for the game of pac-man," in *2009 Fifth International Conference on Natural Computation*, IEEE, vol. 5, 2009, pp. 598–602.

[106] X. Li *et al.*, "To create dda by the approach of ann from uct-created data," in *2010 international Conference on computer application and system modeling (ICCASM 2010)*, IEEE, vol. 8, 2010, pp. V8–475.

[107] J. Fraser, M. Katchabaw, and R. E. Mercer, "An experimental approach to identifying prominent factors in video game difficulty," in *Advances in Computer Entertainment: 10th International Conference, ACE 2013, Boekelo, The Netherlands, November 12-15, 2013. Proceedings 10*, Springer, 2013, pp. 270–283.

[108] A. Ebrahimi and M.-R. Akbarzadeh-T, "Dynamic difficulty adjustment in games by using an interactive self-organizing architecture," in *2014 Iranian conference on intelligent systems (ICIS)*, IEEE, 2014, pp. 1–6.

[109] M. Wolterink and S. Bakkes, "Towards explainable prediction of player frustration in video games.," in *Proceedings of the 16th International Conference on the Foundations of Digital Games*, 2021, pp. 1–10.

[110] E. De Bono, *De Bono's thinking course*. Pearson Education, 1994.

[111] C. C. Preston and A. M. Colman, "Optimal number of response categories in rating scales: Reliability, validity, discriminating power, and respondent preferences," *Acta psychologica*, vol. 104, no. 1, pp. 1–15, 2000.

[112] K. Finstad, "Response interpolation and scale sensitivity: Evidence against 5-point scales," *Journal of usability studies*, vol. 5, no. 3, pp. 104–110, 2010.

[113] M. Alberts, E. Smets, J. Vercoulen, B. Garssen, and G. Bleijenberg, "'verkorte vermoeidheidsvragenlijst': Een praktisch hulpmiddel bij het scoren van vermoeidheid," *Nederlands Tijdschrift voor Geneeskunde*, vol. 141, no. 31, 1997.

[114] E. A. Matthews, I. Koleva, and S. Basnet, "Consistent gaming skill demographics in academic research," *IADIS International Journal on Computer Science & Information Systems*, vol. 17, no. 2, pp. 16–30, 2022.

[115] A. Haider *et al.*, "Minipxi: Development and validation of an eleven-item measure of the player experience inventory," *Proceedings of the ACM on Human-Computer Interaction*, vol. 6, no. CHI PLAY, pp. 1–26, 2022.

[116] W. A. IJsselsteijn, Y. A. De Kort, and K. Poels, "The game experience questionnaire," 2013.

[117] Scikit-learn. "Randomforestclassifier," Accessed: Aug. 20, 2024. [Online]. Available: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`.

[118] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[119] T. Hastie, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009.

[120] M. Schonlau, "Applied statistical learning," *Statistics and computing. Springer: Cham*, pp. 143–160, 2023.

[121] P. Probst and A.-L. Boulesteix, "To tune or not to tune the number of trees in random forest," *Journal of Machine Learning Research*, vol. 18, no. 181, pp. 1–18, 2018.

[122] P. Probst, M. N. Wright, and A.-L. Boulesteix, "Hyperparameters and tuning strategies for random forest," *Wiley Interdisciplinary Reviews: data mining and knowledge discovery*, vol. 9, no. 3, 2019.

[123] M. Segal, "Machine learning benchmarks and random forest regression," *Technical Report, Center for Bioinformatics & Molecular Biostatistics, University of California, San Francisco*, May 2003.

[124] S. Bernard, L. Heutte, and S. Adam, "Influence of hyperparameters on random forest accuracy," in *Multiple Classifier Systems: 8th International Workshop, MCS 2009, Reykjavik, Iceland, June 10-12, 2009. Proceedings 8*, Springer, 2009, pp. 171–180.

[125] Scikit-learn. "Permutation importance with multicollinear or correlated features," Accessed: Aug. 20, 2024. [Online]. Available: `https://scikit-learn.org/stable/auto_examples/inspection/plot_permutation_importance_multicollinear.html#sphx-glr-auto-examples-inspection-plot-permutation-importance-multicollinear-py`.

[126] C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn, "Bias in random forest variable importance measures: Illustrations, sources and a solution," *BMC bioinformatics*, vol. 8, pp. 1–21, 2007.