Universität Bremen

Behaviour-based detection of Visual Interaction Obstacles with 1D and 2D Convolutional Neural Networks

Bachelor Thesis at the Cognitive Systems Lab
Prof. Dr.-Ing. Tanja Schultz
Faculty 3: Mathematics and Computer Science
University of Bremen

by

Anthony Mendil

Supervisor:

Mazen Salouz

Examiner:

Felix Putze
Unknown

Day of registration:	1. Mustermonat 1851
Day of submission:	3. Mustermonat 1963

I hereby declare that I am writing this work independently and have not used any sources or resources other than those specified.

Bremen, the 3. Mustermonat 1963

Zusammenfassung

... deutsch ...

Der deutsche Abstract wird in jedem Fall benötigt.

Abstract

... english ...

Der englische Abstract wird nur benötigt, wenn die Arbeit in englischer Sprache verfasst wird.

Contents

1 Notes **1**
- 1.1 Anmerkungen . . . 3

2 Introduction **5**

3 Memory Game **7**

4 CMM-based Cognitive User Simulation **9**

5 Collected Data **11**
- 5.1 Behavioural data . . . 11
- 5.2 Brain activity . . . 13

6 Data Preparation **15**
- 6.1 Similarity matrix creation . . . 15
  - 6.1.1 Conceptualization . . . 15
  - 6.1.2 Screenshot extraction . . . 19
  - 6.1.3 Implementation . . . 20
  - 6.1.4 Testing . . . 21
  - 6.1.5 Final Matrix . . . 21
- 6.2 Incorporation of the new similarity matrix . . . 23
- 6.3 Removal of invalid logs . . . 23
- 6.4 Simulation of user behaviour . . . 24
- 6.5 Sorting logs by quality . . . 24
- 6.6 Evaluation of simulation . . . 25
  - 6.6.1 Results before and after changing the simulator . . . 25
  - 6.6.2 Search for a subset of the data . . . 28
  - 6.6.3 Conclusion of the evaluation . . . 31
- 6.7 Feature generation . . . 32
  - 6.7.1 1D CNN features . . . 32
  - 6.7.2 2D CNN features . . . 32

7 Convolutional neural networks **39**

8 Classification **41**
- 8.1 Hardware resources . . . 41
- 8.2 Models . . . 41
  - 8.2.1 1D CNN . . . 42
  - 8.2.2 2D CNN . . . 43
- 8.3 Training setup . . . 44

8.4 Training results and evaluation 45

9 Voting based system **51**

10 Conclusion **55**

10.1 Prospect 55

11 Appendix **57**

Bibliography **59**

Index **61**

List of Figures

1.1 Bild kurz 4

3.1 Bild kurz 8
3.2 Bild kurz 8

6.1 Bild kurz 16
6.2 Bild kurz 17
6.3 Bild kurz 22
6.4 Bild kurz 25
6.5 Bild kurz 25
6.6 Bild kurz 26
6.7 Bild kurz 26
6.8 Bild kurz 30
6.9 Bild kurz 30
6.10 Bild kurz 34
6.11 Bild kurz 34
6.12 Bild kurz 36
6.13 Bild kurz 36

8.1 Bild kurz 42
8.2 Bild kurz 43
8.3 Bild kurz 48
8.4 Bild kurz 49
8.5 Bild kurz 49
8.6 Bild kurz 49
8.7 Bild kurz 49

List of Tables

1.1 Tabelle mit kurzer Unterschrift 3

5.1 add caption. 11

6.1 add caption. 22

6.2 Before change to simulator. p and t values in paired t-test for different comparissons of matching pairs per round. All 1000 simulated games per real game were used. The following abreviatrions are used: real glare effect - r_g, real no obstacle - r_n, simulated glare effect - s_g, simulated no obstacle - s_n 27

6.3 Before change to simulator. p ant t values in paired t-test for different comparissons of penalties per round. All 1000 simulated games per real game were used. The following abreviatrions are used: real glare effect - r_g, real no obstacle - r_n, simulated glare effect - s_g, simulated no obstacle - s_n 27

6.4 p and t values in paired t-test for different comparissons of matching pairs per round. All 1000 simulated games per real game were used. The following abreviatrions are used: real glare effect - r_g, real no obstacle - r_n, simulated glare effect - s_g, simulated no obstacle - s_n 28

6.5 p ant t values in paired t-test for different comparissons of penalties per round. All 1000 simulated games per real game were used. The following abreviatrions are used: real glare effect - r_g, real no obstacle - r_n, simulated glare effect - s_g, simulated no obstacle - s_n 28

6.6 Before change to simulator. p and t values in paired t-test for different comparissons of matching pairs per round. For the simualted data only the best 10 simulated logs from each real log are used. Only the first 10 rounds are used. The following abreviatrions are used: real glare effect - r_g, real no obstacle - r_n, simulated glare effect - s_g, simulated no obstacle - s_n 29

6.7 Before change to simulator. p values in paired t-test for different comparissons of penalties per round. For the simualted data only the best 10 simulated logs from each real log are used. Only the first 10 rounds are used. The following abreviatrions are used: real glare effect - r_g, real no obstacle - r_n, simulated glare effect - s_g, simulated no obstacle - s_n 29

6.8 p and t values in paired t-test for different comparissons of matching pairs per round. For the simualted data only the best 10 simulated logs from each real log are used. Only the first 10 rounds are used. The following abreviatrions are used: real glare effect - r_g, real no obstacle - r_n, simulated glare effect - s_g, simulated no obstacle - s_n 30
6.9 p values in paired t-test for different comparissons of penalties per round. For the simualted data only the best 10 simulated logs from each real log are used. Only the first 10 rounds are used. The following abreviatrions are used: real glare effect - r_g, real no obstacle - r_n, simulated glare effect - s_g, simulated no obstacle - s_n 30
6.10 mean values of matching pairs for real and simulated games. sd10x. for the first 10 rounds 31
6.11 add caption. Upper and lower boundaries are inclusive. 35

8.1 add caption 41
8.2 add caption 41
8.3 1D CNN. Best accuracies and losses for 5, 10, 15 and 20 steps. Before the changes to the simulator. 46
8.4 2D CNN. Best accuracies and losses for 5, 10, 15 and 20 steps. Before the changes to the simulator. 46
8.5 1D CNN. Best accuracies and losses for 5, 10, 15 and 20 steps. After the changes to the simulator. 46
8.6 2D CNN. Best accuracies and losses for 5, 10, 15 and 20 steps. After the changes to the simulator. 47
8.7 add caption. Upper and lower boundaries are inclusive. 48
8.8 add caption. Upper and lower boundaries are inclusive. 49

9.1 add caption. 52

1. Notes

- fragen an mazen:
- demographic
- original mapping of cards to colours
- nachfragen ob static mapping so ist wie ich denke
- sind whickser nur standardabweichung oder was ist das?
- ist es ok dass ich meine eigene tabelle gemacht habe für delta e interpretation? im interent stand es gibt keine allgemeine..hab auch keine von cielab gefunden
- ich brauche mll knecht

- Einleitung: Was es für verschiedene Obstacles gibt, was schon gemacht wurde, was ich mache, wieso keine eeg daten obwohl sie da sind (sind interaktion obstacle zu erkennen und nutzung zu verbessern ist nicht da wenn man eeg maske tragen muss)

- memory game kurz erklären

- libraries und so die ich benutz habe? pandas keras..

- **TODO: vizualizing of intermediate activations reinnehmen? (ist auskommentiert!)**

- **simulation**:
- simulatin damit man mehr daten hat
- generell simulator erklären und sinn von similarity matrix
- similaity matrix erstelung und gedanken (plus anpassungen an memory game, anpassungen am simulator damit similarity matrix benutzt werden kann)
- es gab invalid logs und hab code geschrieben der das überprüft damit man die rausnehmen kann. vor simulation (validLogsCollecot)
- similarity matrix mapper, waren vorher nicht gmapped und mapper macht das und ersetz alte matrix durch die neue für glare effect
- erklären wie ich korrektheit der matrix überprüft habe
- erklären wie ich korrektheit der farbextraktion und unterschied berechnung überprüft habe (online rechner und anfangs matrix für no obst erstellt und mit alten verglichen aber gibg nicht weil struktur nicht zu erkennen war bei alter matrx von vorherigen arbeieten)
- initially simulationergebnisse mit plots für qualität (plus erklärung)
- sagen was noch nicht optimal und, was am simulator dafür geändert wurde (2 sachen: random decay ab 10 zügen plus eine andere sache) mit begründung und wie die ergebnisse am ende aussahen (zwischenschritte eher niht glaube ich. nur kurz erwähnen)
- darüber reden wie es im realfall ist: wir benutzen nicht alle simualtionen sondern nur die besten, plots zeigen mi nur den besten (vor und nach änderung vielleicht, oder nur nach änderung (aber dann sieht man nicht das änderung sinnvoll war))
- paried t-test kram um signifikante unterschiede /keine unterschiede zu zeigen für verschiedene bedingenen (siehe mazens nachricht)

- **modelle**:
- feature engenierring: also was die komponenten sind und so, wie sie berechnet wurden
- für 1d cnn wuren die so übernommen
- erklärung komponenten von 1d cnn
- (villeicht auch mal nicht mit allen featires probieren, aber da hatte ich problme mit dem cnn. man müsste glaube ich die struktur ändern)
- struktur 1 d cnn mit begründung
- 2d cnn komponenten erklärunen (snythetic image erklären und zeigen wie berechnet wurde und wie es aussieht)

dies	ist	eine	Tabelle
mit		zwei	Zeilen

Table 1.1: Tabelle mit einer langen Unterschrift

- idee von 2d cnn für diesen fall
- 2d cnn struktur erklären und begründen

- **Training und analyse**:
- vielleiht gucken welche falsch erkannt werden und woran es liegt, also wenn die zum beispiel echt schlect oder gut sind obwohl es nicht so sein sollte (rausnhemen und gucken wie ergebnisse sind, vielleicht nur bei bestem modell)
- in gleichen tabellen ergebnise vor änderung am simulator und nach änderung betrachten und vergleichen
- training auf welchem rechner/n,was von: cpu oder gpu oder beides, hardware kurz erwähnen, vpn (fernzugriff)
- train test splt, leave one out k fold (+ begründung mit deep learing ..reliable results etc, randomness)
- kurz erklären wieso weniger züge besser sind bei dieser hci erkennung
- (vielleicht kram zu adaptive learning rate ändern und gucken wie es so ist, ansonsten begründen wieso ich das nicht brauche)
- 20 steps und 40 steps für beide trainieren mit sd0x bis sd10x plus sd20x
- (entweder so dass 1d cnn mit 20 steps auch konvergence erreicht oder danach nochmal anpassung von 20 steps 1d cnn damit es kovergiert)
- darüber schreiben dass letzen n züge nicht mehr so gut simuliert sind (zeigen) und deshalb 40 steps nicht so viel besser ist
- mit maximalen guten steps (glaube 16 züge oder so) trainieren und vergleichen
- statistical tests um signifikante unterschiede /keine unterschiede zu zeigen für verschiedene modelle etc (siehe mazens nachricht)

- in anhang alle skripte aufzählen wie markus

- **TODO: tabbelen unterschrift muss glaube ich dadrunter und nicht drüber!**

- special thanks to mazen for the great support. And special thanks to the csl for letting me use their servers for data storing and training.

1.1 Anmerkungen

Zitationen [Rab89] sind keine Wörter sondern Referenzen und stellen somit keinen Teil des Satzes dar. In anderen Worten: Der Satz muss auch noch funktionieren, wenn die Zitation einfach entfernt wird.

Index-Einträge

Wir haben Tabellen 1.1 und Bilder 1.1.

Figure 1.1: Bildunterschrift

2. Introduction

- papers von denen lesen. da sind richtig gute ineleitunegn an denen man sich orierntieren kann.
-

- relevance of topic - glare detectin has been done before with for example a light detector but not with behavioural data (volatile?) -

Neural networks greatly benefit from lots of data. As the number of participants that provided data is very limited, more data can potentially improve the classification results. The cognitive systems lab already implemented a system that takes the original game logs and simulates user behaviour in order to create new logs. As a result it is possible to create any number of games out of a single original one, from which some may be better than others. However, in order to simulate glare effect games multiple addtional steps and changes are neccesary.

- gereal approach, simulation, matrix creation, often used 1d cnn, novel approach of creating synthetic images and using 2d cnns shows in many cases improved classiiction resuts. Finally four voting based system one each for 5, 10, 15 and 20 rounds are implemented, combining the prediction of 400 models each to decide the final predictions.

- vielleict begrüdung: referenz zu denen. weil es zietgt dass man unter verwnedung simulierter spiele eine deutliche verberseerung der ergebnisse auf den echten spelen sehen kann...

3. Memory Game

The game used for the data collection is a matching pairs game written in Kotlin for Android devices. The original version was developed by Rafael Miranda **TODO: ref** in the context of a bachelor thesis. Originally, the game consisted of cards picturing animals and was only used to model general poor eyesight. Later is was expanded by adding a coloured card set to also model colour blindness. In the following the current functionalities of the game will be explained, only covering those related to the coloured cards. One game consists of 14 cards and each round consists of two cards being turned face up. If the cards match, they will be removed from the field. Otherwise they are turned face down again. The player wins once all 7 pairs have been discovered.

The game differentiates between two groups of obstacles: memory obstacles and visual obstacles. Additionally there is a classic mode providing the possibility to play without any obstacles which uses the red-green card set, shown in fig **TODO: ref**. The memory obstacle mode consists of the same card set and an additional side task: Every time a card is flipped a random number between 0 ans 10 is called out and the player must add all these numbers. After the final pair was found the sum of all numbers is requested. The game contains modes for two types of visual obstacles: Colour blindness and the glare effect. Colour blindness, more precisely a red-green weakness, is simulated by replacing the red-green card set with a brown shifted card set. The glare effect describes a sczenatio in which a lightsource shines onto the display, reflects from it and makes it more difficult to differnentiate the colours of the cards. This effect can be seen in figure **TODO: ref**.

To help in case of interaction obstacles the game provides multiple ways of assistance. Although they are not relevent for this thesis as only data is used where probants had no assistance, they will be explained briefly. There are two types of assistance: memory and visual assistance. Memory assistance is provided by flipping the cards from the previous turn once two non matching cards are flipped. After a short period of time they are turned face down again. If the game is played with visual assistance, each pair gets assigned a letter that is called out once a card is flipped. This introduces a new way of orientation that does not rely on the sense of sight, but instead on the sense of hearing. There is a game mode for each combination of obstacle (no obstacle, memory obstacle, colour blindness, glare effect) and assistance (no assistance, memory assistance, visual assistance). There where also other assistances implemented such as increasing the size of cards if they are revelaed, but they were only used for tests. The glare effect no assistance and the no obstacle no assistance modes are the only ones relevant in this thesis. Figure **TODO: ref** and **TODO: ref** show screenshots in the two modes.

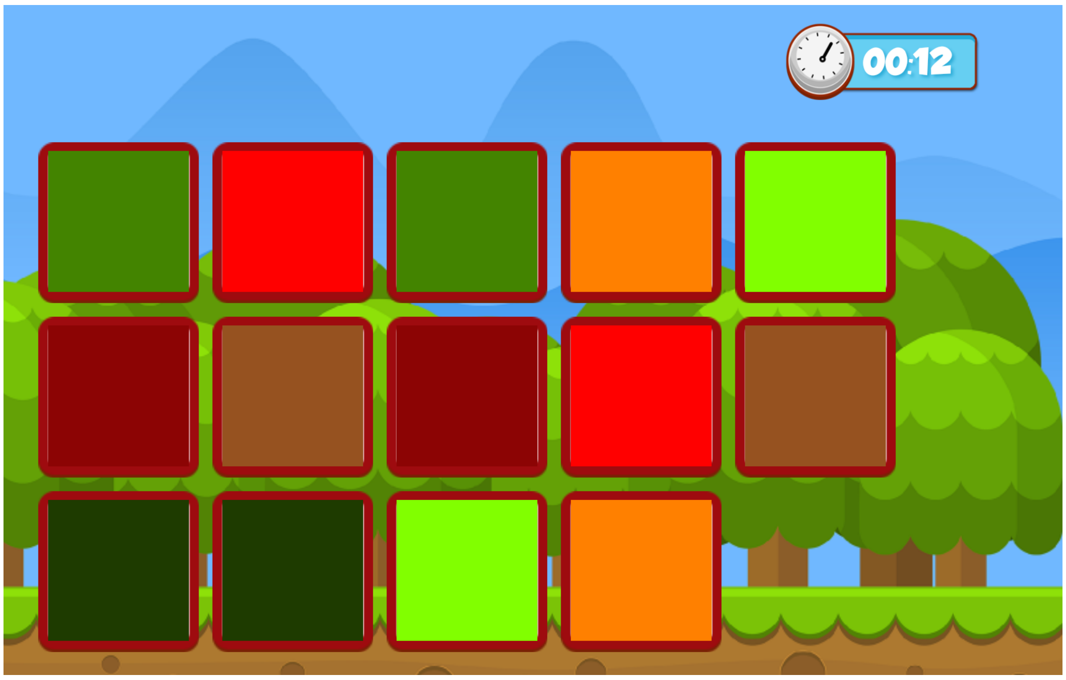

Figure 3.1: Screenshot of the game without obstacles. All cards are turned face up.

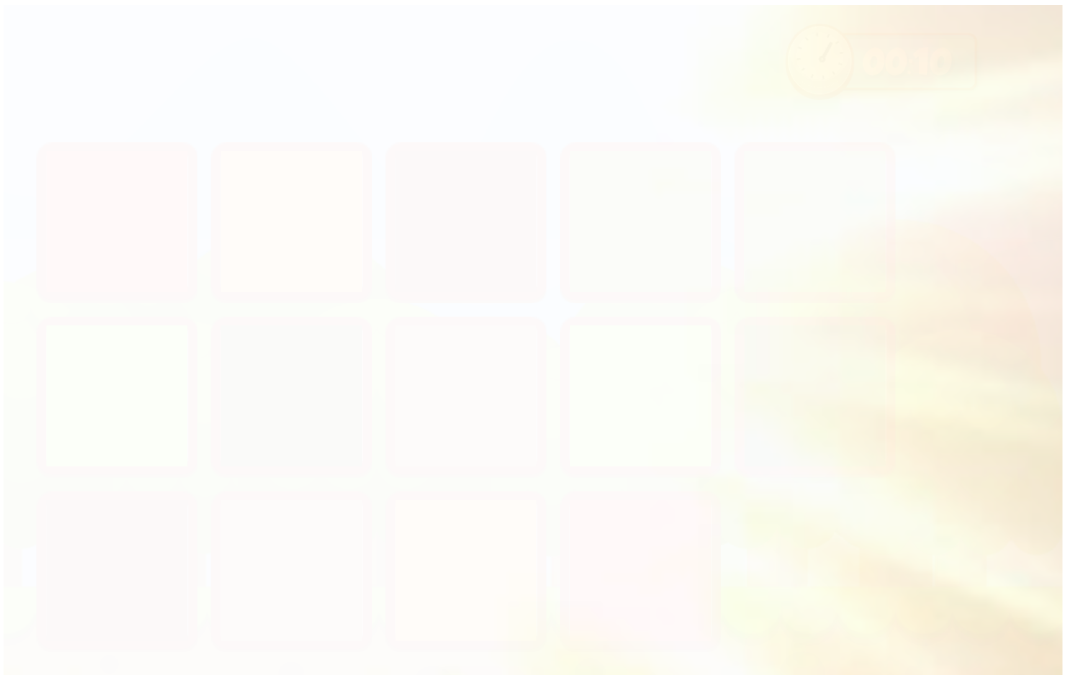

Figure 3.2: Screenshot of the game field with simulated sunlight. Sunbeams are especially visible on the right hand side. All cards are turned face up.

4. CMM-based Cognitive User Simulation

The cognitive system lab developed the computer program cognitive memory model (CMM). It utilizes concepts of the act-r-theory to model the cognitive human memory. ACT-R is a cognitive architecture that aims to describe and integrate central basic mechanisms of human cognition. **TODO: ref:** https://www.google.com/search?q=intimates+deutsch&oq=iminates+&aqs=chrome.1.69i57j0l7.2543j0j15&sourceid=chrome&ie=UTF-8 In the context of the matching pairs game the CMM was used to implement a generative model. By modelling memorized and forgotten revealed cards and deciding in each turn whether to explore unknown cards or to exploit the gathered knowledge to reveal matching pairs, the course of a matching pairs game can be simulated taking the capability of human memory into account. Furthermore it is possible to define a similarity matrix to simulate similarities between cards in matching pairs games. This matrix includes similarity values between 0 and 1 for each colour combination of cards. Such a similarity matrix can be used in the context of visual interaction obstacles to emulate human confusion. In related research it was used to simulate the confusion caused by colour blindness and the usage of a red-green card set. For this thesis the only two modes that are relevant are the glare effect (no assistance) and the no obstacle (no assistance) modes. In order to simualte no obstacle games no simialrity matrix is needed, but for glare effect games a new similarity matrix is created in section **TODO: ref**.

To accurately simulate the performance of human memory multiple parameters are randomly initialized and then repetedly optimized using a genetic optimization algorithm. These parameters can be further extended. The genetic algorithm works by repeadily simulating game sessions and updating the parameters by applying mutation and selection operations. To select the best parameter values two performance measurements are defined and used to compare the simulated game sessions and a real game as reference. These perfomarmce measurements are the number of matching pairs and the number of penalties per round. Penalties are given if a card is revealed whose partner has already been seen before but the pair was not picked up. The optimization of the parameter population is repeated over many generations so that the performance in the simalted games best fits that in the real game. As example one of the paramameters optimimzed by the genetic optimization algorithm is the similarity decay. It reduces the similarity effects during the course of the game, since the user may learn to adapt to the visual interaction obstacle and therefore be less effected by it. Furthermore the similarity effect is continuelasly reduced as there are less cards in the game after pairs were discovered.

As input the simulator takes a number of game logs and simulated a specified amount of new sessions using the genetic optimization algorithm. For instance 20 game logs can be passed to simulate 1000 new sessions out of each game log, resulting in 20000 simulated games. The data contained in the game logs is shown in section **TODO: ref**. In the context of simulating no obstacle games changes to the simulator were made that are explained in section **TODO: ref**. In section **TODO: ref** can be seen whether these changes resulted in improvements of the actual classification result.

5. Collected Data

While the probands played the memory game, behavioral data and brain activity data was collected. Relevant for this work is only the behavioural data recorded in glare effect and no obstacle games. The data was collected from 22 probands. **TODO: grobe demographic herausfinden, mazen fragen**. Each participant except one recorded 2 no obstacle and 1 glare effect game. The one exception did not record any glare effect games. As a result there are 44 no obstacle and 21 glare effect games. The data was already provided and was not collected during this bachelor thesis.

5.1 Behavioural data

In order to record bavioural data, each of the 14 cards was initally given a fixed card code and is was recorded which pairs of cards were turned face up in each round. The card code consists of two numbers separated by a dot. The first number specifies the colour of the card (between 1 and 7 as there are 7 colours) and second number specifies if it is the first or second card with that colour. Once a game is started, all cards are shuffled. The initial assignment of colours to numbers between 1 and 7 is shown in table **TODO: ref**.

Table 5.1: add caption.

Colour	Assigned number
Dark green	1
Brown	2
Red	3
Light green	4
Green	5
Orange	6
Dark red	7

The bevioral data was saved in logs, consisting of the card codes of each round followed by the unix timestamps of when the cards were flipped. Data of 20 rounds and therefore at maximum 40 flipped cards was recorded. If the game was completed in less than 20 turns, the remaining card code entries were filled with 0.0 and the remaining timestamp entries were filled with 0. An example of the sequence of card codes recorded during a game can be seen below. The timestamps are irrelevant for this work and therefore not shown.

5.2,2.2,4.2,4.1,6.1,7.2,6.2,6.1,1.2,1.1,
7.1,2.2,2.1,7.1,5.2,5.1,3.1,2.1,7.1,3.1,
7.2,7.1,3.2,2.2,2.1,2.2,3.1,3.2,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0

With the assignments shown in table **TODO: ref**, the first card code in the sequence above means that the first card to be flipped was the second green card. In the same turn, the second card that was turned around was the second brown card. This mapping is static as colours have the same numbers across all recorded games. However, the simulator requires a dynamic mapping of the card codes. Additionally each of the dynamically mapped card codes has to receives a similarity value for the simulation mentioned in chapter **TODO: ref**. Each card codes describes the two cards that were flipped in a turn and the similarity value describes how similar the colours of the two cards are. These similarity assignments for each combination of colours needed to be added to the logs. These assignments had to be made according to the dynamic mapping of card codes. The similarity values are extracted from a similarity matrix.

By dynamic mapping of the card codes is meant, that the colours are not assigned to fixed numbers but that the numbers are assigned in the reveal order specific to each game. This becomes clear, by looking the card code sequence from above, but dynamically mapped.

1.1,2.1,3.1,3.2,4.1,5.1,4.2,4.1,6.1,6.2,
5.2,2.1,2.2,5.2,1.1,1.2,7.1,2.2,5.2,7.1,
5.1,5.2,7.2,2.1,2.2,2.1,7.1,7.2,0.0,0.0,
0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0

In this dynamic mapping of the card codes, each entry likewise consists of 2 numbers that are seperated by a dot. However, the numbers are differently interpreted. The first number specifies what number of colour it is to be revealed and the second number specifies if it was the first or the second card of that colour. In the example above the colour green is the first to be revealed and therefore assigned to the value one. As it is the first green card revealed the second component of the card code is 1 as well. As a result the first entry is 1.1. Then a brown card is turned face up, meaning the second entry is 2.1 since is was the first brown card. Once the other red card is discovered, the entry for that step will be 2.2. This pattern continues throughout the whole sequence. This means that the mapping is specific to each game and depends on the reveal order of the cards.

Logs with remapped card codes and added similarity values were already provided, but there was a change to be made regarding the glare effect logs before they could be used. The similarity matrix for the glare effect game was just a placeholder and did not correclty portray the color differneces under the influence of sunlight. To successfully simulate glare effect games the simulator requires such an similarity matrix. Therefore a new one is created in section ref. The old similarity values in the glare effect logs are then replaced with the new values in section ref. Contrary to the glare effect logs, in the no obstacle logs, no changes need to be made, as the simulator does not use any similarity values when simulating no obstacle games. It should also be noted that the provided logs with remapped card codes additionally include four statistical features for the last turn. These consist of the number of remaining cards, the number of never revealed cards, the highest number of times the same card was revealed and the number of rounds since all pairs were found. These four values are ignored, as they are newly calculated for every turn in section ref. Last but not least all logs end with a label, describing in which game mode the log was created.

5.2 Brain activity

Eeg data collected from all proabnds during the game. However, there is one problem with using the eeg data: The overlyiong context of this work is to find interaction obstacles and ultimatley imporve the user interaction. This means that the way of discovering such an interaction obstacle should not worsen the interaction experience. If all people had to waer eeg masks when interacting with software just for the purpose of noticing interaction obstacles, the interaction experience itself would suffer. A method of recording brain activity without an deterioration of the interaction experience has not been discovered yet. Additionally it is not cost efficient for every user to use and eeg sensor. As a result the decision was made not to use eeg data and instead only use the behvioural data that is collected direclty through the interaction and stays unnoticed by the user. As eeg data is not used in this work it will not be further explained.

6. Data Preparation

To prepare the data for the training many steps were neccesary. One big step is to increse the available dta by simulaing user behaviour. Therefore the simulator tor in chapter **TODO: ref** is used. For simulating games with no opbstacle there is no need for a similarity matrix. However, in order to sucessfully simulate glare effect games, a special similarity matrix is required. While without any obstacles the color differnes are independent from the position of the crads, this is not the case in glare effect games since the intensity of the light is not the same across the whole field. As a result the first step is to create a similarity matrix that descirbes the differneces of colours under the influence of the glare effect. Once completed, it replaces the old similarity matrix in the glare effect logs. As not all logs are valid and can be used in the simulator, the invalid log must be removed. These valid logs are then used to simulate user behaviour and thereby create new logs. These simulated logs are sorted by their quality and the simualtion is evaluateed. If the performance in the simulated games is similar to that in the original games it is preceeded with the gerneration of the features for the training. Otherwise changes to the simulator ar made and the simulation, the sorting and the evaluation are repeated. In case of very bad simulations of glare effcet logs it would have also been possible to change the approch of creating the similarity matrix or find a new approch. However, the simualtinos of glare effect were very good, which can be seen in **TODO: ref**, which is why the spproach was kept.

6.1 Similarity matrix creation

6.1.1 Conceptualization

The aim is to create a similarity matrix that descibes the differneces between colours under the influence of the simulated sun light. The difficulty hereby is that the intensity of the light is not spread evenly across the whole field. Instead, as it would be if a real sun would shine onto the display, a certain are has the highest intensity and the intensity is lower the further away from that area. Adittionally there are wide lines of higher intensity comming from the brightest area, that simulate sunbeams. This influence of the simulated sunlight can be seen in figute 3.2.

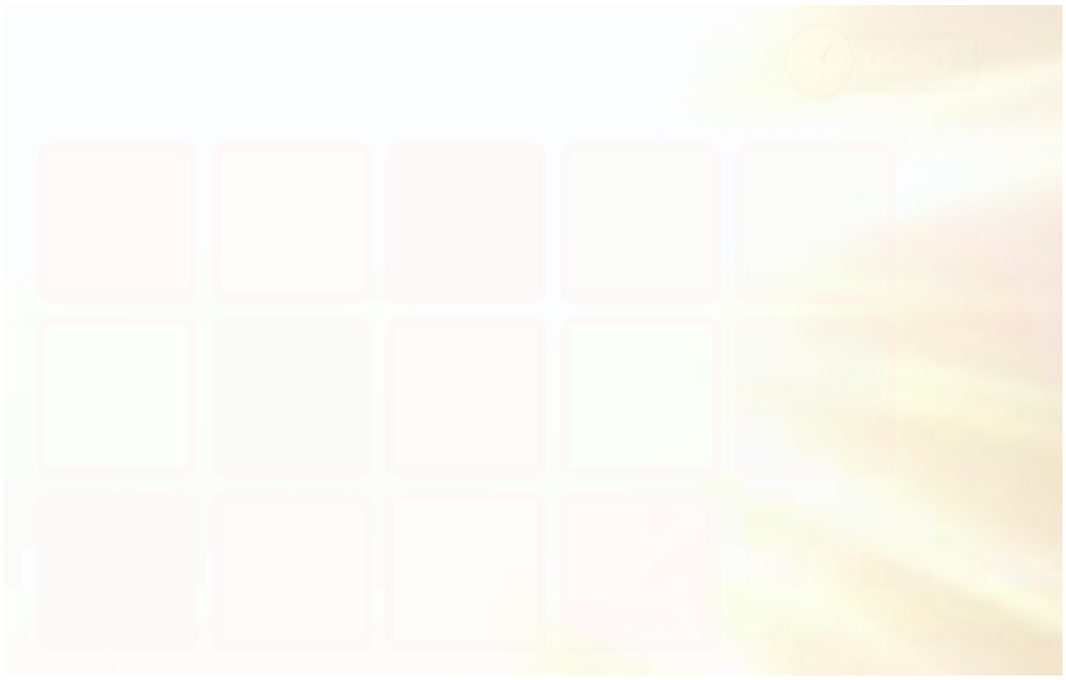

Figure 6.1: Screenshot of the game field with simulated sunlight. Sunbeams are especially visible on the right hand side. All cards are turned face up.

TODO: vielleicht bild raus nehmen weil es oben schon einmal ist und referenzien nach oben

As a result simply extracting the rgb values for one pixel of each card in one game and calculating the differneces has two major problems: Firstly, the differences are highly influenced by the position of the specific cards. If the matrix is calculated using a single glare effect game the differneces are only representative for exactly that game and may be completely differnet if the cards are positioned differently. Secondly, extracting single pixels for each card in an image could lead to color values that do not represent the overall observed colour due to varying brightnesses on differnet points of a card. This behaviour is undesired, as the final similarity matrix is supposed to represent the differneces of the observed colours of the cards. **TODO: vielleicht quelle suchen die sagt das menschen farbenm ion bereichen mitteln oder so** In order to solve these problems a more complex approach then simply extracting single pixels out of a game was needed.

The problem of the positional influence can be solved by creating similarity matrices for more than one game and calculating the mean color differences of those matrices. The idea is that by creating using so many games that all or most combinations of positions and colours are included and calculating the mean of all comparissons for , the result will not be influenced by the position of the cards. However, first needs to be determined which number of games satisfies this condition. Each game consists of 14 cards, with each two cards having the same colours. When determining the difference between two colours there are two differnet cases:

- 1. The colours are not the same: In each game there are exactly 4 combinations of positions for those two colours, because there are two cards for each colour.

- 2. The colours are the same: In each game there is only one combination of positions for those colours.

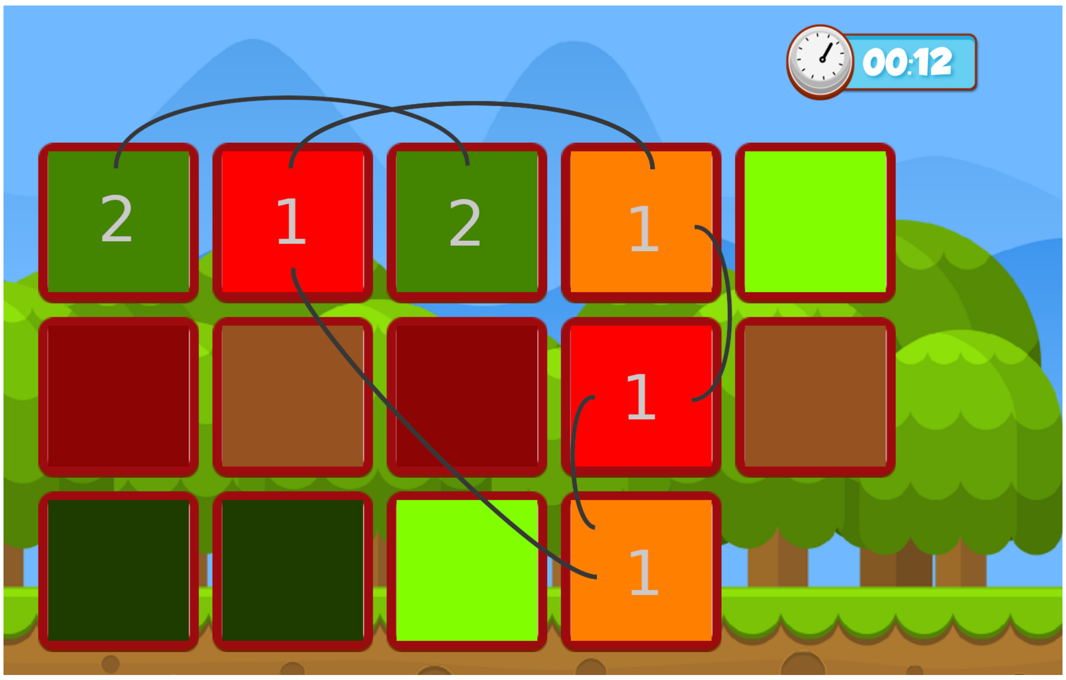

Figure 6.2: Examplary showing the number of different comparissons for case 1 and 2. The numbers on the cards represent the case and the edges indicate unique comparissons. All cards are turned face up. A game without any obstacles is used only for the purpose of clarifying the differnet comparissons. All screenshots used for the actual calculation are taken from glare effect games.

First it is calculated how many screenshots are necessary for the first case. For two different coloured arbitrary but fixed cards there can be

$$C_1 = 14 \cdot 13 = 182$$

combinations of positions are possible. The reason that it is not not $14 \cdot 14$ is that 14 comparissons of cards with themself would be included. We formulate the condition that enough screenshots need to be taken so that a arbitrary but fixed combination of positions for two different colours is included with a probability of 95%.

As there are 4 such combinations of positions in each game the probanility P(A) to get a specific combination in a game is

$$P(A) = \frac{4}{182}$$

The counter probability P(¬A) is

$$P(\neg A) = 1 - \frac{4}{182} = \frac{178}{182}$$

The number of necceary screenshots to include a specific combination with 95% probaility is calculated by solving

$$\begin{aligned}
1 - P(\neg A)^n &\geq 0.95 \\
1 - \left(\frac{178}{182}\right)^n &\geq 0.95 \\
1 &\geq 0.95 + \left(\frac{178}{182}\right)^n \\
\left(\frac{178}{182}\right)^n &\leq 0.05 \\
n &\geq log_{(\frac{178}{182})}(0.05) \\
n &\geq 134.8024
\end{aligned}$$

This means that about 135 Screenshots of games are needed in the first case. Now we perform the same calculation for the second case where there is only one combination of positions for the colours. However, there are less combinations of positions that are relevant than in the first case. For example for two green cards at index 0 and 1 the 182 comparissons would include a comparison of the first green card and the second green card as well as a comparisson of the second green card and the first green card. This goes for every two cards with the same colour, meaning there are only half as many different comparissons in the second case than in the first case. As a result the number of possible combinations in the second case is

$$C_2 = \frac{C_1}{2} = \frac{14 \cdot 13}{2} = 91.$$

Now we calculate how many screenshots must be taken to include a specific combination of positions for two equal colours with a probability of 95%.

As there is one such combinations of positions in each game the probanility P(A) to get a specific combination in a game is

$$P(A) = \frac{1}{91}$$

The counter probability P(¬A) is

$$P(\neg A) = 1 - \frac{1}{91} = \frac{90}{91}.$$

The number of necceary screenshots to include a specific combination with 95% probaility is calculated by solving

$$\begin{aligned}
1 - P(\neg A)^n &\geq 0.95 \\
1 - \left(\frac{90}{91}\right)^n &\geq 0.95 \\
1 &\geq 0.95 + \left(\frac{90}{91}\right)^n \\
\left(\frac{90}{91}\right)^n &\leq 0.05 \\
n &\geq log_{(\frac{90}{91})}(0.05) \\
n &\geq 271.111
\end{aligned}$$

This means that about 272 screenshots of games are needed in the second case. As there are more screenshots needed for the second case, the first case is also included. This inclusion is caused by the fact that each screenshot includes comparissons for the fist as well as the second case. To have a buffer the decision was made to take screenshots of 300 games. In theory these screenshot include any arbitrary but fixed combinations of positions and colours with a probability of over 95%, which should eliminate the problem of the positional influence.

Nonetheless, this does not solve the second problem of extracting single pixels that can potentially be directly in the area of a sunbeam. This can be fixed, by extracting all pixels in a certain are of the card and averaging their rgb values.

Last but not least it needs to be clarified how the colour differneces are calculated. To capture how colour differences are observed by humans, the rgb colour scale is not suitable. Using the rgb colour scale simirlarly stromng perseaved color differnces do not neccessarily have the same eucliidean distance. Contrary to that the CIELAB colour space aims to do exaclty that. Although not perfect, it more accurately descirbes human colour perception than the rgb colour space. Therefore the colour differences described in the similarity matrix are calculated after converting the colours into the CIELAB colour space. The colour distance is called Delta E score. The lower this score is the more similar appear the colours to human eyes. In related work, when creating the similarity matrix for the effect of colour blindness, the CIE1976 colour model was used for the calculations. However, the formula for calculating the colour difference has since then been improved multiple times, resulting in the CIE2000 colour model. Through multiple modifications of the formula it got closer to a visual equidistance, meaning simirlarly stromng perseaved color differnces have more similar Delta E scores than before. Therefore when creating the similarity matrix for the glare effect the CIE2000 colour model is used instead of the old one.

To sum it up, the chosen approach is to take 300 screenshots of glare effect games with all cards turned face up, extracting areas of rgb values for each card, converting the average rgb values of ares into CIELAB colours, calculating a similarity matrix for each game that contains the delta E scores and finally averaging the delta E scores across all similarity matrices. To achieve values between 0 and 1, the colour differneces additionally need to be scaled down in the end.

6.1.2 Screenshot extraction

To play the memory game and take screenshots an emulator for the pixel 2 in android studio was used. In order to take the screenshots in the needed way and collect additionally necessary information some changes to the memory game were made. In the way the game is intended there are always only maximum two cards turned face up. However, for the screenshots all cards need to be turned around so that the colour differneces can be calculated. Therefore the memory game was adjusted so that all cards are turned face up once the player turns a card. Additionally it was changed so that cards stay face up once they get turned around for the first time. This makes it possible to take screenshots with the colours of all cards visible. Furthermore, for the creation of the similarity matrix it needs to be known what the original colours without the influnece of the simulated sun are. To collect the screenshots and the according colours of the cards in each game a semi-automatic approach was chosen. Once a game is started and a card is flipped, resulting in all cards being turned face up, the colours of all cards are saved in a list. Then a screenshot is manually taken and afterwards the game is exited to enter the main menu. From there on the process is repeated 300 times. As a result there are 300 hundred screenshots of games and a two dimensional list that includes the colours of the cards in the 300 games. The first dimension specifies the game and the second dimension the index of the card. The values of this list are stored in a text file before the game is completely closed. To assure that no mistakes were made when taking the screenshots, the number of collected screenshots is observed during the whole process and afterwards all screenshots are manually checked so that all cards are turned face up.

6.1.3 Implementation

This section will show and explain the implementation of the similarity matrix generation. Code is only included if it helps to further understand what is being explained. First of all the screenshots and the information about the original colours of the cards are loaded. Before the actual calculation of a similarity matrix for each image, the average rgb values of the cards on the field need to be determined.

```python
def determine_glare_rgb_values(image):
'''
Calculates the rgb average rbg values for each card in an image.
:param image: The screenshot of the memory game with all cards
turned face up.
:return: The average rbg values in the specified 150 x 150 pixel
areas for each card.
'''
glare_rgb_values = []
for corner in card_corners:
  x_border = corner[0] + 150
  y_border = corner[1] + 150
  card_values = []
  for x in range(corner[0], x_border, 1):
    for y in range(corner[1], y_border, 1):
      coordinates = x, y
      pixel_values = image.getpixel(coordinates)
      card_values.append(pixel_values[:-1])
  card_r = int(round(np.mean([color[0] for color in card_values])))
  card_g = int(round(np.mean([color[1] for color in card_values])))
  card_b = int(round(np.mean([color[2] for color in card_values])))
  glare_rgb_values.append((card_r, card_g, card_b))
return glare_rgb_values
```

Listing 6.1: Add caption

The cards are each $250 \cdot 250$ pixels big and the rgb values are extracted from squared $150 \cdot 150$ pixel areas. The coordinates the pixels are extracted from are manually selected in gimp. As a result the areas are only correct for the used resolution of 1080p. Once the mean rgb values in those areas are calculated and converted into LAB colours the similarity matrix can be created. LAB colours are neccessary to calculate the colour differnece using the CIE2000 colour model. Each of the 28 cells in the similarity matrix falls into one of the two cases explained in **TODO: ref**, meaning there are either 4 or only 1 unique combination for the calculation of each cell value. For every combination the lab colour distance is determined and the values for each cell are averaged. This completes the steps for a single image, resulting in a similarity matrix with unscaled values. Once a similarity matrix with unscaled values for every image is created, the average of all matrices is calculated. During the whole calculation of the single matrices it is being kept track of the highest occurring colour differnece. The last step neccessary to complete the final similarity matrix is to divide all values in the matrix by use the highest colour differnce to scale them down to be between 0 and 1.

6.1.4 Testing

To verify the correcteness of the color extraction and the calculation of the similarity matrix, the main funtionalities are manually tested. The colour extraction was tested by extracting colours of a screenshot with the skript and compoaring the rgb values to the actual values in the images using gimp. Furthermore the calculation of the delta e score was tested, by comaring results of the script with those of an online calculator.

To assure that the 300 games actually include most of the combinations and therefore eliminate the positional influence of cards, the coverage of combinations can be additionally calculated. Therefore the number of unique combinations included in the 300 screenshots must be divided by the overall number of possible combinations. The number of unique combinations can be counted during the process of creating the similarity matrix, but the number of possible combinations must seperatly be determined. The similarity matrix for glare effect has 28 entries, from which 7 are comparissons between same and 21 between different colours. With C_1 and C_2 being the possible combinations for two arbitrayry but fixed coloured cards in the two cases mentioned above, the number of overall possible combinations C is

$$\begin{aligned} C &= 21 \cdot C_1 + 7 \cdot C_2 \\ &= 21 \cdot 182 + 7 \cdot 91 \\ &= 4459. \end{aligned}$$

The 300 games used for creating this matrix include 99.57% of all possible combinations. The final test of the created similarity matrix, however, will be to actually use it to simulation glare effect games and see whether they are simulated accurately. This is done in section **TODO: ref**.

6.1.5 Final Matrix

In figure 6.3 the final similarity matrix can be seen, dsiplaying how colour differneces are observed under the influence of the simulated sunlight, averaged over 300 games. As mentioned before, the lower the delta e score the more similar the cards appear to the human eye. The maximum delta E score that occured during the calculation was 8.861. All values are divided by that number to achieve scores between 0 and 1.

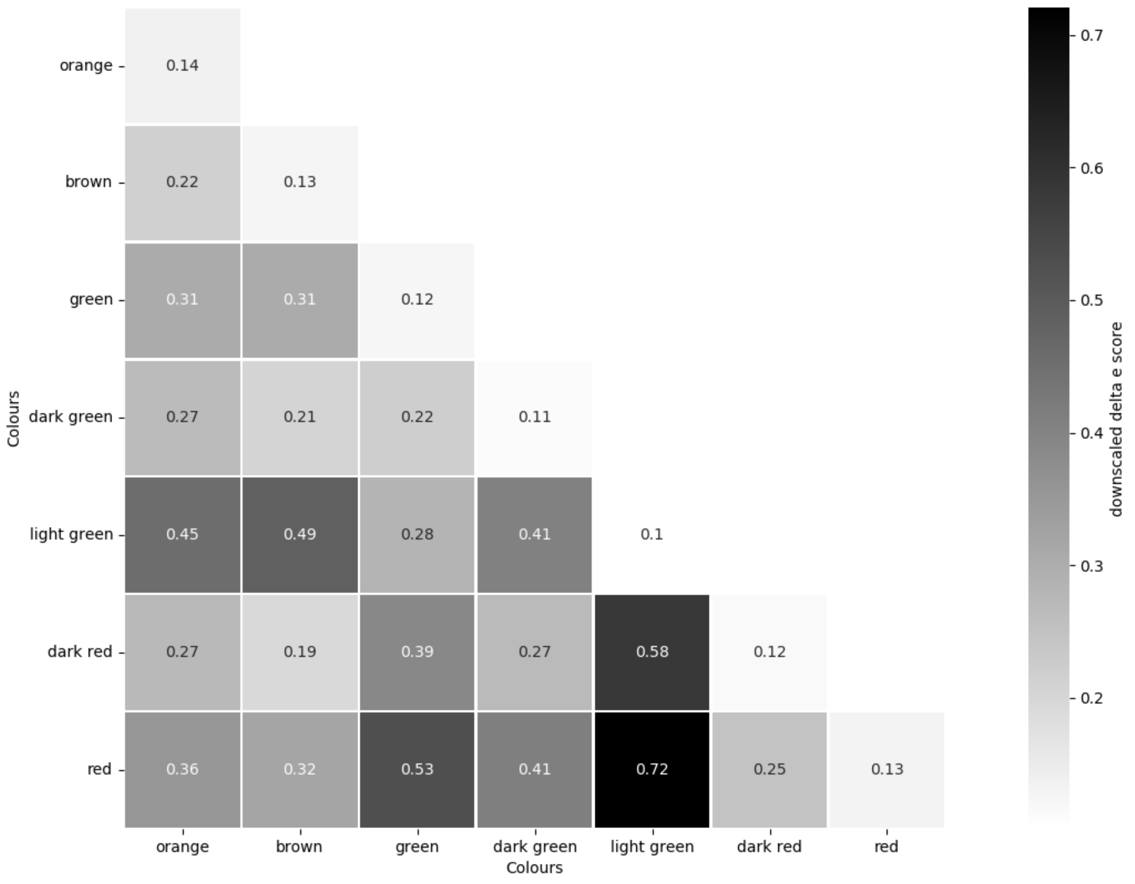

Figure 6.3: Add caption. The lighter the colour the less a difference is noticeable

The delta E score is less a definitive answer, and instead a helpflu metric to apply to a specific use case. Although there are tendencies regarding the interpretation of colour differences, there is no definitive table that descriebes what the different scores mean. One reason for this is that the perceived colour differnece may vary in different situations and circumstances. For instance, is the colour difference perceived differently depending on how long the colours are exposed to the human eye, as humans can over time adapt to the colours. This means that in a setup where all colours are always visible the colour difference will likely be perceived weaker than in the case of the memory game, as the cards are only flipped momentarily, leaving little time for adaptation. Table ref was created through own observation and in that sense is highly influenced by the strength of the sense of sight from the creator of this work. Therefore it should not be taken definitive but only serve the purpose of claryfying how the values in the matrix can roughly be interpreted.

Table 6.1: add caption.

Delta E	Downscaled values	Perception of difference
≤ 1.33	≤ 0.15	Not perceptible by human eyes
$1.33 - 2.66$	$0.15 - 0.3$	Only perceptible through close observation
$2.66 - 4.43$	$0.3 - 0.5$	Perceptible colour difference
≥ 4.43	≥ 0.5	Major colour difference

Before actually using the constructed simialry matrix in the simulator it is unknown if the chosen approach results in high quality simulations. If this is not the case it is possible to change the concept or try other approches.

It should be also noted that the similarity matrix has 28 entries. The one that was used for describing the confusion that happens during colour blindness only has 21 entries, because all 7 values describing cards of the same colour can be left out as the colours are equal. Since in glare effect games, originally equally coloured cards can look due to the influence of the simulated sunlight, these 7 values must be included in the similarity matrix for the glare effect.

6.2 Incorporation of the new similarity matrix

The former similarity matrix in the glare effect logs that was just a placeholder can now be replaced with the newly created one. To do so there was already a script available, that was used after some small adjustments. The main changes were to replace the deprecated functions from libraries with supported ones and incorporate the new similarity matrix. As the code was initially written for similarityg matrices with 21 entries and the one for glare effcet has 28 entries, the code for that created the dynamic similarity assignments the logs had to be adapted. Finally all remapped logs are saved in new files. To check whether the remapping is still done correctly, the resulting logs were compared to logs that were remapped by the original script. The only thing that was not original were the replaced deprecated functions as this was neccesary to execute the script. Besides differnet similarity values due to differnet matrices as well as the fact that the logs now also contained entries for same coloured cards, the mapping was identical.

6.3 Removal of invalid logs

For the simualation to work, the game log that is used for the simulation must have had all cards turned around at least once. If this is not the case the log is classified as invalid. Additionally to removing invalid logs, the aim is to create a balanced data set for training. Therefore the number of games in each game mode has to be equal. Using twice as many no obstacle games for training than glare effect games can lead to games being more often classified to have no obstacle only because they appeared more often during training. As a result the model would be less capable of detecting actual visual interaction obstacles. Furthermore the decision was made that the number of games from each participant in each game mode should be equal, too. This means that the data should not contain more no obstacle games from a particiapnt than glare effect games, and vice versa.

As stated in chapter **TODO: ref** from the 22 participants there are 44 no obstacle and 21 glare effect game logs. To collect the data used for the simulation, a script was written that collects one no obstacle and one glare effect log from each participant. Half of the available no obstacle logs are not collected, because there are not as many glare effect logs. Once the logs are collected, they are validated and the valid ones are saved in new files. If at least one the two logs from a participant from different game modes is invalid, both are removed. Otherwise the training data would be inconsistent in that sense that the logs from different game modes could be from different participants. From the 22 no obstacle and 21 glare effect logs collected by the script, one glare effect log was invalid. This resulted in 20 no obstacle and 20 glare effect logs being used for simulation. These 40 real logs combined with those simulated form the data used for training.

```python
def validate_log(log):
  '''
  Validates logs.
  :param log: The log to validate.
  :return: If the log is valid.
  '''
  needed_entries = ['1.1,', '2.1,', '3.1,', '4.1,', '5.1,', '6.1,',
      '7.1,', '1.2,', '2.2,', '3.2,', '4.2,', '5.2,', '6.2,', '7.2,
     ']
  for needed_entry in needed_entries:
    if needed_entry not in log:
      return False
  return True
```

Listing 6.2: Add caption

6.4 Simulation of user behaviour

The whole process of simulating user behaviour is divided into two parts: First configuartion files are created, using a generic optimization algorithm, that contain the optimized parameter values and secondly these configuration files are used to simulate games. How the simulator functions is explained in chapter **TODO: ref**.

Multiple additions were made to the simulator. In total four classes were added. Two are for generating configuration files for the simulation of game with and without the glare effect obstacle and the other two for using those files to simulate new games based on the user behaviour in the original games. These classes utilize the functionalities already implemented in the simulator. However, as the simulator only worked with similarity matrices that have 21 entries, instead of the 28 of the matrix for glare effect games, the simulator was adjusted so that it can also handle matrices with 28 entries. After all changes and additions were completed, each log was used to simulate 1000 games, resulting in 40000 logs. From these logs 20000 are no obstacle games and the other 20000 are glare effect games. The 40000 logs contain 1000 no obstcle and 1000 glare effect logs for each of the 20 probants.

6.5 Sorting logs by quality

The siulated logs ware sorted by how close their performance is to that in the real game used for their simulation. The value by which is sorted, is the average of two values. The first one is the root mean squared error (rmse) between the matching pairs in the real and the simulated game for each round. The second one is the rmse between the penalties in the real and the simulated game for each round. These two performance measurements are explained in section **TODO: ref**. The formula for calculating the rmse is shown below.

$$rmse = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2}$$

Averaging the two rmse's for each simulated game and sorting the logs accordingly, makes it possible to only use acertain number of the best simulated logs during the training instead of using all of them or a random subset.

6.6 Evaluation of simulation

To evaluate whether the performance in the simlated logs is similar to the one in the original logs two perofmance measurements are used: The matching pairs in each round and the penalties in each round. Penalties are given if a card is revealed whose partner has already been seen before but the pair was not picked up. Some initial observations inspired changes to the simulator. The quality of the simulations regarding the performnce measurements is analysed before and after the changes to the simulator, by comparing the performance between simulations and real games. Furthermore a subset of the simulated data was discovered that mostly shows no significant differnece between the perfromnce of the simulated and the real games.

6.6.1 Results before and after changing the simulator

The Initial simulation results can be seen in figure 6.4 and 6.5. The horizontal line extending from the curves, also calles whiskers, describe the standard deviation of the corresponding value.

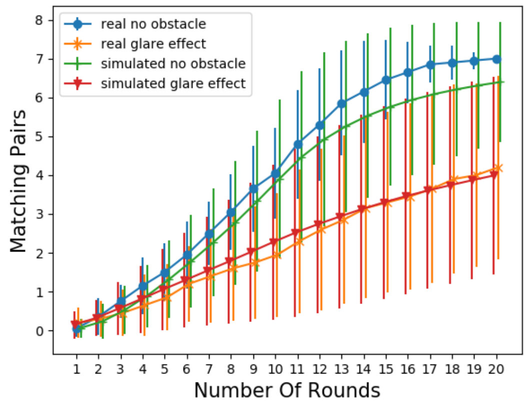

Figure 6.4: Add caption

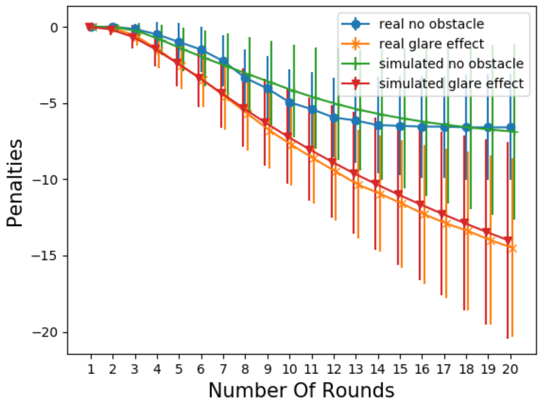

Figure 6.5: Add caption

At first glance it can be seen that the simualtions of glare effect games are very similar to the real games, meaning that the constructed similarity matrix creates good simulation results. As a result there is no need to find a different approach for creating the similarity matrix, and no changes to the simulator regarding the simulations of glare effect games are made. However, the simulation of no obstacle games is not quite as accurate. Especially the number of matching pairs after the tenth round in the simulated games is lower than in the real games. Additionally the standard deviation indicated by the whiskers regarding the no obstacle games is noticeably higher in simulated than in the real games. This is not very string in the first turns but gets worse in later turns. Furthermore, however less noticeable, is are the penalties in the simulated games between round 8 and 18 lower than in the real games.

The fact that matching pairs per round for the no obstacle simulations are noticeably lower than in the real games inspired some changes to the simulator. As there are less cards on the field in later rounds and the experience of the user may have improved over the course of the game, in reality the degree of randomness is likely decreased in later turns. To simulate this a new parameter was introduced that is optimized by the generic optimization algorithm. It is called RANDOMIZING DECAY and is a value between 0 and 1. After the 10th round this value is used to reduce the amount of randomness in the choice of the next card and by that increases the performance in those turns. Another value that also influences the random behaviour in the simulalations was optimized so that there is overall less randomness. The results of these changes can be seen in figure 6.6 and 6.7.

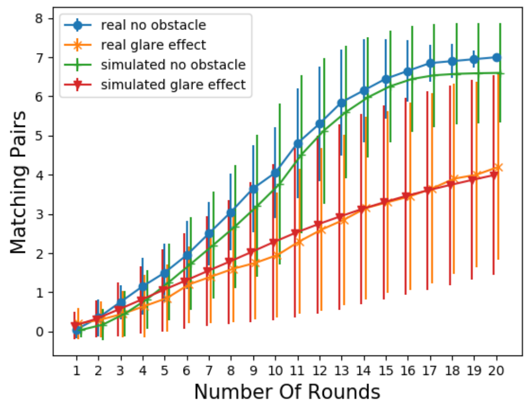

Figure 6.6: Add caption

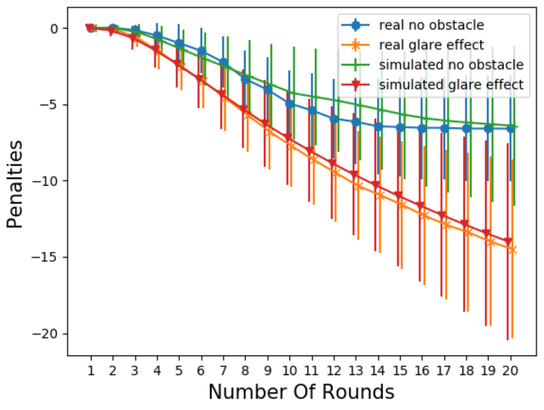

Figure 6.7: Add caption

As the changes only affected the simulation of no obstacle games, the following statements only refer to those games. It is noticeable that the number of matching pairs per round in the simulated games is closer to the real games. Additionally the standard deviation of the simulated games regarding the matching pairs per round gets noticeably smaller after the 10th round and by that is closer to that in the real games. However, the reduction of randomness also resulted in slightly less penalties after round 10 and by that the differnece in penalties between the simulations and the real games becomes bigger. On the contracry, the standard deviation of the penalties in the last 6 rounds of the simulated no obstacle games got slightliy closer to that in the real games. Reasons for the negative impact on the penalties are unknown. It seems that the improvements regarding the matching pairs overweight the deteriorations of the penalties, but if these changes actually improve the overall quality of the simulations can not be said without further analysis.

However, another interesting observation can be made. The standard deviations of the two perfromance measurements in the last rounds of the simulated no obstacle games is notably higher than in the real games. This might be caused by the fact that there are in generel less or even no cards on the field if all pairs were discovered. This theory also fits to the fact that this is not an issue in the glare effect games, as there are more cards on the field in later turns. This concludes, that there is a possibility that the simulator struggles to accurately simulate rounds once there only a few or no cards remain on the field.

Although major tendencies of similarities and differences are visually noticable in figure **TODO: ref** and figure **TODO: ref**, a more accurate analysis is needed to to make reliable statements. Therefore multiple paired samples t-test are performed, comparing real no obstacle, simulated no obstacel, real glare effect and simualted glare effect games with each other. For each combination two tests are performed. One comparing the mean penalties per round and the other one comparing the mean numbers of matching pairs per round. The resulting p values can bee seen in table **TODO: ref** and table **TODO: ref**. These values express whether the compared lists of mean values are significantly different or not. Values higher than 0.05 indicate no significant difference, while values below 0.05 indicate a significant difference. It would be optimal if all comparissons between different game mode show significant difference, while comparissons between real and simulted games of the same type show no significant difference. All paired sample t-tests were performed before and after the changes to the simulator. This was done to see whether the changes to the simulator improved or deteriorated the results of the test or if they had any impact on them at all. First the results of the paired sample t-tests, when including all 20 rounds and all simulated games, are analysed. Tables **TODO: ref** and **TODO: ref** show the resuts before changing the simulator, while table **TODO: ref** and **TODO: ref** show the results afterwards.

Table 6.2: Before change to simulator. p and t values in paired t-test for different comparissons of matching pairs per round. All 1000 simulated games per real game were used. The following abreviatrions are used: real glare effect - r_g, real no obstacle - r_n, simulated glare effect - s_g, simulated no obstacle - s_n

	s_n		s_g		r_n	
	p	t	p	t	p	t
r_g	$1.7e-06$	-6.8179	0.0688	-1.9292	$9.3e-07$	7.1041
r_n	$5.7e-07$	7.3571	$2.7e-06$	6.5765		
s_g	$5.3e-06$	6.2480				

Table 6.3: Before change to simulator. p ant t values in paired t-test for different comparissons of penalties per round. All 1000 simulated games per real game were used. The following abreviatrions are used: real glare effect - r_g, real no obstacle - r_n, simulated glare effect - s_g, simulated no obstacle - s_n

	s_n		s_g		r_n	
	p	t	p	t	p	t
r_g	$3.6e-06$	-6.4297	$1.1e-06$	-7.0423	$3.2e-06$	6.4897
r_n	0.2011	-1.3242	$5.6e-06$	6.2238		
s_g	$5.4e-06$	6.2394				

Table 6.4: p and t values in paired t-test for different comparissons of matching pairs per round. All 1000 simulated games per real game were used. The following abreviatrions are used: real glare effect - r_g, real no obstacle - r_n, simulated glare effect - s_g, simulated no obstacle - s_n

	s_n		s_g		r_n	
	p	t	p	t	p	t
r_g	$5.7e-06$	-6.2181	0.0688	-1.9292	$9.3e-07$	7.1041
r_n	$1.8e-10$	12.2469	$2.7e-06$	6.5765		
s_g	$1.7e-05$	5.7107				

Table 6.5: p ant t values in paired t-test for different comparissons of penalties per round. All 1000 simulated games per real game were used. The following abreviatrions are used: real glare effect - r_g, real no obstacle - r_n, simulated glare effect - s_g, simulated no obstacle - s_n

	s_n		s_g		r_n	
	p	t	p	t	p	t
r_g	$4.8e-06$	-6.2967	$1.1e-06$	-7.0423	$3.2e-06$	6.4897
r_n	0.0195	-2.5519	$5.6e-06$	6.2238		
s_g	$7.2e-06$	6.1045				

It can be seen that all tests comparing different game mods show significant difference in penalties per round as well as matching pairs per round, which is desired. However significant difference is not dersired in the four test comparing real and simulated games of the same mode, which are highlighted by colours. Before the changes to the simulator, the matching pairs per round in simulated and real no obstacle games show significant differneces. The same goes for the penalties per round in simulated and real glare effect games. As the changes to the simulator only affected the simulation of no obstacle games the significant difference between the matchiung pairs in simulated and real glare effect games prevales. Despite the fact that the matching pairs per round in simulated no obstacle games seem closer to the real games in figure **TODO: ref**, the statistical tests still show a significant differnece between them. Additionally, the small deterioration of the penalties that were caused by the changes to the simulator and can be observed in figure **TODO: ref**, led to a significant difference in the penalties per round between simulated and real no obstacle games. Before the changes to the simulator this difference was not significant.

In total, before the changes to the simulator 2 out of 4 tests comparing games of the same mode, result in undesired significant differences. After the changes to the simulator this increased to 3 out of 4 tests. However, this does not lead to the conclusion that the changes to the simulator decreased the quality of the simulations, as the matching pairs per round shown in figure **TODO: ref** in simulated no obstacle game got noticeably more accurate.

6.6.2 Search for a subset of the data

In the statictical tests in section **TODO: ref**, all all 1000 simulations per participant are used. As some simulations are better than others, using a subset only containing a number of best simulations could lead to further findings. Since the simulated games are sorted by their quality in section **TODO: ref**, it is possible to only use a certain number of best simulations for the comparissons, and see whether this improves the undesired properties mentioned above. Furthermore the number of rounds in the comparisson can be changed, so that not all 20 rounds are included. The concideration of including less rounds is of importance, beacuse it can reveal which parts of the game are simulated accurately and which are not. The search for an configuration of those two settings was manually done by varying the number of steps and the ratio between simulated and real data. A configuration of those two settings that produces interesting results consisted of using only the best 10 simulated games per game mode from each participant and only including the first 10 rounds of each game. The results before the changes to the simulator can be seen in table **TODO: ref** and **TODO: ref**, while the results afterwards are shown in table **TODO: ref** and **TODO: ref**. The two performance measurements after the changes to the simulator when only using the best 10 simulated games per game mode from each participant are aditionally portrayed in figure **TODO: ref** and **TODO: ref**.

Table 6.6: Before change to simulator. p and t values in paired t-test for different comparissons of matching pairs per round. For the simualted data only the best 10 simulated logs from each real log are used. Only the first 10 rounds are used. The following abreviatrions are used: real glare effect - r_g, real no obstacle - r_n, simulated glare effect - s_g, simulated no obstacle - s_n

	s_n		s_g		r_n	
	p	t	p	t	p	t
r_g	0.0080	-3.3879	$6.3e-05$	7.0000	0.0058	3.5985
r_n	0.0047	3.7292	0.0038	3.8679		
s_g	0.0052	3.6587				

Table 6.7: Before change to simulator. p values in paired t-test for different comparissons of penalties per round. For the simualted data only the best 10 simulated logs from each real log are used. Only the first 10 rounds are used. The following abreviatrions are used: real glare effect - r_g, real no obstacle - r_n, simulated glare effect - s_g, simulated no obstacle - s_n

	s_n		s_g		r_n	
	p	t	p	t	p	t
r_g	0.0024	-4.1873	0.0491	2.2730	0.0014	4.5577
r_n	0.9426	0.0740	0.0012	4.6629		
s_g	0.0020	4.2932				

Table 6.8: p and t values in paired t-test for different comparissons of matching pairs per round. For the simualted data only the best 10 simulated logs from each real log are used. Only the first 10 rounds are used. The following abreviatrions are used: real glare effect - r_g, real no obstacle - r_n, simulated glare effect - s_g, simulated no obstacle - s_n

	s_n		s_g		r_n	
	p	t	p	t	p	t
r_g	0.0064	-3.5291	$1.5e-05$	8.3768	0.0058	3.5985
r_n	0.0882	1.9120	0.0039	3.8430		
s_g	0.0044	3.7734				

Table 6.9: p values in paired t-test for different comparissons of penalties per round. For the simualted data only the best 10 simulated logs from each real log are used. Only the first 10 rounds are used. The following abreviatrions are used: real glare effect - r_g, real no obstacle - r_n, simulated glare effect - s_g, simulated no obstacle - s_n

	s_n		s_g		r_n	
	p	t	p	t	p	t
r_g	0.0020	-4.3052	0.0564	2.1881	0.0014	4.5577
r_n	0.9005	-0.1286	0.0012	4.6771		
s_g	0.0017	4.4200				

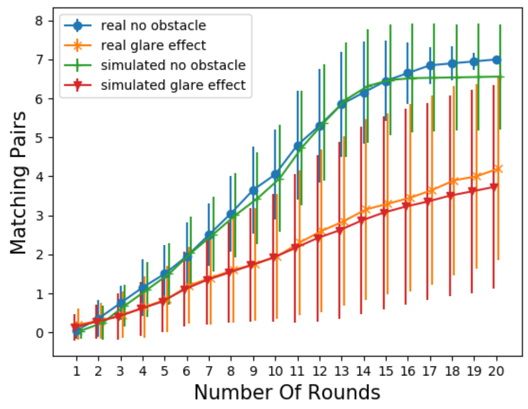

Figure 6.8: Add caption

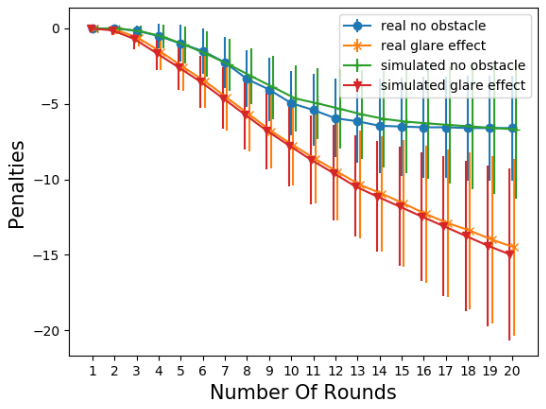

Figure 6.9: Add caption

Interestingly, including all the simulated data in the tests, results in less undesired result before the changes to the simulator compared to afterwards, but using only a small subset of the data results in the opposite. Before the changes to the simulator, when including only the best 10 games and the first 10 rounds in the tests, only one of the 4 tests comparing real and simulated games of the same mode show no significant difference. This is shown in table **TODO: ref** and **TODO: ref**. Contrary to that, after the changes to the simulator, the test on the chosen subset result in 3 out of those 4 tests show no significant difference, shown in table **TODO: ref** and **TODO: ref**.

As all but one test produce desired results after the changes to the simulator, additional searches for a configuration that only shows desired resultrs were done. However, such a configuration was not found. If less simulations are used, the three formerly significant differences become insignificant, but the difference in matching pairs bewtween simulated and real glare eeffect games becomes significant. Therefore using the 10 best simulations and the first 10 steps is a good compromise. It should be empghazised that this does not mean that the simulation of the glaree ffect games is bad reagrding the number of matching pairs per round. This only means that the paired sample t-test finds a significant difference in that comparisson. If the two lists of mean values that supposedly are significantly different are manually compared it can be seen that in reality the difference is very small. These values can be seen in table **TODO: ref**.

Table 6.10: mean values of matching pairs for real and simulated games. sd10x. for the first 10 rounds

	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10
real	0.2	0.3	0.45	0.65	0.85	1.2	1.4	1.6	1.75	1.95
simulated	0.135	0.27	0.405	0.6	0.79	1.105	1.34	1.53	1.72	1.915

The highest mean differnece is in round 6 with a differnece of 0.095 matching pairs in that round. This shows that it does not mean that all glare effect simulations used are bad regarding the matching pairs, only because the statistical test finds a significant difference.

6.6.3 Conclusion of the evaluation

Although no configuration in which all simulations have no significant difference to the according real games regarding the paired sample t-test was found, it can be said that the simulator is capable to simulate the first 10 rounds very accurately, given only the 10 best simulations are used. The statistical tests and the previous consciderations regarding the too large standard deviation in later turns indicate, that the simulator performs less good in later turns. Having said that, these are only indications that would need further tuning and testing of the simulator to be confirmed. Doing so might lead to improving the simulator.

Regarding the examination whether the changes to the simulator improved the overall quality of the simulations, no clear statement can be made. When all simualted data is used, the paired sample t-tests indicate, that the quality of the simulations may have gotten worse, while when using only the best 10 simulations and only the first 10 rounds, the tests indicate the opposite.

This analysis also indicates, that it might not ideal to use all simulated data for training as that could results in a significant difference between the simulated and the real games. Using to many simulations compared to real games could decrease the performance of the classifeirs on real data if they adapt too much to simulated games. Furthermore is should be analysed how using differnet amounts of turns in the classification impacts the classification results. In section **TODO: ref** different configurations regarding ratios of simulated to real games and the number of rounds includes, are tested.

Nonetheless it should be emphasized that the aim of no significant difference between real and simuleted games is set very high. As shown in table **TODO: ref**, the difference can be very little and the paired sample t-test still concludes a significant difference. Overall the simulations and real games are very similar when comparing them with the two performance measurements. If only the best simulations are used the performance becomes even more similar.

There are many options in choosing a subset of the data or adjusting the various parameters of the simulator, which could encance the quality of the simulations or reveal further aspects of the simulator that could be improved. In that sense, this analysis does not claim to be complete.

6.7 Feature generation

6.7.1 1D CNN features

For the 1D CNN five statistcal features for each step are used: The card codes, the number of cards left, the number of never revealed cards, the highest number of times the same card was revealed and the number of steps since all pairs were found. They are calculated using the game logs from **TODO: ref**. These features can be direclty fed into the 1d CNN explained in chapter **TODO: ref**.

The code for calculating the statistal features out of game logs was already given. A script was written that incorparates this funtionality in order to calculate the features for all logs. Additionally, after creating the neccessary directory structure the script saves the files for the splits of the raw data and the features. The reason for saving the features is that they do not have to calculated before evrey training and instead can be loaded from files. The reason for also saving the raw data even though this work does not need them anymore is, that the working group that was collaborated with also trained other models and does not directly load the features but instead caalculates them before each training.

6.7.2 2D CNN features

For the second approach of using a 2D CNN further steps are taken. Therefore synthetic images in gray scale colours are created using the features mentioned above. As the images are in gray scale colours, only one color channel is needed. One can visually think of this approach as creating graphs for each feature that display their values in each timestep, taking a photograph of each graph and stacking them on top of eachother. This can be clarified by looking at the code that produces the synthetic images.

```
def create_image(game, components=[True, True, True, True, True]):
    '''
    Creates synthetic images out of the five staticial features.
    :param game: The statical features in each step.
    :param components: Five values describing which of the features
    should be used to create the image.
    :return: The synthetic image.
    '''
    card_codes = np.zeros((7, steps))
    cards_left = np.zeros((8, steps))
    never_revealed_cards = np.zeros((14, steps))
    max_same_card_reveals = np.zeros((20, steps))
    rounds_since_done = np.zeros((27, steps))

    x_position = 0
    for step in game:
      card_code = math.floor(step[0])
      first_or_second = int(round((step[0] % 1) * 10))
      if card_code != 0:
        card_codes[card_code - 1][x_position] = first_or_second
      pairs_left[int(step[1] / 2)][x_position] = 1
      never_revealed_cards[int(step[2])][x_position] = 1
      max_same_card_reveals[int(step[3])][x_position] = 1
      rounds_since_done[int(step[4])][x_position] = 1
      x_position += 1

    image = np.zeros((0, steps))
    if components[0]:
      image = np.vstack((image, card_codes))
    if components[1]:
      image = np.vstack((image, max_same_card_reveals))
    if components[2]:
      image = np.vstack((image, rounds_since_done))
    if components[3]:
      image = np.vstack((image, cards_left))
    if components[4]:
      image = np.vstack((image, never_revealed_cards))

    return image
```

Listing 6.3: Add caption

By using pseudo colors, the images can be displayed with colours, like in figure 6.10. These $76 \cdot 40 \cdot 1$ (height · width · colour channels) images are direclty fed to the 2D CNN explained in chapter .. **TODO: rerf**. By setting the origin of the image to the lower instead of the upper left corner a more naural looking image is created and can be seen in figure 6.11.

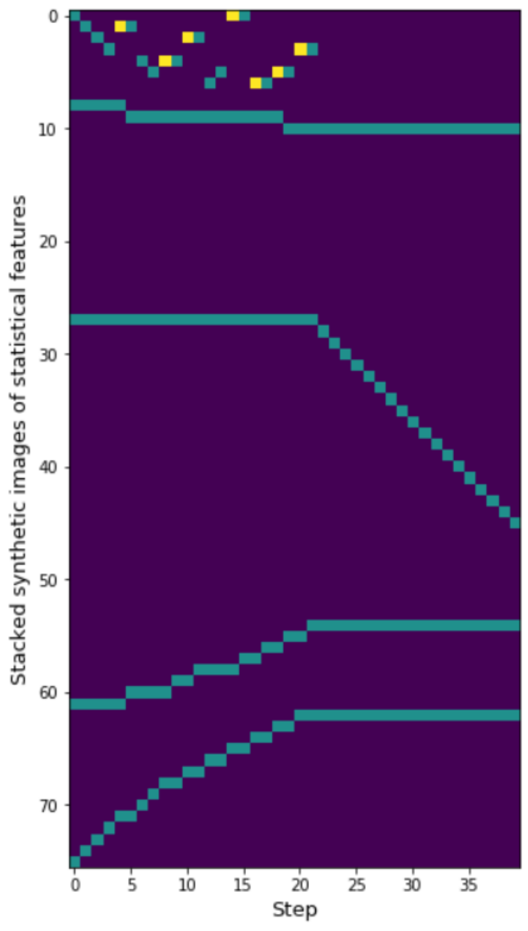

Figure 6.10: Add caption

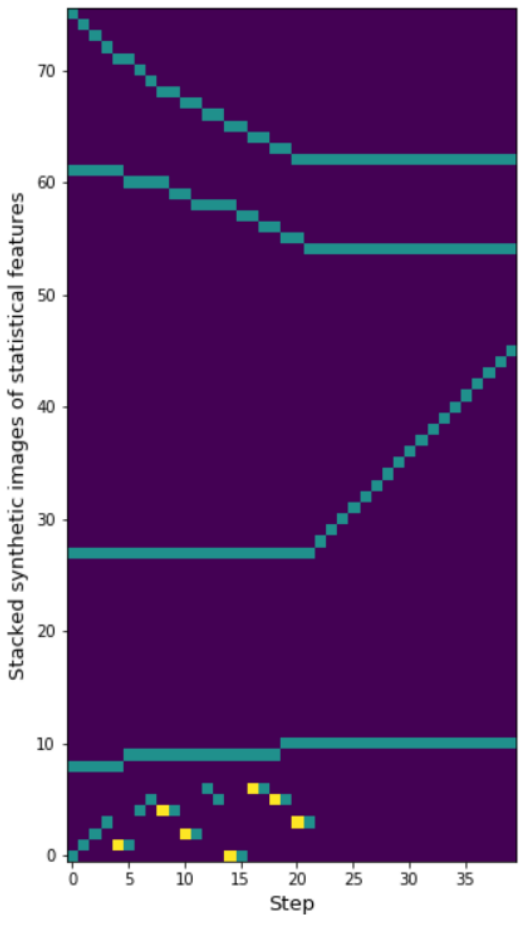

Figure 6.11: Add caption

The width of the image is 40 because that the number of steps recorded. Table **TODO: ref** shows which of the areas in the image describe which statical feature. The different vertical spaces given to statistical features were purposefully chosen. The aim was to use as much space as neccessary but as little as possible. As each round consists of two cards being turned face up and the impossibility to chose the same card twice in one turn, a card can be at maximum revealed 20 times during 20 rounds. Considering that this value is at minimum 1 because it is not possible to flip no cards, the range of 20 values is sufficient for this value. Furthermore 14 steps are at minimum needed to complete the game, since there are 14 cards. As a result this value can range from 0 to 26, meaning a vertical space of 27 is needed. In order to save space, the statistcal feature of the number of cards left was converted to the number of pairs left. By dividing by 2 the vertical space neccessary to visualize this feature is halved, without information being lost. Instead of representing the numbers left on the field, the value now represents the number of pairs left. Furthermore the number of never revealed cards can range from 0 to 13. The value 14 is not possible because two cards have to be chosen each turn, meaning that it is impossible to not reveal any card. The stacking order was chosen so that the coloured pixels of different statistical features rarely touch each other.

Table 6.11: add caption. Upper and lower boundaries are inclusive.

Statistical feature	Range	Vertical space in the image
Card codes	1-7	0-6
Maximum of same card reveals	1-20	7-26
Steps since game ended	0-26	27-53
Pairs left	0-7	54-61
Never revealed cards	0-13	62-75

Regarding the visual representatoin of he card codes two two decisions were made: Using different colours for the first and the second card of a colour and combining the information about the 14 cards in an vertical area of size 7. For each card code the vertical position of the representing pixel is decided by the first number and the colour is chosen according to the second number. As a result each row contains the information about cards of a specific colour. This representation of the card codes is not chosen arbirtrarily. On the contrary, it was inspired by reasons that need explanation.

Insrtead of combining the information about the 14 cards in an vertical area of size 7, it would have been also possible to use double the vertical space so that the first and the second card of a colour each have separate areas. The first cards and the second cards of a colour would each have individual vertical spaces with sizes of 7. However, it was decided against it. This decision was based on a conscideration regarding the way convolutional neural networks function and how they are used in this work. How cnns function is discussed in detail in chapter **TODO: ref** and therefore the following explanation is kept short. During a convolution, each filter has a fixed size and step by step walks over the image, extracting features from areas to create a feature map. This means that a neuron in this layer only reacts to stimuli in a local area of the previous layer. **TODO: satz selsbt verstehen und gucken ob ich es umschreieb dass es verständlicher ist. auf jeden fall auch in cnn chapter aufgreifen** This behaviour follows the biological model of the receptive field. As cnns consist of multiple layers, the receptive field, and therefore the distance in which dependencies between the pixels in the original input image can be learned, grows with each layer. This also means that if the structure of the cnn is chosen small, its receptive field might not grow enough for the model to learn dependencies between far distant pixels of the original input image. In this work, the 2d cnn used is intentionally constructed with a small number of layers for reasons explained in chapter **TODO: ref**. Therefore it is desireable that pixels in between an important dependency can be assumed are close to each other so that the small cnn is able to learn that dependency. Unlike in classical image classification the images used are created synthetically, and in that sense their structure can be chosen freely in the favour of the task at hand. The chosen representation of the card codes exploits exactly this freedom. As the dependencies between cards of the same colours can be estimated to be very important for the classification, the representation is chosen so that if cards of the same colour are flipped in quick succession the pixels describing them are also locally close to each other in the synthetic image. The dependency between such close pixels does not require a large receptive field in order to be learned by the network. Therefore this representation is better suited for the small structure of the cnn that is being utilized.

The main reason that motivated the use of different colours derived from the first decision to combine information about 14 cards in a vertical space of 7. If the same colour was used for all card codes, there would be no visual differnece between flipping the same card in consequtive turns and flipping two different cards with the same colour. This of course would also mean that the model could not differentiate bewteen those two cases, although they stem from completely differnet behaviour. Using different colours for the first and the second card of a colour fixes this issue. Furthermore the colours visually emphasize some behavioural characteristics that could be helpful for the eventual classifier. By comparing the the card code sections of two images from which one is created using a glare effect and the other one with a no obstacle game, using different colours results in noticeably different visual representations.

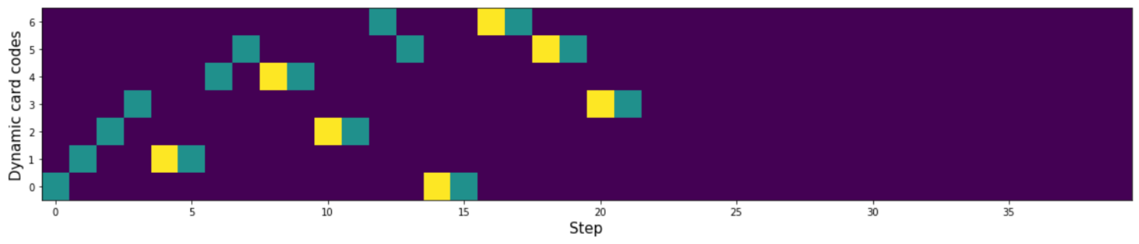

Figure 6.12: Add caption

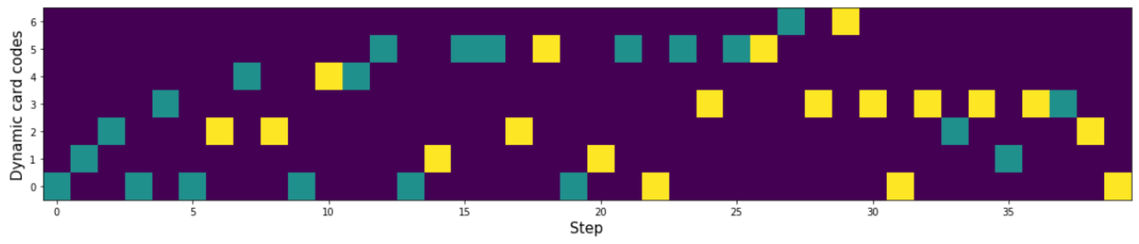

Figure 6.13: Add caption

In **??**, showing the image for an no obstacle game, there are yellow pixels directly left of green ones, meaning that the probant flipped a card, knew that he had already seen the matching card and direclty flipped it. However, this happens less often in figure 6.13 which shows the image for an glare effect game. This is not the case in every game, but the theory is that separations of different coloured cells in the image are more likely in glare effect games than in no obstacel games. Therefore it could be beneficial for the model to learn these characteristics. It should be noted that what the cnn exactly learns or what exact influence the chosen representation has can only be speculated.

Especially the uncertainty in the influence of the different colours shows, that the approach of creating synthetic images is very experimental. There is no such thing as a standard approach in creating synthetic images for cnns. At least there is none, yet. Although the way cnns function can be conscidered when deciding how to create the synthetic image, this does not guarantee good classification results.

umschrieben: grund ist, dass es zusätzliche characteristekn des spiels visualisiert, die von den convolutional genauer gesgat dem rezepivem feld erfasst werden können. ...

zeigen und erklären .. bei 14 reihen und getrennt jeweils für erste ode zweite wären die pixel die gefundene paare beschrieben nicht in der nähe voneonader. Ein kleines rezeptive feld würde könnte den zusammenhanmg nicht erfassen. Und selbst wenn man ein größeres wählt gibt es in dem bereich des feldes dann deutlich mehr pixel mit dem selben farben die den wichtigen zusammenhang zwischen 2 spezifischen pixeln in den schatten stellen könnten. Würde man gleiche farben benutzen gäbe es keinen untershcied zwischen zweimal die selbe karte in nah einanderfolgenden zügen und die zueinandergehörigen beiden karetn. Es heißt jedoch nicht dass das netz nicht in der lage ist featurees (oder so) aus verschiedenen bereichen in verbindung zu setzen. Ganz im gegenteil: die fullyy connected oder auch dense layer tun genau dies. Jedoch suchen sich die fully connected layer quasi die feautures aus dem letzen vorherigen layer aus die wichtig sind. Aber angenommen die beiden angesprochen pixel sind sich nicht nah genug um vom rezeptiven field erfasst zu werden und zwischen den beiden pixeln bzw. generell in dem bereich der card codes sind noch viele weitere solcher pixel, dann gibt es keine features die den zusammenhang enthalten und die aufgabe diesem ztusammenhang zu finden liegt dann an den dense layern. Das die dense layer einen zusammenhang zwischen genau den beiden pixln finden und das für gänzlich unterschiedliche card code bereiche je spiel ist äußerst undwahrscheinlciu. **TODO: das muss biscchen genauer und verständlicher alles.**

TODO: text

TODO: vielleicht mit mazen drüber reden ob das sinn ergibt. Ich glaube das ergibt sinn, aber ich müsste nochmal mehr das quellen suchen. Auch mir receptive field und so

7. Convolutional neural networks

As convolutional layers are complex and have many variations for differnet use cases, only concepts and functionaslities that are relevant for this work will be covered. This also menas that 3 dimensional cnns will not be explained, but in generel cnns function the same way whether they have 1, 2 or 3 dimensions. The main differnce lies in how the filter, also known as kernel or festure detector, moves acrross the data. This is later explained in more detail. (**TODO: ich glaube input data ist keine regel sondern nur meist so..zum beispiel kann man auch 1d convoltion bei 1d, 2d, oder 3d daten machen**).
First of all the core concepts of cnns will be explained and afterwards the models used in this work will be described.

On of the core concepts of convoluntional neural networks are the concolutions.

Unituitievly, a 2d cnn must not neccerailty have 2 dimensional data as input.

notes bei models:
- stride = wie viele schritte auf einmal, default 1 deshalb vermute ich bei mir 1
- maxpooling: pool size = 2*2, und bei keras wird stride dann defauklt auch zu 2*2 (=pool size)
- input size = 85*40 glaube ich
- filter weights are randomly initialized, so that they dont all initially learn the same feauters. To briefly explain the behaviour of filters a seperation bettween two cases can be made: The first case is that not all features of high quality have been learned yet, high quality meaning that learning them would lower the cost function. In that case it is highly unlikely that each filter would begin to resemble other filters, as that would most certainly result in an increase of the cost function and therefore no gradient descent algorithm would head in that direction. The other case is that all features of high quality have already been leanred by a subset of the avaibable filters. In this case the cost function would not be increased if the remaining filters learn similar features than the other filters.
- in einem der schritte wird eine zeile usgelassen, aber ich trainiere nicht nochmal

alles neu (oder?)

- 1d and 2d convolution - reduction of dimension - relu - number od filter, random initializion, etc - kernel and kernel size - loss function and optimizer, stochastic gradeint decent - Dropout layers - Pooling, max pooling - flatten - dense relu, softmax - droput layer, pooling ändert nichts an der anzahl der fetsure maps - **TODO: bessere quelle finden, buch oder so** - stride (ganz kurz, beschriebt verhalte wenn etwas über bleibt, zero padding oder weglassen) - back propagation - wie schaffen die es bspw. 64 feature maps auf 32 zu reduzieren? also wie funktionieren die filter in dem fall? - receptive field

8. Classification

8.1 Hardware resources

Multiple servers, provided by the cognitive system lab, were used. One of the provided servers is optimized for storing large amounts of data and was therefore used to store the histories and the results from the models during training while the other servers focus on computing power and were used to execute that training and evaluation. Mainly two servers were used for the training. One has the hardware specification shown in table **TODO: ref** and the other one those shown in table **TODO: ref**.

Table 8.1: add caption

Component	Model	Additional details
Processing Unit	Intel® Xeon® Gold Prozessor 5218	64 logical cores, turbo boosts to 3,90 GHz
System Memory	-	377 GB
Grafics	2 x NVIDIA RTX 2080 ti	11 GB graphics memory

Table 8.2: add caption

Component	Model	Additional details
Processing Unit	Intel® Xeon® Prozessor E5-2650 v4	24 logical cores, turbo boosts to 2,90 GHz
System Memory	-	252 GB
Grafics	NVIDIA GTX 1080 ti	11 GB graphics memory

TODO: specs nochmal checken sobald ich ins cartesium kann und da nachfragen. vor allem grafikkarte und hesteller des rams

8.2 Models

For determining whether probands were blinded, one dimensional and two dimensional convolutional neural networks were utilized. One dimensional convolutional neural networks have become popular for time series classifications **TODO: ref**, while wo dimensional coovolutinal neural networks are mostly used for image processing. However, a possibility is to create synthetic images from the data and use these images in a two dimensional convolutional network as done in section **TODO: ref**. The dimensions of the data and the feature maps in the visualizations are calculafed for the case that all 20 rounds are used. If a diferenet nuber of rounds is used the values differ. The cnn models are intentionally created not very complex for two reasons: Firstly, a lot of training needs to be done with differnet configurations regarding the steps included and the ratios of simulkated to real data. As mentioned in **TODO: ref**, 76800 models need to be trained. As the time this work is created in is limited, training that many complex models would take too long. Especially the training of the 2d cnns requires a lot of time. Secondly, to assure that the structures are complex enough, more complex models were exemplary trained and they showed either similar or worse performance. The results of the more complex models can be see in the appendix **TODO: ref?**.

8.2.1 1D CNN

The structure of the 1D CNN used is shown in figure 8.1.

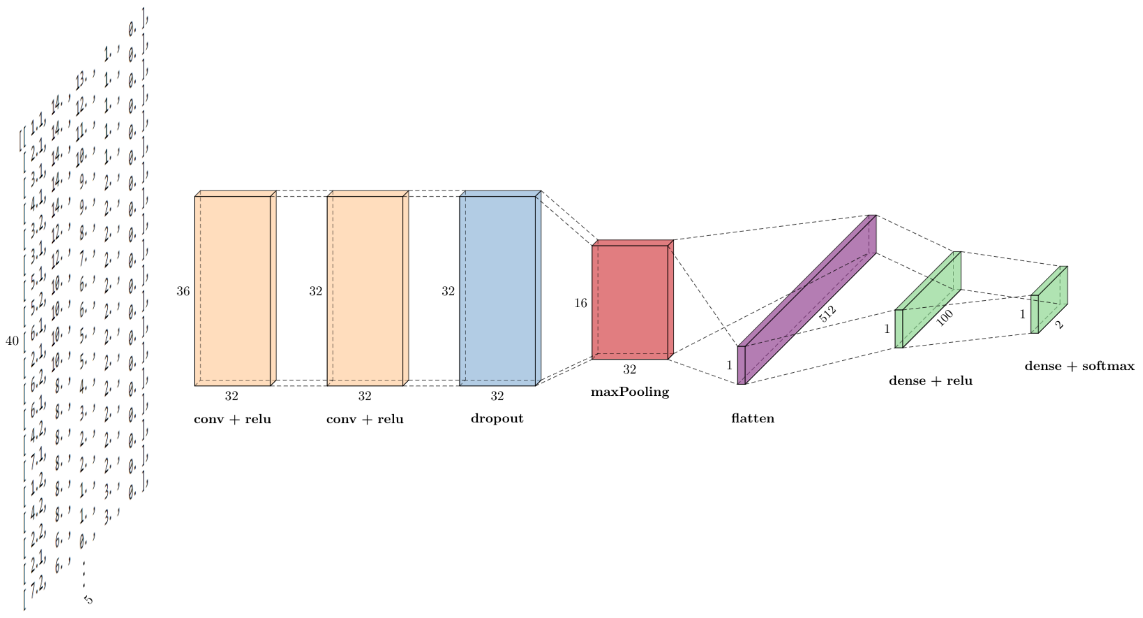

Figure 8.1: The width describes the number of feature maps and the height and depth describe the dimensions of the feature maps. As example after the first convolutional layer the data consists of 32 one dimensional feature maps with the length of 36.

The input data consist of the 5 statical features, explained in section **TODO: ref**, for each of the 40 steps (20 rounds), resulting in the input dimensions $40 \cdot 5$. Initially the data is passed to the first convolutional layer of the network. The first layer has 32 filters and the kernel has a size of 5, meaning that every step from the kernel includes all data from 5 steps in the game. Since the input data is two dimensional and the convolution is one dimensional, the created feature maps are one dimensional. Therefore the first convolutional layer creates 32 festure maps each with a size of

36. These feature maps are passed to the second convolutional layer with the same number of filters and the same kernel size as the first layer. This again results in 32 feature maps with a decresed size of 32. Afterwards a dropout layer with a rate of 0.5 randomly sets input units to 0 and by that helps to prevent overfitting. Then a one dimensional max pooling layer with a pool size of 2 reduces the size of the 32 feature maps to 16. And finally the network is completed with a flatten layer and two dense layers. **TODO: vielleicht genauer erklären was dense macht, aber eigentlich ja oben auch schon, nur einen groben satz hier**. Both convolutional layers, and the first dense layer use a rectified linear unit as activation function. The last dense layer uses softmax normalization to bring the data down two two values for the two classes. The elements of the output vector are between 0 and 1 and sum up to 1 and specify the result of the classification. In total 57486 parameters are trained in this model.

As optimizer the Adam optimizer from keras is used that implements the Adam algorithm. Adam optimization is a stochastic gradient descent method **TODO: ref zu keras adam seite**. The learning rate is set to 0.0001 and not changed during the training. The batch size is set to 32 and the model is trained for 2500 epochs.

8.2.2 2D CNN

The structure of the 1D CNN used is shown in figure 8.2.

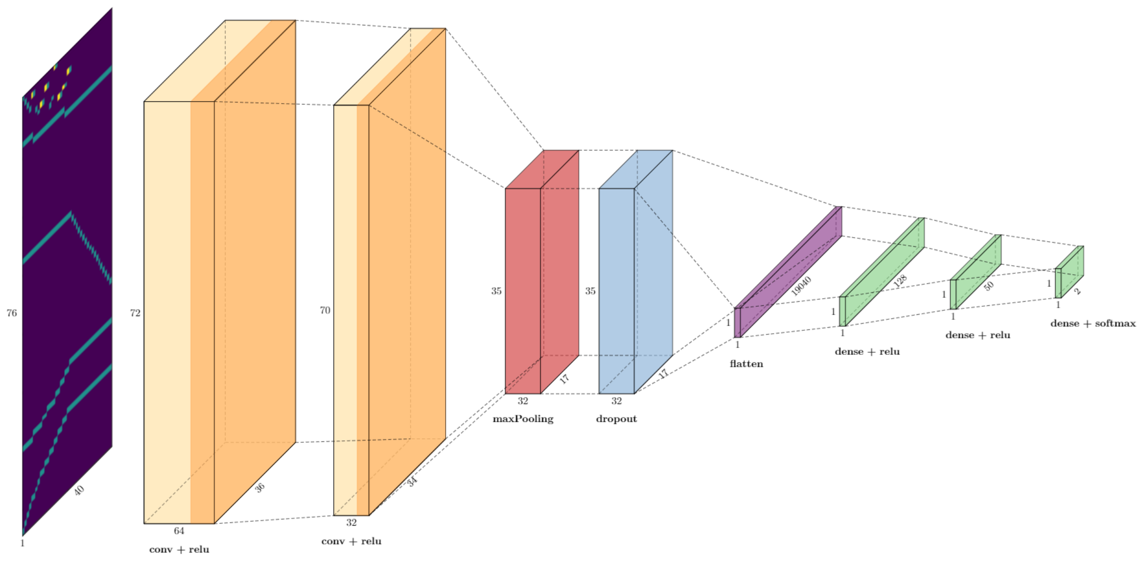

Figure 8.2: The width describes the number of feature maps and the height and depth describe the dimensions of the feature maps. As example after the first convolutional layer the data consists of 64 two dimensional feature maps with the size of $72 \cdot 36$.

The input data consist of the synthetic images generated in .. **TODO: ref** using the 5 statistical features, resulting in the input dimensions $76 \cdot 40 \cdot 1$. Initially the data is passed to the first convolutional layer of the network. The first layer has 64 filters and the kernel has a size of $5 \cdot 5$, meaning that each step from the kernel includes a $5 \cdot 5$ sized area of the synthetic image. This convolution results in 64 feature maps each with dimensions of $72 \cdot 36$. These feature maps are passed to the second convolutional layer with 32 filters and a kernel size of $3 \cdot 3$. This results in 32 feature maps and a decresed dimensionality of $70 \cdot 34$. Then a two dimensional max pooling layer with a pool size of $2 \cdot 2$ reduces the dimensinalty to $35 \cdot 17$. Afterwards a dropout layer with a rate of 0.2 randomly sets input units to 0 and by that helps to prevent overfitting. And finally the network is completed with a flatten layer and three dense layers **TODO: vielleicht genauer erklären was dense macht, aber eigentlich ja oben auch schon, nur einen groben satz hier**. Both convolutional layers, and the first two dense layers use a rectified linear unit as activation function. The last dense layer uses softmax normalization to bring the data down two two values for the two classes. The elements of the output vector are between 0 and 1 and sum up to 1 and specify the result of the classification. In none of the different operations zero padding was required in order to prevent information loss. In total 2463928 parameters are trained in this model.

As optimizer the Adam optimizer from keras is used that implements the Adam algorithm. Adam optimization is a stochastic gradient descent method **TODO: ref zu keras adam seite**. The learning rate is set to 0.00001 and not changed during the training. The batch size is set to 32 and the model is trained for 1000 epochs.

8.3 Training setup

There is one script each for training the 1D cnn and the 2d cnn. The only differences are in the structure and the hyper parameters of the cnns, explained in section **TODO: ref**, and that before training the 2d cnn the loaded statistical feaures must first be used to create the synthetic images. Both scripts load the same statistical features, already divided into 20 splits. As there are statistical features for 20 real and 20000 simlauted no obstacles games as well as the same amount for glare effects games, this results in each split containing data from 19 real and 19000 simualted games for each of the two classes, resulting in 38 real and 38000 simulated games available for the training. From 2 real and 2000 simulated games not used for training, only the data from the real games is used for testing. This is done becauuse it is not desirable that the models adapt to the simualted data as in reality they will be only used in real games. Furthermore it can be chosen for which ratios between simualted and real games models should be trained. It was chosen to use ratios of one real game to 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, and 20 simulated games. Moreover it can be chosen how many turns in the game should be used (a turn consists of flipping 2 cards). In real world scenarios, the less the faster the decision is made and assistance can be applied, but if waited the longer the decision will likely be correct more often. To analyse this, models for different amounts of turns are trained. The turns chosen are the first 5, 10, 15 and 20. More than 20 turns were not recorded. **TODO: Vielleicht rein: Training models for ranges of turns in the middle, for instance from turn 10 to 15, was left out, because this is not very interesting when conscidering real world usage.** All models are trained twice. Once on the simulated data created before modifying the simulator and once afterwards. This can potentially show whether the changes improved the classification results.

The structure and hyper parameters of the modells for all 1d cnns are identical. Same goes for all 2d models. The only differnece is the input shape of the modells since it depends on teh number of steps used in the training. Identical structure for modells of the same type are used so that different training results for different amount of steps can be reliably compared. It should be noted the structure and hyper paramters are not the same between the 1d and 2d cnns.

Once one of the scripts is executed, required directory structures are created and training processes for the different ratios of real to simulated games are performed successively. The training for one ratio consists of parallelized training of 20 splits using multiprocessing. The training for each split is repeated 20 times. As a result for each ratio 20 models are created for every split. This means that 400 models are trained for every ratio. As there are 12 different ratios trained, this results in 4800 models trained for each execution of one of the scripts. As mentioned above, the training is done for 4 different amounts of steps for the 1d cnn and the 2d cnn. This adds up to 38400 models being trained. Furthermore this done twice: Once before the changes to the simulator and once afterwards. All in all this results in 76800 models. For each model the training history is saved in a file. Saving and loading the historeis is neccessary due to the parallel training of multiple models through multiprocessing. Once the training of the 4800 models of one script is completed, the training histories are loaded from the files. The results from the 400 training histories for each ratio of real to simulated games are averaged, plotted and the final figures are saved for the analysis. The reason for averaging the results over many models is, that deep neural networks by default initialize their weights randomly, resulting in potentially different outcomes of each training. By averaging the results, the values and therefore the interpretation becomes more reliable and meaningful. All models are trained on the cpu. The fact that 76800 models are trained led to the decision, not to save all models but instead first evaluate the results and then retrain and save the models that create the best results.

8.4 Training results and evaluation

The training results for the various configurations can be seen in table **TODO: ref**, **TODO: ref**, **TODO: ref** and **TODO: ref**. The first two tables contain the results before the changes to the simulator and the last to the results afterwards. The analyiss will only focus on the accuracy, but the loss is also listed for the sake of completeness. The best accuracies for each number of turns in each table are highlighted in green. If multiple ratios of real to simulated game for a certain number of turns produce the same accuracy only the one with the smallest amount of simulations is highlighted. The accuracies in the tables are those from the best epochs. In most of the cases the models start overfitting after a certain epoch resulting in a deterioration of the accuracy after reaching a local maximum. All models except the 1d cnns for 5 rounds and ratios of sd0x and sd1x reach such a local maximum in accuracy before the training is stopped.

It should be claryfied that the chosen configurations of models to be trained make no claim to completeness. The models are not individually optimized and therefore do not neccessarily produce the best results possible. As the amount of collected real data with 20 recordings for each label is very limited, it was decided not to split the data into training, test and validation data, and instead only into training and test data. In order to optimize the models such a validation set would be neccessary. It could be used to tune the hyper parameters of the model. The reason why this is done on a separate validation set is, that the models can then be tested on the test data which was not part of the process of selecting the best hyper parameters. If the models were optimized using the test data, the results would not be representitive of the performance of the models on unknown data.

Table 8.3: 1D CNN. Best accuracies and losses for 5, 10, 15 and 20 steps. Before the changes to the simulator.

	5 r.		10 r.		15 r.		20 r.	
	Acc.	Loss	Acc.	Loss	Acc.	Loss	Acc.	Loss
sd0x	65.12%	0.6584	81.12%	0.5116	78.37%	0.5377	79.62%	0.492
sd1x	71.25%	0.6385	80.88%	0.4777	77.25%	0.5355	80.12%	0.4999
sd2x	68.5%	0.6476	77.38%	0.5283	78.25%	0.5275	81.25%	0.4887
sd3x	69.38%	0.6494	75.5%	0.5252	76.13%	0.5329	79.75%	0.4999
sd4x	67.5%	0.6436	75.88%	0.538	77.25%	0.5322	81.25%	0.4975
sd5x	72.25%	0.6212	76.5%	0.5252	77.75%	0.5338	81.25%	0.4985
sd6x	71.87%	0.6346	74.75%	0.547	76.25%	0.5376	79.38%	0.5075
sd7x	70.25%	0.6252	73.62%	0.5395	76.0%	0.5326	80.0%	0.5078
sd8x	74.25%	0.6121	72.5%	0.5509	77.0%	0.535	79.75%	0.4993
sd9x	74.88%	0.5826	74.63%	0.5483	76.0%	0.542	81.75%	0.4987
sd10x	74.87%	0.5971	74.5%	0.5558	77.12%	0.5394	80.87%	0.5036
sd20x	73.25%	0.5831	73.5%	0.5493	75.63%	0.5355	80.75%	0.5013

Table 8.4: 2D CNN. Best accuracies and losses for 5, 10, 15 and 20 steps. Before the changes to the simulator.

	5 r.		10 r.		15 r.		20 r.	
	Acc.	Loss	Acc.	Loss	Acc.	Loss	Acc.	Loss
sd0x	63.5%	0.6471	74.0%	0.4991	79.62%	0.4726	79.0%	0.4778
sd1x	68.12%	0.619	76.62%	0.5171	79.5%	0.4984	81.75%	0.4735
sd2x	68.25%	0.6382	80.0%	0.5243	80.38%	0.4851	84.12%	0.4573
sd3x	69.5%	0.6392	78.88%	0.5518	80.25%	0.5038	84.62%	0.4718
sd4x	67.63%	0.6415	76.38%	0.5685	79.62%	0.5106	85.0%	0.4726
sd5x	70.88%	0.6339	74.63%	0.5668	80.0%	0.51	84.88%	0.474
sd6x	70.62%	0.638	77.0%	0.5623	80.12%	0.5077	84.88%	0.4776
sd7x	69.75%	0.6543	75.38%	0.5762	79.5%	0.5176	85.0%	0.4839
sd8x	70.0%	0.6467	74.25%	0.5775	80.38%	0.5164	85.0%	0.484
sd9x	71.25%	0.6395	78.75%	0.5696	81.5%	0.5064	85.0%	0.4814
sd10x	68.75%	0.6296	77.75%	0.5674	80.12%	0.5098	85.0%	0.4802
sd20x	**73.0%**	0.5833	76.63%	0.5486	79.75%	0.5064	84.88%	04704

Table 8.5: 1D CNN. Best accuracies and losses for 5, 10, 15 and 20 steps. After the changes to the simulator.

	5 r.		10 r.		15 r.		20 r.	
	Acc.	Loss	Acc.	Loss	Acc.	Loss	Acc.	Loss
sd0x	63.87%	0.6698	81.88%	0.5014	78.5%	0.5401	78.62%	0.5107
sd1x	67.75%	0.6648	79.0%	0.5095	75.75%	0.5305	79.38%	0.5101
sd2x	66.0%	0.6506	79.25%	0.4845	76.0%	0.5325	78.37%	0.514
sd3x	64.38%	0.6585	77.38%	0.5037	77.62%	0.5261	79.25%	0.5113
sd4x	67.0%	0.6377	75.38%	0.5289	77.62%	0.5331	80.25%	0.5027
sd5x	67.0%	0.6452	76.75%	0.5146	78.75%	0.5226	79.0	0.5042
sd6x	70.25%	0.6427	75.25%	0.5335	78.75%	0.5208	79.38%	0.5049
sd7x	70.25%	0.6449	76.88%	0.5114	78.0%	0.5129	78.75%	0.5115
sd8x	71%	0.628	76.25%	0.523	77.25%	0.5261	79.12%	0.5156
sd9x	70%	0.6276	75.37%	0.5226	77.62%	0.5281	78.63%	0.522
sd10x	71.5%	0.6242	76.25%	0.5261	76.38%	0.5331	78.62%	0.4986
sd20x	71.88%	0.6246	72.38%	0.5617	76.25%	0.5416	79.12%	0.5

Table 8.6: 2D CNN. Best accuracies and losses for 5, 10, 15 and 20 steps. After the changes to the simulator.

	5 r.		10 r.		15 r.		20 r.	
	Acc.	Loss	Acc.	Loss	Acc.	Loss	Acc.	Loss
sd0x	63.62%	0.6461	73.87%	0.4929	80.0%	0.4557	78.5%	0.476
sd1x	67.88%	0.6446	76.12%	0.531	79.5%	0.4911	79.75%	0.4874
sd2x	69.75%	0.6463	79.62%	0.5243	80.87%	0.4905	80.0%	0.4859
sd3x	69.5%	0.6449	77.5%	0.5507	82.25%	0.4994	80.0%	0.4865
sd4x	69.0%	0.6347	72.75%	0.5684	80.12%	0.5156	80.0%	0.495
sd5x	69.38%	0.6342	73.62%	0.5634	80.5%	0.5122	80.0%	0.4917
sd6x	71.0%	0.6326	74.5%	0.5586	80.62%	0.5038	79.5%	0.4874
sd7x	70.0%	0.6328	71.62%	0.555	80.0%	0.5061	79.5%	0.4874
sd8x	69.88%	0.6287	72.75%	0.5567	80.25%	0.5057	79.62%	0.4874
sd9x	69.12%	0.6238	72.38%	0.5532	79.5%	0.5067	78.88%	0.4887
sd10x	69.5%	0.6203	73.25%	0.5536	79.5%	0.5062	79.0%	0.4904
sd20x	69.88%	0.5935	73.0%	0.5494	79.62%	0.5005	0%	0

The results show that using a certain number of simulated games for the training can often significantly improve the classificatin resuls over using only the real data. A good example is the accuracy for sd0x compared to sd4x for 20 rounds for the 2d cnn in table **TODO: ref**. Using no simulations the accuracy is 79.0% while for a ratio of 9 simulations for each real game the accuracy is 85.0%. Furthermore is can be seen that using to many simulations compared to real games can decrease the accuracy. In table **TODO: ref**, for 10 rounds, the accuracy for sd2x is 80.0% and less when using more simulations. This behaviour highly varies depending on the configuration of the model. Sometimes the models produce the best results without any simulations, while other times using simulations strongly outperforms using only real data. It is possible that the optimal ratios of simulated to real games are not included the tested configurations. Especially the 2d cnn for 5 rounds before the changes to the simulator achieves the best accuracy with the highest ratio that was tested. Therefore the optimal ratio could be higher than 20 simulations for each real game.

When looking at the best accuracies of 1d and 2d cnns for each number of rounds, it can be observed, that the accuracy of the 1d cnns do not get much better and sometimes even worse after more than 10 rounds, while this is not the case for 2d cnns. This can be seen in figure **TODO: ref**. The fact that the 1d cnns do not profit from more than 10 rounds might be related to the findings from section **TODO: ref**, that indicate that the simulaor perfroms worse in later rounds. However, if the case, this begs the questions why this does not apply to 2d cnns. There is a possibility, that the worse perfromance in later turns has a strong impact on the statistacal features fed to the 1d cnn, but once these feautures are used to create the synthetic images for the 2d cnn this negative impact decreases. Figure **TODO: ref** additionally shows the ratios of simulated to real data for each of the best results. The fact that the 2d cnn after 10 rounds still have the best results when using a substantial amount of simulations while the the best results for the 1d cnns after the 10th round use very little simulations, supports the theory from above. Nonetheless the findings are not sufficient to verify the theory. For that more research would be required. However, if found to be true, it could mean that 2d cnns using synthetic images have the potential to handle inacurate simulations better than 1d cnns. From a different perspective, this could also indicate that, assuming the simulator does actually perform worse in later rounds and is improved in that regard, the accuracy of the 1d cnns after 20 rounds could be improved.

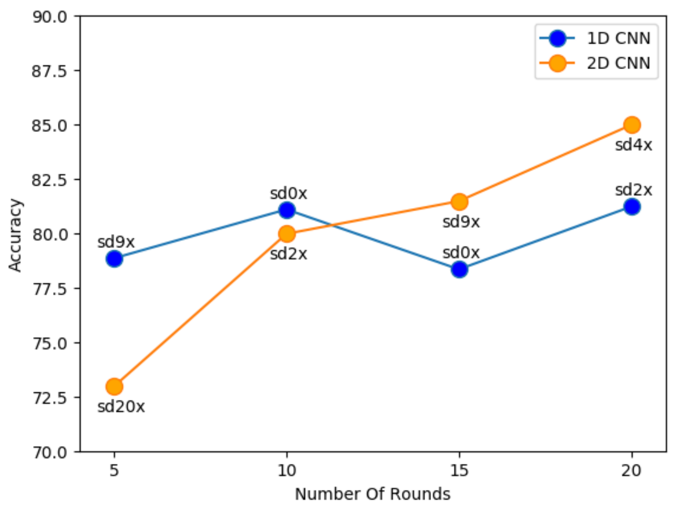

Figure 8.3: add caption

It was conscidered to perform statical tests comparing different models and ratios to reveal significant or insignificant differeneces. However, the decision was made that this does not realy gives reliable results, because the values in the tables **TODO: ref** are not neccesarily the best accuracies achievable. As mentioned above the models were not individually optimized. Therefoe the results of ststistical tests performed on the achieved accuracies in the tables, would not justify any staements regarding whether 1d cnns or 2d cnns perform significntly better in different configurations. Such statistical test would only show significant differences or indiffierences in the results, but could not be interpreted for any valuable findings, as not all models are used to their full potential. For choosing the best configurations from those tested no statistical tests are used. Instead simply those with the highest accuracies are chosen.

Table 8.7: add caption. Upper and lower boundaries are inclusive.

	5 r.	10 r.	15 r.	20 r.	Average
1D CNN - before	74.88% (sd9x)	81.12% (sd0x)	78.37% (sd0x)	81.25% (sd2x)	78.91%
1D CNN - afterwards	71.88% (sd20x)	81.88% (sd0x)	78.75% (sd5x)	80.25% (sd4x)	78.19%
2D CNN - before	73.0% (sd20x)	80.0% (sd2x)	81.5% (sd9x)	85.0% (sd4x)	79.88%
2D CNN - afterwards	71.0% (sd6x)	79.62% (sd2x)	82.25% (sd3x)	80.0% (sd2x)	78.22%

Table **TODO: ref** shoes the best result for all categories before and after training. It can be seen that although some accuracies are slightly better after the changes to the simulator the results are on average noticeably worse than before. For instance is the accuracy of 85.0% from the 2d cnn for 20 rounds decreased to 80.0%. The reason for this is unknown and can be highly complex, due to the many factors involved. A theory is, that reucing the randomness of the simulator and therefore making the simulated games more similar to the real games, may have resulted in decreased generalisation in the models and therefore less robustness on unknown data. However, this theory can hardly be varified. The descreased performce after the changes leads to the decision to only use the models trained on the simulated data created before the changes to the simulator. The chosen configurations can be seen in table **TODO: ref**.

Table 8.8: add caption. Upper and lower boundaries are inclusive.

	5 r.	10 r.	15 r.	20 r.
1D CNN	74.88% (sd9x)	81.12% (sd0x)		
2D CNN			81.5% (sd9x)	85.0% (sd4x)

The 1dnns trained, perform better for 5 and 10 rounds than 2d cnns while 2d cnns perform better for 15 and 12 rounds. Depending on the number of rounds it can therefore be benefitial to use either 1d cnns or 2d cnns. The 4 best configurations for the differnet numbers of rounds are used in section **TODO: ref zu voting**. Figures **TODO: ref**, **TODO: ref**, **TODO: ref** and **TODO: ref** show the training and test accurcy during the course of training these 4 models.

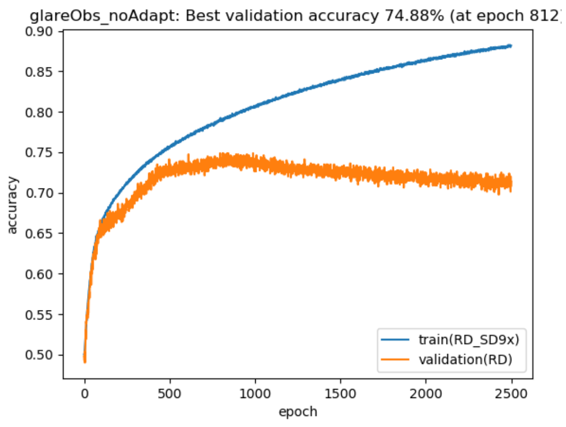

Figure 8.4: Add caption

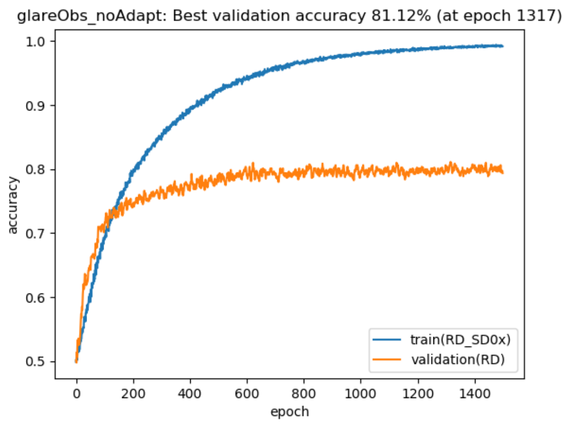

Figure 8.5: Add caption

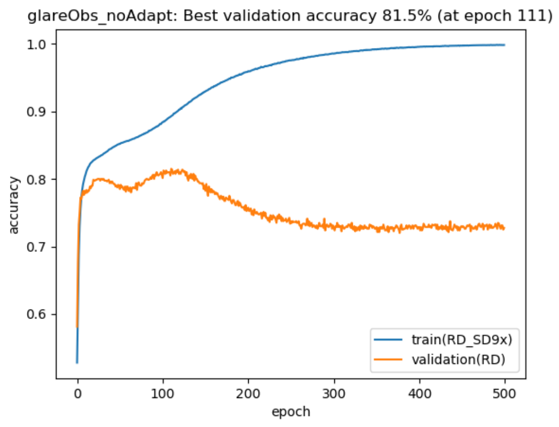

Figure 8.6: Add caption

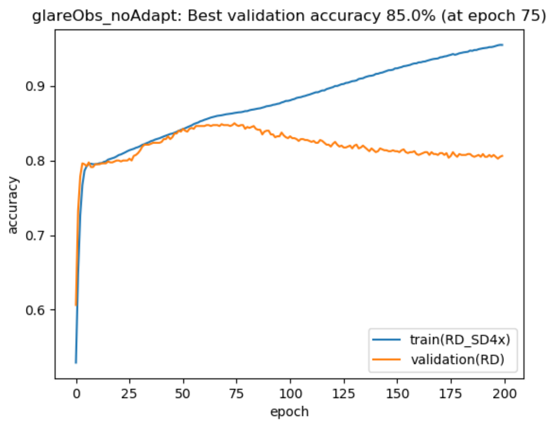

Figure 8.7: Add caption

9. Voting based system

The accuracies shown in section **TODO: ref** are not representitive of how the models would perform in real life scenarios. Since the accuracy is the mean accuracy of 20 repititions for each split there is no single modeel that produces that accuracy. And due to the fact that stochastic models can produce different outcomes each time they are trained, using only one of the models for the prediction does not produce relaible results **TODO: quelle**. For analysing how the modles would perform in real life scenarios a voting based system is deployed **TODO: quelle die sagt dass man sowas benutzt**. It consists of multple models classifying the input data and then voting for the final classification. A voting based sytsem for each of the best cnns and ratios of simulated to real data for each number of step is created. As mentioned in section **TODO: ref** the 1d cnn creates the best accuracy for 5 and 10 rounds while the 2d cnn creates the best accuracies for 15 and 20 rounds. Hence, in total 4 voting based systems are created, one for each of the 4 number of rounds. **TODO: besser ref zu einer tabl machen wo die drin stehen** Each of the 20 repititions of a split is tested as before, only with the two real games (one no obstacle and one glare) that were not included in the training. However instead of direclty calculating the accuracy of each repitition of that split, the output labels are saved for all repitions and used to perform a voting for the final label. Meaning that if for instance 15 out of 20 repitions classyfied the game as a glare effect game, while the other 5 classified it as a no obstacle game, the final classification will be that of a glare effect game. This is done for the two test games specific to each split. Using the final labels created by the voting, the accuracy of that split is calculated. In the example above, if assumed that both final labels would be correct, the accuracy of that split would be 100%. This procedure is repeated for each of the 20 splits and the accuracies of the splits are averaged.

As none of the models have been saved after the training, the 400 models for each of the 4 voting based systems, are retrained and saved. In total 1600 models are trained and saved for the voting based systems. However, they are trained for less epochs than in section **TODO: ref**, as the models would otherwise overfit. For each of the 4 configurations the number of epochs in chosen so that the accuracy of the models in the last epoch, and therefore the accuracy of the saved model, is in the area of the local maxima. The number of epochs was intentionally chosen only in a rough area of the local maximum of the test accuracy and not exactly at the maximum. As mentioned in section **TODO: ref** there is no validation set and therefore using the exact number of epochs as optimal for the test data would mean that the models would be optimized according to the test data and the results would be less representative of the real world performance of the systems. The **TODO: ref** shows amongst other things the average accuracy after the last training epoch for each of the 4 configurations with and without voting. The slight difference of the average accuracies without voting shown in the table and the accuracies shown in table **TODO: ref zu oben wo optimum ist** are caused by the fact that the models needed to be retrained in order to save them. As the accuracies in table **TODO: ref zu oben wo optimum ist** were those from the best epoch and due to the nature of stochastic models their accuracy does not neccessarily reach that identical value after exactly the epoch after which the training for the models was stopped in.

Table 9.1: add caption.

	avg. acc.	acc. of VBS	Overruled correct classifications	Overruled false classifications
5 r.	73.13%	72.5%	38	33
10 r.	80.13%	80.0%	32	25
15 r.	80.63%	80.0%	9	4
20 r.	85.0%	85.0%	0	0

When comparing the average accuracies without voting with the accuracies of the voting based systems, it can bee seen that the accuracy of the voting based systems is eithher sligly lower or identical. In some cases such a system may have a better accuracy that without voting, but this is not always the case. Two key factors that this is influenced by are the numbers of correct and false classifications that are overruled by the majority in the voting. If more correct classification are overruled by wrong ones than wrong by correct ones, the accuracy of the voting based system will be worse than the average accuracy without voting. The reason is that once a correct classification is overruled by the majority, it has no influence on the calculated accuracy any more, because the accuracy of the voting based system is determined by the final labels. An good example of such a case occured in the voting based system for 5 rounds, where in one of the splits, 11 out of the 20 models voted incorrect, resulting in 9 correct predicitoins being ignored. Table **TODO: ref** also shows how many correct and false predictions were overruled by their opposition. In every system except the one for 20 rounds, more correct predictions are overruled than false ones, resulting in a lower accuracy of those voting based systems compared to the accuracy calculated without voting.

As shown by the accuracies that are all higher than 50%, the majority of the 400 models in each system make the correct prediction. However, their votes are not evenly distributed across all splits. A voting based systems highly profits from low fluctuation in the voting distribution between the models. If the voting in the majority of splits has similar results, such that for instance around 15 models vote correct and 5 vote wrong, the accuracy of those splits will all be 100% and the average accuracy of the voting will be higher than the average over all models without voting. But if in for instance in the first split 20 models vote correct and 0 incorrect, while in the second split 9 vote correct and 11 incorrect, the overall performance of the voting based system suffers. Instead it would be better if in the example above, in the first split 15 vote correct and 5 incorrect and in the second split 14 vote correct and 6 incorrect, resulting in the accuracy of both split being 100%. In total in both explained distributions, 29 models vote correct and 11 incorrect, but the performance of the voting based system that has less fluctuation in the distribution performs massively better. A way to achieve less fluctuation in the voting distribution can be to optimize the hyper parameters and the topology of the models.

It is important to mention that these voting based systems are not meant for production use, even if the fact that they are not optimized is overlooked. The training of models for production use, unlike before, should not be divided into 20 splits from which each split excludes real data for testing, but instead all of the real data should be included in the training. The validation of these systems would be done with be done by collecting data from new participants.

In the context of detecting interaction obstacles, models that use less rounds are of higher relevance, as they make their prediction earlier. If only one of the 4 voting based systems was used, the one for 10 rounds with an accuracy of 80% should be the preferred choice. Making the prediction after 5 rounds with an accuracy of 72.5% seem less ideal. Same goes for the two voting based systems for 15 and 20 rounds. The system trained on 15 rounds is not more accurate than the system for 10 rounds and waiting 20 rounds before making a prediction is too long. By then the user might have already left the game, due to a bad experience caused by the interaction obstacle. However, it is also possible to use all 4 systems. Each 5 rounds could be predicted whether an interaction obstacle is involved. This could be especially usefull as the glare effect might only be an issue for a couple turn until the player reaches shade or enters a building. If predicted that the sun is no issue anymore, the eventual adaptiation could be removed again.

10. Conclusion

85 % etc.

10.1 Prospect

- es wurde zwar impolementiert nicht alle features zu verwenden, aber es wurde nie ausprobiert. Hätte man testweise machn können.
-Vielleicht hätte man ein weiteres statistical feature für penalties machen können, so wie es beim qualitätscheck der simulationnen berechnet wird.
- statistical tests could be used to validate whether the changed to the simluator iproved the quality o the simuations. Howevr, as shown by the classification results even though they seemed to improve the quality or at least match it, the classification results were noticebly worse when usidm the simulated games ceaed by the modified simultor. There ae many variables and factors that influnece the final result, which makes it hard to validate whether changes to the simulator improved the overall results, even when using statistical tests. - a user study could be done to validate the voting based systems in real scenarios. As mentioned in chapter **TODO: ref** the results shown in tabel **TODO: ref zu table mit voting ergebnissen** - real world sacenario 20 modelle ohne splits, offline test leace one out, online test future work (alle data 20 repitions) (vielleicht wird es sogar minimal besser weil man die 20 modelle auf 20 statt auf 19 daten + simulierte ratio trainiert)

- man hätte ausgehend von intialen trainingsergebnissen noch mehr trainieren können in die richtigunen, aber wegen zeitgründen nicht gemacht

- outlook in study: man fragt die leute ab wann sie etwas nicht mehr unterschieldoch wahrnehmen und berechnet dauraus eine neue bedeutung in der tabelle herleiten

- mehrfach simulieren und immer nur beste nehmen. Aber: sehr zeitaufwendig

- weiter forshcne ob synthetic images besser mit inacurate simulations klarkommen

- online test machen, hier war nur offline

11. Appendix

gucken wie man appendix richtig macht in latex

Bibliography

[Rab89] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Index

example intro, 3