university of groningen / faculty of mathematics and natural sciences / artificial intelligence

UNIVERSITY OF GRONINGEN

MASTERS THESIS

Combining online and offline machine learning to create an interruption management system using eye data

Author:
Peter Reddingius BSc

Supervisors:
Dr. Jelmer P. Borst
Prof. Dr. Niels A. Taatgen

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

Human Machine Communication
Artificial Intelligence

January 6, 2021

UNIVERSITY OF GRONINGEN

Abstract

Combining online and offline machine learning to create an interruption management system using eye data

by Peter Reddingius BSc

In our current COVID-19 society everyone is advised to work from home as much as possible. That can lead to many new problems, such as a whole new set of possible distractions and interruptions. Interruptions have been shown to be disruptive, which can lead to errors and stress. Interruptions during low workload moments are much less disruptive than during high workload moments. Unfortunately, interruptions do not usually take the current workload into account. Using pupil dilation, a measure of working memory usage which is strongly related to workload, better moments for interruptions can be found. This paper proposes a task-independent machine learning approach to manage interruptions at low workload moments. A mondrian forest classifier was pre-trained on eye data from participants of another interruption experiment, and it also continues learning while the participant performs the experiment. To test the viability of the interruption management system (IMS) the paper also proposes a new experimental task to test the viability of an IMS using the game of sudoku. The proposed experimental task uses sudokus which provide a clear distinction between high and low workload moments. While the experiment could not be completed due to the pandemic, a small pilot experiment showed promising results. The IMS seems to be able to predict moments of low workload and managed to improve at finding good moments for interruptions while the participant was performing the experiment.

Acknowledgements

Throughout this whole long project I have received a great deal of support and guidance. I really want to thank my supervisor, Jelmer Borst. Not only for all his input and expertise, which was invaluable to create this project, but also for encouraging me to keep on going through the tough first couple of months of the pandemic, where he arranged the necessary supplies so that I could keep on making progress during that strange time.
I'd like to thank everyone that visited my colloquium and gave me feedback to improve this thesis, it really helped to talk about this research to other experienced cognitive modelling experts. I also want to thank all the participants of the experiment that, even during the pandemic, wanted to help me to complete the pilots. Thanks to you I still have a results section to discuss!
Additionally, I'd like to thank all my friends for listening to me talk about this project (even visiting my colloquium) and for providing the necessary distractions to rest outside of research. Finally, I want to thank my mother for encouraging me through this entire journey and for proofreading this thesis, along with helping me to sort out the final issues.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

At the time of writing the COVID-19 virus is still holding the world in its grip and this has a huge effect on everyone's lives. The government strongly advises everyone to work from home as much as they can, a new situation for many. Working from home has its advantages, but there are also many possible disadvantages. Working form home leads to a whole new set of possible distractions: instead of co-workers you might now be interrupted by family members or roommates, who might not realise that you are working.
Interruptions at work also are different than interruptions at home: a family interruption at work is mostly viewed as family intervening with work, but a work interruption while working at home can be perceived as work interfering with their family (Edwards and Rothbard 2000). This means that while working at home a family distraction could potentially be more disruptive than if it occurs at work. Since the crisis is a new development there is no official research yet as to how people experienced working from home, but it stands to reason that everyone who suddenly had to work from home will have experienced trouble dealing with new interruptions in an unfamiliar work-environment.
Interruptions are a part of all of our daily lives and can have positive and negative effects. Some interruptions can provide the necessary knowledge to complete a task and can help to resolve issues quickly (Isaacs et al. 1997). But for most interruptions that is not the case, these interruptions distract from the main task and have negative effects on task performance (Adamczyk and Bailey 2004). These negative effects include increased task performance times, increased errors and increase in stress (Hodgetts and Jones 2006). This can have major consequences as interruptions are stated as a significant factor in accidents at places of high risk, such as flight decks (McFarlane and Latorella 2002).
The best time to interrupt someone, which is when an interruption is least disruptive to the task performance, would be during a moment of low workload (Iqbal, Adamczyk, et al. 2005). Workload is however not very well defined and is hard to measure, but working memory usage seems to be the most important factor (Nijboer et al. 2016). This effect was seen when participants were performing their main task and a new task would interrupt and interfere with this main task. Four conditions were tested, one were both required working memory, one were none required working memory, and two were only one of the two required working memory. If neither the main nor the interrupting task required working memory then the interrupting task will not be disruptive when switching from one task to the other. If both tasks require working memory then the switch can lead to a reduction in efficiency and an increase in error (Borst, Taatgen, and Rijn 2015).
An interruption often does not come at an opportune moment, which means that a shift from the main task to the interrupting task disrupts the working memory for the main task (Mark, Gonzalez, and Harris 2005). It can take a long time before someone is recuperated from an interruption and more errors are made after the interruption (Gillie and Broadbent 1989; Altmann and Trafton 2007). Interruptions can

also lead to higher levels of stress and frustration in the participants which can cause other negative effects (Mark, Gudith, and Klocke 2008). On average people switch tasks, which includes interruptions, every three minutes (González and Mark 2004). The longer an interruption lasts the longer it also takes to recuperate after a task (Monk, Trafton, and Boehm-Davis 2008), but it takes less long if the person was interrupted at a low workload moment (Iqbal and Bailey 2006).
A simple solution might be to carefully watch other people and try to refrain from interrupting them at inopportune moments. Unfortunately, we cannot clearly see another persons working memory and know when they can be interrupted without negative effects. In a previous experiment participants were shown a 15 to 30 second video of someone at work and were asked to rate their interruptibility. The participants rated the interruptibility on a five point scale but they were correct only slightly more often than chance (Fogarty et al. 2005). With the current online work environment it is even harder to estimate whether someone can be interrupted because you cannot directly see the person you are trying to contact.
So, an interruption is costly and we cannot see for ourselves when someone is interruptible. This is what lead to the creation of Interruption Management Systems (IMS), which are computer systems that can schedule interruptions for us. In this paper such a system will be developed based on eye data and two small experiments will be conducted using this system. Before this IMS is explained the paper will explain previous research on interruption management systems and why eye data is used for this project.

1.1 Interruption Management Systems

The goal of an interruption management system is to find the least disruptive moment for an interruption and to automatically schedule an interruption at that moment. One of the first experiments with an IMS was by McFarlane (2002). In this experiment they used four different kinds of interruptions. There were two control groups where interruptions could be scheduled randomly or scheduled at an even pace. These were compared to a condition where the participant could decide when to schedule the interruptions and a condition where the interruption could be mediated using an IMS. The IMS calculated the workload metric by getting information from the state of the task to find the interruptibility of the user, this made this IMS completely task-dependent. Their results showed that the best results came from the condition where participants could decide when the interruptions could be scheduled, with the IMS mediated condition second. The author suggested that combining these systems could prove to be most effective.
Others investigated interruptibility using an IMS with several different layers. These layers included knowledge about the current task, knowledge about the users current stage and low level information such as mouse movement. Using this information the model was able to gain a performance benefit for urgent and important tasks (Arroyo and Selker 2011). While it showed good results it was a complex system that used both general information, such as mouse movement, but also task-dependent information. This means that this IMS, just like the one described above, cannot be easily used for another task.
Others investigated interruptibility using psycho-physiological data such as eyeblinks and EEG data (Züger and Fritz 2015). This research showed that a classifier trained on these features was able to measure the interruptibility of the participant

with high accuracy. While this methdo could be used for a task-independent IMS, in the paper it was not used to schedule interruptions. The main problem with this research for use in an IMS is that it required multiple uncommon measures, such as an EEG, heart rate monitor and sensors to measure body temperature. An IMS that could train on psycho-physiological data without having to use these uncommon measures would make it much cheaper and comfortable for the participant. This thesis aims to build upon this research and use psycho-physiological data to find moments of low workload, but tries to make use of less invasive measures.

1.1.1 A task-independent IMS

Katidioti, Borst, Bierens de Haan, et al. (2016) worked on a task-independent interruption management system that used real time pupil dilation measurements to interrupt participants at moments that would be the least disruptive to them. This system used the pupil dilation in order to identify low workload moments during the experimental task. The IMS was tested on an email task which could be interrupted by chat messages. While a participant makes progress in the task the Workload Identifier Value (WIV) is constantly calculated along with the Percentage Change in Pupil Size (PCPS). The WIV is the current average times a threshold value, which was set after a pilot. The PCPS is the percentage change of the current pupil dilation compared to the average of the last minute. If the PCPS stays under the WIV for more than 200 ms it was assumed to be a low workload moment and an interruption would be scheduled.
The experimental task used to test the IMS was an email task that was created based on previous research. The email task simulates the working environment of an employee of an electronics corporation who has to answer emails (the main task) while being interrupted by chat messages. The participants had to remember certain information about a product, which allowed high and low workload moments to be identified in the task. A task occupying working memory was labelled as a high workload moment and moments were participants did not have to remember task specific information were labelled as a low workload moment. The experimental task was based on an experiment by Salvucci and Bogunovich (2010) where these low and high workload moments were tested and confirmed.
The IMS showed good results: it managed to significantly reduce the amount of interruptions scheduled during high workload moments. The results can be seen below in Figure 1.1. The improvements are, however, marginal and not all high workload moments showed a decrease in interruptions.

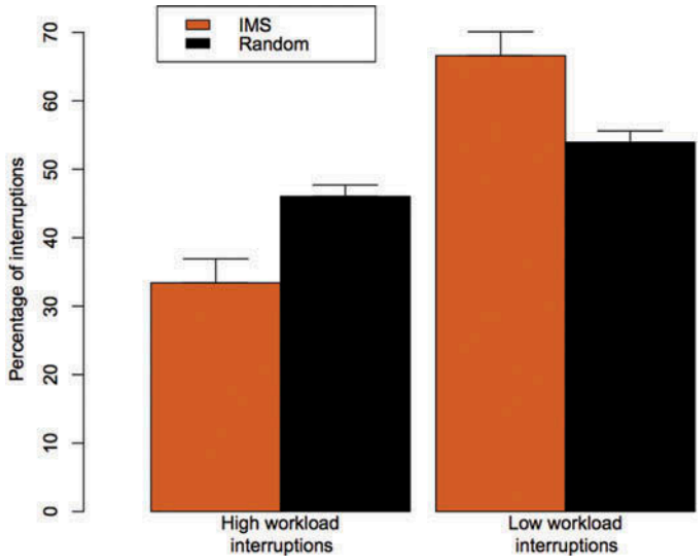


FIGURE 1.1: Average number of interruptions on high and low workload moments per block for both conditions (Katidioti, Borst, Bierens de Haan, et al. 2016)

What makes this research particularly interesting is that it is completely task-independent. The only factor that is used is the pupil size of the participant, which is a good indicator of workload. This thesis aims to build upon this work and make a system that can also be used at with any task in any environment. The system built here will also be able to use new data sources, makes use of a machine learning classifier and will continue training on the participant to achieve even better results.

1.1.2 Pupil dilation and workload

It is common knowledge that light affects our pupil size, but there are many more factors that can increase or decrease the size of the pupil. Covert cognitive variables, for instance, have a effect on the size of the pupil too (Hess and Polt 1960). Early experiments from that time also show that workload affects pupil size, such as that pupil size increases with difficulty of mathematical equations (Kahneman, Tursky, et al. 1969) or when the number of digits to remember increased (Kahneman and Beatty 1966). The pupil size kept on increasing until the participants had to remember seven or eight digits, this also reflects the limits of working memory.
Pupil size is now a well established measure of working memory load (Beatty 1982; Laeng, Sirois, and Gredebäck 2012) and has been used often in interruption research (Iqbal, Adamczyk, et al. 2005; Katidioti, Borst, and Taatgen 2014; Katidioti, Borst, Bierens de Haan, et al. 2016). It was shown that pupil dilation decreased when a participant was switching between subtasks, which is a low workload moment (Iqbal and Bailey 2005). A follow up study also showed that the time cost of the interruption was lower when participants were interrupted at these low workload moments compared to high workload and random interruptions (Iqbal, Adamczyk, et al. 2005).
One major notable drawback of pupil dilation is that it takes approximately one second after the onset of the stimulus for the pupil dilation to reach its maximum size

(Hoeks and Levelt 1993). To prevent this from having to great of an effect, the classifier will make use of the slope of the pupil dilation. This will allow the classifier to identify to small changes in the pupil size so it doesn't have to wait for the absolute dilation size. This will be used in conjunction with the PCPS (Iqbal, Adamczyk, et al. 2005), which helps to minimise the effect of other factors that can influence pupil dilation.

1.2 Aim of the project

The aim of this project is to develop an Interruption Management System that can use pupil dilation data to determine suitable moments for when a participant can be interrupted. The system will be pre-trained on pupil dilation data from participants of another interruption experiment. While this is normal practice in machine learning, this research will go further. There are large differences between the pupil sizes of different people and the pupil size can also be affected by many more factors than just the workload. This makes it hard to develop a universal system to schedule interruptions, unless it could train on the specific participant and environment as well. The IMS developed in this paper will continue to learn while the participant is performing the task to maximise the efficiency of the system.
The secondary goal of the project is to create a program that can be used to test the IMS. The program will contain a main task which requires the participant to use their working memory, but will also include moments when working memory is not required by the participant. The main task can be stopped when the IMS schedules an interruption, which also requires working memory.
This results in the following research questions:

- Can a new experimental test program be constructed that shows clear moments of high and low workload?
- Can an IMS be constructed that is pre-trained on data from another experiment to increase the number of interruptions scheduled at low workload moments?
- Can the IMS use online learning to further decrease the disruption of the interruptions?

In the next two chapters the information required to understand the research is explained in more detail. This is split into two chapters, the first of which discusses the creation of the experimental task that was developed for this research and the second chapter provides information about the machine learning classifier used in the IMS. Afterwards, the methods of the experiment will be explained and the results will be discussed.

Chapter 2

The experimental task

Previous IMS research has used a variety of experimental tasks to test these systems. The requirement of all these systems is that there is a main task and a subtask which both require working memory investments from the participant. Many of these experiments have used an email task to test their IMS (Arroyo and Selker 2011; Salvucci and Bogunovich 2010; Katidioti, Borst, Bierens de Haan, et al. 2016). Another experiment used a programming task but in that experiment solely computer science graduates were used (Züger and Fritz 2015). One experiment even used three main tasks in order to generalise the results: a document editing task, a video task and a web searching task (Iqbal and Bailey 2006).
What all of these tasks have in common is that all are rather long and similar tasks that simulate work environments. Here, the goal is to develop a task-independent IMS. The IMS will be pre-trained on data from the experiment by Katidioti, Borst, Bierens de Haan, et al. (2016), which also used pupil dilation. To properly examine the task-independence of the IMS developed in this thesis an experimental task should be used that is different than the email task of the experiment by Katidioti, Borst, Bierens de Haan, et al. (2016).
This paper will therefore go in a different direction and will make use of a simple game, which allows for natural high and low workload moments and also has the benefit of possibly being more fun for the participant. Eventually it was decided to use the game of sudoku for this new experimental task. A sudoku is a fun puzzle that many know but few are really proficient in, which makes it excellent to use as an experimental task.
A sudoku consists of a square of nine by nine cells. These cells can either have a number in them or be empty. The goal is to fill in all the empty squares using the following three rules:

- All rows must contain every number from one to nine
- All columns must contain every number from one to nine
- All three by three blocks must contain every number from one to nine

Normally one would solve a sudoku by examining the cells that have numbers in them and try to use that information to find open cells that can be filled in using the rules above. Every empty cell in the sudoku is a subtask for the participant to solve. The current experiment however, requires that the difficulty of the subtask is known to get an idea of when a participant is in a low or in a high workload moment. So instead of having the freedom to choose which cell to solve, the experimental task program will decide for the participant. In the sections below the design of this task will be explained.

2.1 Design of the sudoku task

The sudoku task was programmed in java. For an experiment that uses pupil dilation it is important to manage the light levels of the program so that this would only have a limited influence on the pupil dilation. This means that the interface of the task needs to look very simple and doesn't include too many different colours. The cell that the program selected would stay at the same light level but the row, column and block that the cell was in would be highlighted in a light grey colour. This would help the participant to find the solution for that cell and would also keep the light level as constant as possible. The final design can be seen in Image 2.1.

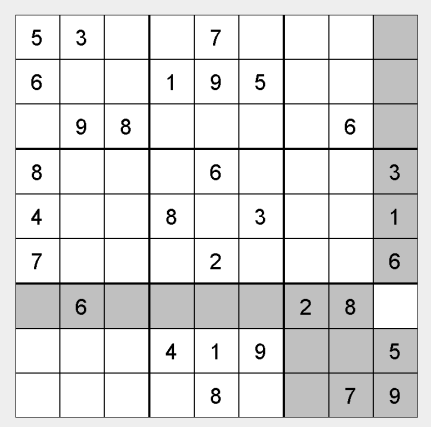


FIGURE 2.1: The design of the sudoku task, the currently selected cell is in the seventh row of the ninth column.

2.1.1 Controls

Every step of the experiment needs to be able to be solved without having to look at the keyboard, so the eyetracker can see the eyes of the participant. This means that the controls need to be really simple and easy to find by feeling alone. The participant needs to be able to select the correct number and to confirm their answer. Using only the left and right arrow keys the participant can swap through the available options (1 to 9) and can confirm their choice with spacebar. If the participant did not enter the correct value for the cell it would be labelled and shown as a mistake. Their incorrect answer would be substituted for the correct value, this is necessary since a mistake early on could result in an unsolvable sudoku. The participants were instructed to use their right hand for the arrow keys and their left hand for the spacebar.

2.2 Steps to solve the sudoku

Since the sudokus needed to be solved rather quickly and participants did not need to have any previous knowledge of sudokus in order to perform the task, only easy or medium sudokus were selected. The sudokus were selected from an online website and can be found in Appendix B. In order to determine the difficulty several sudoku techniques needed to be identified and ordered on difficulty (*Sudoku techniques* 2002). Only techniques that can be used without taking notes were used so

that participants had to use their memory for the task. Each of these techniques received a value to show the difficulty of the step on a scale from 0 to 100, where 0 requires no effort at all and 100 is very complicated. Each of these techniques were implemented into the program so it could find out if a cell can be solved with one or more of the techniques. Using this method the program could select the cells that are currently solvable, and choose one random cell from the list of solvable cells that participant would have to solve next. The five possible techniques are explained below.

2.2.1 Single option - 10

The simplest solution to solve a target cell is when a row, column or block is completely filled in except for the target cell. The answer has to be the only number that does not appear in the row, column or block. In the example in Figure 2.2 all numbers are present in the first row except for the number 1, so the answer is 1. As this was the easiest option it received a difficulty value of 10.

5	6	8	9	3		4	2	7
3		2	6			9		
		7		4			8	
	2	6				8		1
			7	6			4	
4	7					2		
2	1					7		
		4			6	5		
				9	7		3	

FIGURE 2.2: Example of the single option technique.

2.2.2 Scanning in one direction - 30

A slightly harder technique is to scan in one direction, either horizontally or vertically. Since there can only be one occurrence of a number in a row or column, scanning can be used to determine where a number has to occur in a block. By looking at the example in Figure 2.3 it can be observed that there are two occurrences of a 4 in the rows next to the target. Since there can be no fours in the top and bottom row of the block and the other two squares in the block are filled in the number 4 has to occur in the target cell. This received the difficulty value of 30.

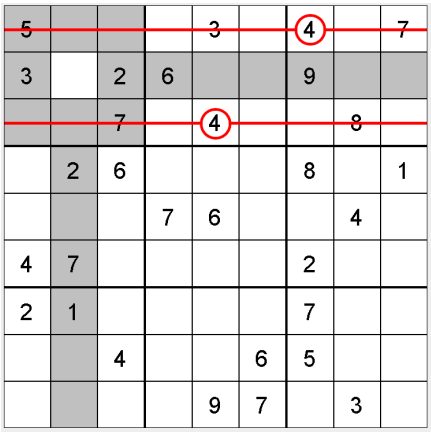


FIGURE 2.3: Example of scanning in one direction.

2.2.3 Scanning in two directions - 40

In some situations, just one single direction of scanning is not enough. In these cases horizontal and vertical scanning can be combined to determine the only possible cell in a block that can contain a certain number. In the example in Figure 2.4 it can be observed that the number 7 is already present in all rows and all columns of the block except for the row and column of the target. Which means that the number 7 has to be filled in here. Scanning in two directions received the difficulty value of 40 as it is only slightly harder than scanning in one direction.

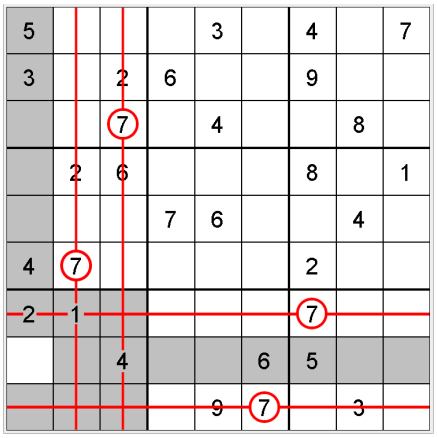


FIGURE 2.4: Example of scanning in two directions.

2.2.4 Single candidate - 60

A single candidate is when only one number can be filled into the designated cell because the remaining eight numbers are already present in the row, column and block. In the example in Figure 2.5 it can be observed that the numbers 1, 2, 4 and 8 are in the block; 4,5,7 and 9 in the column, and 6 and 7 in the row. This means that the numbers 1,2,4,5,6,7,8 and 9 are all present which means that the only number that can be filled in here is 3. This technique can be hard to spot which is why it

received the value of 60.

5				3		4		7
3		2	6			9		
		7		4			8	
	2	6				8		1
			7	6			4	
4	7					2		
2	1					7		
		4			6	5		
				9	7		3	

FIGURE 2.5: Example of the single candidate technique.

2.2.5 Missing number - 80

The toughest technique is when a row or column has no other option for a certain number than the target cell. By looking at the rows, columns or blocks crossing the column or row of the target there is sometimes only one possibility left for the location of a number in that row or column. In the example shown in figure 2.6 below, the rows crossing the column of the target all eliminate the possibility of the number 2 occurring in any other location than the target, which means that the answer must be 2. This technique received the difficulty value of 80 as it can be very difficult to identify for someone who has no experience with sudokus.

5				3		4		7
3		2	6			9		
		7		4			8	
	2	6				8		1
			7	6			4	
4	7					2		
2	1					7		
		4			6	5		
				9	7		3	

FIGURE 2.6: Example of the missing number technique.

2.3 Interruptions

The sudoku would only be one part of the test program, there also needs to be something that interrupts the participant while performing the task. This did not have to

be complicated, as long as it would take up sufficient workload that the participant could not continue their train of thought from the sudoku. A simple algebraic formula would be sufficient. All formulas look like the equation given below, the participant would be asked to solve for x.

$$ax + b = c \tag{2.1}$$

A random formula is generated each time, where the answer is between 1 and 9. This was necessary so the same method to fill in the cells for the sudoku can be used to fill in the answer to the interruption. First three numbers would be generated randomly between certain ranges. The requirements of the algorithm to create a formula are:

- a (range 2-9), b (range -9 to 9, excluding 0), x (range 2-9) are randomly generated. For a and x the number 1 is taken out since it would make the formula too easy.
- Solving the formula will result in a c of range 1 to 9.

Once all numbers were generated and had the correct values the interruption would be displayed. The eventual interruption would look like the example given below in figure 2.7. After the formula in the interruption was solved, a question appears that asked the participant for feedback on the interruption, which will be useful for the classifier later. Participants were told in the document explaining the task (see Appendix C) how to answer this question using a five point Likert scale. Selecting 1 meant that the interruption was not disruptive at all and selecting 5 meant that the interruption was very disruptive.

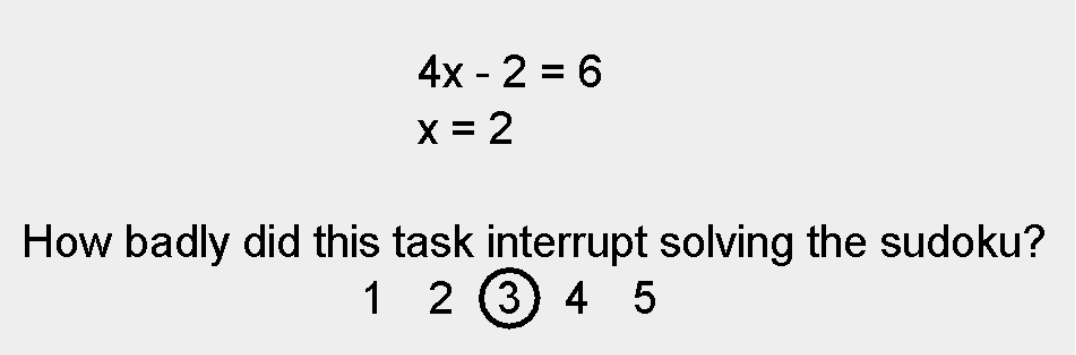


FIGURE 2.7: An image of an interruption as presented to the user. The question and the five point Likert scale only appears after the interruption is correctly answered.

Chapter 3

Machine Learning

In this thesis an IMS will be developed that uses a machine learning classifier. Machine learning is very useful for this experiment since it can be used to find novel connections in large amounts of data and classify the data based on these connections (Alpaydin 2020). For this IMS there will be huge amounts of eye data that needs to be processed in an instant so that a decision can be made when someone can be interrupted or not. Machine learning can be used to process this data and can quickly make a prediction. It can also take in different kinds of data, which makes the IMS capable of expanding to additional or new data for future research.
In this chapter the data that is used to pre-train the classifier is explained along with the way this is processed to form an input for the classifier. Afterwards, different algorithms are examined for their capabilities in classifying the data. Finally, the algorithm that was chosen is examined in detail.

3.1 Data

The data used in this experiment was from the experiment by Katidioti, Borst, Bierens de Haan, et al. (2016). The data included all the eye tracking data from all 22 participants of the experiment. The data obtained from this experiment is:

- **Part of the task:** The trial was divided into ten different steps, half of those steps were high workload and the other half low workload moments.
- **Pupil dilation:** The pupil dilation at the current point, sampled by the eye-tracker at 250 Hz per second.
- **Baseline:** This is the median of the last minute of pupil dilation samples.
- **Events:** Saccades, blinks and fixation are categorised as events.

The data, as obtained, contained updates on the pupil dilation at 250 Hz (one sample every 4 ms). This is far faster than the pupil can react since it can take up to a second to reach the peak (Hoeks and Levelt 1993). On top of that, unless using a supercomputer, an algorithm cannot process 250 inputs per second continuously. So the best option would be to change the data to a lower frequency that can be used for this classification task. The classifier could profit from being able to use the development of the pupil dilation over a time period. To achieve this, the input of the classifier will be a vector of pupil dilation at different moments during a time window. To find the ideal time and the ideal frequency a test was done with three different options for each of the two variables.
The test accuracy of the seven potentially best algorithms (which will be explained below in 3.2) was examined when given one of the nine different input sets. From the results in figure 3.1 it can be seen that the test accuracy keeps on increasing with every step. This suggests that there might even be better options than 2 seconds and

20 Hz (one sample every 50 ms), but to go higher than two seconds would reduce the amount of potential trigger moments for interruptions and going higher than 20 Hz would make the input size too large to handle for the computer that had to run this algorithm. As such it was decided to not increase further than 2 seconds and 20 Hz, and to keep this last result for the rest of the experiment.

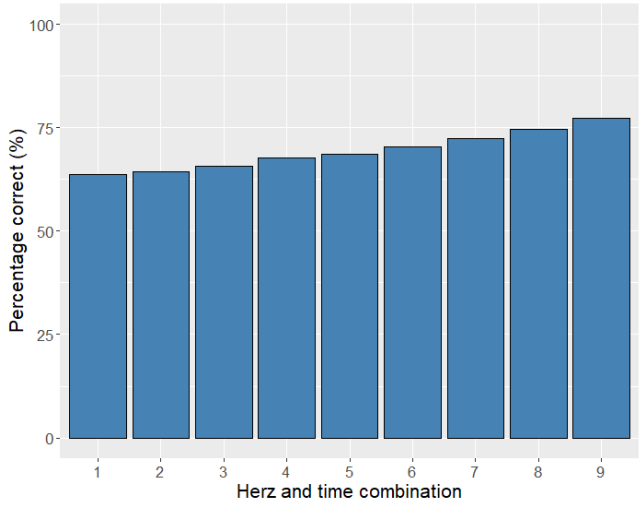


FIGURE 3.1: Test accuracy of different combinations of the input.

This resulted in one data point for each of the 20 inputs per second (20 Hz), times two since it encompasses two seconds, results in 40 inputs. The first group of 40 inputs are the percentage change of the pupil dilation (PCPS). The second group of 40 inputs are the direction of the pupil dilation. The direction is calculated by subtracting the median pupil dilation of the last first half of the pupil sizes in the input to the second half of the pupil sizes of the input. If this value was positive it meant that the pupil dilation was increasing during the last 50 miliseconds, and was decreasing if the value was negative. Additionally, this value could also be larger and smaller which showed how quickly the pupil changed during these 50 milliseconds.
To these 80 inputs there were three additions: direction of the full input, saccades and blinks. The direction of the full input is calculated by subtracting the median pupil dilation of the first half of the full two second duration of the input, from the second half of the data. Saccades and blinks were the percentage of the amount of inputs that were labelled as saccades or blinks by the eyetracker. Adding this together means that every input value is a vector of 83 values.
Before being entered into the algorithm the data would be normalized since there were large differences between the values of the inputs. This was done using min-max normalization, the same method as that was used in the code provided of the algorithm that would eventually be used for the classifier (Lakshminarayanan, Roy, and Teh 2014). While doing the experiment the new data would be added to the stored data that the algorithm was pre-trained on to calculate the normalization. Min-max normalization causes each data point to become a value between 0 and 1, this would prevent large differences between different features from becoming a problem. During testing the test accuracy when using normalization was slightly higher, about a one to two percent increase, so the same normalization method will be used to normalize all data in this experiment.

3.2 Possible classifiers

The machine learning algorithm needs to decide between two different scenarios, that currently the user of the IMS can be interrupted or that the user cannot be interrupted. This is perfectly suited to a classification algorithm, such as a multilayer perceptron, support vector machines and decision trees. To test which algorithms would be best suited to the data, a test was done using the scikit-learn package by Pedregosa et al. (2011) which offers many different machine learning algorithms for python.
All the algorithms were tested with the presets of the package. For this test the algorithms were trained on the data of all 22 participants with the exception of one, the data from this last participant would be the test set. The test was constructed in this way so that it would most resemble the between-participants design of the final experiment. The test was repeated 22 times so that each participant was once in the test set. Some algorithms used in the test contain random factors, such as the random initialization of the weights of a multilayer perceptron, that have the potential to cause a bias. To decrease this potential bias all participants were tested three times, which resulted in the average of 66 tests.
The results of these tests can be seen in Figure 3.2, the complete list of algorithms used can be found in Appendix A, here only the ones with the best test accuracy will be examined.

FIGURE 3.2: Classification accuracy of the best algorithms that were tested.

The clear winners were all based on the concept of decision trees. There was not much difference between these winners, all of them had a test accuracy between 65% and 70%. From these seven winners, two seemed to have the most promising online algorithms, these two would be Random Forest and the Gradient Booster. In this experiment only the Random Forest was examined as there seemed to be better online counterparts of this algorithm, but gradient booster remains a good option as well.

3.3 Random forests and decision trees

Before the online algorithm will be introduced, the next sections will explain decision trees and random forests, and how these can show important features in the

input. Afterwards, online algorithms will be introduced to make the transition to the online random forest that is used in the IMS.

3.3.1 Decision trees and important features

Decision trees try to find a value of one of the variables in the input that can create the largest split in the data. This means that this variable can causes the most uneven split in the true (can be interrupted) and the false (cannot be interrupted) inputs. In other words, it would show which variables were most important in classification. This process can be visualised which provides a tree such as the one that can be seen in image below 3.3.

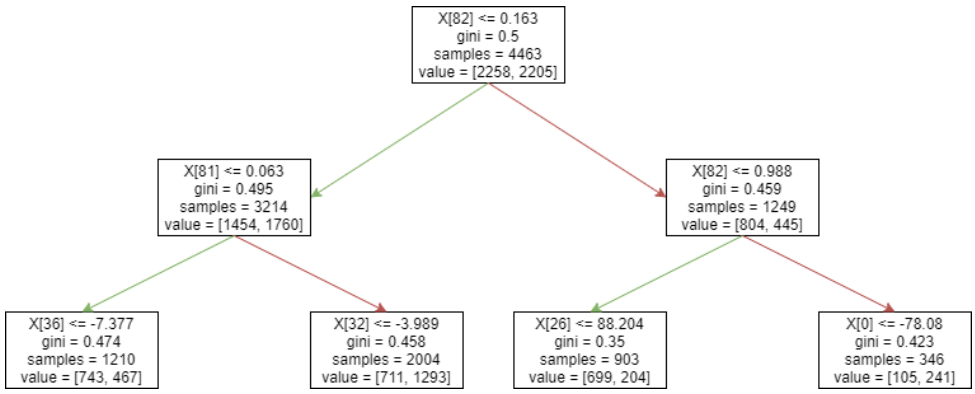


FIGURE 3.3: Image of the root and the first two layers of a decision tree that was trained on the data. Most importantly, it shows which feature is being used for the split between brackets.

The sci-kit implementation of the decision tree randomised the inputs, which meant that the decision tree was slightly different each time. But all of them showed a similar result as the one shown above. The first three splits (the first two rows of the tree) were generally the 81st and the 82nd variable in the input. These correspond to the percentage of inputs that contained saccades (X[81]) and the percentage of inputs that contained blinks (X[82]). Apparently these two generalised variables are important to determine whether the participant was interruptible or not.

3.3.2 Random forest

As stated before, random forest classifiers are based on the the decision tree algorithm. A decision tree algorithm is very useful as it can break up a larger more complex problem into a set of smaller decisions. Decision trees can be used very efficiently in classification, especially binary classification (Safavian and Landgrebe 1991). One large problem with this algorithm is that it tends to overfit because it tries to classify every data point. This problems is what the random forest algorithm aims to fix. An ensemble algorithm like random forest splits up the data in different parts and trains a decision tree on each of those parts. The random forest algorithm allows each decision tree to overfit on their data, which causes differences in how each tree would classify an input. The random forest then combines the answers of each tree, which results in more generalized classification (Breiman 2001).

3.4 Online algorithms

The IMS needs to be able to learn from the participant while the participant is performing the experiment, so the data is continuously coming in one input at a time. But algorithms such as the decision tree, random forest and gradient booster are all batch algorithms. Batch algorithms process the entire set of training samples all as one set (Oza and Russell 2001). Batch algorithms suffer in real world scenarios as data is often not all immediately available, and to fully retrain a batch algorithm on new data is very inefficient. Making algorithms that can learn from data that is continuously coming in is one of the largest current challenges in machine learning (Krawczyk et al. 2017). Online algorithms, algorithms that can learn incrementally from data, can learn from data that is continuously coming in and can be updated instantly and efficiently (Hoi et al. 2018).
In recent years, interest in online algorithms has increased as it can solve many limitations of batch learning. Ensemble algorithms, like the random forest algorithm, seem to be one of the most promising directions for online learning (Krawczyk et al. 2017). One of these new online ensemble algorithms are mondrian forests, which is very similar to random forest algorithms. Mondrian forests are reported to outperform other online random forest classifiers and can even achieve results close to batch random forest classifiers. It is also well adaptable to learning from data that is continuously coming in, making it suitable for this experiment (Lakshminarayanan, Roy, and Teh 2014).

3.5 Mondrian forest

The mondrian forest algorithm is, like the random forest algorithm, an ensemble technique. Instead of decision trees it uses a slightly different algorithm called mondrian trees. Mondrian trees in turn are based upon the concept of the mondrian process. The mondrian process was first defined by Roy, Teh, et al. (2008). The mondrian process is named after the famous Dutch painter for a reason, as his paintings can help to understand how the process works. A painting by Mondrian can be seen as a two-dimensional space with x and y axis ranging from 0 to 1, or in short $[0,1]^2$. In this space the mondrian process recursively partitions the plane using several axis-aligned cuts. This is similar to how decision trees operate, but the mondrian process includes a 'budget' that it uses and if a split is more costly than the current budget the split is not made. This last process is similar to how normal decision trees can have a maximum depth.
While the mondrian process operates similarly to decision trees, it is a generally infinite structure. Since classification uses a finite set of observed data, the mondrian process is restricted to a finite set of points, resulting in mondrian trees (Lakshminarayanan, Roy, and Teh 2014). Mondrian trees, as compared to general decision trees, introduce a 'time of split' or 'cost' variable (τ), this split time increases with the depth of the tree and starts at the root at zero. Additionally, it adds a new parameter 'lifetime' or 'budget' (λ), a non-negative number set at the start that limits the number of splits that can be made by the algorithm. Finally, it checks the distance between the upper ($u^x_{\delta j}$) and lower ($l^x_{\delta j}$) bounds of each dimension (δ) of the data in each node (j), only regarding the input data (x) and not the assigned labels (y). This distance is denoted as:

$$E = \frac{1}{\sum_\delta u^x_{\delta j} - l^x_{\delta j}} \tag{3.1}$$

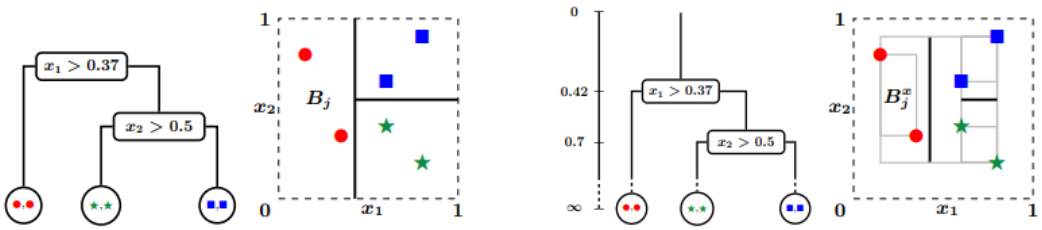
Simply put this calculates the size of the smallest rectangle (B_j^x) that can enclose all the data of the node as shown in figure 3.4. Each time a split can be made, the algorithm checks if the current budget is larger than the cost of the split and the distance between the data in the node. This leads to the following if statement:

$$\textbf{if } \tau_{parent(j)} + E < \lambda \tag{3.2}$$

If the statement is not true then node j will be added as a leaf node in the mondrian tree. If this statement is true, a new node j is created in the decision tree with its cost being the cost of the parent ($\tau_{parent(j)}$) plus the distance (E), so $\tau_j = \tau_{parent(j)} + E$. The data is then split according to the uniform split distribution and the algorithm continues recursively with the two child nodes.
The main differences between mondrian trees and decision trees are:

- The nodes are sampled independently from the labels (Y).
- Each node has a split time.
- The lifetime parameter controls the maximum number of splits possible (like the maximum depth parameter for decision trees).
- A split in a node is only made on the basis of the extend of the training data in the node.

These differences can also be observed in Figure 3.4. Important to notice is the grey rectangle (B_j^x) that is drawn around the current maximum values of the data that the split is made on. This shows how the mondrian tree is continuously making the minimal split that is required to split the data, instead of the large split that is made with a decision tree.



(a) Decision Tree (b) Mondrian Tree

FIGURE 3.4: Example of a decision and mondrian tree in $[0, 1]^2$, with horizontal axis x_1 and vertical axis x_2. The figure on the left shows the structure and partition of a decision tree. The right figure shows a mondrian tree, which also shows a time axis including the time of splits. (Lakshminarayanan, Roy, and Teh 2014)

The described algorithm is not yet capable of taking in continuous data, in other words, this is not an online algorithm. In order to make it function online it needs methods that can extend the tree when new data comes in. When a new data point comes in there are three different operations that may be executed:

- Introduction of a new split 'above' an existing split.
- Extension of an existing split to the updated range of the data in the node.

- Splitting an existing leaf node into two new children.

Most other online decision tree algorithms only use the last method. While it is possible to implement the first two methods for a normal decision tree, it works better for a mondrian tree. Because the mondrian tree makes the minimal split required to divide the data, it is easier to extend an existing split. The mondrian tree also keeps track of the cost of making a split in order to make it easier to introduce a split 'above' the one that was already present. An example of the functioning of the Mondrian Tree algorithm can be seen in Figure 3.5. In this image a mondrian tree is extended when new data points come in. For data point c a new split 'above' an existing split is made. Data point d requires the extension of the previous split between a and b before a new split can be made in what previously was a leaf node.

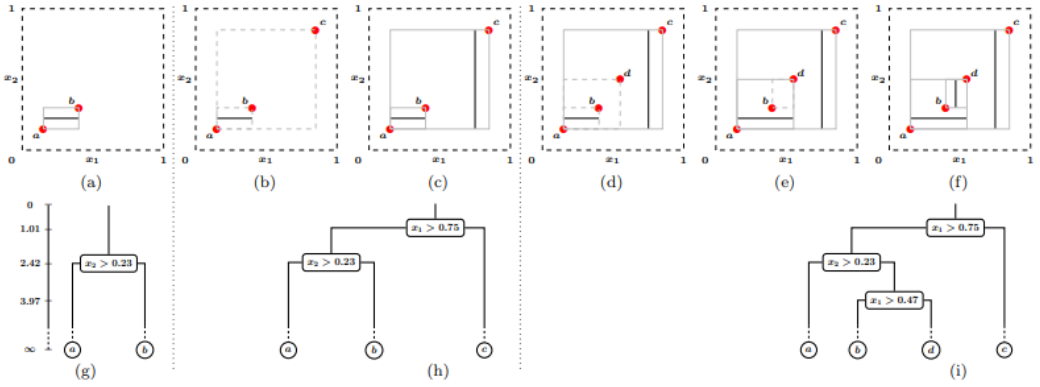


FIGURE 3.5: This figure shows an example of the Mondrian Tree algorithm in a two dimensional plane. The grey rectangles (B_j^x) show the upper ($u_{\delta j}^x$) and lower ($l_{\delta j}^x$) bounds of data in the block. The black lines show the splits that were made to divide the data in the blocks. (Lakshminarayanan, Roy, and Teh 2014)

Finally this process is combined with the ensemble technique of random forests. One Mondrian forest is made up of multiple Mondrian trees. Each Mondrian Tree makes a prediction when given an input, the prediction is the distribution of class labels of the inputs in the leaf node. The Mondrian Forest simply takes the average of the predictions made by the individual Mondrian Trees.

3.6 Programming the Mondrian Classifier

The mondrian classifier (written in python), as obtained from the github of one of the authors at (Lakshminarayanan, Roy, and Teh 2014), was able to take in a batch and to continue training on the rest of the data. However, the algorithm was not designed to continue training on one new input at a time. This is required for this research because each new input needs to be processed as it comes in. A large part of the obtained code had to be changed to allow the data of this experiment to work with this algorithm. For instance, a way to save and load a trained algorithm was added so that it could be pre-trained in advance. Some of the inner workings of the algorithm were changed to allow it to take in a continuous data stream, this however did not change the core characteristics of the algorithm as described in section 3.5.

Even though it worked, the test accuracy was about ten percent lower than offline random forests (resulting in about 60% test accuracy) and the pre-training of the algorithm became quite slow. This might be due to the slight changes of the algorithm but generally online algorithms just perform less well than their offline counterparts (Krawczyk et al. 2017).

Chapter 4

Methods

4.1 Pilot 1: The sudoku experimental task

A pilot was done with the finished sudoku program to see if it would be suitable for this experiment. The main questions were to find out if the program worked correctly, if the program is a good task for creating high and low workload situations and to find out if there were any quality of use improvements that could be made to the program. No interruptions were scheduled for this experiment because it was purely to test the validity of the sudoku task. The results of this experiment are explained in section 5.1.

Participants
It was tested on four participants (1 female) between the ages of 20 and 25. They all also received a financial compensation of eigth euros.

Procedure
The total experiment lasted for about one hour. Reading the explanation document and setup for the experiment took on average ten to fifteen minutes, leaving at most 50 minutes for the experiment. The participants were asked to work on the sudokus quickly and accurately, guessing was allowed if the participant felt that they could not figure it out. The participants were asked to solve three sudokus in an hour. Each participant received the same sudokus but the cells to fill in were presented in a different order for each participant because it was randomly decided by the program.

Apparatus
The participants were seated in a small room with constant lighting. The eyetracker data was not necessary for this pilot, but to simulate conditions for the final experiment the participants were asked to use a chin rest to stabilise their head.

4.2 Preparation for the final experiment

To finalise the project all parts had to be integrated. This meant that the java program, which was running the sudokus, had to be connected to the python program, which was running the classifier. The classifier had to be connected to the eyetracker in order to receive the eye data. This means that all these components need to be able to relay the relevant information to each other. First the connection between the eyetracker and the classifier is examined, and afterwards the connection between the experimental task and the classifier.

4.2.1 The eyetracker and the python program

The eyetracker came with python packages to retrieve information from the eyetracker. The python program retrieves pupil dilation information from the eyetracker as soon as there is a new update. These new updates are saved until a complete input vector is available to put into the algorithm. The data is then put into the algorithm every two seconds to check if it is time for an interruption.
The program will not take in new data after an interruption was scheduled and will wait for a response from the java program. The response is the answer the user provided to the question if the interruption was well scheduled. The mondrian forest would then be updated using this response.
Due to a technical limitation the program did not take in event data from the eyetracker. This would allow the program to react to the saccades and blinks of the participant, which seems to be an important feature as explained in section 3.3.1. The implications of this will be discussed in the discussion.

4.2.2 Deciding for an interruption

The program has to decide when an interruption can be scheduled. To this end, a threshold (θ) is introduced. The mondrian forest returns the chance of the input belonging to a certain class, if the output of the forest was higher than the threshold an interruption would be scheduled.
The algorithm checks if it has been long enough since the last interruption, which is to prevent the possibility of constant interruptions and, simultaneously, to keep a regular pace of interruptions. The amount of time that the program waits after the last interruption is I_{min}. After this time has passed an interruption can be scheduled, but the threshold will be high to ensure that the program only schedules interruptions at the most opportune moments. However, this could result in too few interruptions being scheduled, since the threshold is very high. The program needs to schedule interruptions often in order to train the algorithm, so at some point an interruption needs to be forced. To solve this there is a set amount of time after the last interruption (I_{wait}), once this is reached the threshold will start decreasing with small steps (step size = h) every two seconds (every time a new input is checked). The threshold will go down quickly, this makes a threshold more likely to happen the longer it has been since the last interruption.

4.2.3 Switching between subtasks

After each cell of the sudoku that was solved a short break was introduced before the next cell was shown and the next subtask started. The break was chosen randomly from a value between 2000 and 3000, which are the amount of miliseconds the break would last. This was done to create a moment when the participant would have nothing to do. This created the lowest possible workload moment during the task, a moment when the participant should definitely be interruptible. Ideally, the IMS would schedule most interruptions during these short moments. This moment is shown in the results section as difficulty zero (0).

4.2.4 The experimental task and the algorithm

The python and the java program communicate via a messaging system which updates text files on the computer that the other program monitors for changes. While this is not a very efficient solution, it is still sufficiently fast enough and it also keeps an automatic log of everything that happened during the experiment. The following messages are sent back and forth, this is the only communication that happened during the task:

- The python program, which contains the algorithm, writes a message to the task each time an interruption can be scheduled.
- The java program, containing the experimental task, writes a return message to the algorithm with the answer the user provided
- The algorithm sends a message when the participant has lost fixation

The last one remain largely unused due to problems in communicating with the eyetracker, but these can be fixed and used to make the program function better. This method can be used to stop the program when the participant loses fixation so that, if necessary, a new calibration can be done.

4.3 Pilot 2: The online classifier

The second pilot was to test the complete experiment. Aside from making sure everything works as intended the goal was to gather data to see the viability of this program. The results of the pilot could show what kind of results would come out of the final experiment.

Participants
The pilot was done with six participants (3 female) between the ages of 20 and 28. All participants received a monetary compensation of eight euros for the experiment. The participants were asked to read the explanation document which was by now extended to include information on interruptions, this document can be found in appendix C.

Procedure
The experiment lasted for about one hour. Reading the explanation document and setup for the experiment took on average fifteen minutes, leaving 45 minutes for the experiment. Participants completed three whole sudokus during that time and answered about 30 interruptions. Each participant received the same sudokus but the order of the cells was different for each one.

Apparatus
For the experiment an EyeLink Portable Duo eyetracker was used. The participants were asked to use a chin rest to stabilise their head and eyes for the experiment. Pupil dilation was measured at a sample rate of 250 Hz. Calibration and drift correction was performed at the start of the experiment.
The eyetracker was connected a HP 14-bp020nd laptop. Due to the having a dual core CPU with a speed of 2GHz and only 4GB of RAM, this limited the complexity

of the mondrian forest.
Because this pilot took place during the COVID-19 crisis there were several measures in place to keep both the examiner and the participant safe. Most notably the entire setup was made so that the examiner and participant could stay 1.5 meters away from each other at all times.

Settings

- θ = 0.8
- h = 0.01
- I_{min} = 20 seconds
- I_{wait} = 40 seconds
- Size of the forest = 5 / 20 trees

Before the start of the experiment a forest of 50 trees was used, but this would take too long to load for the laptop. For testing purposes the first four participants used a forest of 5 trees, the final two participants could use a forest of 20 trees, which was the maximum the laptop could handle.
The results of this experiment will be detailed in results section 5.2.

Chapter 5

Results

Due to the COVID-19 crisis no full experiment could be completed for this thesis. The proposed plan for a full experiment will be explained in the discussion. Here, only the results obtained so far, the results of the two pilots, will be discussed. The goal for the pilots was to test if everything worked as intended. Four participants for each experiment did not make it possible to conduct meaningful statistical tests, especially with a between-subject design. The small amount of data did not even make it possible to pass all of the assumptions for the statistical models. Therefore the data is only presented as an evaluation of the potential of the experiment.

5.1 Pilot 1: The sudoku experimental taks

The first Figure 5.1 shows the percentage of cells that were answered correctly by the participants. This shows that participants managed to solve most of the cells, but had to guess more often in the 30, 40 and 80 difficulty gategories. In general all participants answered all techniques correctly between 80% and 100% of the time. The hardest technique, missing number (80), is also the one that has the lowest amount of correct answers, which is to be expected. Technique 60, single candidate, is slightly higher than expected. This difficulty will show unexpected results more often, the implications of this will be discussed later.

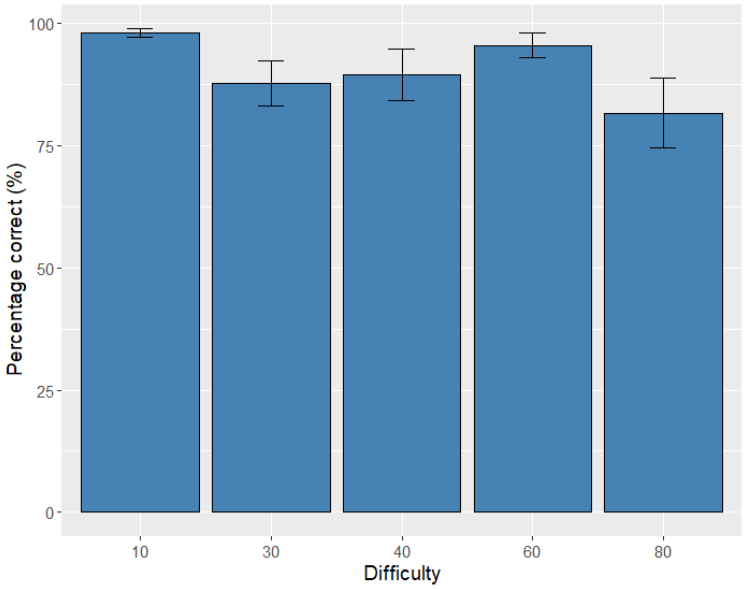


FIGURE 5.1: Percentage correct per technique.

Secondly it was important to look at how much time it would take the participants to solve each technique. If it took more time, it will probably be a harder technique to use. The results are in figure 5.2, most noticeable there is much variation between techniques. The easiest technique took only 5 seconds to solve, and the hardest one nearly 40 seconds. It is almost a perfect staircase of increasing reaction times, except for the technique labelled 60, single candidate. It is actually understandable that this technique is easier, which is mostly since this sudoku is solved differently than a normal sudoku. When solving a sudoku normally it is recommended to look for cells that are filled in and to see if these can help solve empty cells. When solving a sudoku like that it is hard to spot a cell that can be filled in using the single candidate technique. For this experiment the participant is asked to solve one specific empty cell, and in that case it is a good strategy to start by looking for numbers that surround it to see if it can be solved using the single candidate technique.

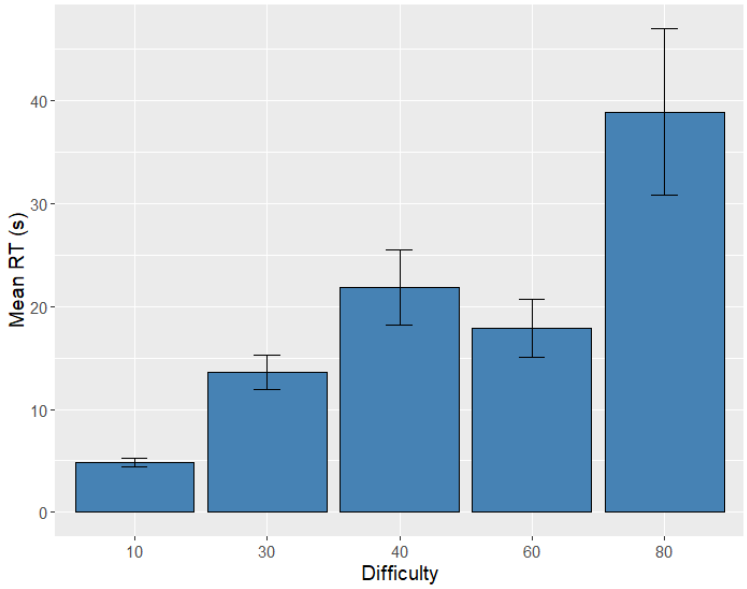


FIGURE 5.2: Mean reaction time (ms) per technique.

Figure 5.3 shows the percentage of time participants spent on average trying to solve each technique. On average participants spent the most time in the most difficult technique and the least time in the least difficult technique. The other techniques did not differ much.
The participants provided feedback that it felt frustrating when there was a row, column or block with a single option, one that was complete full except for one cell, and that the algorithm selected another cell to solve. To make their experience better it was decided to give preference to the single options technique if a cell that could be solved with this technique was available. This would alleviate the frustration and would increase the amount of time spent in the single option technique (10). Theoretically it should also help to make the participants spend slightly less time in the hardest technique because every cell that gets filled in reduces the chance of the hardest technique appearing, as there is more information present to use an easier technique. The results of this change can be observed in the second part of the results.

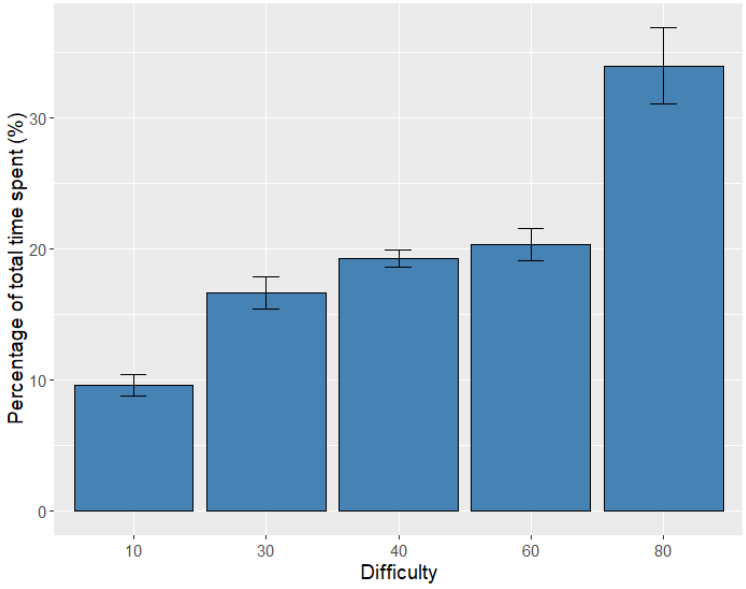


FIGURE 5.3: Percentage of total time spent on each difficulty category.

In conclusion, it seemed that the sudoku program worked really well. There were no problems with the functioning of the program and a slight annoyance of the participants was removed because of the feedback to this pilot. Participants answered most of the questions correctly and all participants managed to finish the experiment within an hour, including setup. There was also a clear difference in reaction times between techniques which seems to suggest that there is more workload required for the harder techniques then for the easier techniques.

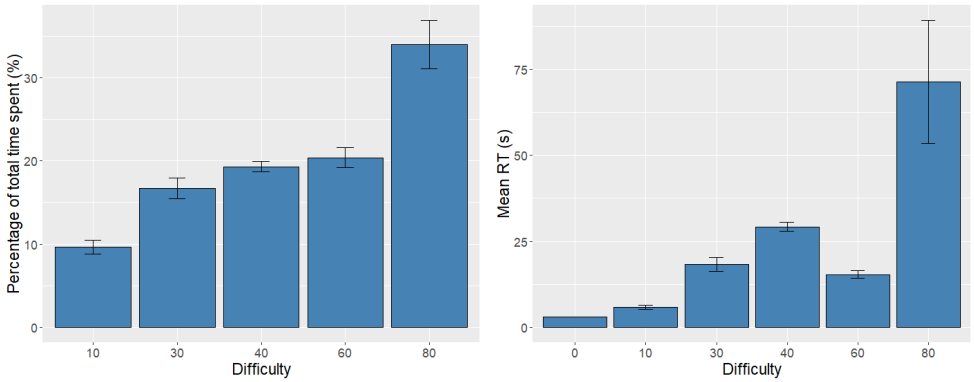
5.2 Pilot 2: The online classifier

For the first two participants there were still some issues to work out that caused some restarts in the program, which resulted in the data being incomplete. These two participants were therefore excluded from analysis. After these problems the programs worked well and no large issues occurred.
As stated before, the first four participants used a mondrian forest made up of 5 mondrian trees. This is quite a small forest but the load times were still about thirty seconds to load in the entire pre-trained forest on the laptop. This faster forest was created so that for the first couple participants some changes could quickly be made to fix issues that might arrive during the experiment, while retaining as much experiment time as possible. The other prepared forest was made up of 50 mondrians, this was a size that worked well on the computer it was created on, but the laptop struggled with it. When the python program was already active (which was the necessary order), the load times to get the pre-trained forest took up to 20 minutes, which caused major planning issues. Between the fourth end fifth participant a new forest was pre-trained with 20 mondrians, which had a load time of 5 to 10 minutes. This still took some time but it could load while the participant was getting seated and reading the instructions.
All results are between subjects, the comparisons show the average of the means of the four participants including the standard error. First a comparison between the

two pilots will be made to see how comparable they are and what the effect of the interruptions is. Afterwards the results will be examined to detect the effect of the online learning of the algorithm.

5.2.1 Comparison between the two pilots

Because of the introduction of the interruptions the participants would be expected to be slower because the interruption takes over their working memory. This causes the participant to have to restart after the interruption which would make the reaction time of the participants slower. The actual time spent on the interruption was taken out for this comparison. The comparison can be seen below, note that the y axis is very different. The two figures below 5.4 show that both pilots show a similar comparison between difficulties. Only difficulty 60, single candidate, shows a slight change in that it had a lower reaction time than difficulty 30, which was not the case in the first pilot.



Results first pilot Results second pilot

FIGURE 5.4: Reaction time to finish subtasks of each difficulty category

To inspect the differences more precisely, Table 5.1 shows a comparison of the means of both pilots. The final column shows the change from pilot one to pilot two. Most noteable is that in nearly all difficulties the reaction time increased, which is to be expected because of the interruptions. Getting back to work after a disrupting interruption will take a while, as discussed in the introduction. Interestingly it can be observed that difficulty 60 decreased slightly, even with the introduction of interruptions. There is no clear explanation as to why this decrease would happen. Finally, it is noticeable that the reaction time to finish a cell of the hardest difficulty nearly doubled. This is the toughest condition so interruptions can be the most disruptive, but there is another effect at play here. It did happen occasionally that because participants took a long time to complete this difficulty that multiple interruptions happened during one subtask. In the next paragraph this effect will be further examined.

Difficulty	Mean P1	Mean P2	Change
0	- (±)	2.90 (±0.02)	-
10	4.83 (±0.41)	5.78 (±0.67)	1.20
30	13.62 (±1.69)	18.31 (±2.01)	1.34
40	21.85 (±3.63)	29.14 (±1.31)	1.33
60	17.86 (±2.85)	15.31 (±1.10)	0.86
80	38.87 (±8.05)	71.40 (±17.88)	1.84

TABLE 5.1: The means (and standard error) of the reaction times of the participants of pilot 1 (P1) and pilot 2 (P2) are compared. The change signifies the difference between the means of the first and second pilot.

A comparison of the subtasks that had no interruptions and all subtasks that had one interruption gives a new perspective on the previously examined results. This comparison can be seen in table 5.2. First of all, difficulty 60 shows that reaction time does increase when an interruption happens as expected, it seems that the participants of the second pilot were just on average faster. Difficulty 30 shows a huge difference though, an interruption during that specific subtask more than doubles the reaction time, making it similar to difficulty 40. Why interruptions could potentially have such a large effect on this subtasks of this difficulty is unknown.

Difficulty	Mean 0 Interruptions	Mean 1 interruption	Change
0	2.90 (±0.02)	2.98 (±0.08)	1.03
10	5.63 (±0.64)	9.01 (±2.29)	1.60
30	13.89 (±1.48)	32.85 (±2.97)	2.36
40	24.82 (±3.63)	35.81 (±4.98)	1.44
60	14.71 (±1.27)	20.03 (±2.54)	1.36
80	45.18 (±11.07)	69.29 (±12.82)	1.53

TABLE 5.2: The means (and standard error) of the reaction times of the participants of pilot 2 of all subtasks that had no interruptions and all cells that had only one interruption.

As in pilot one it is important to look at the amount of time participants spent on subtasks of each difficulty. Three changes are present since the first pilot, the results of which can be seen in figure 5.3. The first difference is the introduction of interruptions, which can increase the time spent in a technique when a participant takes a long time to recover from the interruption. The second difference is the addition of difficulty zero, the artificially introduced waiting period. Finally, difficulty 10 is favoured now, to help balance the time spent on all difficulties. Figure 5.5 shows the percentage of time spent on each difficulty for the second pilot.

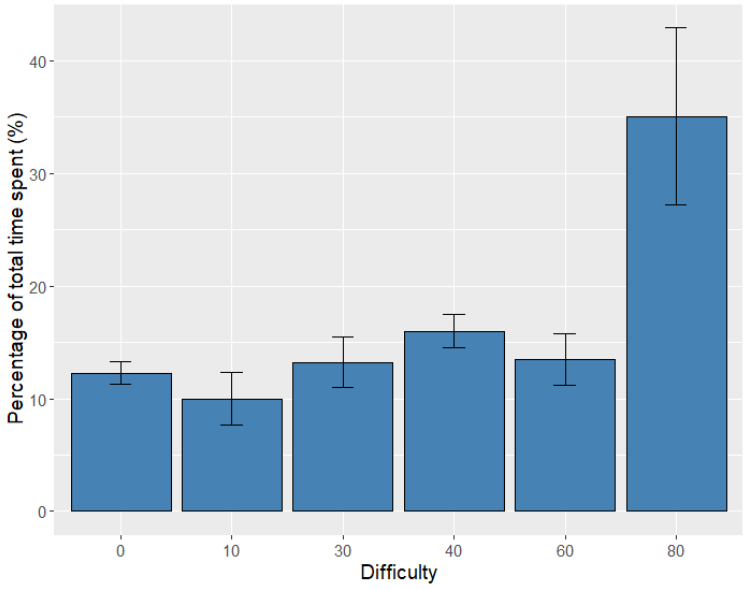


FIGURE 5.5: Time spent (ms) on each difficulty category during the experiment.

It shows a couple differences since the first pilot. First of all, the amount of time spent on each category is pretty similar, difficulties 0 to 60 all take up between 10% to 17% of the total time spent. Participants spent more than a third of the time on difficulty 80, about two to three times as much as the other difficulties.

5.2.2 Interruptibility judgement

The participants would always rate an interruption with an integer value ranging from one to five. An answer of one meant that the interruption was very well timed and five very badly timed. The average feedback of the participants can show during which subtasks the participants felt like an interruption was more disruptive than in other subtasks. This, combined with the average amount of interruptions during subtasks of that difficulty, would show if the algorithm worked, as it should reduce interruptions during subtasks that participants generally rated above the average. The results can be seen in Table 5.3.

Difficulty	Mean number of interruptions	Mean number of interruptions per minute of time spent	Mean Answer
0	6.50 (±1.04)	1.21 (±0.21)	1.84 (±0.50)
10	1.75 (±0.75)	0.46 (±0.13)	3.38 (±0.85)
30	3.75 (±1.60)	0.71 (±0.19)	2.63 (±0.63)
40	4.00 (±1.22)	0.58 (±0.12)	2.93 (±0.12)
60	2.50 (±0.87)	0.45 (±0.10)	3.45 (±0.32)
80	8.25 (±2.25)	0.60 (±0.14)	3.38 (±0.14)

TABLE 5.3: The means (and standard error) of the amount of interruptions during each difficulty category, the number of interruptions during that category per minute of time spent in that category and the mean answer that the participants gave after an interruption during that category.

As expected, difficulty zero scored far below the average. This is to be expected since the participants were during this category simply waiting for the next square to be selected by the program. The other difficulties received on average rather similar answers. Difficulty 30 and 40 scored slightly lower than average, and interruptions during difficulty thirty were generally considered less disruptive than difficulty 40, which falls within expectations. Difficulty 10 and 60 show answers slightly above the average, which does not conform with the general expectations. There might be many reasons for the high rating of these categories, which will be explored further in the discussion. The hardest difficulty (80) also scores above average, which is to be expected, but it is not much higher than average. This might suggest that the algorithm found good times during this difficult subtask, or simply that participants did not often resort to responding very negatively to the question.
The most interesting observations come from the comparison with the average amount of interruptions per difficulty. As the average answer increases the amount of interruptions seems to decrease. This can be further explained by examining the mean answer of the participants and correlating that to mean number of interruptions of time spent. The Pearson's correlation coefficient was calculated which showed that the two conditions were negatively correlated ($r(4)=-0.95$, $p<0.01$). This results in an R^2 of 90%, meaning that the two predictors explained 90% of the variance. While the test shows a significant results, with only four participants this results might not be reliable. This does seem to suggest that the algorithm responds well to the feedback from the participant and can identify moments that participants respond well to, even though their answers might not directly compare to expectations.

5.2.3 Impact on the interruptions

Earlier in the results section (5.4) the amount of time spent on a subtask of each difficulty of pilot one was compared the second pilot. It seemed to show that generally it took longer to complete subtasks when the participants were interrupted. But the effect of needing more time to switch tasks that both require working memory also works the other way round. It is therefore expected that for difficulties with a higher workload, the time to complete an interruption would also be larger. The time needed to complete an interruption can be seen in Figure 5.6.

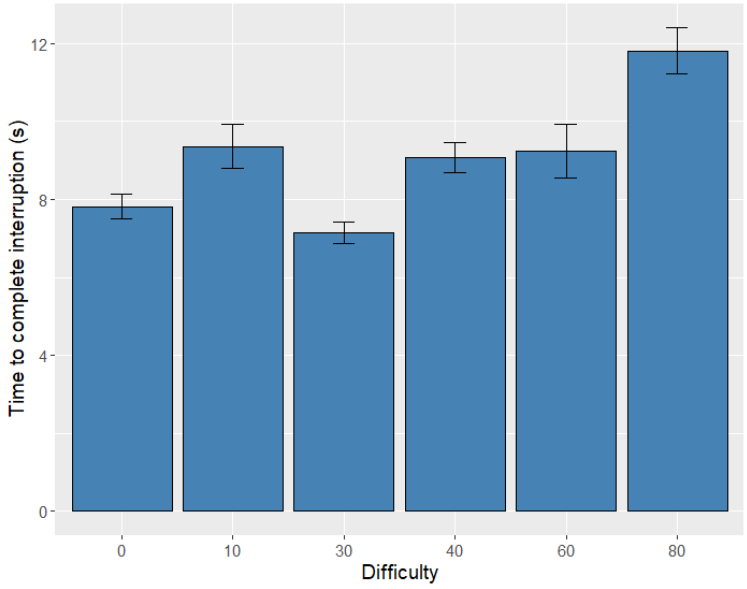


FIGURE 5.6: Time (s) to complete an interruption for each difficulty category.

As expected easier conditions results in a lower interruption time, and harder conditions in a longer interruption. Difficulty 10 seems like the biggest outlier here, this might be due to the subjective experience as mentioned in table 5.3. The other outlier is possible difficulty 30 which drops just below difficulty 0.

5.2.4 Learning to schedule interruptions during the task

The final subsection of the results will look at the effect of the training that the algorithm did while the participant performed the experiment. The averages of the first sudoku will be compared with the averages of the second sudoku, the third sudoku is excluded here since not all participants managed to complete it in time. The expectations are that the algorithm will learn during the first sudoku and the effects of that learning will be visible in the second sudoku, where it will continue to get better. The expectation is that most noticeably the number of interruptions in the zero difficulty category will rise substantially and in the hardest difficulty will greatly decrease.

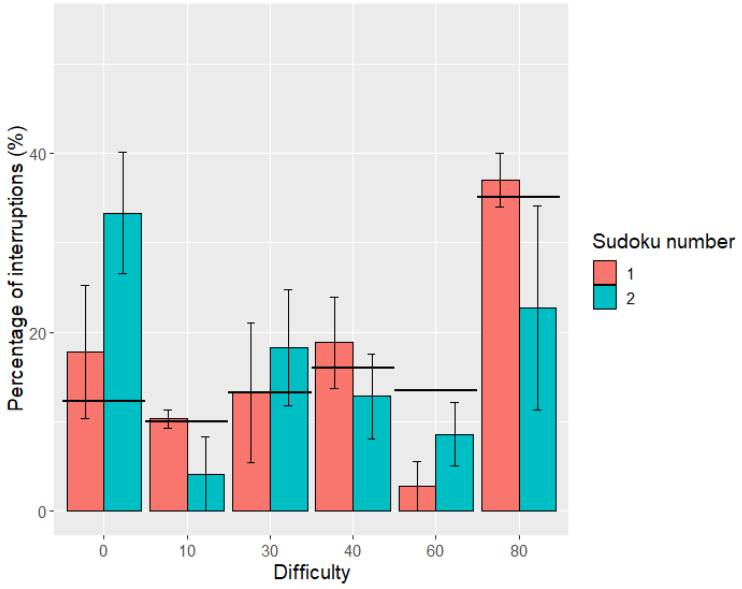
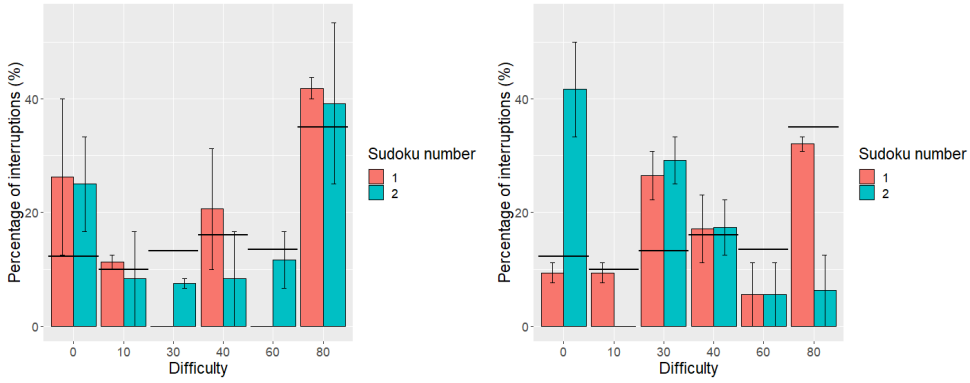


FIGURE 5.7: Percentage of interruptions scheduled during each difficulty category. The black lines show the average amount of time spent on each difficulty as shown in figure 5.5

Figure 5.7 shows some interesting differences between the two sudokus. The first is that the number of interruptions scheduled during the easiest difficulty category (0) increased from 18% to 34%, which is very high considering that only 12.5% of the time is spent on this difficulty. To compensate for this increase the amount of interruptions in the highest difficulty (80) has decreased, from about 37% to 23% . The algorithm also learned to avoid difficulty 10, where participants did not like to be interrupted. Difficulty 60 did increase a little, but was already very low during the first sudoku. Participants rated this highly so that could explain why it is far below the expected value.
Overall it shows that most interruptions took place when expected considering the amount of time spent on each category during the first sudoku, but it learned to avoid the highest difficulty and to schedule during the easiest difficulty. Because of the change in the algorithm that took place from participant four to five, where the amount of trees in the mondrian forest was increased from five to twenty, there might also be a difference in the observed effects.



Five mondrian trees
Participants 3&4

Twenty mondrian trees
Participants 5&6

FIGURE 5.8: Percentage of interruptions scheduled during each difficulty category split based on the size of the mondrian tree. The black lines show the average amount of time spent on each difficulty.

Figure 5.8 shows a clear difference between the two groups of two participants. The first two participants seem to have little to no improvement for most categories, while the final two participants show the improvements seen in the overall figure amplified. This could be due to the small number of trees in the mondrian forest that was used for the first two participants. For the final two participants the amount of interruptions in the easiest difficulty (0) has increased from 10% to 42%, while the amount of time spent on this category is only 12.5%. The hardest difficulty (80) has decreased from 32% to 6%, which again is huge considering that the amount of time spent on this category is 35%. Other difficulties behaved mostly as expected, difficulty 10, which participants rated highly, went down to nothing. Difficulty 40 did not change, with the average answer being very close to three this is to be expected. Difficulty 60 did not change but is still well below the average, which is to be expected since the average response was higher than the average. Finally difficulty 30 slightly increased and was already well above the average, this also corresponds to the comparable low response given by participants to this category.

Chapter 6

Discussion

In this paper it was proposed to develop a new interruption management system that is task-independent and able to learn using feedback from participants. In order to achieve this three research questions were established. The first research question was about developing a new experimental task to test the task-independent IMS developed in this paper. The second research question was aimed at developing a classifier that was pre-trained on data from a previous experiment that could reduce the amount of interruptions scheduled at high workload moments. Finally, the third research question was about extending the classifier so it could learn from the participant while performing the task using online learning, which would further reduce the amount of interruptions at high workload moments.
A new experimental task was developed that was based on sudokus to test the task independence of this new system. Using the difficulty of different techniques used to solve sudokus, high and low workload moments were established. Along with the new experimental task an IMS was developed that made use of a classifier that is capable of both offline and online training. The classifier was pre-trained on eye data of another experiment, and could continue training on the eye-data of the participant performing the experiment. The participant could provide feedback when an interruption was scheduled, which was used to improve the system based on the participants preferences. A message to schedule an interruption and the feedback of the participants are the only two connections required to run the IMS, making it completely task-independent.
While not enough data could be gathered due to the pandemic to obtain significant results, the pilots gave promising results. The experimental task seemed to have high and low workload moments based on the difficulty of the technique required to solve a cell. Unfortunately, no results were obtained that could indicate whether the pre-training of the classifier caused an increase in the number of interruptions scheduled at low workload moments. Finally, the results seemed to indicate that the online training of the classifier caused it to avoid scheduling interruptions at moments that the participant labelled as disruptive and instead schedule the interruptions at moments that the participant labelled as less disruptive.
To definitely answer the research questions a full experiment needs to be conducted, in the first section the design of the final experiment will be explained. Afterwards, the results of the pilots are examined in more detail to investigate what changes could be made before the final experiment. Finally, the implications of this study will be discussed.

6.1 Design of the final experiment

As stated before, the pilots do not provide enough data to answer the research questions. Not only are more participants required, there should also be more conditions

to investigate the effects of the IMS. These four conditions are stated below:

- No interruptions
- Random interruptions
- Pre-trained algorithm only
- Pre-trained algorithm including online training

The four conditions will be compared with each other to find the true effect of the offline and online learning of the classifier, including the possible effects of the new experimental task. Comparing both variations of the classifier (the last two conditions) to the random condition should show whether the IMS managed to significantly reduce the number of interruptions during high workload moments, and whether these were scheduled during low workload moments instead. The 'no interruptions' category is used to determine any potential effects that the new experimental sudoku task might have had.

6.2 Possible improvements

The first improvement is the number of interruptions. The second pilot was slightly lacking in interruptions as it sometimes took too long for the threshold to go down or if the algorithm could continuously not find a good moment for an interruption. While the plan was to have an interruption every minute at the least, there were on average about 0.7 interruptions per minute. Since the threshold mechanism is rather simple, there are many improvements that can be made to rectify that. Currently it starts at 0.8 and stays there for twenty seconds, after that it will decrease by 0.01 every two seconds. This did not decrease fast enough, after sixty seconds had passed since the last interruption this threshold would be at 0.7, which was not enough to force an interruption. An alternative solution would be to work with brackets, after twenty seconds have passed the threshold decreases by 0.1 (to 0.7). Afterwards the threshold reduces by 0.1 every fifteen seconds. After 55 seconds have passed the chance of an interruption would be very likely (threshold at 0.6), and after another fifteen seconds the threshold drops to below chance, basically forcing an interruption. This would provide a better spread of interruptions, and the brackets can be monitored, so the threshold level can be taken into account to see if the interruption was forced or not.
There were a couple communication issues with the eyetracker, pupil size was retrieved from the eyetracker but sadly eye events were not. The program automatically filled in the value zero because of this problem for the final two inputs of the input vector (the percentage of saccades and blinks). In section 3.3.1 it was observed that these last inputs were often very important to the decision tree process and these were also part of the psycho-physiological features used in the experiment by Züger and Fritz (2015). Having these values always at zero might have caused a potential bias in the algorithm. Since there is no insight into the way the mondrian trees split the data, the true effects of this bias are unknown. Including these eye events for the final experiment can be beneficial to the accuracy of the classifier.
While observing the participants performing the task the output of the mondrian

forest could be observed compared to the threshold. This showed how the algorithm developed over time, and how strongly it reacted to updates. The observations showed that many times the algorithm reacted very strongly to a new update, whether the feedback of the participant to the interruption was positive or negative. There were, for instance, moments when the algorithm managed to schedule interruptions in the difficulty zero category four times in a row, but after that an interruption was forced at an inopportune time as judged by the participant. This resulted in a severe effect on the output of the algorithm and caused it to lose some of the progress that it made during the streak before. By adjusting the severity of the update mechanism the algorithm should be able to learn better and more efficiently. Unfortunately, the effect of the pre-trained algorithm cannot be compared to a random condition to find the effect of pre-training. By examining the online learning effect from sudoku one to sudoku two, however, it does not seem to have a very large impact. The number of interruptions scheduled during each difficulty category in Figure 5.7 are close to the average amount of time spent in that condition. This seems to suggest that the pre-training did not have a large effect, and possibly that the pupil data from one participant might not generalise well to another participant. The true effect however will need to be examined in a full experiment before any conclusions about this can be made.
For future experiments a powerful enough machine should be used to run the experiment. Using a laptop for the experiment provided a huge bottleneck in the amount of trees that can be used in the mondrian forest, which generally improves with the amount of trees. Five trees for the first two participants that were evaluated was too small, and the results were also not impressive. Twenty trees showed much better results compared to only five. The original plan was to run the experiment on a mondrian forest made of fifty trees which could probably further increase the accuracy of the classifier.
In the results (section 5.2.2) the feedback that the participants provided on when the interruption was scheduled was examined. While difficulty ten is a really easy category to complete, because you only need to count from one to nine and find the number missing, it was experienced as badly as the hardest difficulty category. A similar effect also happened in difficulty sixty, which was easy to solve but received negative feedback overall. This could be due to the fact that in this experimental task difficulty and workload do not completely align. Often during the task the participants might be searching for the right technique to use, which might require less workload than while the participant is using the technique to solve it. Since difficulty sixty, and especially difficulty ten, are easy to detect, most of the reaction time is spent solving the cell, and therefore gets worse feedback. Another solution could be that participants just do not like to be interrupted while solving these shorter tasks, resulting in them rating it worse. Since the algorithm did respond to the feedback, interruptions during these two difficulties greatly decreased, resulting in even less data to find out what causes this disparity. This suggests that even though these results were unexpected, the algorithm learned regardless and avoided the moments that the participant labelled as disruptive. This is exactly what an IMS aims to achieve, and is a positive results regardless of the unexpected feedback to the difficulty categories.

6.3 Implications of the study

This task independent IMS of this study aims to improve upon of previous research. Previous interruption management systems that were developed often relied upon task specific information (McFarlane 2002; Arroyo and Selker 2011), or used less available measures such as an EEG (Züger and Fritz 2015). A previous IMS study developed a system that was solely based on pupil dilation to find good moments for interruptions (Katidioti, Borst, Bierens de Haan, et al. 2016), the results were promising but the overall improvement over the random condition was not large. The IMS developed in this research solved these issues by using a classifier that only relies on eye data and is therefore task-independent. Additionally, the IMS described in this paper makes use of a classifier that can use additional data sources and that can continue training on the participant to increase the performance of the classifier. Although definite results were not achieved during this thesis, the IMS shows great promise that it indeed does improve on previous research.
Eye-data is an important measure of working memory, and therefore workload (Beatty 1982). Previous research, such as Katidioti, Borst, Bierens de Haan, et al. (2016) has used only the percentage change in pupil dilation or only eye-blinks (Züger and Fritz 2015). The IMS described in this paper is capable of using both pupil dilation and eye-blinks. It can also take in more data, such as the mouse and keyboard information used in the system of Arroyo and Selker (2011) to increase accuracy.
By planning interruptions at low workload moments the disruption of the interruption can be minimised (Iqbal and Bailey 2006). Unfortunately, we are not capable of determining whether another person is in a state of high or low workload (Fogarty et al. 2005). An IMS can monitor small signs that indicate a state of low workload, and schedule interruptions during this time period to decrease the time necessary to recuperate after the interruption and minimise the errors made due to the interruption (Katidioti, Borst, Bierens de Haan, et al. 2016). The IMS presented in this paper has potential to achieve better results than the other interruption management systems that preceded it.
If the IMS presented in this paper will achieve similar, or even better, results in a full experiment, it could help to increase efficiency and reduce errors made for many people. The IMS described in this study makes use of an eyetracker to classify moments as a high or low workload moment, which is something that most people do not have access to. For some jobs, such as air traffic controller or someone operating a military drone, the negative effects of interruptions might not be acceptable, making the use of an eye-tracker worthwhile. Our mobile phones also come equipped with increasingly better cameras. A study by McAnany et al. (2018) suggested that mobile phone based eye-trackers are already highly promising and will continue to get better, which would make eye-tracking possible for all. Finally, the IMS presented in the paper can also use different inputs, so it could potentially use any available inputs to classify high and low workload moments.
As it stands the IMS presented in this paper is not yet usable in large scale applications, but it does provide the next step in achieving the goal of having an IMS available for all to reduce the cost and frustration of interruptions. While everyone is still working from home due to the pandemic, everyone is dealing with many new interruptions during their work. The current system might not help those in need of an interruption management system now, but paves the way to have one available in the future. Since the system can be specified to the user and the environments the user works in, it could be used in any situation, whether working at the office or at home.

Chapter 7

Conclusion

The pilots showed very promising results, even with the low number of participants. The possible inverse correlation between the answer of the participants and the amount of interruptions seems to indicate that a classifier can be trained to learn from the feedback of a participant. Most noticeably the increase in interruptions scheduled during the easiest difficulty category and the decrease in the hardest difficulty show that the algorithm learned to find good moments to schedule interruptions. Making the algorithm larger by adding more mondrian trees seemed to increase this effect, suggesting that an even larger mondrian forest might achieve even better results.
The experimental task presented in the paper seems to work well for interruption research. There are some unexpected results in this experimental task though, where some difficulties were easier or harder than expected, but the IMS seemed to be able to learn these categories nonetheless based on participant feedback. The algorithm, which received no input from the experimental task except for the participant feedback, seems to have learned to avoid scheduling interruptions during these categories because participants did not like to be interrupted at that moment.
Unfortunately, none of the research questions can be answered definitely in this paper due to the lack of data. Currently only the results from the pilots can be used to examine what the potential answers to the research questions could be. The research questions stated in section 1.2 were:

- Can a new experimental test program be constructed that shows clear moments of high and low workload?
- Can an IMS be constructed that is pre-trained on data from another experiment to increase the number of interruptions scheduled at low workload moments?
- Can the IMS use online learning to further decrease the disruption of the interruptions?

Preliminary results suggest that the first research question can be answered positively, while there is some unexpected behaviour the overall structure does suggest that there is a difficulty curve for the different categories.
The second research question cannot be answered from the current results. The pretrained algorithm without the online training was not tested and compared to a random condition, which makes it impossible to give an answer at this moment.
Finally, the last research question seems to have the most positive results. The decrease in the disruption of interruptions seems to be very clear in the final figures discussed in the results. For the full effects this model needs to be compared to a random condition but the results seem so strong that a full experiment might show similar positive results.

Bibliography

Adamczyk, Piotr D and Brian P Bailey (2004). "If not now, when? The effects of interruption at different moments within task execution". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 271–278.

Alpaydin, Ethem (2020). *Introduction to machine learning*. MIT press.

Altmann, Erik M and J Gregory Trafton (2007). "Timecourse of recovery from task interruption: Data and a model". In: *Psychonomic Bulletin & Review* 14.6, pp. 1079–1084.

Arroyo, Ernesto and Ted Selker (2011). "Attention and intention goals can mediate disruption in human-computer interaction". In: *IFIP Conference on Human-Computer Interaction*. Springer, pp. 454–470.

Beatty, Jackson (1982). "Task-evoked pupillary responses, processing load, and the structure of processing resources." In: *Psychological bulletin* 91.2, p. 276.

Borst, Jelmer P, Niels A Taatgen, and Hedderik van Rijn (2015). "What makes interruptions disruptive?: A process-model account of the effects of the problem state bottleneck on task interruption and resumption". In: *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. ACM, pp. 2971–2980.

Breiman, Leo (2001). "Random forests". In: *Machine learning* 45.1, pp. 5–32.

Edwards, Jeffrey R and Nancy P Rothbard (2000). "Mechanisms linking work and family: Clarifying the relationship between work and family constructs". In: *Academy of management review* 25.1, pp. 178–199.

Fogarty, James et al. (2005). "Predicting human interruptibility with sensors". In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 12.1, pp. 119–146.

Gillie, Tony and Donald Broadbent (1989). "What makes interruptions disruptive? A study of length, similarity, and complexity". In: *Psychological research* 50.4, pp. 243–250.

González, Victor M and Gloria Mark (2004). ""Constant, constant, multi-tasking craziness" managing multiple working spheres". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 113–120.

Hess, Eckhard H and James M Polt (1960). "Pupil size as related to interest value of visual stimuli". In: *Science* 132.3423, pp. 349–350.

Hodgetts, Helen M and Dylan M Jones (2006). "Interruption of the Tower of London task: support for a goal-activation approach." In: *Journal of Experimental Psychology: General* 135.1, p. 103.

Hoeks, Bert and Willem JM Levelt (1993). "Pupillary dilation as a measure of attention: A quantitative system analysis". In: *Behavior Research Methods, Instruments, & Computers* 25.1, pp. 16–26.

Hoi, Steven CH et al. (2018). "Online learning: A comprehensive survey". In: *arXiv preprint arXiv:1802.02871*.

Iqbal, Shamsi T, Piotr D Adamczyk, et al. (2005). "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*. ACM, pp. 311–320.

Iqbal, Shamsi T and Brian P Bailey (2005). "Investigating the effectiveness of mental workload as a predictor of opportune moments for interruption". In: *CHI'05 extended abstracts on Human factors in computing systems*, pp. 1489–1492.

— (2006). "Leveraging characteristics of task structure to predict the cost of interruption". In: *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pp. 741–750.

Isaacs, Ellen et al. (1997). "Informal communication re-examined: New functions for video in supporting opportunistic encounters". In: *Video-mediated communication* 997, pp. 459–485.

Kahneman, Daniel and Jackson Beatty (1966). "Pupil diameter and load on memory". In: *Science* 154.3756, pp. 1583–1585.

Kahneman, Daniel, Bernard Tursky, et al. (1969). "Pupillary, heart rate, and skin resistance changes during a mental task." In: *Journal of experimental psychology* 79.1p1, p. 164.

Katidioti, Ioanna, Jelmer P Borst, Douwe J Bierens de Haan, et al. (2016). "Interrupted by your pupil: An interruption management system based on pupil dilation". In: *International Journal of Human–Computer Interaction* 32.10, pp. 791–801.

Katidioti, Ioanna, Jelmer P Borst, and Niels A Taatgen (2014). "What happens when we switch tasks: Pupil dilation in multitasking." In: *Journal of experimental psychology: applied* 20.4, p. 380.

Krawczyk, Bartosz et al. (2017). "Ensemble learning for data stream analysis: A survey". In: *Information Fusion* 37, pp. 132–156.

Laeng, Bruno, Sylvain Sirois, and Gustaf Gredebäck (2012). "Pupillometry: A window to the preconscious?" In: *Perspectives on psychological science* 7.1, pp. 18–27.

Lakshminarayanan, Balaji, Daniel M Roy, and Yee Whye Teh (2014). "Mondrian forests: Efficient online random forests". In: *Advances in neural information processing systems*, pp. 3140–3148.

Mark, Gloria, Victor M Gonzalez, and Justin Harris (2005). "No task left behind?: examining the nature of fragmented work". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, pp. 321–330.

Mark, Gloria, Daniela Gudith, and Ulrich Klocke (2008). "The cost of interrupted work: more speed and stress". In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, pp. 107–110.

McAnany, J Jason et al. (2018). "iPhone-based pupillometry: a novel approach for assessing the pupillary light reflex". In: *Optometry and Vision Science* 95.10, p. 953.

McFarlane, Daniel C (2002). "Comparison of four primary methods for coordinating the interruption of people in human-computer interaction". In: *Human-Computer Interaction* 17.1, pp. 63–139.

McFarlane, Daniel C and Kara A Latorella (2002). "The scope and importance of human interruption in human-computer interaction design". In: *Human-Computer Interaction* 17.1, pp. 1–61.

Monk, Christopher A, J Gregory Trafton, and Deborah A Boehm-Davis (2008). "The effect of interruption duration and demand on resuming suspended goals." In: *Journal of experimental psychology: Applied* 14.4, p. 299.

Nijboer, Menno et al. (2016). "Contrasting single and multi-component working-memory systems in dual tasking". In: *Cognitive psychology* 86, pp. 1–26.

Oza, Nikunj Chandrakant and Stuart Russell (2001). *Online ensemble learning*. University of California, Berkeley.

Pedregosa, Fabian et al. (2011). "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12, pp. 2825–2830.

Roy, Daniel M, Yee Whye Teh, et al. (2008). "The Mondrian Process." In: *NIPS*, pp. 1377–1384.

Safavian, S Rasoul and David Landgrebe (1991). "A survey of decision tree classifier methodology". In: *IEEE transactions on systems, man, and cybernetics* 21.3, pp. 660–674.

Salvucci, Dario D and Peter Bogunovich (2010). "Multitasking and monotasking: the effects of mental workload on deferred task interruptions". In: *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 85–88.

Sudoku techniques (2002). URL: https://www.conceptispuzzles.com/index.aspx?uri=puzzle%5C%2Fsudoku%5C%2Ftechniques.

Züger, Manuela and Thomas Fritz (2015). "Interruptibility of software developers and its prediction using psycho-physiological sensors". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 2981–2990.

Appendix A

Algorithm test

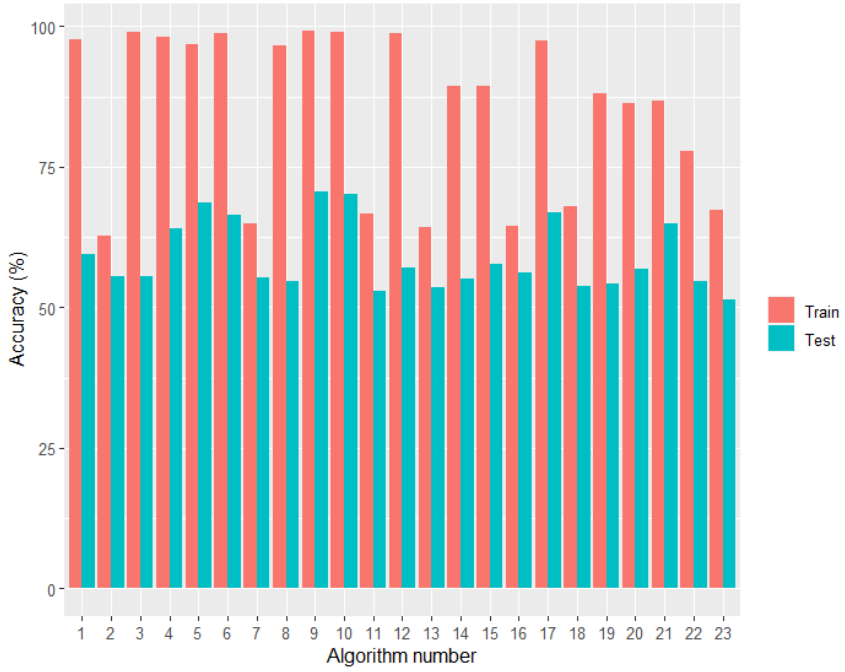


FIGURE A.1: All Algorithms used for the test in section 3.2

1. MultiLayer Perceptron (MLP)
2. K-nearest neighbours
3. Support Vector Machines
4. Decision Tree
5. Random Forest
6. Adaboost
7. Gaussian Bayes
8. Quadratic Discriminant Analysis
9. Gradient Booster
10. Extremely Randomised Forest
11. Stochastic Gradient Descent
12. NuSVC
13. Nearest Centroid
14. Logistic Regression
15. Linear Discriminant Analysis
16. Bernouilli Naive Bayes
17. Bagging Classifier
18. Passive Aggressive
19. Ridge Classifier
20. Ridge Classifier CV
21. Extra Trees
22. LinearSVC
23. Perceptron

Appendix B

Sudokus used

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Sudoku 1 [1]

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Sudoku 1 - answer

6	8				5			
1			4	7		3	6	
7							4	9
				8			5	
	7		1	9	3		8	
	2			4				
8	4							7
	3	7		2	9			6
			7	8			3	8

Sudoku 2 [1]

6	8	4	9	3	5	7	2	1
1	9	2	4	7	8	3	6	5
7	5	3	6	1	2	8	4	9
3	6	1	2	8	7	9	5	4
4	7	5	1	9	3	6	8	2
9	2	8	5	4	6	1	7	3
8	4	6	3	5	1	2	9	7
5	3	7	8	2	9	4	1	6
2	1	9	7	6	4	5	3	8

Sudoku 2 - answer

5				3		4		7
3		2	6			9		
		7		4			8	
	2	6				8		1
			7	6			4	
4	7					2		
2	1					7		
		4			6	5		
				9	7		3	

Sudoku 3 [1]

5	6	8	9	3	1	4	2	7
3	4	2	6	7	8	9	1	5
1	9	7	2	4	5	6	8	3
9	2	6	3	5	4	8	7	1
8	5	1	7	6	2	3	4	9
4	7	3	8	1	9	2	5	6
2	1	9	5	8	3	7	6	4
7	3	4	1	2	6	5	9	8
6	8	5	4	9	7	1	3	2

Sudoku 3 - answer

[1] Sudokus obtained from https://www.websudoku.com/

Appendix C

Explanation Document

Introduction:

In this task you will be asked to solve the sudokus that appear on screen. The program will determine the square that you will need to fill in. You can move through the possible answers, ranging from 1 to 9, by pressing the left and right arrow keys. To lock your answer, press the spacebar. A green or red ring will appear around the square if your answer was correct or incorrect. After a short pause the program will fill in the correct number in the spot, so you'll never have to worry about past mistakes. After the break a new square will be selected for you to fill in until the sudoku is complete. After one sudoku is completed a new one will appear after a short pause. There are a total of three sudokus.

Basic knowledge:

A sudoku consists of nine by nine squares that either contain a number or are empty. The goal of the puzzle is to fill in all the numbers according to the rules of a sudoku. In each row, column or 3x3 box every digit needs to be present once, so it can't happen that a number appears twice or more in the same row, column or box.

Sudoku techniques:

Below I've listed the different techniques that you are expected to know to finish the sudokus. Please read the explanations carefully and look at all the examples.

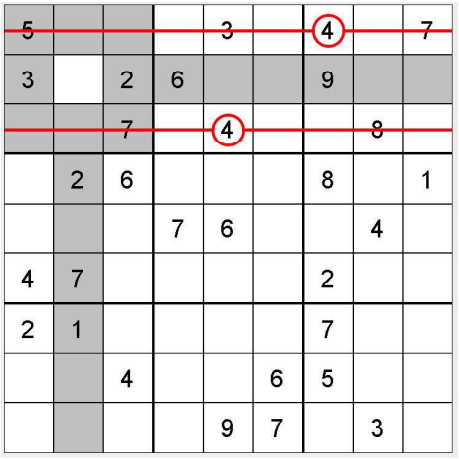
Singe option:

If a row, column or box is completely filled in except for the target square the answer has to be the only number that does not appear in the row, column or box. In the example below all numbers are present in the first row except for the number 1, so the answer is 1.

5	6	8	9	3		4	2	7
3		2	6			9		
		7		4			8	
	2	6				8		1
			7	6			4	
4	7					2		
2	1					7		
		4			6	5		
				9	7		3	

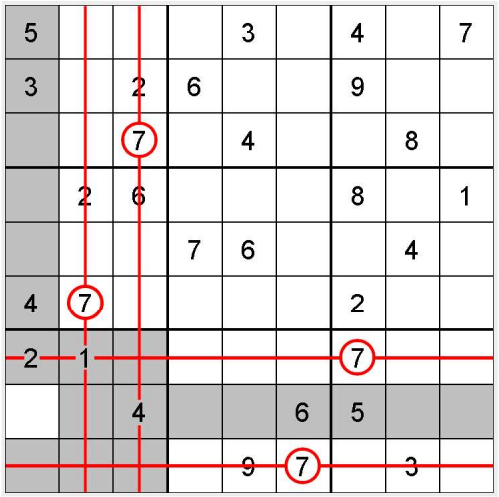
Scanning in one direction:

Since there can only be one occurrence of a number in a row or column, you can scan to determine where a number has to occur in a box. By looking at the example you see that there are two occurrences of a 4 in the rows next to the target. Since there can be no fours in the top and bottom row of the box and the other two squares in the box are filled in the number 4 has to occur in the target square.



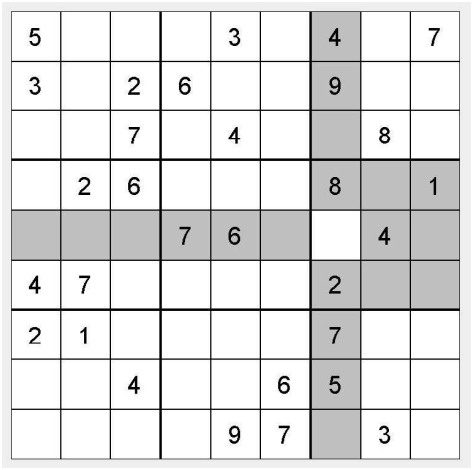
Scanning in two directions:

In some situations, just one single direction of scanning is not enough, in these cases you can combine horizontal and vertical scanning to determine the only possible square in a box that can contain a certain number. In the example below you see that the number 7 is already present in all rows and all columns of the box except for the row and column of the target, so the number 7 has to be filled in here.



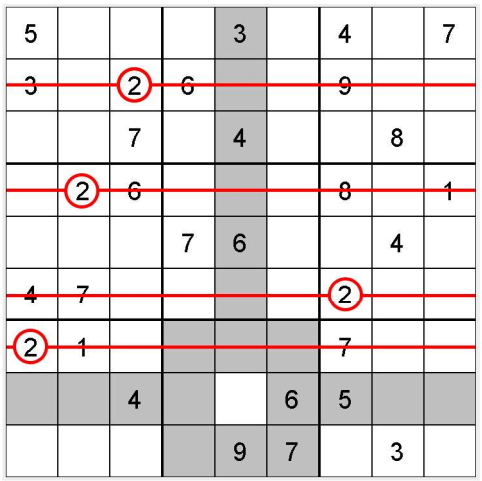
Single candidate:

A single candidate is when only one number can be filled into the designated square because the remaining eight numbers are already present in the row, column and box. In this example you see that the numbers 1, 2, 4 and 8 are in the box; 4 5,7 and 9 in the column, and 6 and 7 in the row. This means that the numbers 1,2,4,5,6,7,8 and 9 are all present which means that the only number that can be filled in here is 3.



Missing number:

A missing number is when a row or column has no other option for a certain number than the target location. By looking at the rows, columns or boxes crossing the column or row of the target you can sometimes find that there is only one possibility left for the location of a number in that row or column. In the example below, the rows crossing the column of the target all eliminate the possibility of the number 2 occurring in any other location than the target, which means that the answer must be 2.



Interruptions:

While performing this task there will be occasional interruptions in the form of a simple mathematical function. An example of which will be given below:

4x - 3 = 9

You'll be asked to find what the variable x is in this equation. You can answer in a similar manner as the sudoku, by pressing the right, to add one to the current digit, or left, to subtract one from the current digit. You answer by pressing spacebar, only a correct answer will be accepted.

Afterwards there will be a short evaluation of this interruption. It will ask you how much this task interrupted your thinking (on a scale from 1 to 5) on the main task, the sudoku. If it hardly interrupted your thinking you answer 1 and if it completely ruined your train of thought answer 5. After submitting your answer with another spacebar press the sudoku will continue as normal.

Pilot:

As this is a pilot study there is still a lot that can (and most likely will) go wrong. The purpose of the pilot is for me to observe what happens so I can figure out what might be going wrong. Please bear with me during this process.

Before you start:

Please keep your eyes focused on the screen as much as possible and try not to move your head from the stand.

Please try to balance time and accuracy, try to fill in each square as quickly as possible but try not to make any mistakes. If you really cannot figure out a target then you can just make a guess but try not to do so.

If you have any questions please let me know!