Deep Neural Network Analysis of Face Expressions Linked To Emotions and Continuation Desire

By

Dines Rae Selvig

AALBORG UNIVERSITY
STUDENT REPORT

Medialogy
Aalborg University, Copenhagen

Supervisor: Henrik Schønau Fog

May 2019

Abstract

Prior research suggests and reveals that there is a correlation between human emotional responses and the subjective qualities of digital interactive experiences, using facial analysis done by deep neural networks propose a true non-intrusive way of measuring emotional responses and continuation desire of a player. This thesis proposes a tool to measure emotional responses across eight different emotions and continuation desire in real time of any game. The emotional recognition system achieves an accuracy of 98% and the continuation desire system achieves 93.3% accuracy. This forms a strong tool that shows a correlation between emotions and continuation desire, which can be used to evaluate digital interactive experiences, in critical stages of development of said content.

Aalborg University Copenhagen

Semester:
10. Semester Medialogy (4th Master)

Title:
Deep Neural Network Analysis of Face Expressions Linked To Emotions and Continuation Desire

Project Period:
01-02-2019 to 28-05-2019

Semester Theme:
Master Thesis

Aalborg University Copenhagen

Frederikskaj 12,

DK-2450 Copenhagen SV

Semester Coordinator: Stefania Serafin

Secretary: Lisbeth Nykjær

Abstract:

Prior research suggests and reveals that there is a correlation between human emotional responses and the subjective qualities of digital interactive experiences, using facial analysis done by deep neural networks propose a true non-intrusive way of measuring emotional responses and continuation desire of a player. This thesis proposes a tool to measure emotional responses across eight different emotions and continuation desire in real time of any game. The emotional recognition system achieves an accuracy of 98% and the continuation desire system achieves 93.3% accuracy. This forms a strong tool that shows a correlation between emotions and continuation desire, which can be used to evaluate digital interactive experiences, in critical stages of development of said content.

Supervisor(s):
Henrik Schønau-Fog

Project group no.:
N/A

Member:

Dines Rae Selvig

Contents

List of Figures

Preface

The following project is a mater thesis in Medialogy at Aalborg University Copenhagen. The thesis is written by Dines Rae Selvig, and supervised by Henrik Schønau-Fog.

The reference style for the thesis is Harvard AD style.

All links and cross references are hyperlinks, meaning that are clickable and will navigate you to the referenced spot.

The thesis have developed two algorithms to infer emotions and continuation desire from video data in real time, and the reader will first gain knowledge in the field of game testing, emotions, continuation desire, machine learning and measurement of physiological signals. The project consist of two iterations, first iteration is the development of the emotions recognition system, and the 2nd iteration will focus on the development of the continuation desire algorithm. All systems will be tested and results presented.

The AV production for the thesis can be viewed here: `https://youtu.be/mLPeqHa4xpA`

Abbreviations
FER - Facial Expression Recognition
LSTM - Long Short Term Memory
NN - Neural Network
DNN - Deep Neural Network
AI - Artificial Intelligence
QA - Quality Insurance
HR - Heart Rate
EEG - Electroencephalography
EMG - Electromyography
ECG - Electrocardiography
EDA - Electrodermal Activity
GSR - Galvanic Skin Response
SGD - Stochastic Gradient Descent
CD - Continuation Desire

Motivation

The motivation for this thesis is to create and develop a software system to evaluate test sessions in the interactive experiences development process, which can be used to measure both emotions experienced and continuation desire during a play session. Then use this data as evaluation metrics during development of media content such as interactive games or films. To get a deeper understanding of the user base, catch narrative bugs and irregularities in the game play mechanics.

Therefore, in my opinion, having a non obtrusive way of measuring emotions and continuation desire is of good value for the developer both during and after a developing sprint.

Currently many companies in the industry of films and games, test their content on test viewers or game testers and evaluate if they enjoy that content in terms of fun and engagement. But these tests is usually made with asking about how much they enjoyed the specific content or by observing the testee and see how they react to said content. This way of using obtrusive measures can bias the data, likewise a persons recall of one hour or more of experience is often poor. This usually bias the data, if only parts of the experience is remembered, and more specifically if only unsatisfying parts is remembered [1]. Leading to a need for a way to capture, process and store each individual experience in terms of bio-data, so a deeper analysis of the journey from start to finish can be used to evaluate the entire experience.

The use case for this type of software is vast, and can be used in many stages of the development process for reactions to mechanics of the game or narrative of the film. As a real time system can be developed, it can be used for adaptive changing content in interactive experiences, which is shown to increase the engagement of a user.

1 Introduction

The Game industry is a billion dollar industry, with video game revenue predicted to exceed over 112 billion US dollars in 2019[2], and continue to grow. Moreover the film industry with global revenue exceeding 285 billion US dollars[3], is also a large market, where content quality many times dictates the revenue. It is estimated that with an improved infrastructure for testing and removal of bugs or low quality scenes, millions of dollars can be saved [Sagi and Silvestrini, 2017]. The QA (Quality Insurance) department needs more tools at hands for testing, not only for bugs that makes a game unplayable but also for narrative bugs, which is an increasingly part of every game and is always present in films. In the current research, a vast majority of interactive experience testing revolves around using interviews and observation of play testers or test viewers to evaluate interactive experiences in terms of fun and engagement. Few attempts have been made to develop a framework for a more reliable testing using different data gathering techniques. Such data could give a deeper understanding of what a users experience during a test session. Many rely on heavy equipment and complicated setups, which is not always applicable for smaller studios to use, or even to replicate a specific scientific paper. As the rise of Machine Learning and computational power, it is now more accessible to use algorithms to find correlations in high dimensional data signals and give a better understanding of what these signals mean. Other than for testing purposes, such data could also be used for adaptive experiences in a real-time setting, such as changing the narrative or character behaviour or even lighting inside an interactive experience, to create a more tailored experience to the user.

When evaluating interactive experiences, one can measure, flow[Buchanan et al., 1991, Chen, 2007], presence[Lombard and Ditton, 1997], immersion [Jennett et al., 2008] or enjoyment [Ijsselsteijn et al., 2008], but one method of measure specifically developed for evaluating interactive experiences is continuation desire [Schoenau-Fog, 2011a,b, Schoenau-Fog et al., 2013, Schoenau-Fog, 2014a]. Continuation desire is the desire to continue playing/watching digital content, which evaluate the real metric for content to be popular, namely retention [Debeauvais, 2016, Vyvey et al., 2018].

Emotion recognition is the process to identify an affective state of a person, and is getting increasing attention from researchers in human-computer interaction [Seng et al., 2018]. Such as the use for gaming experience [Gam, 2009], mental diagnosis [Hea, 2011], driving safety [dri, 2011]. There is a vast amount of different data measures that can be used for emotion recognition such as audio, video and physiological measures. Using audio as emotion recognition have been used by researchers in different works, using spectral analysis as features to predict emotions in speech [Bitouk et al., 2010]. In terms of visual based recognition, different approaches have been explored such as older systems using binary patterns [Moore and Bowden, 2011, Shan et al., 2009], and image processing techniques to track landmark points [Tawari and Trivedi, 2013] and newer methods such as convolutional neural networks[Arriaga et al., 2017, Goodfellow et al., 2013]. Typically individual researchers achieve high accuracies, in the area of 90-95%, within their own testings, but it is a result of sparse data, and commonly systems learn specific individuals which is used in the experiment, rather than a generalised solution [Jang et al., 2011, Lee et al., 2005, Yu and Chen, 2015, Pollreisz and TaheriNejad, 2017].

In this thesis, the scope and focus is to develop such a tool to measure emotions and continuation desire both during development of interactive experiences and finished experiences. The measured data from a test session is then visualised to pinpoint emotional spikes and changes in continuation desire, these points can show elements in the interactive experience which spark these spikes.

The tool is equally relevant and applicable in adaptive interaction experiences, as an input to control certain aspects of the environment such as the narrative, character behaviour and mood lighting. The developed emotion recognition system achieves an accuracy of 98% trained on two million images, moreover a continuation desire system is developed which achieves 95.1% accuracy trained on 2.6 million samples. An expert interview reveals that a tool to measure emotions and continuation desire is applicable in the production phase of a game, and can give great value to the developers in early stages as well as later in the production phase.

2 Related Works

Creating personalised video games is drawing more attention from both players and game studios, as the research have found that stimulating the mind, with signals that create an emotional response is an import component of game design [Christy and Kuncheva, 2013, Picard, 2010].

Machine learning have also been used to predict emotions from different kind of biometrics, such as heart rate, video data, galvanic skin response, EEG, eyetracking etc. [Wang et al., 2018, Zhang et al., 2018]. It is a promising way for seamless and accurate measures of emotions during game play, and real time measures.

When working with different measures, both physiological and video feeds, one has to choose a subjective measure which best fit the way of evaluating the experience of play. [Schoenau-Fog et al., 2012, Schoenau-Fog, 2011a] shows a model for evaluating continuation desire, as a useful tool for game design and analysis of games [Schoenau-Fog, 2014b]. As of now, the connection between continuation desire and different biometrics have not been explored in any academic sense to its fullest, apart from [Selvig et al., 2018]. Which used Galvanic skin response, Eyetracking and Heart rate to determine the level of continuation desire a user was feeling, during game play, the project achieved an accuracy of 75%.

Outside of academia, there is multiple products to find related to emotion recognition, especially targeted affective gaming, and product analysis. A common denominator is using biometric signals and a processing method, either in terms of signal processing or using neural networks to model the data to an emotion. Typically in this space, arousal and valence is used to describe each emotion, instead of the actual emotion itself. This is due that valence and arousal is a continuous values which in theory can map any values to a direct emotion, as seen on the circumplex model [Russell, 1980].

Company	Input Data	Output	Features	Use Case
iMotion[4]	Eye Tracking, Facial Images, EEG, GSR, ECG, EMG	7 Emotions, Valence, Emotion Channels, Engagement	Emotion Recognition, Face Landmaks	Primarily Marketing
Modl.ai[5]	Virtual Player Behaviour Metrics	Competence, Autonomy, Relatedness, and Presence	Player Motivation, Optimise Player Experience, Retention, Track Player Base	Game Behaviour
Crowd-Emotion[6]	Eye Tracking, Facial Coding	6 Emotions	Product Response Analysis	Primarily Marketing
Sensum[7]	Eye Tracking, Facial Images, GSR, Laser, Radar, Sound	7 Emotions, Valence, Arousal	Complete Emotion Recognition	Primarily Marketing
Affectiva[8]	Video Feed	7 Emotions, 20 Expressions	Complete Emotion Recognition	Primarily Marketing

Table 1: Table showing different companies and their solutions in terms of input and output data

On table 1, different companies and their products is shown, and they all accept different input modalities from heart rate to electroencephalography (EEG). The most complete and complex setup is from Sensum, where they can provide tailored emotion recognition setups for your specific project, and use almost any kind of input, and output various data. Only one of the companies focus on testing within games and experiences, which is Modl.ai, where their products is within player experience, retention etc. But they use player behaviour inside the game, and does not measure any physiological signals. Therefore no company which specifically focus on the games or film industry, and especially not continuation desire, where the current gap is.

Based on the current research specific areas is extracted and elaborated upon in more detail in the following analysis section, explicitly Continuation Desire, Emotions, Human Behaviour Measurements, Machine Learning and Affective Interactive Experience and lastly a delimitation.

3 Analysis

From the related works in the previous section, the following section will clarify continuation desire as a qualitative measure for evaluating games, as well as which emotions relates to this affect. Then different emotional states, and how to measure such states with different measures, most specifically using video, also describing different expression and how humans show emotions. Then an overview of which machine learning algorithms that exits as of now to predict emotions, and lastly how these measurements can be used in affective computing.

3.1 Continuation Desire

Conation was first defined in the eighteenth century, as on of three parts of the mind: Affection, Cognition and Conation (Continuation Desire) [Hilgard, 1980]. Conation was then and still is defined as the desire and will to strive for a goal, and as this connection between knowledge and affection which leads us to act [Huitt and Cain, 2005]. Huitt describes conation as the following:

> "The personal, intentional, planful, deliberate, goal-oriented, or striving component of motivation, the proactive (as opposed to reactive or habitual) aspect of behaviour".

This means that conation is the intrinsic motivation that is displayed when attempting to achieve a goal through volition. [Schoenau-Fog, 2011a, Schoenau-Fog et al., 2013] redefines conation in the context of digital media, more explicitly video games as continuation desire, and formalises it as the player engagement process framework[Schoenau-Fog, 2011b].

Continuation desire is a way to describe a players engagement based on different triggers which cause players to engage in disengage from a interactive experience. The player engagement process, presents a comprehensive connection between four elements which forms continuation desire, objectives, activities accomplishments and affect[Schoenau-Fog, 2014a]. The relationship between these pillars can be seen in figure 1.

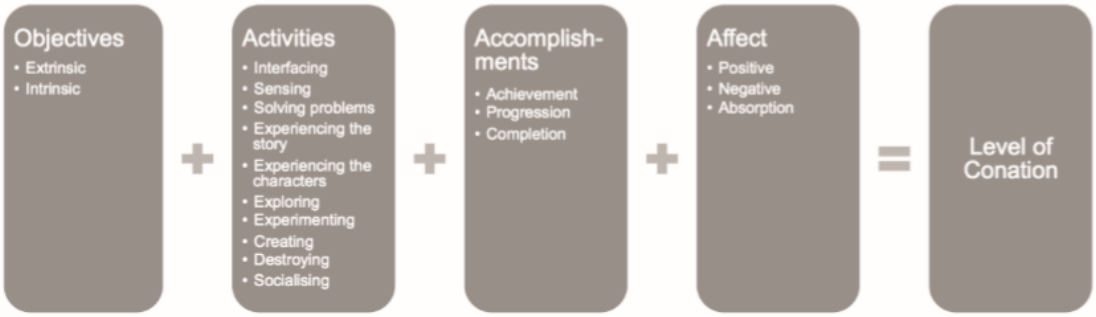


Figure 1: Causes of continuation desire add to the level of conation
Source: Schoenau-Fog [2014b]

Objective of a interactive experience is the extrinsic and intrinsic motivations a player experiences during play. These includes the explicit objectives of a game as well as any preconceived objectives that a player brings to the experience. [Schoenau-Fog, 2014a].

The activities define the ways a player can engage with the experience. These are both the ways that a game allows the player to interact with and experience the game. Therefore the activities are the different possibilities the game allow the player to pursue and experience [Schoenau-Fog, 2014a].

The accomplishments is defined as the rewards that the game can provide the player, they can be in different forms as receiving rewards, experiencing progression in the narrative, and completing objectives [Schoenau-Fog, 2014a].

Lastly the affect is defined as the absorption of the player, into some activity and the emotions which are experienced during play. The affect can also be described as the conscious or sub conscious emotional response experienced by the player. This response can cause a physiological, cognitive and behavioural reaction, such as facial expressions, rise/fall in heart rate, etc. Brett et al. [2003], Schoenau-Fog [2014a].

This can be translated into the positive or negative emotions which makes or breaks the experience, or in other words pull the player in or out of the experience.

These four parts of the Player Engagement Process in combination describes the level of conation a person is currently experiencing. The activities, objectives and accomplishments of a game comes from either the conscious design of the developer, which have designed the experience that the player experiences or act to specific items, narratives, etc. or it can come from the preconceived expectations of the game that a player can have.

These four parts of continuation desire, interacts sequentially with each other in cyclical rotation. This cycle is the player engagement process, and the process always begins with the affect of the player. The reason is that the player will initially have a form of intrinsic motivation before experiencing an experience.

The experience provide the player with different objectives, which is accomplish-able trough defined activities. These accomplishments can either lead to a negative or positive affect which makes the player continue or disengage in the experience Schoenau-Fog [2011b]..

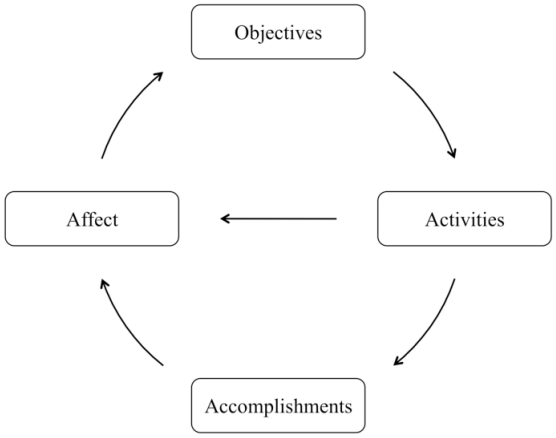


Figure 2: The Relation between Objectives, Accomplishments, Activities and Affect. (The OA3 framework)

Source: Schoenau-Fog [2011b]

The emotions which relate to affect can cover the whole spectrum of emotional responses, from low to high arousal and low to high valence, these changes in human behaviour can be measured and it is therefore possible to classify them using machine learning [Jang et al., 2011], using such a method it is possible to obtain continuation desire without using subjective measures. It is therefore not without basis that the emotions in the affective space of continuation desire can be measured

This is also supported by [Drachen et al., 2010], who found that there is a significant correlation between physiological measures and self reported measures revolving a players experience in a FPS (First Person Shooter) game.

Continuation desire is a combination of different aspects it is import to note that it is possible to measure the affective state of a player, as all actions will lead to a response, as seen on figure 2. Engagement have a close relation to continuation desire, in the next section engagement will be presented and discussed, and the differences between the two will be mapped out.

This thesis will use conation and continuation desire interchangeably, and the definition of the terms is the definition of continuation desire.

3.2 Engagement

Engagement is an element of the player experience which is highly important, and it is of utter importance for games to be engaging to be considered a good game. It is not enough to only motivate a player to keep playing, they have to be engaged to keep playing a game [Schoenau-Fog, 2011b]. Player engagement is one facet of the player experience, and can be closely related to flow[Buchanan et al., 1991, Chen, 2007], presence[Lombard and Ditton, 1997], immersion [Jennett et al., 2008] and enjoyment [Ijsselsteijn et al., 2008].

Flow refers to being in 'the zone', which is a mental state of mind, where a person is performing an activity, such as playing a game, fully immersed in a feeling of full involvement and enjoyment.

Presence can best be described as the sense of belief that the user has now left the real world, and is 'present' in the virtual world [Stanney et al., 2002].

Immersion, closely related to presence, often described as the acceptance of being able to be present in a virtual world. As such, removing as many real world sensations, and substituting it with sensations from the virtual world. Immersion is then by essence related to the multi modal nature of the perceptual senses [Mestre, 2019, Lombard and Ditton, 1997]. This can in other words be described as a state of mind of a player when the attention to the real world fades out when they focus on the game world. The player is drawn to the game world, and hours of play feels like 10 minutes.

A significant predictor of enjoyment, the perceived autonomy during game play which refers to the extent to which a player feels free to make their own choices [Zhang et al., 2016, Trepte and Reinecke, 2011].

By these concepts engagement is an intertwined concept and closely relates to all of them. Research by [O'Brien and Toms, 2008] has defined engagement as

> "A value of user-experience that is dependent on numerous dimensions, comprising aesthetic appeal, novelty, usability of the system, the ability of the user to attend to and become involved in the experience and the user's overall evaluation of the salience of the experience."[O'Brien and Toms, 2008]

Other researchers have defined engagement with terms such as presence, immersion, flow, affective dimensions, satisfaction [Amir Zaib Abbasi and Hlavacs, 2017]. Meaning that with these measures one can fully define and measure engagement. This can play a large role in evaluating video games, as engagement, enjoyment and immersion arise from a volition to experience the game.

In relation to continuation desire, engagement is entwined with continuation, as for an engaging player experience to occur, the player must have the desire to continue. According to [Brown and Vaughan, 2009], in the context of play, the desire to continue is a product of play and that the pleasure of the experience makes a player continue playing. [Schoenau-Fog, 2011b] sets the desire to continue in the context with player engagement, in the paper, Schoenau elaborates which characteristics of a players engagement that makes a player want to continue playing.

To be engaged, and feeling immersed or being in the 'flow', can be described by emotions, as often a person react to content with an emotional response. The next section will focus on emotions, and how to measure a persons emotional state and which cognitive processes and state can influence a persons emotional response.

3.3 Emotions

Over 130 years ago in 1872, Charles Darwin wrote in his book called 'The Expression of the Emotions in Man and Animals', the

> "Facial expressions of emotion are universal, not learned differently in each culture" [Darwin, 1872.].

Fast forward 130 years to statement as since then been much debated, however, psychologist Paul Ekman, who is a pioneer and renowned researcher in the studies of emotions [Steven J. Haggbloom, 2002] proposed six universal emotions, sadness, disgust, fear, anger, surprise and happiness [Ekman, 1964, 1992, Posner et al., 2005], see 3.

Figure 3: Figure showing the six different universal emotions put forth by Paul Ekman.

In both the field of psychiatric and neuroscience research, there is a general consensus that through evolution human have been supplied with a range of basic emotions [Posner et al., 2005, Ekman, 1992]. Each of the emotions is unique in the physiological and behavioural expression, and each of them emerges from a activation within the particular neural pathways in the central nervous system.

The emotional system is complex, and it can be described by the emotional continuum, see 4. A human, have three orders of emotion, the first order is the automatic processes with the human body, such as appetites. As it is bodily responses, which is a automatic process, that is hard to control by cognitive processes, and is a responses or reaction to a stimuli and is therefore categorised as uncontrollable responses.

The second order is basic emotions, which as mentioned, is the universal emotions: emotions, sadness, disgust, fear, anger, surprise and happiness. These emotions can also be translated to high/low arousal and positive/negative valence [Posner et al., 2005].

The third order is higher order emotions which requires cognitive process to be activated, these are emotional responses which is activated by complex neural signals. Higher order emotions can be pride, anxiety, remorse, etc. As these emotional responses is produced by complex signal and cognitive processes makes them significantly more subjective than the basic emotions. The third order is also tertiary emotions, which means that they require higher level cognitive processing, as the emotions is usually self-conscious, and require self reflection and evaluation.

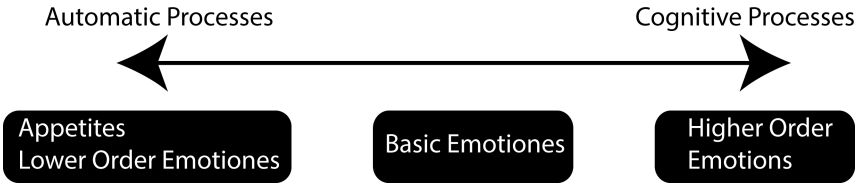


Figure 4: Figure showing the emotional continuum and the different emotion orders.

There exists six different states, which cover the phenomena concerning the emotional responses, as many emotions and responses is depended on the state of mind a person [Russell, 1980, Russell and Barrett, 1999].

Emotion refers to a 'brief' also known as macro or even micro expressions and is an autonomic brain and behavioural change, which produces a response in form of a facial expression. An emotion is often also a strong reaction to a stimuli.

Feelings refers to the subjective representation of an emotion. Typically a feeling can, and in most cases do, last longer than an emotion. As a feeling can persist over hours, an emotion typically does not last longer than one minute. A feeling have a lower intensity than an emotion.

Moods is the affective state of a person that can effect a persons emotional response. They are of a lower intensity than an emotion and feeling, but they last significantly longer, as moods can be active over multiple days. Moods often occur with no expressive signs and apparent cause, as they are of low intensity.

Attitudes is an enduring effect, which can be controlled by beliefs and preferences, such as religious beliefs and cultural differences. An example of this can be people from different levels in the society.

Affective style is the stable dispositions that can bias a person toward perceiving and responding to objects or people with a certain emotion or mood. Affective styles can be developed from previous living styles, closely relating to attitude. The affective style is developed throughout life and is changed slowly over time, adapting to the current living situation.

Temperament is typically a specific affective style that can be observed at a young age, suggesting that they may be determined by genetic factors. This is still a debated state, but it is typically referred to either genetic factors or how parents is teaching children at a young age how to behave can bias their temperament.

These six affective states could and in many cases will affect the measuring of emotional responses, both with self reported measures or measured with physiological measures.

As mentioned emotions can be segmented into arousal and valence, by having two axes a model can be made to plot emotions on a two dimensional system. Such as model was made by [Russell, 1980], called the circumplex model of affect, see 5. The model describes how emotions have both arousal and valence properties which relate to the neural circuitry.

Being able to concise emotions in a visual way and compare emotions on a mapped figure makes it more accessible to compare emotions. According to Russell [1980], Russell and Barrett [1999], Posner et al. [2005], is it possible to sub serve all emotions on the model, as valence and arousal cover all affective states.

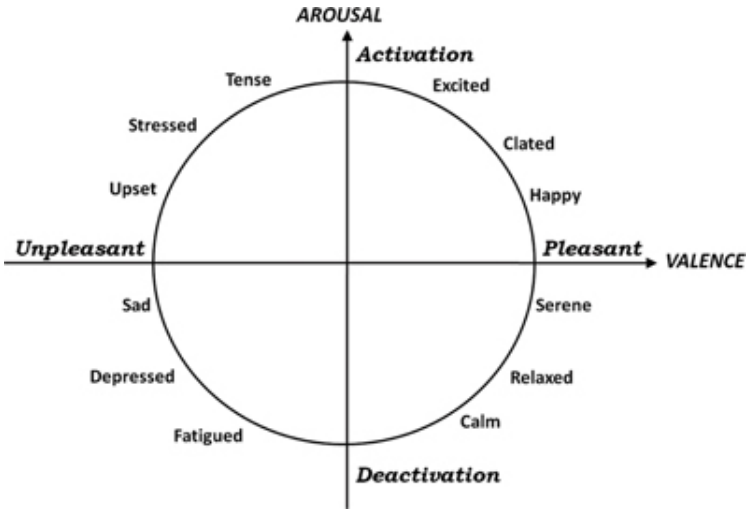


Figure 5: Circumplex Model of Affect [Russell, 1980] - Displaying emotions placed on a two dimensional system with the y-axis being arousal and the x-axis being valence.

Source: `www.emotiondevelopmentlab.weebly.com/circumplex-model-of-affect.html`

Researchers have for a long time written and researched the difficulties people assessing the describing their emotions Saarni [1999]. When conducting self reported measures of emotions, people tend to either not experience the emotion or recognise emotions when isolated from each other. [Cacioppo et al., 2007], describes that the psycho physiology as a field that is based on a assumption that emotion and cognition are embodied phenomenon rooted deep in the physical substrate of the body and brain. It must then be possible to measure characteristics of the human

body to elucidate and infer the understanding of the structures of third order cognitive functions.

Another model of emotion, is the OCC (Ortony, Clore and Collins's [Ortony et al. 1988]) model of emotion, see 6 (A), which is a widely used model, that states the strength of an emotion, based on events, agents or objects. Events, agents or objects is present in the environment of the person which is experiencing an emotion. The model have 22 different emotion categories, and consist of five processes that makes up the complete system. The model defines the process from which a character/person goes from an initial event to an emotion [DeLancey, 2009, Steunebrink et al.].

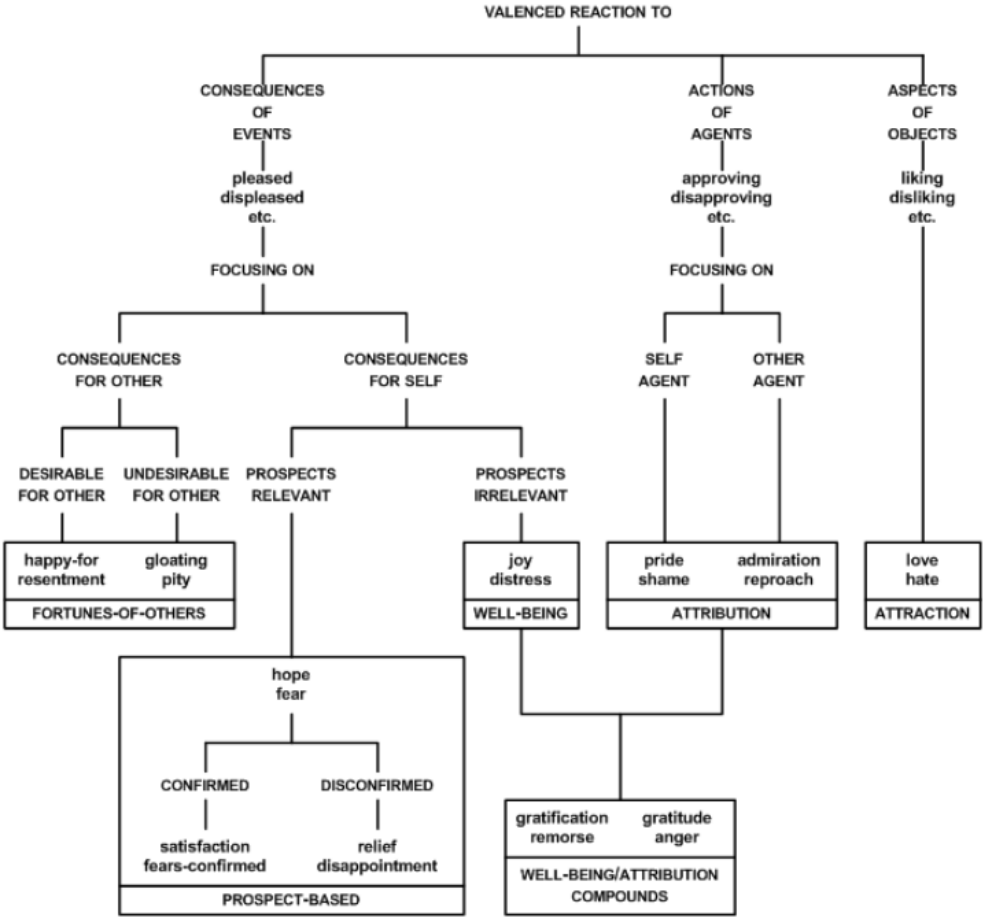


Figure 6: Original OCC model
Source: [DeLancey, 2009, Steunebrink et al.]

The model consist of different processes, and a list of the flow through the model is listed next.

- Classifying an event, action or object that is encountered.
- Quantifying the intensity of the emotion.
- Interaction of the generated emotion with the already existing emotion.
- Mapping that emotional state to an expression.
- Expressing the emotional state.

Firstly a person has to make sense of either an event, action or object that is present in the environment, then try to quantify how intensive the emotion should be, then the new emotional expression have to be merged together with the current emotional state that a given person is in or a compound emotion when an event is caused by an action of an agent. Then this compound or pure emotion, have to be formed into an expression, and lastly express the emotional expression [DeLancey, 2009, Steunebrink et al.].

The OCC model is different than the circumplex model, as the circumplex model does not generate a flow of how an emotion is formed, but rather maps out emotions based on valence and arousal. They both have different use cases, but can both be used to analyse emotional responses.

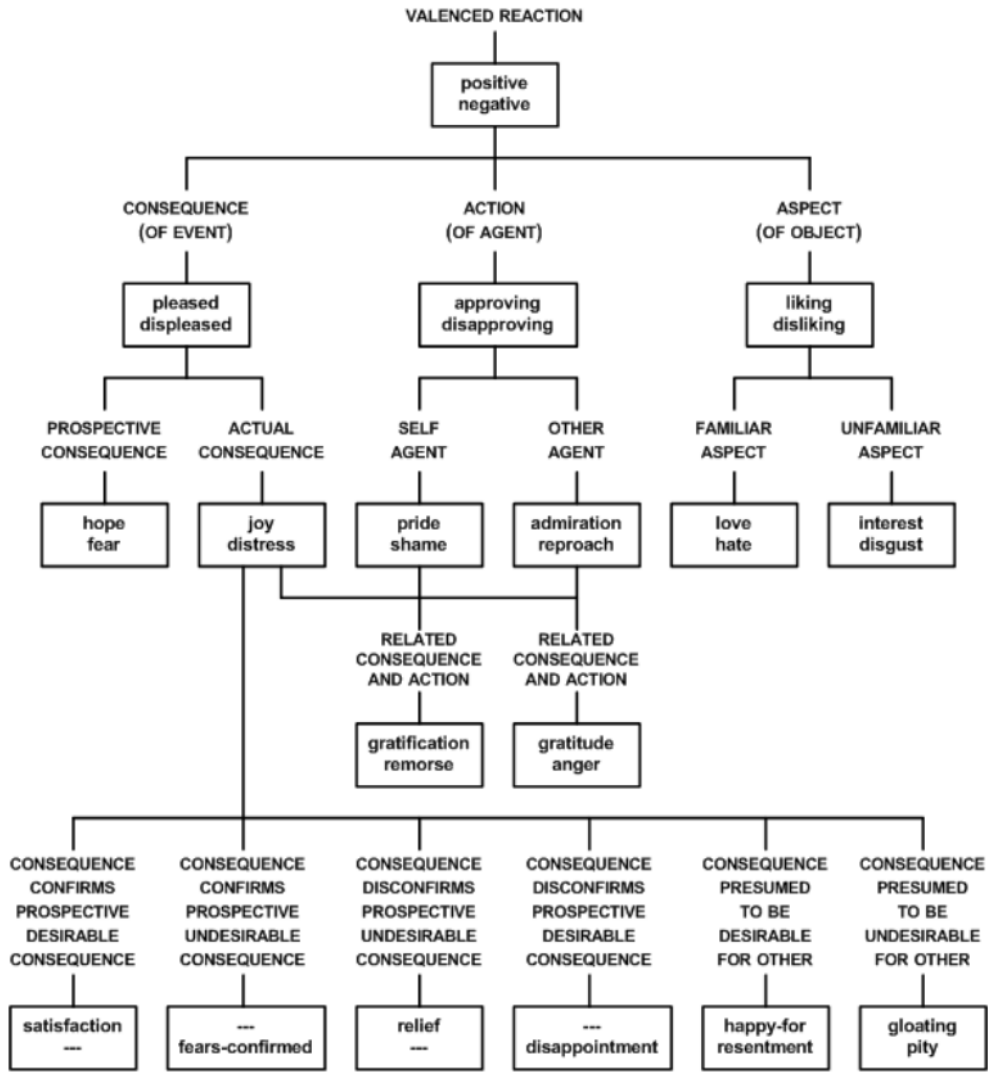


Figure 7: Altered OCC model
Source: [DeLancey, 2009, Steunebrink et al.]

The model is commonly used in controlling artificial agents, as the model is build up around logical states, and is therefore directly implementable in a logical system. On 7, an altered OCC model is figured, which is the original model transformed into an even more logical diagram, based on UML, as the process is as inheritance in programming. To see all emotional responses the model can generate, see appendix 11.1.

From this section it is found that emotion have roots in the physiological state of mind, and that basic emotions is closely to uncontrollable, meaning that it is possible to measure an emotional response. The basic emotions is also referred as universal emotions, which makes it possible for one system to measure emotions from every human on earth, as emotional responses should be largely identical.

Closely related to emotions is facial expression, the next section will focus on the visual emotional responses that occurs when a person reacts to stimuli.

3.3.1 Facial Expressions

As Darwin wrote, all facial expressions are universal, and even the debate that some could be culturally different, the basic emotions is universal to a high degree. From newborns to blind persons to the average person, all show the basic emotions in the exact same way [Matsumoto and Willingham, 2009]. This project concur with Darwin in some manner, that most facial expressions is universal, especially macro expressions, but micro expressions might have a slight variance depending on the cultural background of a subject. Two different kinds of expressions exists, marco and micro-expressions, which denotes both the duration and the area of the facial musculature that is used. Macro-expressions often last 0.5-4 seconds and involves the entire face [Ekman, 2004]. Macro-expressions is mostly single emotional responses and last said duration when the emotion is not being concealed, by higher cognitive functions. Macro-expressions occur mostly when we are alone or with friends or family, when we feel most open and familiar.

Figure 8: Image showing eight different emotions and their corresponding macro expressions. Source: RAVDESS Image dataset [Ryerson, 2018]

Micro-expressions is however different, as they last as little as 1/30 of a second and involves only small parts of the face. Micro-expressions is therefore arduous to see, especially for a human. They are often categorised as concealed emotions as they occur when two parts of the brain, the cortical motor strip and subcortical areas, oppose over the neural pathways to take control over the facial expression. This ends with quick fleeting micro expressions.

An evaluation test developed by Matsumoto & Hwang concludes that humans can on average classify micro-expressions and subtle macro-expressions with a 48% accuracy, and when both joy and surprise is removed, which is the easiest to recognise, it falls to 35%. This means that most people is not very good at recognising facial expressions, when the emotional response is short.

Several researchers have investigated how high cognitive load affects the response of an emotional expression, and usually high cognitive load tends to suppress emotional responses yielding a more neutral expression or of lower intensity. This means that, in relation to games, that during a challenging level, players might not express high intensity emotions in the moment, or even any. Expressions can be enforced by being in a social setting, when people playing a game together, expressions occur more often than when alone. Which mean being in a social setting, amplifies your emotional expressions as the context is more appropriate to express emotional responses [Frith, 2009].

From this section, measuring emotions from facial expressions can introduce challenges as expressions is depended on the environment and setting around the person. It is therefore important to create a environment and setting that facilitate responses in a safe manner.

Different methods exists to capture and recognise emotions, both measuring external signal, but also internal signals known as psychological measures. The next section will present different

kinds of measures, and what they can measure.

3.4 Human Behaviour Measurements

Measuring on humans have been a long lasting field, and it is still trying to map out how the human body works and reacts to different stimuli. According to [Cacioppo et al., 2007], psycho-physiology is based on the general assumption that emotion and cognition both are embodied phenomena which is rooted in the physical substrate of the human body and brain. The body and brain is both objects that can be measured and can therefore be used to elucidate our understanding of the relation between response of higher cognitive responses and stimuli.

This section will focus on presenting some of the most used methods to extract metrics about the human body. Such as external measures focusing on video feed and pose detection. Internal measure focusing on heart rate (HR), electrodermal activity (EDA) also know as galvanic skin response (GSR) and ocular events such as eyetracking.

3.4.1 External Measures

Video feeds have for a long time been used to analyse human behaviour in different situations, to pinpoint what different kinds of stimuli correlates with which responses. Especially the face of a person reveals vast amounts of information about the cognitive state of mind of a person. After all the face is the most expressive part of the body [Ekman, 2004, 1992, Frith, 2009]. But the face is also the one of the most complex systems, as many reactions is very situation based, making it difficult to correlate stimuli to certain expressions. Researchers have explored the field, mostly related to emotions recognition from video data, such as [Huang et al., 2018] which propose a end-to-end machine learning algorithm to classify emotions, from image sequences, achieving state-of-the-art results on the AVEC[9] data set.

A method for measuring heart rate without using on-body instruments, was developed to make a truly unobtrusive measure. It was developed by [Wang et al., 2018], and can measure HR using a frontal face camera, using colour-intensity or even motion based methods to measure it using machine learning, making it completely remote. The technique is novel and still not mature, making it unreliable in natural conditions for now. But in the near future, this method might be more robust, and usable in natural conditions.

Audio, is also a way to externally measure the state of a person, especially related to emotion, as voice frequency and patterns tend to change when we are feeling an emotion. [Huang et al., 2018], uses both image and audio data to correlate expressions from humans to valence/arousal of emotions. [Sin, 2015] uses pitch and energy as well as spectral features, to classify four different emotions, using support vector machines, resulting in a high accuracy of 95 % on their own dataset.

Emotion recognition from pose, is also a moving field, as information about ones emotional state can be inferred from the pose of a person. One paper by [Suj, 2015], uses 3D images of persons, excluding frontal to avoid using facial expressions, to classify emotions. The paper results in an 83 % accuracy, which is high for a pose-based approach.

3.4.2 Internal Measures

According to [Cacioppo et al., 2007] heart rate has been a long research focus of the psycho-physiology field, as its a behaviour which is easy to measure but also a complex system which reacts in a sensitive manner to neuro-behavioural processes.

A common measurement technique for HR is Electrocardiography (ECG), which uses electrodes placed on the body to measure the electrical activity around the heart region on the body, from this signal heart period and heart rate can be extracted. Heart period is the time between each heart beat, and heart rate is heart beats per minute.

This method is very intrusive, therefore another method was developed named photo-pletyhsmography, to produce the same data but using a LED and a transistor, to record the backsplatter from the skin instead. Making the measurement less intrusive. This is important as the more a person is aware of something being measured, the more the results can be biased.

Another highly used method is GSR, being a method to measure skin conductance, and was found that the skins conductance changes to different kinds of visual and auditory stimuli and dates back to 1880 [Cacioppo et al., 2007]. The skins conductance levels, higher conductance means larger amounts of sweat on the skin, relates to arousal [Egan et al., 2016], this means that it

is possible to, almost, directly link the conductance levels with arousal. The eccrine glands located at both the palm of hands and feet, respond to physiological signals rather than temperature changes such as the back, armpits and head glands. This means that for the measurement to be correctly measured, needs to be located at the palm or feet. A device to both handle HR and GSR, which is also rather unobtrusive is the Naos QG mouse from Mionix [10]. Such a device can help measuring physiological signals more subtle, than older devices [Dekker and Champion, 2007].

Different investigations by researchers have focused on linking HR and GSR with both valence and arousal, and have found that HR typically relates highly to valence, and GSR to arousal [Drachen et al., 2010, Lang et al., 1993, Frijda, 1986, Mandryk and Atkins, 2007b, Egan et al., 2016]. It was also found that these metrics can be correlated with the experience of play, more specifically to fun, immersion and flow. The methodology still needs robustness and verification, but the results are compelling towards that humans show and express enough information about their state of mind, to be able to measure accurately on higher cognitive functions.

Its important to note that when working with HR and GSR, sufficient baselines of the signals should be measured, as the baseline dictates the quality of the following data.

Eye tracking can also extract valuable data, as it was found that pupil dilation directly linked to a high cognitive load, and arousal [Bradley et al., 2008]. Another study [Selvig et al., 2018], found that pupil dilation also can be related to continuation desire, as arousal and cognitive load correlates with the level of continuation desire as the two metrics relates to the affect as well.

From this section it can be extracted that there is multiple ways of measuring data regarding the human body and brain, both from external and internal measures. They both vary in cost and system complexity, but occurs promising. Due to the scope of this project using the some accessible hardware from a general consumer, using video data to determine facial expressions seems like to most likely approach.

This data that can be obtained from such signals can be used to develop machine learning algorithms, to classify different cognitive state of a given person. More specifically emotions, which is both possible from video data, but also HR and GSR. From the previous sections there is now a link between continuation desire, emotions, and how to measure it. Continuation desires affect aspect is directly linked to emotions, which then again is measurable by different metrics which cover both internal and external signals, by this link it can be concluded that it is possible to some degree to measure continuation desire by measuring emotions through the use of for example video feed.

The next section will focus on the link between machine learning and external and internal data signals, and how to measure emotions from said signals.

3.5 Machine Learning

In the following section different machine learning approaches and methods will be presented and discussed in relation to facial expression recognition and continuation desire prediction. How machine learning algorithms learning and why it works will also be presented.

Classifying and predicting human emotions is a hard task, even for humans. Creating machine learning algorithm which can predict and classify such emotions is equally hard. As the era of machine learning is upon us, research in this area is moving at enormous speeds, also within predicting and mimicking human behaviour.

There is endless designs of neural networks, and usually they are tuned to the specific problem they try to solve. In the area of classifying human emotions, some network designs can achieve higher accuracy, as some methods converges and represents data in a more robust and accurate manner. Which methods to use is greatly depended on the structure of the data, and what type of data it is, as text and numerical data requires distinct different approaches.

Artificial neural networks is a supervised learning paradigm, which means that it needs the 'answers' to the data upon training. The network maps input to output, and it does that by finding correlations between the input variables and the output. A neural network is known as an universal approximator, because it learns to model an unknown function between the input and output. This means that in the learning phase, a neural network finds the right function to transform input (x) to output (y).

A neural network can be constructed in many ways. A model is constructed by layers, with each layer consisting of n neurons. The input layer is the first layer of the model, here the data enters the model. Next is hidden layers with neurons in each, and lastly an output layer. Each neuron has a weight and bias, which is the variables that is being tuned in training. This structure can be seen on figure 9. The amount of layers and neurons in each layer in a model varies depending on the the feature space of the input and of the complexity in the data structure.

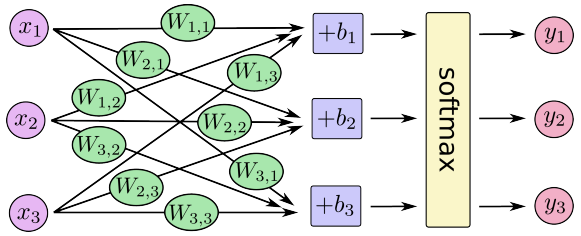


Figure 9: A simple fully connected neural network model, with one layer before the output layer. The figure shows neurons, neuron weights, bias, activation and output.
Source: [11]

When a layer is being presented to some data, neurons combine the input with the weight, which either dampens or amplifies the signal. These weights are being calculated by a gradient, which is tuned by the network it self with a method called back propagation, back propagation stands for 'the backward propagation of errors'. Different algorithms can perform back propagation, and they all use a method called gradient decent, which calculates the gradient of the loss function through automatic differentiation. This optimisation is the backbone in machine learning, and is the reason why an algorithm can learn from data.

The loss function can vary as well, but the loss functions role is to calculate an error value for how the model performs, which means that when a model during training makes many wrong 'answers' the loss becomes large. If however the model is having a large amount of correct answers the loss will be low.

Different gradient decent algorithms have been developed, one of the first is called Stochastic Gradient Descent (SGD) which was developed in the 1951 [Robbins and Monro, 1951]. SGD is still very much used, but newer algorithms have been developed which are more intelligent. Such an algorithm is Adam, derived from adaptive moment estimation. Adam uses an adaptive learning rate value for each weight in the network, instead of having a single learning rate value for the whole network as SGD uses. Adam, is using two different methods both build on top of SGD, namely AdaGrad (Adaptive Gradient Algorithm) and RMSProp (Root Mean Square Propagation). Using more intelligent optimisers such as Adam, a machine learning algorithm can learn faster, and much less likely to find a local minimum. Performance of different optimisers can be seen on figure 10.

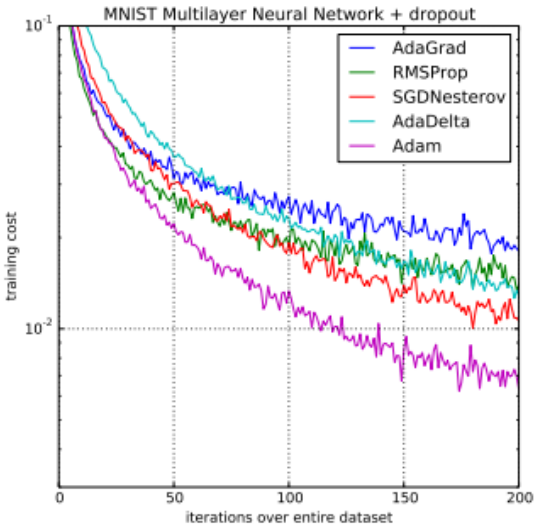


Figure 10: Comparison of Adam to other optimisation algorithms training a multi-layer perceptron.
Source: [Kingma and Ba, 2014]

On figure 11, a 3D plot of the loss landscape is visualised. It can be seen how the a gradient decent algorithm optimises the loss of a machine learning algorithm. The goal of a any optimiser is to find the minimum in the loss landscape, as seen two routes is present, one finding the global minimum, and one finding a local minimum. It is always the goal to find the global minimum as this hold the lowest loss, meaning that the network can achieve the highest accuracy.

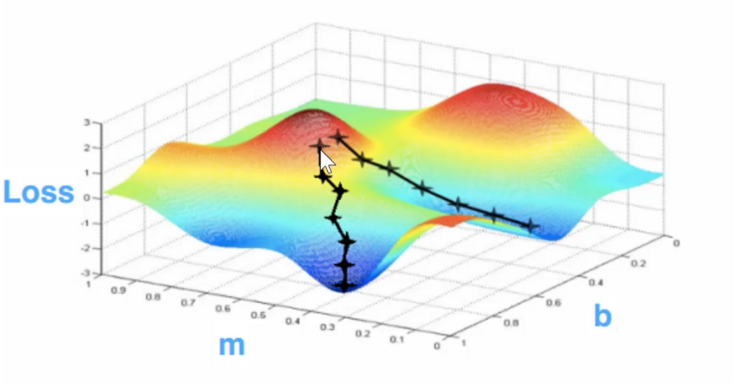


Figure 11: Gradient Decent Algorithm Visualisation. Plot showing how the gradient decent algorithm, finds a minimum in the loss.
Source: Jeremy Howard @ Fast.ai [12]

When working with images, or image sequences (video), convolutional layers is commonly used to extract features of each image. As images are 2D, having a width and height, 2D convolutional layers is used, which takes a kernel and literately processes the image. The task of convolutional layers is as said, to extract information from the picture, and in most cases it extract lines, shapes and texture patterns from the image, so the essence of a convolutional layer is to extract feature patterns from a picture. in terms of FER, the kernel will learn to look for specific lines in the face, such as eye placement, eyebrow angle, mount width and angle, as these landmarks usually relates to a specific expression.

On figure 12, it can be seen how a convolutional network processes an image of a car to output that the image is of a car. As seen, there is a series of different layers, conv which is convolutional layers, ReLU activation layers and Pooling layers.

Convolutional layers consist of filters, where a convolution iteratively processes the image, for each filter a feature map is produced which is the dot product between the convolution kernel and the image. A kernel can be any size but is typically in region of 3x3 pixels. This means that

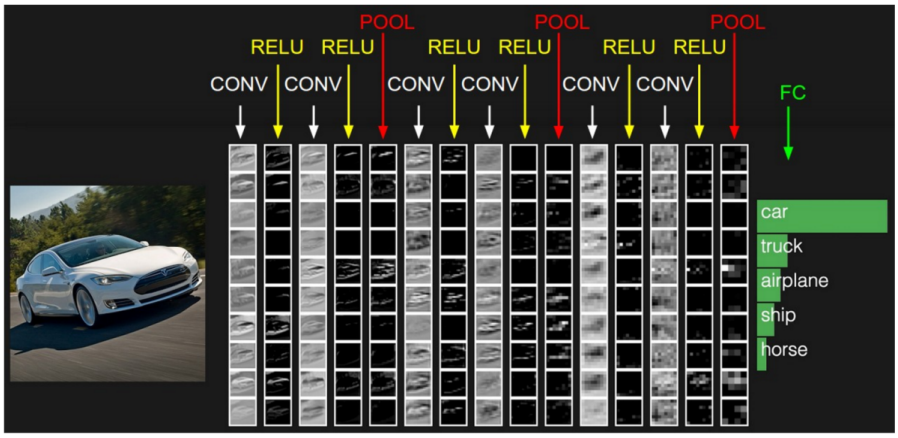


Figure 12: A convolutional network processing and image of a car to output that the image is a car.

Source: [13]

convolutional layers extract important information from an image, why they are also known as feature extractors.

Activation layers, is a function which 'activates' a signal in a certain way, there exists multiple activation functions, but a popular one is the ReLU function. ReLU maps all negative values to zero while all positive values is kept. The reason for using activation functions is that the network would without it just be a linear function (Linear Regression), as we only multiply and add values. A linear function cannot solve complex data structures and find complex correlations, therefore an activation is needed to make the function non linear, as a Universal Function Approximator (Neural Network), needs this complexity to work with for example images, audio, large dataset, etc.

A pooling layer, pools values together from a kernel, this is an important step in Convolutional networks, as the features needs to be extracted to higher level representations. Another reason is that the image needs to shrink from the input to output as this saves computational resources needed, as large images is not needed downstream in the network structure. A pooling layer takes a kernel of a size, such as 3x3 and output one value from these nine values, this means that the image is shrunken three times on both the height and width of the image.

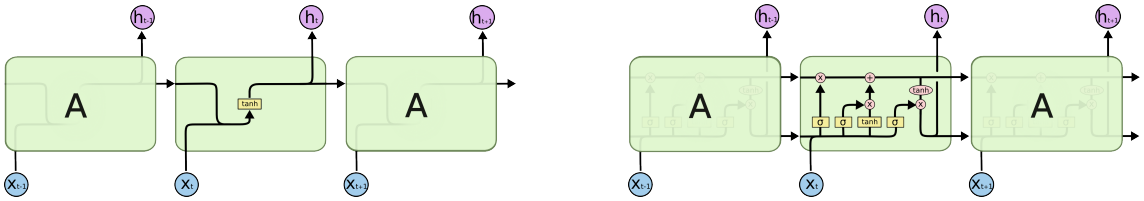
In the image columns the extracted feature maps is visualised, and it can be seen what the algorithm sees throughout the network. It can be seen that the further that the image travels through the network, the images changes significantly, and becoming increasingly higher levels representations of the input image. It is also seen that what becomes apparent is the texture patterns and the lines which is represent in the image. The outline of the car is quickly drawn, while the background of the image is filtered out.

As seen in the emotion section 3.3 on figure 8, there is a distinct different in landmarks across the face, such as mouth angle, eye shape, and eye brow angle. It is also seen that some facial expressions is closely related in expression but opposite on the emotional space. An example is calm and neutral which is both closely related in the emotional space, and in expression. but angry and disgust is also close in terms of expression, but both negatively charged.

For evaluating conation, which is evolving over time, and changing depending on past events, temporal data is needed for processing.

When working with temporal data, meaning that there is a relation between each time step in a sample pool, recurrent units is often used to process data in relation to each other. Normally, fully connected layers and convolutional layers, only process one sample at a time as these layers does not expect any relation between the current and the past sample. Recurrent units, will process batches of data in relation to each other. See figure 13.

The problem with recurrent networks is that it only remembers the past input, this is good enough for many applications, but if the relationship between the data is important over 100 samples, this approach will not work. [Hochreiter and Schmidhuber, 1997] proposed an improved version of the recurrent model, with a type called Long Short-Term Memory (LSTM), which can



(A) (B)

Figure 13: (A) Recurrent Neural Network (RNN). (B) Long Short-Term Memory (LSTM)
Source: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/#fn1`

remember relations between samples up to 1000 time steps. This is a immense improvement over recurrent networks. LSTM also learns much faster and can tackle complex problem with long-time-lags which no recurring units can [Hochreiter and Schmidhuber, 1997].

A study by [Zhang et al., 2018] used EEG data and a LSTM model to predict emotions with a 90% accuracy. The network was constructed with a quad-directional spatial LSTM layer, after this layer a bi-directional temporal RNN layer is used to spot more subtle emotion patterns, lastly a softmax layer creates probability for each class.

Another study using LSTM for emotion recognition is [Chao et al., 2015], this network predicts continues arousal & valence values using EEG, GSR, imaging. The accuracy for this approach ended at 60%.

Two projects using a video feed for facial recognition to classify emotions, have achieved high accuracies, +80% range. One using a Kinect [Mao et al., 2015], another a normal web cam [Ouellet, 2014], both state that it would be possible to use in real time, but no such tests is conducted in the studies. Leaving the question running in real time unconfirmed. A common factor is that the dataset is often small, and consists of only a few people, meaning that the algorithm is learning specific responses relating to the person. This means that the algorithm is not learning a generalised way of predicting emotions, which will make the algorithm useless in a real environment. Therefore it is important to have as many different samples as possible in the data set.

The problem as of now is that the field is testing different algorithms and methods but there is no universal protocol or established model yet, so all results is very independent and specific. This is a problem recognised by [Jang et al., 2011, Lee et al., 2005, Yu and Chen, 2015, Pollreisz and TaheriNejad, 2017] and this needs to be addressed before emotion recognition can be fully used in a natural setting.

From this section it can be extracted that for image/video analysis an algorithm consisting of convolutional layers is needed to process images, while using Adam as optimiser. These methods is sufficient to create a facial expression recognition algorithm to extract emotions from images. As mentioned previously continuation desire is a feeling that can fall and rise over time, meaning that it was a temporal dependence. To process this kind of data, an algorithm needs to remember long spacial relations, which the LSTM unit can solve.

Data extracted from an algorithm such as emotion recognition, can be used in real time to control different aspects of an interactive experience, such as NPC (Non Person Character) behaviour, mood lighting, narrative adaptation. This leads to the next section about affective interactive experience, using human physiological signals.

3.6 Affective Interactive Experience

Affective computing is no new field, in fact it was been around for decades, but mostly in the form of a concept. As true affective experience is still not applicable. Rosalind Picard, which is credited for starting the whole field of affective computing, states that "All computing that relates to, arises from, or deliberately influences emotions or other affective phenomena" is in the field of affective computing [Picard, 1997].

As stated by [Hudlicka, 2003] "Affective gaming means adapting to the player's emotions, to minimise frustration while ensuring a challenging and an enjoyable experience." This can be illustrated by figure 14. Showing a person interacting with an experience, the person is then reacting to said experience in an emotionally way, which then again affects the physiological state of the human body. This reaction from the human body can be measured with HR, GSR, EEG, Video data, eyetracking, etc. These signals can be analysed which leads to what kind of emotional state the person experiencing is currently in. This state can then be used to alter the experience to create a deeper connection between the person and the experience.

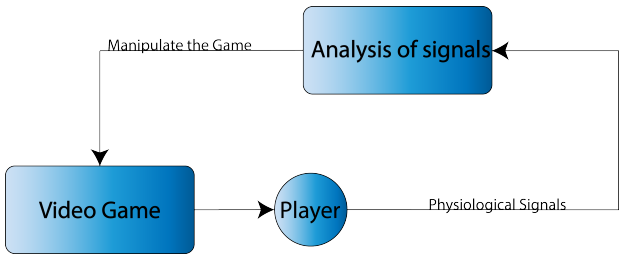


Figure 14: Model showing an affective gaming system.

Affective experiences still needs more development before it can create an emotional impact on a person, as stated

> "The problem is that conventional video games do not have the technical possibility of assessment to the player's emotion beyond the simplistic control instruction. For it to truly be able to be an affective video game, the system needs to be able to recognise and "sense" to a certain degree the player's reactional/emotional state more precisely, and directly." [Modic, 2017]

These difficulties is also mentioned by [Christy and Kuncheva, 2013], including the same uncertainty on what to measure from a person, as different persons may react emotionally different, and what kind of equipment to use.

For the commercial market to adopt affective computing in interactive experiences equipment needs to be developed that is reliable and easy to use.

A couple of application have been using affective feedback loops in a game setting. One using HR, EMG and GSR, processed by fuzzy logics to match input to valence/arousal levels, real time during gameplay [Mandryk and Atkins, 2007a]. One similar application used GSR and HR to create a game focused on relaxation, where the person playing, had to focus his/her calm levels to proceed in the game. With training, a person would be able to control calmness [Munekata et al., 2012]. Yet another application used HR to control the game in three different affective states. More precisely: 'Assist Me', 'Challenge Me' and 'Emote me', the person would then shift between these states to adaptivly change the experience [Mark et al., 2005].

The research is focusing on vastly on which psychophysiological hardware to use for the moment. Showing that the need for the right hardware is there. Different types of hardware is already available in the commercial market, such as a portable Electroencephalography (EEG) headband from Myndplay[14], a optical mouse from Miomix[15] capturing GSR and HR, and lastly eyetracking from Tobii[16], which now can be integrated into the computer screen. Currently 100+ games [17] is using the eyetracking from Tobii, but only as an input device to control movement, visual effects etc.. As this project focuses on using a video feed, to determine emotional states of a person experiencing digital content, the need for a special developed hardware product is not needed, as most individuals have a web camera, in either their laptop, external web cam or in their smart phones. This eliminates the need to find a product which then needs to be adapted into the market, which vastly improves the use case for this product.

This concludes the analysis, and in the next section a delimitation of the scope for this product will be presented, leading to the final problem statement of this thesis.

4 Delimitation

When designing engaging experiences, levels of conation is a good measure of player experience as the PEP is a cohesive model, rooted in affect. Affect correlates closely with valence and arousal which makes up emotions described by the circumplex model. Therefore if measuring emotions, either with biometric data, or a video feed, it is possible to determine levels of continuation desire through the use of said data. To measure emotions truly unobtrusive, using a camera to capture facial expressions is the optimal way, since other biometrics such as heart rate and galvanic skin response require equipment attached to the testee. The setup can quickly become complex and expensive, and the scope of the project is to create and test a software system which is accessible and applicable for smaller development studios.

In terms of facial expressions during high cognitive loads, where faces usually become very neutral, this project seeks the co operational play style in games to enforce interaction, even through challenging play levels. A neutral face may occur, and it is not said that, that does not relate to conation. Both is determined usable to infer conation from said data.

Using faces as features, it is optimal to use machine learning, especially deep learning to extract deep facial features during experiencing digital content, these deep features can both be used for emotion recognition but also to determine the level of conation. This could be a powerful tool for both affective and adaptive gaming, for changing narrative behaviour, but also for game or film evaluation, to investigate the underlying emotions and continuation desire during a game or film. The use such a system for adaptive system, the developed recognition system should be able to process data in real-time, therefore the developed system needs to be lightweight in terms of compute power needed, and usable cross platforms for PC, Unix based systems and mobile operating systems.

With this scope, a more precise problem statement have formed, and is as follows:

> **To which extend is it possible to predict a players continuation desire and emotions during an interactive experience, using only a video feed as input in real time? And is this data driven approach use-able when evaluating an interactive experience?**

5 Methodology

The following section will argue for the choice of methods, what informed the choice of test and a final list of requirements for answering the problem statement.

The methodology for the project is both deductive confirmatory and exploratory research [Patton, 2001, Easterby-Smith et al., 2002], as this thesis build on top of existing technologies and methods of how to perform emotion recognition from facial expressions, and which methods is best suited for this, but at the same time exploring novel approaches, especially related to continuation desire prediction, as this area is new, and unexplored. This thesis does confirm usability of other methods and approaches, but explores the field, and introduce new use cases and novel techniques to solve problems in emotion recognition and present new ways of using continuation desire and deeper insight during development of interactive experiences. The philosophical approach is of positivism as the theory builds on top of what can be measured, and the outcome should be reproducible by other researchers [Patton, 2001, Easterby-Smith et al., 2002]. The thesis, uses experimental design and a case study as the strategies to test the proposed theory and hypotheses. These approaches will be tested with a mixed-method approach, both using qualitative and quantitative data to support the proposed hypotheses [Patton, 2001, Easterby-Smith et al., 2002]. Both studies is cross-sectional time frames, as no long testing period is needed to produce data for the analysis. Later in this section, the different testing scenarios will be explained, and the used methods and data analysis methods will be presented and discussed.

Listicle of Requirements
In order to answer the problem statement, the problem statement is broken into parts which needs to be present to be able to answer it successfully. A list of requirement is formed:

- *A system to capture and store video data and real time emotion predictions of minimum two people*
- *A system of predicting the users level of Conation, both post play and in real time, based on the collected video data*
- *A system to visualise and present data in a usable format to the game tester*

The thesis will use the iterative design process to develop and evaluate in two iterations.

The first of two iterations in this thesis will focus on the development of a classifier to predict emotions based on facial features from video data, and collecting data for the development of the machine learning algorithm for predicting continuation desire.

The second of two iterations will focus on validating the produced algorithms in terms of accuracy in a real environment, this means that the algorithm will be used to classify emotion and continuation desire and this data will be compared to the users own self reported data. This can validate the produced model on how it would perform in a real scenario. Also a qualitative evaluation of the data produced from the algorithms in relation to game development testing will be conducted with a case study with a game studio. The test will focus on how a system as the one developed in this project can be used, and if it is even feasible to use a data driven approach.

In the following sections three different test procedures will be presented, to both gather data, but also test the system and see if the final problem statement can be answered in a significant manner from the tests conducted.

5.1 Test Procedure of Data Collection - Test one

The first test, is not a test of the system per se, but to collect data for the continuation desire classification algorithm, which needs data of people playing a co-op game and their emotional response during game play.

The game chosen for the test, is a Play Station 4 game which is called 'Little Big Planet'[18]. The game is a platformer in semi 3D, which can be played in a co-op setting. As discussed in the analysis, using a co-op or multiplayer game where there is direct interaction, either playing together, or communication through voice, tends to make players more impassioned and expressive about their emotional state.

The test uses convenience sampling as students close at hand is the most accessible test subjects, its a non-probabilistic sampling method, but it is assumed that the subject falls within the acceptable boundaries of what is needed for this test. As this test does not require any skill sets prior to playing a game, it is argued that the test subjects background is of non importance.

Before the test begins subjects will be asked to fill out an consent form explaining the data gathered and usage thereof, see appendix 11.8.

Next a demographic questionnaire to gather information about the participants were conducted electronically which included the questions:

- Age - Numerical Response.
- Gender - Multiple Choice
  - Male
  - Female
  - Prefer Not to Say
  - Other
- Occupation - Free Text.
- How many hours a week do you spend playing video games? - Multiple Choice
  - 0-3 Hours a Week
  - 4-6 Hours a Week
  - 7-9 Hours a Week
  - 10-13 Hours a Week
  - 13+ Hours a Week
  - Other
- Favourite Video Game Genre - Multiple Choice
  - Action
  - Adventure
  - Strategy
  - RPG
  - Indie
  - MMO
  - Simulation
  - Sport
  - Point and Click
  - FPS

- Other

The questions was made to get an indication of each participants background in games and to get data on age and gender for later data analysis and to visualise distributions.

After the questionnaire an introduction to the test was conducted. The introduction was introduced the same way with the following text.

> "The test you are about to start, is to gather data regarding how people respond to gameplay in terms of emotion and continuation desire. You are to play the game called Big Little Planet 3 on PlayStation, while playing I will ask every five minutes about your desire to continue rated on a scale from one to seven, where one is do not want to continue, and seven is want to continue playing. You are to answer truthfully. You will be asked individually so your teammate can't hear your response. While playing a camera which is situated at the top of the screens will record your reactions to the game during your play through. You will be playing for at least 30 minutes, and after that you are free to stop at anytime, but free to continue as well."

After the introduction to the test, the subjects will start playing from the first level, first getting an introduction to the game trough introductory levels. As mentioned the continuation desire reporting from the participants was made isolated from the other participant to avoid the participants being biased by each others responses. The continuation desire was asked just before subjects started playing, and then every five minutes throughout the testing session.

After the test, the participants were free to go, as no further questions was needed to be answered.

The data from this test will be used to develop the algorithm to infer continuation desire from emotional expressions from the video data. During the tests, the video data will both be stored locally, and the video will be analysed in real time to extract facial expressions from the participants and stored locally as well. This data is to be used in the 2nd iteration for the development of the algorithm. During the 2nd iteration correlations between the data points will also be investigated to see which emotions relates the most to continuation desire.

5.1.1 Test Setup

The test setup used can be seen on figure 15, showing nine Samsung 55" 4k display (UD55E-B) forming a video wall of 165". The PlayStation 4 is attached though HDMI, and two standard issue PlayStation wireless controllers is used.

(A) (B)

Figure 15: (A) Test Setup, play screen, camera and computer for recording. (B) Same setup, used during play testing

As seen on figure 15, the camera is placed on top of the lower middle screen, the camera is a standard issue web camera from Logitech filming in 720p30fps (Logitech C525 Webcam[19]), the video output is sufficient for the purpose of the recordings. The video feed is captured by Open Broadcaster Software (OBS [20]) on a desktop with an i7-7700k, Nvidia GTX1080TI, 16GB RAM running Windows 10. All recordings is saved in a mp4 file format.

5.2 Test Procedure of Case study: Six-player Video Game - Test 2

When the algorithm have been produced from the 2nd iteration, the algorithms accuracy in a real world scenario and the usage of produced data will be tested and evaluated in collaboration with a company called Invisible Walls[21], which is currently in the development phase of their new game called Cainwoord[22], which is a multiplayer game with up to six players. Invisible Walls is also behind the game Aporia: Beyond The Vally[23], which is a single player, story/puzzle game.

Figure 16: Concept keyart from Cainwood
Source: Invisble Walls [24]

To describe the game and what the objective of the game is, is best explained by the intro from Invisble Walls:

> "With Cainwood we want to invoke a super strong social experience, filled with trust and deception. Players are working together, but a few players are not who they seem and are actively working against you. With Cainwood we want to create situations where you can never fully trust your companions, but you will have to trust someone in order to progress through the game. This means that you constantly have to look over your shoulder, uncover information on players and in the end, you will have to figure out who the Synth players are to win the game. We aim to create a rich playground of possibilities fueled by social interactions with other players. This will result in a new player experience each time you press "find a game"."[25]

The game starts with six players spawning in pairs, two of the six players is evil while the others are good. The job of the evil players is to sabotage the other players in the game. The evil wins if all good players is eliminated, and the good wins if all bad players is eliminated. Elimination can either occur by being killed by a weapon or strangulation, or you can be voted off by other players during a voting phase. Communication in the game is done by voice, and the mechanics of the game enforces the need for communication as many of the puzzles requires two or more players to complete. As said in the analysis using a co operative game can enforce subjects to express emotions more vividly than regular single player games. Choosing to test on Cainwood is a direct derivative of this research.

Two different test sessions conducted by Invisible Walls will be used to evaluate the algorithm, both consisting of six players, in three to four play throughs of the game. The length of the play through will vary per player as the length of the game is not fixed, and you are able to be eliminated by other players.

In each session both gameplay and the face of the player will be recorded using OBS (Open Broadcaster Software)[26], and saved for data processing after the session. Each computer used use approximately the same hardware, and all of the cameras are the same. The camera is located at the top of the screen, giving a good camera angle to capture facial expressions.

5.2.1 Interview

The test session, will use the data collected, and the corresponding analysis, to facilitate an interview about game testing, and the possible need for deeper insight into the players reactions to the game. The interview will use a mixed method approach between informal conversational interview and a standardised open-ended interview [Patton, 2001]. The informal conversational will give flexibility to touch upon subjects or interesting areas of conversation, which could be of value, but without asking specific questions, but few questions is needed in a specific way so a standardised open-ended interview approach is used as well, to have exact worded questions, but still be able to ask elaborations on answers this is also know as probing, and will be used throughout the interview to stay on point, but still be exploratory [Patton, 2001, Easterby-Smith et al., 2002].

The reason for using this approach over a structured interview or questionnaire, is to explore and adapt to the interviewee's responses, and explore different angles relating to the answers, and this could lead to views or use cases which were unforeseen. As this interview will only happen once, a questionnaire would not give enough sufficient data, but would in return give more structured data.

The goal of the expert interview with guidance from [Bogner et al., 2009], is to understand current testing methods in an established game studio, and to see if tools that provide deeper insights in the testees is of value, especially if the system is easy to use. It is also to understand the use case of tests in a game studio as well as what they use from current methods to further develop their game.

The interview guide, in form of questions in an semi-structured setting:

- Q: What is your current testing methodology.
- Q: What do you use your results from a testing session for?
- Q: Do your current methodology encompass everything you seek from your test? Do you reach your testing goals?
- Q: Do you need or think that there is a need for deeper insight into a player in a play test.
- * Present Data *
- Q: Seeing this data, is there some areas that surprises or interests you?
- Q: Does this kind of data and analysis give you a deeper insight into how the players react to the game?
- Q: Do you believe that using this data driven approach could enhance your game testing?

All questions is either knowledge questions or background questions, which is worded to extract as much knowledge from the interviewed subject as possible.

The first half of the questions before the present data stage, is to gather information about their current testing methodology, and understand what they are testing to achieve, moreover also to get an understanding of their tests in the future as well as getting an indication if more and deeper insights into their test participants is needed.

At the 'present data' stage, an analysis of three players will be performed on data that I have received from Invisible Walls before the interview. The data consists of game play and frontal video of each participant. The video will be analysed by the system developed in this project, and graphs will show emotional responses and continuation desire over time. At this stage the results will be discussed as well, in relation to the questions asked.

Next questions is about the data outcome and if such data can be used in the development process of a game, and how it can impact the way games are developed.

The interview will be conducted with Sebastian Hurup Bevensee, Creative Director at Invisible Walls, and currently managing and designing Cainwood. He has a vast knowledge in game testing, and game development, and will therefore provide a substantial knowledge about this area, which can yield good responses to the questions formed in the previous list.

The interview session will last approximately one hour, and will be video recorded for further data analysis. The qualitative data will be analysed with regard to the final problem statement, to see if answers can support the need for the system developed.

5.3 Test Procedure of Model Validation - Test 3

The two machine learning algorithms developed in this thesis, namely the emotion recognition system, and the continuation desire predictor will also be validated to access its accuracy in other domains from where it was developed. This validation will be done by another test session similar to the data collection test session, where another interactive experience will be used, and cross validation between the reported continuation desire and the predicted values will be done, to analyse the true accuracy of the system. To analyse the true accuracy of the emotion recognition system, sequences of images of the session will be validated to see if they correspond to the recognised emotion of the system.

The game which will be used for this validation test is a simple 2D single player game called 'There is Only One Level'[27]. As this project is focusing on cooperative game play as these games makes emotional responses more apparent, it is interesting to test whether single player games works as well. Therefore this particular game is used.

To start out the test, the same procedure as the data gathering procedure in test 1 will be used, first signing the consent form, then filling out the questionnaire with the same questions as described in test 1. The method for the players to report continuation desire will be slightly different, as this game consist of levels which is completed at a fast rate, another approach will be used. Instead of asking every five minutes, the players will be asked to report their continuation desire level at each finished level, see 2. More explicitly, the players is asked at each completed level to set a cross at which continuation desire level they current are.

Continuation Desire	1	2	3	4	5	6	7
Stage 1							
Stage 2							
Stage 3							
...							
Stage 15							

Table 2: Table showing the tabular sheet the self reported measures is reported on in test 3.

After the testing of at least 15 subjects the validation of the data produced will be conducted. The reason for having at least 15 subjects is that to get a as large a range as possible of different subjects in terms of expression behaviour, play skill-set and the larger the sample the more accurate the validation becomes.

The validation of the continuation desire will be done by comparing the reported continuation desire values with the predicted values, if the validation achieves a comparable accuracy that lies close to the original validation performed during training, the system can be said to have achieved good generalisation of the data, and is therefore transferable to multiple games as well.

The emotions predicted will also be validated and will be done by manually going through every 30th frame of the video data corresponding to one frame per second and tag that frame with an emotion. During this process the validator will not have access to the recognised emotions by the algorithm to achieve as little bias as possible. After the video have been analysed the emotions manually registered will be compared to the recognised emotion by the algorithm, this will yield an accuracy of the emotion recognition system, which as the continuation desire predictor, will be the true accuracy in the real world.

This data will be used to quantitatively accept or reject the problem statement in regard to of which extent a system can be developed to recognise emotions and continuation desire of an acceptable degree. For the algorithms to be of sufficient quality, they both have to perform above 85% accuracy.

From the methodology section and the listicle of requirements two algorithm needs to be designed and developed this will be done over two iterations, with the first iteration being the design, development and test of the facial expression recognition system, the 2nd iteration will go through the same cycle for creating the continuation desire algorithm.

6 1st Iteration: Facial Expression Recognition

In the first iteration of the project, there is foremost focus on creating a system to obtain facial expressions of people from video both in terms of design and implementation. Expression of people will end out in a system that links facial expressions to emotions. Moreover a test of said system and discussion thereof.

6.1 Design

The system design of creating an application to successfully predict which facial expressions relates to which emotions using machine learning, starts with defining the scope of the system. The system needs to output at least the six basic emotions mentioned in the analysis. The system needs to be lightweight to successfully be able to compute emotions in real time.

6.1.1 System Design

The pipeline for the system to recognise emotions, has to be efficient and be able to an arbitrary video, output any emotion present in each frame. It is important that faces are detected and extracted from each frame, so that when and only when a face is present an emotion is recognised. A pipeline is therefore to be designed.

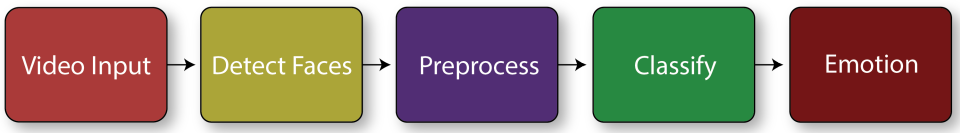


Figure 17: System Pipeline. Video input from arbitrary device, detect faces using Haar Casade, Preprocess image, classify, show emotion

The video input has to accommodate all resolutions, as cameras tend to fluctuate between different resolutions depending on the quality of the camera. This needs to be handled programmatic so any camera can be attached and used in the system.

For detecting face, or more specifically locate faces, one has to use a face localisation technique. There is different methods for doing so, but a well used, and very accurate method is Haar Cascade [Viola and Jones, 2001], which is a deep learning algorithm which can detect objects in images. If trained to localise faces, it will find faces with an accuracy of 99%. The Haar Cascade is a method put forth by [Viola and Jones, 2001] in 2001, but is still one of the most used methods to localise objects, especially faces. Not only does it perform very well, but is also very fast, and this is due to a smart design of how it finds and classify an image. To dive deeper into the model, the model consist of three stages, first a kernel runs over the image, giving labels to each region, negative if no object is found, and positive if an object is found. If the label is negative, nothing more is done to that specific part of the picture. If however a positive label is found, the algorithm use more time on validating that specific part. This way of quickly finding relevant parts, creates a very effective algorithm. The algorithm is also based on a technique called ensemble, which is taking a group of weak classifiers, and have a weighted vote for the final output. Ensemble enables the system to have multiple specialised algorithms to act as one.

During preprocessing when a face is localised, that specific part is extracted and resized to an format that is going to be used for the FER classifier, commonly you want a equal width/height image, as that is most straight forward to work with, as convolutional layers in neural networks expect both sides to be dividable by the same number. Decolourisation of each pixel value is also needed, to convert a normal RGB image to greyscale, to minimise computational power, as one greyscale channel is sufficient for FER. Rescaling of each pixel value from a normal 0-255 scale to 0-1 is also needed to create as little variance in the data as possible.

From the overall system design, specific parts will now be elaborated upon, first designing the model architectures.

6.1.2 Model Architecture

The model architecture will reflect state of the art methods for image recognition and classification, and two different methods will be designed to be tested against each other in terms of accuracy and compute time. Two different model designs will be designed, both based on state of the art concepts, namely residual connections, and densely connections. These concepts will be presented in their proprietary sections.

Generally a models inference time is closely related to the amount of parameters a model consist of, and the higher the amount of parameters the more complex the models is. A more complex model can also model more complex relations between features, meaning that it can in many cases achieve a higher accuracy but at the cost of inference time. Inference time, is the amount of time to which an algorithm take to process an image to a class. Complexity of the model and data needs to be in a optimum space to achieve the best case for training, as a too complex model will quickly overfit the data, meaning that it will 'learn' the data so good that it can remember all samples in the data set. This can mean that the model only works on the training data, but will unusable on any other data.

For models which are used in real time, the parameter count needs to be kept as low as possible to have the fastest inference time.

Both models will strive to use as few parameters as possible to achieve the lowest compute time while still having an acceptable accuracy. The design of the first model, will use a method called residual connections between convolutional blocks see figure 18. The second model will use densely connections, as seen on figure 19.

Residual Connections adds an identity mapping to the output of a block of layers in a model as seen on figure 18. It is more safe to fit the residual mapping, as this is known instead of assuming that stacked layers will fit the underlying mapping in the data. This optimisation is also easier as the residual identity is known at all times, whereas otherwise the mapping is unreferenced. This is to ensure low parameter levels while still producing high accuracies [Szegedy et al., 2016], more over this method ensures that even in cases with vanishing/exploding gradients which is common in deep neural networks, the input data for each layer will remain of high quality as a fresh batch of data is continuously added. This ensures faster convergence as well.

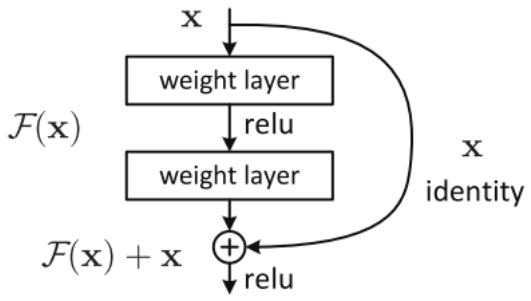


Figure 18: Residual Learning Block, which illustrates a residual identity 'skipping' two layers, and added to the output of those layers.
Source: [He et al., 2015]

The results from [He et al., 2015] is that it is possible to obtain significant higher accuracies by using residual connections between layer block, instead of plain stacked layers in a model, without increasing the parameters of the model. As the model can use the residual identity function if the current layer block is not beneficial in anyway due to vanishing/exploding gradients, the residual identity can be used as well, thus "jumping" over the current layer block. This also helps with overfitting to the data, as no high level abstractions of the data is produced, in the same manner compared to a deep sequential model.

Densely Connection is a connection type proposed by [Huang et al., 2016], which uses another but similar technique to solve the same problem as the previous mentioned residual connection method. Densely connections in a model can be seen on figure 19. The figure illustrates that each layer in the network takes all the preceding feature maps from convolutional layers as input. This means that the output layer, takes all feature maps as input meaning that both low and high level features is used in the final classification when entering the output layer. This is a competent method as completely raw data is usually weak in correlations and having both raw and high level data produces stronger correlations.

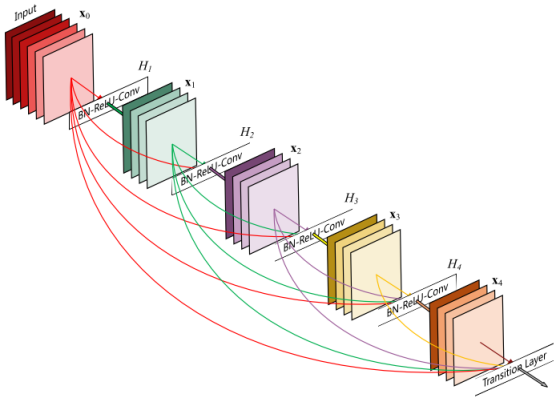


Figure 19: Five layer dense block, each layer takes all preceding feature-maps as input. Source: [Huang et al., 2016]

The authors argue that this type of connection will solve problems like vanishing/exploding gradients, as all outputs from all layers is reused throughout the network. Reuse of features maps, can hamper information if for example two layers have different distributions or if some layers if suffering dead neurons from vanishing/exploding gradients. Concatenating these two layers for example will pass on a much richer combined feature map, than having a singular sequential pass through.

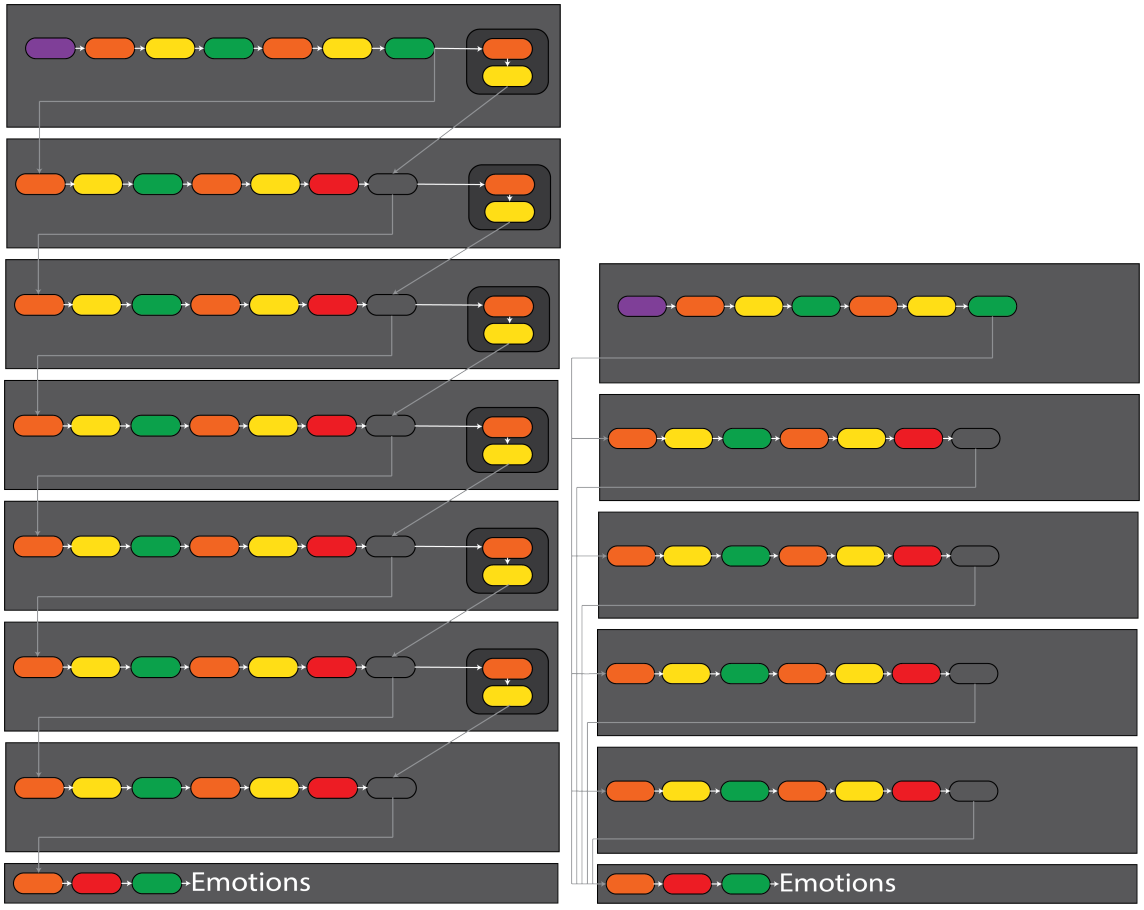
Both these models can significantly reduce the parameter count while still having a comparable accuracy to state of the art models.

From these two approaches, two models will be designed using these methods in the following section, they will from now on be known as the residual and densely model.

Model The model design will be visualised in this section, showing the connection between each layer, and which methods is going to be used. The models will not be presented with corresponding hyper parameters as these will be tuned in the actual implementation. As mentioned, two different techniques will be used, and tested. First the design for the residual model, which will use the method of residual connections. The model design can be seen on figure 20, showing one base block, consisting of two convolutional layers, two batch normalisation layers and two activation layers, as well as on input. Next is six convolutional blocks each consisting of two convolutional layers, two batch normalisation layers, one activation layers, one pooling layer and lastly an addition layer for the residual connection as mentioned earlier. In terms of filters in each of the convolutional blocks, the amount will increase at each block level with a factor of two, this is to capture increasingly smaller details and higher level abstractions of the data.

The residual model, also needs to be fast enough during inference to run real time, on most computers, therefore the depth and the parameter count has to be kept to a minimum. Seven layers should be sufficient to not compromise on accuracy, but a deeper model could perform better, but would be slower at processing.

Next is the densely model, using the techniques discussed on the densely connections paragraph. As seen on figure 20 the model consists of five blocks and one output block, each block consist of the same layers as the residual model. This means that the layers in a block in the densely model is also two convolutional layers, two batch normalisation layers, one activation layers, one pooling layer, what is fundamentally different is what data is given to each block. As seen on the figure, each layer takes as input all preceding layers, which means that the inputs become much



(A) (B)

Figure 20: (A) The model will consist of one base block, and six convolutional blocks, and one output block. Purple: Input Layer, Orange: 2D Convolutional Layer, Yellow: Batch Normalisation, Green: Activation Layer, Red: Pooling Layer, Grey: Residual Addition. Large Grey boxes: Blocks, Smaller Grey Boxes: Residual Computational Box (B) The model will consist of five convolutional blocks, and one output block. Purple: Input Layer, Orange: 2D Convolutional Layer, Yellow: Batch Normalisation, Green: Activation Layer, Red: Pooling Layer, Grey: Residual Addition. Large Grey boxes: Blocks, Grey Lines: Densely Connections

larger proportional to the depth, therefore the model can achieve same results even with being more shallow.

6.1.3 Data

To train the machine learning algorithms discussed in the previous section, a data set is needed which contains images of facial expressions across different emotional states. As discussed in the analysis, the emotions which will be used ranges over should at least be 6 different emotional states including the six basic emotions as discussed in the analysis. The emotions which will be used is: Neutral, Calm, Happy, Sad, Angry, Fearful, Disgust, Surprised. Which is eight different emotions, the reason for adding more emotions to the model is that most of the time humans is in a neutral or calm state when no apparent emotion is being felt, by this argument two more emotions is added.

Datasets can be compiled with different methods, such as Google images search queries, consolidation of different publicly available data sets, or by using a pre-compiled dataset. It is also possible to create a new data set from scratch, but that task is to heavy and time consuming for this thesis, therefore available data sets is going to be used.

To ensure the best scenario for the algorithm to train, there should be a class balance across the dataset. If a skewed class imbalance exists the algorithm might reflect the distribution of classes, instead of directly learning the features of the images. In the latter case, the model might produce untrue accuracies, and recognise false emotions in runtime.

This ends the design section, and the next section will focus on implementing the design choices presented. The overall pipeline has to be programmed to be able to successfully solve the task at hand, next the machine learning algorithm has to be developed as well, lastly the data set have to be compiled from existing data available.

6.2 Implementation

The implementation will be made in Python 3.6[28] with the use of several libraries for data handling and neural network implementation. The implementation will use Keras[29] on top of Tensorflow[30], as Keras offers a high modular block coding style, and is sufficient for this project. Using Tensorflow as the backend, as Tensorflow is currently the best and most used framework for machine learning, as it supports a large range of deployment environments. Tensorflow is currently developed by Google as an open-source project, Keras is also developed and maintained by Google employees, and is open-sourced as well.

For compute a Windows based machine with a AMD Threadripper 2950x 16-core 4.3 GHz CPU, 32 GB ram and a Nvidia RTX 2070 GPU is used as the main system. Slave system used: Windows based machine with a AMD Threadripper 1950x 16-core 4.1 GHz CPU, 64 GB ram and a Nvidia GTX 1080ti GPU.

These system should be sufficient for training all models used in this project.

6.2.1 Dataset

The dataset used for this implementation is compiled from three different dataset, their names and meta data can be seen in table 3. To compile the complete dataset from the three subsets, AffectNet, RAVDESS and Fer-2013 parsing and extracting was needed, as all of the data is in different formats. AffectNet consist of images from the internet primarily Google Images, and has a large variance in image size, which needed to be trimmed to specific dimension to be compatible with the other data. An image size of 48x48 pixels was chosen due to the FER-2013 dataset using this size, moreover using a smaller image size also improves inference time, as lesser pixels equals to a decrease in calculations. As the model needs to be able to run in real time, the inference time is important as stated in the design section. All images had to be converted to one channel grey scale as well. The reason for using grey scale, is that using colour channels for FER is not necessary because line detection works adequate on grey scale images. Reducing the channels from three to one also cuts the computation time significantly.

The RAVDESS dataset consists of video data, therefore each frame from each video needs to be extracted and resized to 48x48 and converted to grey scale in one channel. This was done programmatically, and can be seen in the appendix 11.2.

The FER-2013 dataset is already in 48x48 images and grey scale, so this data did not need any parsing to be used.

The Affectnet images, needed to be resized as well, and converted to grey scale. This was done in a similar matter to the RAVDESS dataset the same logic was used as code example 11.2 in the appendix.

Name	Feature	Labels	Samples	Subjects
AffectNet	Facial Images (Varying sizes)	8 Different Emotions	1,000,000 images	450,000
RAVDESS	Videos of Facial Expressions (1080 x 720)	7 Different Emotions	4904 videos $\approx$ 900,000 images	$\approx$ 24
FER-2013	Facial Images (48x48)	7 Different Emotions	35,887	$\approx$ 28,000

Table 3: Table showing used datasets, and their meta data.
AffectNet [Mollahosseini et al., 2017], RAVDESS [Ryerson, 2018], FER-2013 [Goodfellow et al., 2013]

Combining these three datasets give roughly 1,935,000 images ranging over eight emotions. This is one of the largest combined datasets for FER, and should form a good and generalised algorithm. As the images consists of a large amount of different people, meaning that the algorithm will learn the generalised emotional expression, rather than the emotions related to each specific person.

The ground truth of the data set is an important factor to discuss and review, as the ground truth in the data will be reflected the machine learning algorithm which uses said data. An example of this could be that the data has a ground truth of 90%, meaning that 10% of the data is wrongfully labelled. If a machine learning algorithm is trained on that data, and performs 100% accurate on the test set, the actual accuracy will end up being only 90% as the data hold that accuracy level. Therefore it is important to know the ground truth before using the data.

Each image in the dataset have been validated by four individuals to ensure a high ground truth from the data set creators [Mollahosseini et al., 2017, Goodfellow et al., 2013, Ryerson, 2018], while I personally have validated 30,000 images from random sampling, and ended at a ground truth of 98%. It can be concluded that the dataset is of high quality, and that the ground truth is of high accuracy. But the exact ground truth level is not possible to be calculated.

The ground truth for the whole dataset is unknown, but as the random sub-sample of 30,000 images indicates that the dataset is of high quality, it can be assumed that the rest of the dataset follows the same tendencies.

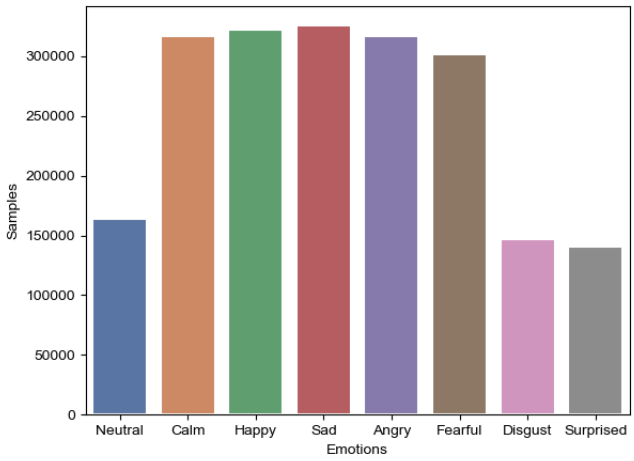


Figure 21: Distribution of classes in the dataset, roughly 1,935,000 labels over eight classes.

On figure 21 the distribution of the different classes in the dataset can be seen, and it shows that there is two levels of alignment in the dataset. First at 150,000 labels each is the neutral, disgust, and surprised emotions, the second distribution line is at around 300,000 which includes calm, happy, sad, angry and fearful. The fact that the second alignment line is at twice as many samples as the first alignment line, is not a problem in terms of dataset imbalance. But it is close to create an imbalance, but it is not seen as a problem in this case.

6.3 Model Implementation

The code snippet 1, showing the model implementation in Python with the use of Keras. The model is developed from the model design of the residual model proposed in the design section, for implementation of the densely connection model see appendix 11.6, training of the model is done in the same way as the residual model presented in this section.

As mentioned the model will use residual connections between convolutional blocks. The model consist of five convolutional blocks, each consisting of a separable convolutional layer, followed by a batch normalisation, then a ReLU (Rectified Linear Unit) activation function. The block will be explained in detail with corresponding code snippets, but for the full code see appendix 11.3.

On line 5, the input shape is set from the function input, the model is using 48, 48 as input shape, this refers to size of the images. Next is the base convolutional block, having first a 2 dimensional convolutional layer with 8 filters with a kernel size of (3, 3) and a stride of (1, 1), moreover it uses a kernel regularisation called l2 regularisation. L2 regularisation is a method also known as ridge regularisation, and it calculates the sum of squares for all features, and forces the weights to be small, so that all pixels receive the same attention. If some weights were significantly larger than others, those signals will be amplified to a larger degree. The kernel which iteratively move with the stride over the input image, processes nine pixels at each step. The kernel regularisation is also to ensure that the model does not over fit during training, and the current value of 0.01 provides the best training parameter for the model.

Next is the batch normalisation, which normalises the batch. Batch normalisation is to ensure better and faster training, as it regularises the batch, much like the l2 normalisation. The normalised output is multiplied by the standard deviation (Gamma) and add the mean (beta) to

each batch. The back propagating algorithm can tweak the gamma and beta values to change or denormalise the output to find the most optimal weights.

Next is an activation layer, using the ReLU activation function, which dictates if a given neuron should fire or not. The ReLU function ranges from 0 to infinity, meaning that if a neurons value is -1, ReLU will shift it to 0 meaning no activation will be given. This can give sparisity in the model which is good, as neural networks is build from the theory of the human brain, not all neurons in the brain is giving a signal at all times. This is also the reason for choosing ReLU over other activation functions such as tanh or sigmoid. ReLU has a lower change of vanishing compared to other functions and is superior to other functions in terms of use cases in convolutional neural network and deep learning.

Then is another 2 dimensional convolutional layer using the same parameters as the previous convolutional layer. Also followed by a batch normalisation and ReLU activation.

Listing 1: Python code showing the machine learning model implemented with Keras - Base Block (models.py)

```
1  def Residual_Model(input_shape, num_classes, l2_regularization=0.01):
2      regularization = l2(l2_regularization)
3
4      # base
5      img_input = Input(input_shape)
6      x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
           use_bias=False)(img_input)
7      x = BatchNormalization()(x)
8      x = Activation('relu')(x)
9      x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
           use_bias=False)(x)
10     x = BatchNormalization()(x)
11     x = Activation('relu')(x)
```

The first two layers of the first convolutional block is where the first residual connection is made, which consist of a 2 dimensional convolutional block, with 16 filters, (1, 1) stride, followed by a batch normalisation layer. The rest of the block is the same as the before mentioned base block, but with a different end, as the output of the batch normalisation on line 14, is fed into a max pooling layer (line 16). A max pooling layer is pooling pixels together to shrink the input dimensionality so higher level features can be extracted, and it improves computational use. Effectively it takes a 3 by 3 square of pixels and outputs 1 value, the value the max of all 9 values, meaning that it shrinks 9 values to 1. The pooling filter size moves iteratively with a stride of (2, 2). Lastly the output of the pooling layer, is added to the residual layer on line 17, as discussed in the design using residual layers improves the training and accuracy of the model, and is a novel technique in deep learning.

The same pattern of convolutional block is present for the next four block, only the amount of filters increase from the base at 8 to 16, 32, 64, 128, 256 and 512 respectively.

Listing 2: Python code showing the machine learning model implemented with Keras - First block(models.py)

```
1      # module 1
2      residual = Conv2D(16, (1, 1), strides=(2, 2), padding='same', use_bias=False)(x
           )
3      residual = BatchNormalization()(residual)
4
5      x = SeparableConv2D(16, (3, 3), padding='same', kernel_regularizer=
           regularization, use_bias=False)(x)
6      x = BatchNormalization()(x)
7      x = Activation('relu')(x)
8      x = SeparableConv2D(16, (3, 3), padding='same', kernel_regularizer=
           regularization, use_bias=False)(x)
9      x = BatchNormalization()(x)
10
11     x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
12     x = layers.add([x, residual])
```

The last block of the model is the output section, as seen on code snippet 3, which consists of three layers. First a convolutional layer, with 'num_classes' as the filter amount, in this case eight, corresponding to the amount of classes the model is trained upon. As the data set consists of eight different emotions. The next layer is a global average pooling layer, which instead of taking the max value, as seen in the previous block, the average of all values in the kernel is the output. Lastly

the output is going through an activation function called softmax, which produces a normalised vector output consisting of eight values. This output is the final predictions, and it outputs the probability for each class at any given sample.

On line 7-8 the model is returned from the function definition.

Listing 3: Python code showing the machine learning model implemented with Keras - Last block(models.py)

```
1     # module 6
2     x = Conv2D(num_classes, (3, 3),
3                padding='same')(x)
4     x = GlobalAveragePooling2D()(x)
5     output = Activation('softmax', name='predictions')(x)
6
7     model = Model(img_input, output)
8     return model
```

To see a complete summary of the model, see appendix 11.4.

6.4 Data Load

For training the residual model, data will be loaded and some preprocessing of the images will be done as well. As three different data sets will be used, they have to be concatenated to one large dataset to be used in the training session. On code snippet 4, the function for loading of the datasets can be seen. As the data come in different formats and structures, there is need for some parsing and preprocessing of the inputs. Inside the function two different functions is also defined, first of is the FER data set, which comes in a comma separated file (csv), with both labels and images. The image size is defined as (48, 48) and the pixels is loaded into a list as seen on line 8. As the images is saved as pixel sequences in a long list, parsing is needed to achieve a regular image format and shape. This is done in the loop at line 11. Each new pixel sequence end with a whitespace, and is therefore splittet at each whitespace as seen on line 12. It is converted to a numpy array, and reshaped to the right width and height, and then resized again with opencv to make sure the format is correct. This is done to all 35,000 images in the FER data set. Moreover the dimension is expanded with line 17 to achieve the shape of (image, width, height, value) for each image. The faces and corresponding labels is then returned.

Listing 4: Python code showing the data loading (training.py)

```
1  def load_dataset():
2
3      def load_fer():
4          dataset_path = 'data/fer2013/fer2013.csv'
5          image_size = (48, 48)
6
7          data = pd.read_csv(dataset_path)
8          pixels = data['pixels'].tolist()
9          width, height = 48, 48
10         faces = []
11         for pixel_sequence in pixels:
12             face = [int(pixel) for pixel in pixel_sequence.split(' ')]
13             face = np.asarray(face).reshape(width, height)
14             face = cv2.resize(face.astype('uint8'),image_size)
15             faces.append(face.astype('float32'))
16         faces = np.asarray(faces)
17         faces = np.expand_dims(faces, -1)
18         return faces, data['emotion']
19
20     def preprocess_input(x, v2=True):
21         x = x.astype('float32')
22         x = x / 255.0
23         if v2:
24             x = x - 0.5
25             x = x * 2.0
26         return x
27
28     with open('data/ryer_data_csv_48/face_images.pkl', 'rb') as f:
29         x_ryer = pickle.load(f)
30
31     with open('data/ryer_data_csv_48/face_images_labels.pkl', 'rb') as f:
32         y_ryer = pickle.load(f)
```

```
33
34      with open('data/affect/affect_face_images.pkl', 'rb') as f:
35          x_affect = pickle.load(f)
36
37      with open('data/affect/affect_face_images_labels.pkl', 'rb') as f:
38          y_affect = pickle.load(f)
39
40      fer_x, fer_y = load_fer()
41      fer_x = preprocess_input(fer_x)
42
43      # Change labeling so each dataset match
44      fer_y = np.where(fer_y == 5, 8, fer_y)
45      fer_y = np.where(fer_y == 1, 7, fer_y)
46      fer_y = np.where(fer_y == 6, 1, fer_y)
47      fer_y = np.where(fer_y == 2, 6, fer_y)
48      fer_y = np.where(fer_y == 0, 5, fer_y)
49
50      x_ryer = preprocess_input(x_ryer)
51
52      x_affect = preprocess_input(x_affect)
53
54      x = np.vstack((x_ryer, fer_x, x_affect))
55      y = np.hstack((y_ryer, np.asarray(fer_y), y_affect))
56
57
58      y = pd.get_dummies(y).as_matrix()
59      return x, y
```

The second nested function is the preprocess_input at line 20 which rescales the grey scale values from being between 0-255 to 0-1 as this centres the data, and makes a more uniform distribution between 0-1, this makes the variance smaller, which then again makes the training easier in terms of convergence. All the datasets will be rescaled between 0-1.

From line 28 to 38 the RAVDESS data set and the affectnet data set is loaded, and both data sets is rescaled, and the FER data set labels is reorganised to match the other data as seen on line 44-48. All datasets is concatenated with numpy and horizontal stacking, and the labels is made to one hot encodings (line 58) and lastly returned for usage.

6.4.1 Training

On code snippet 5, code is showing the Keras training implementation, for training the models. First the model is instantiated on line 1, in this example the residual model is used, the input shape is set, as mentioned (48, 48), and lastly the classes amount, which is eight. Next the model is compiled using categorical crossentropy as the loss function, as the problem is to solve categorical classes. The optimiser used is Adam, which is an adaptive optimiser which is greatly used in convolutional models, the metric specifier is set to accuracy, and is purely which metrics to output during training.

During training data augmentation will be used to further increase accuracy but also robustness. The ImageDataGenerator object has different settings, and dictates how the data augmentation should be handled. The images is both sheared, rotated, shifted, enlarged, and flipped during training. This also means that over fitting is less likely as there is a low chance that each images is the same.

Listing 5: Python code showing training function for training the residual model (training.py)

```
1  model = models.Residual_Model(input_shape, 8)
2
3  model.summary()
4
5  adam_opti = adam()
6
7  model.compile(loss='categorical_crossentropy',
8                optimizer=adam_opti,
9                metrics=['accuracy'])
10
11
12 train_datagen = ImageDataGenerator(
13     shear_range=0.2,
14     featurewise_center=False,
15     featurewise_std_normalization=False,
```

```
16      rotation_range=10,
17      width_shift_range=0.1,
18      height_shift_range=0.1,
19      zoom_range=.1,
20      horizontal_flip=True,
21      )
22
23
24  csv_logger = CSVLogger('models/model_check_point/residual/residual_log.log', append
        =False)
25
26  early_stop = EarlyStopping('val_loss', patience=patience)
27
28  reduce_lr = ReduceLROnPlateau('val_loss', factor=0.1, patience=patience/4, verbose
        =1)
29
30  tensorboard = TensorBoard(log_dir='tensorboard_logs/residual/', histogram_freq=0,
        batch_size=batch_size, write_graph=True, update_freq='epoch')
31
32  trained_models_path = 'models/model_check_point/residual/emotion_model_' +
        model_name
33  model_names =  trained_models_path + '.{epoch:02d}-{val_acc:.2f}.hdf5'
34  model_checkpoint = ModelCheckpoint(model_names, 'val_loss', verbose=1,
        save_best_only=True)
35
36  callbacks = [model_checkpoint, csv_logger, early_stop, reduce_lr, history,
        tensorboard]
37
38  x_train, y_train = load_dataset()
39  xtrain, xtest, ytrain, ytest = train_test_split(x_train, y_train, test_size=0.2,
        shuffle=True, random_state=42)
40
41  with tf.device('/gpu:0'):
42      hist = model.fit_generator(train_datagen.flow(xtrain, ytrain, batch_size),
43                         steps_per_epoch=len(xtrain) // batch_size,
44                         epochs=epochs,
45                         callbacks=callbacks,
46                         validation_data=(xtest, ytest)
47                         )
```

During training, different callbacks is used to monitor and alter training parameters to further improve the overall training session and accuracy of the model. First off, an early stopping mechanism is used as stated on line 26, to stop training when the loss stop improving, this means that the training has stabilised on a plateau, and has reached an optimum. The patience variable dictates how many epochs the loss has to be steady before stopping the training.

On line 28 a callback function for reducing the learning rate when reaching a plateau is used, as when a model is nearing an optimum, reducing the learning rate is often a good altering to ensure that at this stage only small high level correlations should be learned. This ensures that learning at this point is only very general, and that the model should not be making large changes to it weights to accompany some small parts of the data.

Moreover on line 30, Tensorflow board is used to monitor the whole training session, to during and afterwards alter the model design and/or training parameters.

Lastly on line 34, checkpoint saves of the model is used, to save the best performing models during the whole training sessions, this is to check the best trained model, as this sometimes is not the model on the final epoch.

Just before training, the training data is split into a train set and a validation set, with a split of 80 % training data and 20 % validation data. The data is also shuffled to not learn any specific relationship between the persons in the data and the outputs.

Lastly in this snippet on line 41, the model is training with the fit_generator function, which uses the defined data augmentation model, using the training data in batches of 512, with steps per epoch corresponding to the amount of training samples divided by the batch size, resulting in each epoch using all the data. Callback takes a list of all the callbacks that is going to be used, and lastly the validation data is passed as well. The model is trained on a GPU, to accelerate the training many fold corresponding to training on a CPU.

6.4.2 Prediction

On code snippet 6, the code for inference with the trained model. On lines 2-8 parameters for the inference script is instantiated, first, the face detection algorithm Haar Casacade, the previously trained emotion model, and a video to do inference on is loaded. The frame skip variable, is to only use every 15th frame, as predicting every half a second is sufficient.

On line 27, the video capture module is made, and extraction of frames is done done on line 29.

In the while loop, at line 33, the modulus sign is used to get only every 15th frame. Each frame passing the statement, a grey scale frame is extracted, to be used for the face extraction algorithm line 36, which will return a list of faces, even if there is only one face detected. Next a for loop goes through all faces in the list and applies an offset, to centre the face on line 39, using the offset values, the grey scale image of the face is extracted on line 40. Then the image is resized to (48, 48) to be used for the emotion recognition model. The image is preprocessed on line 46 as during training, rescaling the image from 0-255 to 0-1, then dimensionality is altered to correspond to the model input, as previous stated. Then on line 49 the emotion is inferenced from the image resulting in a list of length eight, corresponding to the emotions which are predicted. The values in the list is a probability for each possible emotion. The result is appended to a list, for further processing.

On line 53, a new frame is received from the vidcap.read(), this process is repeated until no further frames is left. The emotion list is saved on line 55-56.

Listing 6: Python code showing inference with the trained model (inference.py)

```
1
2  # parameters for loading data and images
3  detection_model_path = 'models/detection_models/haarcascade_frontalface_default.xml
       '
4  emotion_model_path = 'models/final_model.hdf5'
5  path_to_video = 'data/test_video/test_1_dengi.mp4'
6
7  skip_frames = 15
8
9  emotions_predicted = []
10 # hyper-parameters for bounding boxes shape
11 frame_window = 10
12 emotion_offsets = (20, 40)
13
14 # loading models
15 face_detection = load_detection_model(detection_model_path)
16 emotion_classifier = load_model(emotion_model_path, compile=False)
17
18 # getting input model shapes for inference
19 emotion_target_size = emotion_classifier.input_shape[1:3]
20
21 # starting lists for calculating modes
22 emotion_window = []
23 frame_count = 0
24
25 # starting video streaming
26 vidcap = cv2.VideoCapture(path_to_video)
27
28 success,image = vidcap.read()
29
30 success = True
31
32 while success:
33     if frame_count % skip_frames == 0:
34         gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
35         faces = detect_faces(face_detection, gray_image)
36         for face_coordinates in faces:
37
38             x1, x2, y1, y2 = apply_offsets(face_coordinates, emotion_offsets)
39             gray_face = gray_image[y1:y2, x1:x2]
40             try:
41                 gray_face = cv2.resize(gray_face, (emotion_target_size))
42             except:
43                 continue
44
45             gray_face = preprocess_input(gray_face, True)
```

```
46                 gray_face = np.expand_dims(gray_face, 0)
47                 gray_face = np.expand_dims(gray_face, -1)
48                 emotion_prediction = emotion_classifier.predict(gray_face)
49
50                 emotions_predicted.append(emotion_prediction)
51         frame_count +=1
52         success, image = vidcap.read()
53
54 with open('video_predictions/video_pred.pkl', 'wb') as f:
55         pickle.dump(emotions_predicted, f)
```

It is possible to visualise how the network 'sees' an image by visualising the feature map outputs from a convolution layer, this can be seen in the appendix 11.11. Six figures is shown where each figure consist of eight feature maps, the feature maps is from different depths of the model where it can be seen that the deeper the image flows in the model, the more high level the features become. The images is still recognisable on the first layers, but the deeper the probing of feature maps become, the more pixelated and harder it is to see what the image shows. The last figure shows eight kernels from the 48th layer in the model, which shows how the features is extracted. The image used in the visualisation can be seen on figure 22.

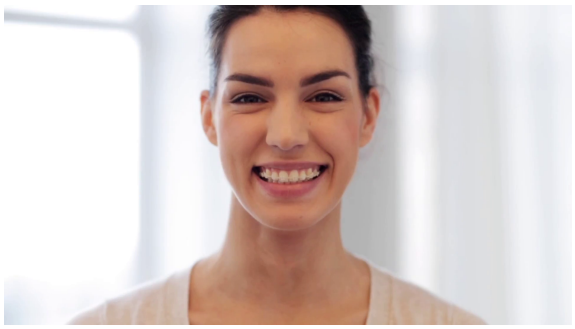


Figure 22: Image showing a happy emotion, used in the visualisation of the emotion recognition model

The implementation section have presented code and the overall program which will create a emotion recognition system, as specified in the design chapter, and thus completing the first item in the requirements list. The next step is to validate the model, as well as gathering data to the next algorithm, which is the continuation desire predictor. The next section fill focus on the results from the training of this algorithm and the test to gather data from subjects playing a video as written in the methodology section.

6.5 Results

This section will present the results from the first test of the emotion recognition system, and present demographic, and data concerning the data gathered as well. The test was conducted on six play sessions each consisting of two people, resulting in 12 people (n=12) finishing the test. All testers was students at Aalborg University Copenhagen. Moreover it will present the training of the algorithm and the achieved accuracy as well as a validation of the model used in a real environment.

6.5.1 Training of Emotion Recognition

The training of the machine learning algorithm for emotion recognition can be seen on figure 23, both accuracy and loss of training and validation is visualised. The curve illustrates a good training session, with the training accuracy starting at around 60% and ending at 97.7%, while the validation accuracy ends at 98% at the end. The training accuracy is in-sample accuracy which means that the sample have been seen and used to training of the algorithm, meaning that this accuracy is not reference for the actual performance of the algorithm. The validation accuracy however is out of sample test set, which is not used for training, but illustrates how the algorithm performs on data that have not been seen and therefore indicates the actual performance of the algorithm in a real scenario.

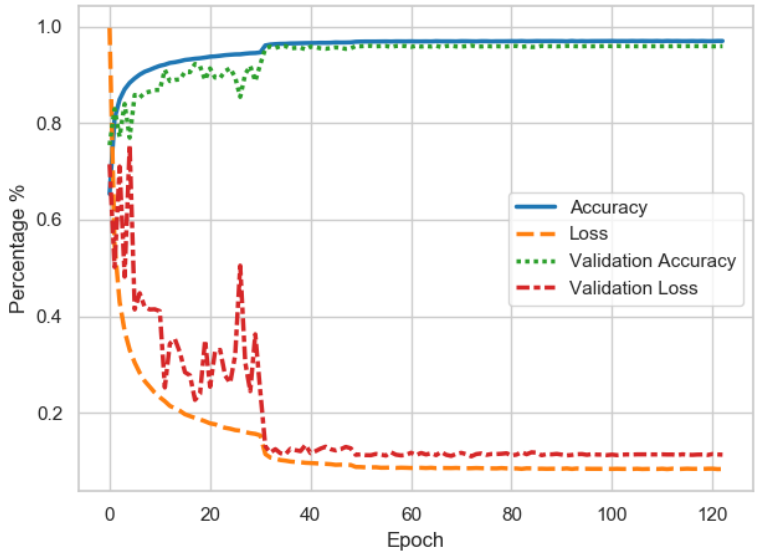


Figure 23: Figure showing training accuracy, training loss, validation accuracy and validation loss development over epochs for the residual connected model. Y-axis ranges from 0 to 100% and x-axis from 0 to 122 epochs

It is also seen that at around the 30th epoch the algorithm finds the minimum optimum weights as the learning stagnates to a plateau, which indicates that no more accuracy can be obtained with the current form of the algorithm. An epoch is one forward and backward pass of the entire data set, so in this instance it is around 2,000,000 images, one batch however is a sub-set of the data set which is set to 512 samples at a time, as the computer can not handle the entire data set at once.

Training of the algorithm took 36 hours, on a high-end machine as stated in the implementation.

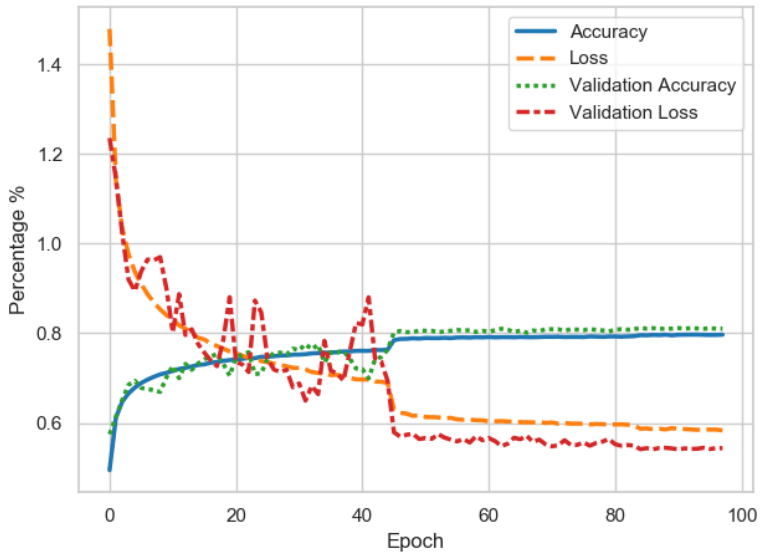


Figure 24: Figure showing training accuracy, training loss, validation accuracy and validation loss development over epochs for the densely connected model. Y-axis ranges from 0 to 100% and x-axis from 0 to 91 epochs

The densely connected model did not train to the same point as the before mentioned model, as seen on figure 24. The accuracy ends at 81% after 91 epochs, the curvature of both the accuracy and loss is comparable to the training session of the residual model, as both find the optimum very suddenly around the same timestep.

The training on the residual model converges more quickly and ends of a higher accuracy than the densely model. The densely model took 23 hours to train, as the model is slightly smaller than the residual model, but due to RAM constraints it was not possible to perform a complete parameter-to-parameter comparison.

6.5.2 Conation Gathering Test

Before the test started, all participants was asked to answer a questionnaire relating to demographics and a few insights into their knowledge and use within games, the specific questions can be seen in the methodology section 5. On figure 25 the age distribution of all participant can be seen, showing an age range from 22-30 ($\bar{x} = 25.5, \sigma = 1.95$). To get an indication of the level of which the players would approach the game, a question about their weekly play time was asked. On figure 26, the distribution of the aggregated occurrences of the different categorise is visualised. Most played more than 13 hours a week, while the second most is 7-9 hours a week, this data suggests that the general level of the players are higher than average, as the average gamer plays six hours a week[31].

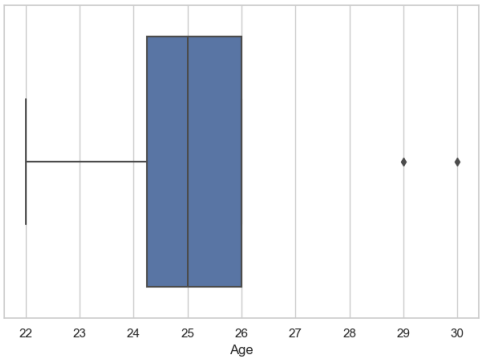


Figure 25: Boxplot showing age from Conation Test Sessions

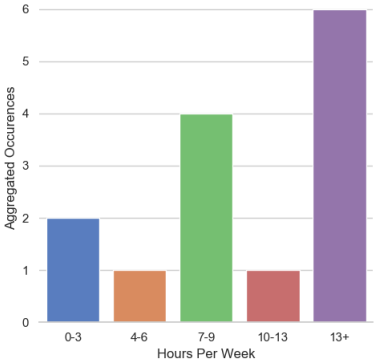


Figure 26: Gameplay time distribution from Conation Test Sessions

On figure 27 and 28, two figures is showing the reported conation values from the test sessions. The figure 27 shows the average conation and variance for each interval which shows declining line over intervals which fades over to a plateau which again declines to the last episode. An interval is five minutes as the players continuation desire was gathered every five minutes.

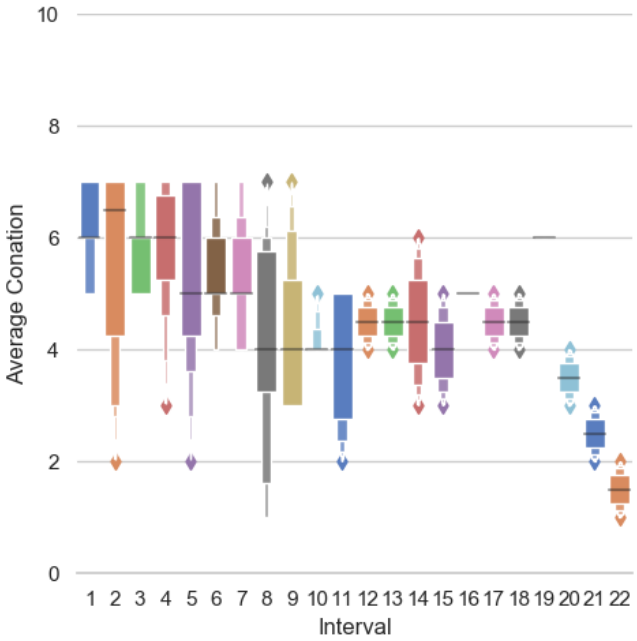


Figure 27: Box plot showing the average continuation desire and variance for each interval from Conation Test Sessions

Figure 27 also shows a large variance in multiple intervals, where testees have been chosen

values over a large span. This is seen on interval two, five and eight, where the variance is large, and potential outliers is seen. It is important to note that each interval is not supported by equal amount of samples, as samples is also declining over intervals, especially after interval eight. The plot shows a black line in typically the middle of each box, which indicates the average, when different size columns indicate the amount of samples in that area. The longer the columns the larger the variance.

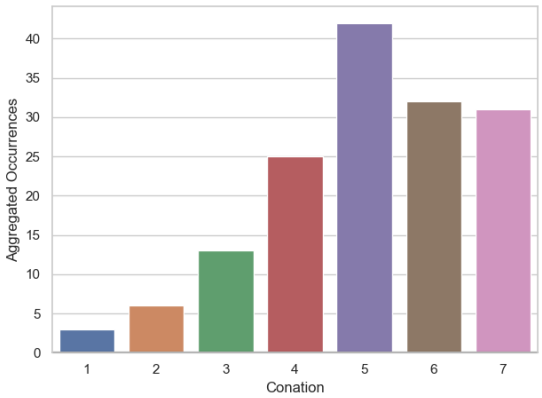


Figure 28: Conation distribution from Conation Test Sessions

The total set of emotions registered is 2,597,174, which corresponds to the amount of frames where a face was detected, if no face is detected in a frame means that the frame will not be used further in the recognition. There was six test sessions, with two participants in each, resulting in 12 unique persons, due to varying play session time, each person have reported different amount of conation values. The variation in the play session length is due to the test being open-ended, players should play a minimum of 30 minutes but was able to continue if wanted. Each interval corresponds to a conation value reported, this results in 128 intervals of around five minutes each, with a frame rate of 60 fps, which results in $\approx 2,597,174$ frames with faces (samples).

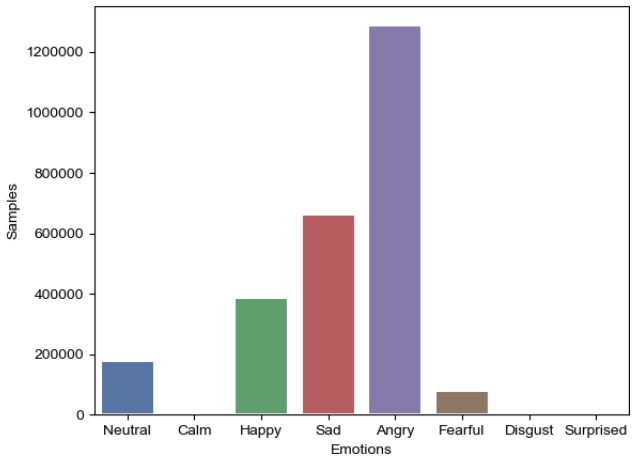


Figure 29: Distribution of predicted emotion during the Conation Test Session

On figure 29, the distribution of the predicted emotions from the conation test session can be seen. It shows that the angry emotion was by far the most present emotion followed by sad, happy, neutral and then fear. Disgust and surprise was not recognised in this test. The reason for the disgust emotion not to be present is that it is by default a rare emotion which is not seen that often, and that the played game does not show content which would make subjects react with disgust. The surprise emotion, is also a rare emotion and is not a long lasting emotion. Again the game play observed does not produce any surprising elements, and this could explain why those two emotions is non existing.

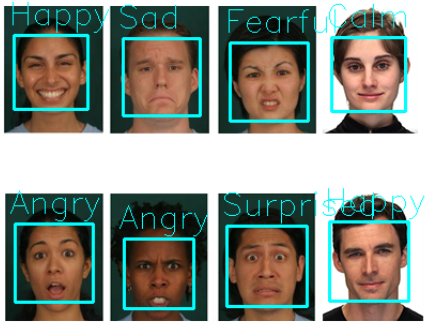


Figure 30: The emotion recognition run on a collection of face images (image from analysis)

An image test was conducted on an image consisting of eight emotion images, see 30. The image is from the analysis section about facial expressions, and shows the predicted emotions from each face in the image. most of them is correct, but as seen some incorrect as well. The first incorrect is the third image from the top left corner, which is showing disgust, but is classified as fearful. The first image from the bottom left is showing surprise, but is classified as angry. The third image from the bottom left, is showing fearful but is classified as surprise.

The test indicates that there is still room for improvement. The faces which are wrongly predicted, is also close in terms of expressions, and could in my opinion be miss classified by a human too. But this image show an accuracy of 62.5 % which is significantly lower than the validation set test which achieved 98%.

This concludes the results section, and the next section will discuss the results in relation to the final problem statement, and the thesis as a whole.

6.6 Discussion

Thus ends the first iteration cycle, and the development of the emotion recognition system, and data gathering for the continuation desire algorithm which is to be developed in the next iteration. The emotion recognition system for frontal face facial expression analysis, ended at 98% accuracy which certainly makes it an algorithm which performs as well, or better than state of the art work [Ryerson, 2018, Mollahosseini et al., 2017, Arriaga et al., 2017, Jang et al., 2011]. Affectnet which performs around 75% in accuracy for FER tasks, trained on 900,000 images [Mollahosseini et al., 2017], to [Arriaga et al., 2017] who performs 78% accuracy trained on 35,000 images, this projects proposed model and data results in state of the art results. It is also important to note that the performance of this algorithm is tested on a large range of different people's faces making the results more robust compared to other researchers who used data sets with small amounts of persons. The sheer scale of the data set is also the biggest that is publicly known for FER tasks, which gives credit to the generalisation obtained on the algorithm.

The accuracy is calculated from the validation test while training the algorithm, this means that the algorithm achieves a correctness of 98% on 10% of the data set, which means that the validation set consists of 200,000 samples. The validation test set is out-of-sample data, which means that the algorithm does not see the data during training but is only used for validation. This is common practise to evaluate machine learning algorithms, as in-sample data would be 'cheating' as the algorithm already knows those data points.

The model is not yet validated on a 'in the wild' test which tests the system in an actual environment, but out of sample data shows promising results. As seen in the results on figure 29, there is a large part of the data which is predicted as Angry, which spikes an interest into why that much of the data is classified into that category. This unforeseen classification could indicate an angry emotion which is prone to activate too easily compared to the other emotions. This have to be addressed to validate that the emotions recognition system is working as intended and does not wrongly classify an emotion. This analysis will be carried out in the 2nd iteration in the validation of the models.

The other model trained, namely the densely model, did not converge to an acceptable accuracy, and is therefore not going to be used further in this project. There was a significant performance different between the residual and the densely model, at around 17% points.

In terms of data obtained from the test session for usage for the next iteration, 2,597,174 samples in 128 sequences should be sufficient to train a continuation desire recognition system, though more samples would be better as the current class distribution as seen on figure 28, most samples is in the high end of the continuation desire spectrum, meaning that the lower end of the spectrum could potentially mean that there is not sufficient samples to make a generalised algorithm based on the data. One solution could be to instead of making a model to classify seven different levels of continuation desire, it could instead classify if a person is in a high or low state of continuation desire, which puts more samples in each bin of possible classes.

Because of the real time aspect of the developed algorithms mostly regarding 48x48 images, a vast amount of pixels is lost going from video steam to classification. Tests where conducted to see if larger images would increase accuracy and more importantly increase the robustness of the algorithm. It is observed that the conditions around the subject have to reach an optimum for the algorithm to reach its full potential, this involves light, colour of the light and shadows on the subjects face. With larger images, this could potentially solve these faults, and from tests the results are of high quality, but the data sample was small, as memory in the compute machine became a limit for how large the data set could be, as 300x300 images would require 40 times as much memory.

Therefore when using the algorithm, the light, background light and settings on the cameras feed have to match as closely to the data as possible, as these irregularities can affect the output of the algorithm.

This ends this iteration, the results mean that the project can continue onward to the development of the continuation desire algorithm, as the emotion recognition system is acceptable as the accuracy is sufficient, as of state of the art results. This means that the first requirement of the project is fulfilled, and the next stage can be developed. Even with the surprising results, there is no reason not to continue, as there is still confidence in the developed method.

The next section will start the 2nd iteration of the thesis, which will design, develop and test the continuation desire predictor system.

7 2nd Iteration - Continuation Desire Predictor

The 2nd iteration will focus on developing the continuation desire algorithm, and present results regarding the case study, and validation of the model developed in this iteration and the last. First a design section will focus on the design of the system, then an implementation of the algorithm followed by results from the tests, and lastly a discussion and conclusion.

7.1 Design

The design section will focus on the system as a whole, and dive into the design of the specific parts which is needed to fur fill the requirements specified in the method section 5.

7.1.1 System Design

The system design of the 2nd iteration builds on top of the 1st iteration, using the whole emotion recognition pipeline as seen on figure 17. The new pipeline is shown on figure 31, compared to the flow in the first iteration, a new layer is added, namely emotion features, pool sequence, classify and continuation desire level. Emotion features is the data from the emotion recognition system, which is the emotion probabilities to an image, and the neuron outputs from the second to last layer in the model as well. As specified in the first iteration implementation section the last layers, a pooling layer and dense layer, which both have eight neurons, these layers aggregated will output 16 features for a sample. The reason for using these specific outputs is that they are the most high level representations of the an input, meaning that there is a good separation between data from each class. This is also known in the field as transfer learning or fine tuning of a model, which is where you use an existing model to perform other but related tasks. But to able to solve the task you have to fine tune new layers to that specific problem, but you are using an already trained model.

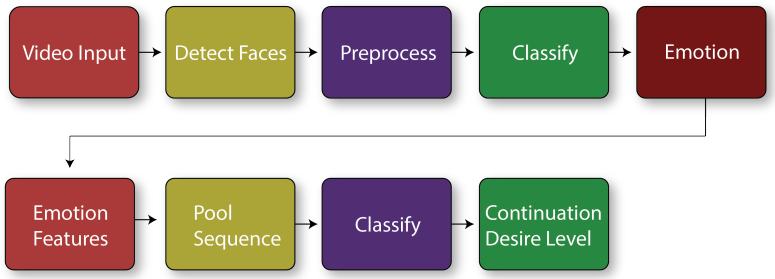


Figure 31: Flow diagram showing the flow through the system. The pipeline consist of input from arbitrary device, detect faces using Haar Casade, Preprocess image, classify, emotion output, emotion features, sequence pooling, continuation desire level classification, final continuation desire level.

The pool sequence step is for doing sequence classification, as one need a sequence of data when there is a temporal relation between data samples, or when the task is spatial depended. More on this in the next section.

The pool sequence step is therefore to pool samples together in sequences of five minutes, which is then fed to the algorithm. The classify step is where the the network infer the continuation level of a subject using the sequence provided, the output is then the level of continuation desire of said subject.

From this, the parts that will be presented is how to obtain the data that is going to be used for training and, a new model design for the continuation desire prediction also needs to be designed. Lastly graphs for visualisation of the data when evaluating a play session will be presented as well.

7.1.2 Data

The data used in this iteration, will be the data obtained during the testing session from the 1st iteration, as mentioned 2,597,174 emotion data points was produced by the data gathering session,

and that will be used to create the data for the machine learning algorithm to detect at which continuation desire level a subject is feeling.

Continuation desire relates to emotions through the affect aspect of the continuation desire model. The affect aspect is controlled by positive and negative reactions to interactive experiences, and therefore can emotions be linked to affect, as a positive or negative reaction to content, usually is related to an emotional response to stimuli.

Each sample consists of 16 features, and is going to be extracted from the emotions recognition model as explained in the previous section.

To calculate the ground truth of the continuation desire data set, will the ground truth of the first dataset multiplied by the second datasets ground truth. The ground truth is a metric which numerically informs about a data set's truth level or correctness in percent. The first dataset had a ground truth of 98% and the dataset used in this iteration is based on the first model achieving 98% resulting in the continuation desire dataset to be at 95.15%. Which means that if the continuation desire predictor achieves 100% accuracy, it would in reality only be 95.15% accurate, as 4.85% is incorrect in the data. The labels of continuation desire is self reported from the subjects, and is not possible to validate, therefore it is assumed that the data is 100% truthful from the participants side.

Emotion Probability Features
6.265e-03, 9.313e-13, 2.498e-01, 1.539e-02, 6.923e-01, 3.609e-02, 3.579e-05, 3.389e-06

Table 4: Table showing one sample from a sequence, showing eight features of the dataset which is extracted from the output layer of the emotion recognition model.

Neuron Features
-3.057, -2.568e+01, 6.284e-01, -2.158, 1.647, -1.306, -8.222, -1.057e+01

Table 5: Table showing one sample from a sequence, showing the eight features of the dataset which is extracted from the second to last layer of the emotion recognition model.

In table 4 and 5, the features from the dataset extracted from the emotion recognition system is seen. The first eight is the probabilities from each emotional expression from the output layer of the emotion recognition model. As mentioned in the implementation in the first iteration, the output layer had eight neurons which is softmax activated, meaning that the sum of the eight values in the last layer is always one. The values is statistical probabilities on each emotion from an image. So the probabilities data tell what kind of state the subject is in, on a numerical representation.

The next eight in table 5 is features from the dataset extracted from the emotion recognition system's second to last layer, which is a pooling layer. The pooling layer shows high level information about the image from which a subject is present, the information is therefore a vector representation of the subjects face, giving information regarding expression from another perspective than emotion probabilities. As the second to last layer feeds data forward to the last layer, it is expected to see high correlations between data points in the first and second table.

7.1.3 Model Architecture

As mentioned in the analysis, continuation desire is a feeling that rises and falls over time and is therefore dependent on past experiences, meaning that the machine learning algorithm have to learn correlations in sequences of data points. This can be done with having a memory cell that can remember past instances of a sequence, because a normal dense layer in a neural network, only looks at one data point in a dataset at a time. As introduced in the analysis, LSTM layers can remember past instances, and use past seen data points to make a more accurate prediction of a sequence. The model for predicting continuation desire, needs to be able to remember past instances, and therefore this method is going to be used in the design of the model.

To sum up, a Long-Short-Term-Memory layer or LSTM for short is a specific implementation of recurrent units, gates and memory cells. A layer which has recurrent properties is used when the data you are working with is of time series format, meaning that there is a relation between sample points. As continuation desire has a relation between frame one and two or frame 10 and 1000 in a sequence, recurrent units is needed. The LSTM, can remember long dependencies in the

data when trained on sequences, and this is useful when continuation desire changes over time, and is dependent on past experiences.

The model is going to be a shallow model, with three LSTM layers and two dense layers. Having three LSTM layers, with two layers returning the whole sequences, the network will be able to remember long spatial relations across all features. The emotions recognition will be the feature extracting network, therefore the Continuation Desire model can be relatively small, as the data used is of high level features and quality.

On figure 32 the model design can be seen, as mentioned it will first receive a sequence of data points as input, corresponding to approximately five minutes of samples, as each label was obtained in five minute intervals. Next is a LSTM layer followed by an activation, there after another LSTM layer also followed by an activation layer. The first two layers will return full sequences, meaning that the following layer will see all samples in a sequence, same as the past layer. The third layer is also a LSTM followed by an activation, this layer will not return a sequence but simply the last data point in the sequence as a aggregated weight summation that will correspond to the aggregate of the whole sequence. Next is two dense layers, both followed by an activation, which lastly results in an output.

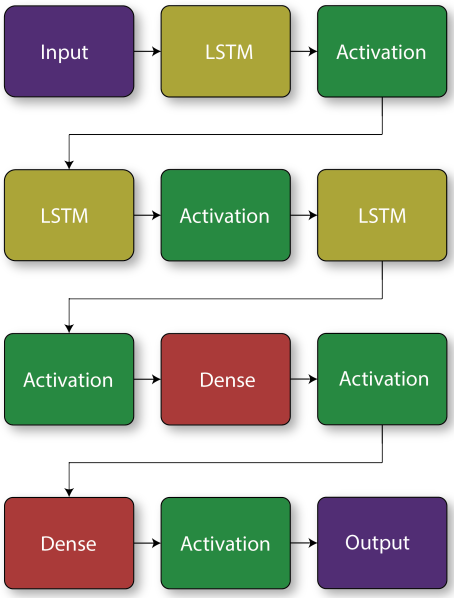


Figure 32: Model showing the model design for the continuation desire classification model, from input to output.

The activations through out the network, will either be ReLU or Leaky ReLU activation, but this will be further tested in the implementation of the model. The reason for using a Leaky ReLU implementation is that ReLU activation can sometimes result in dead neurons when used together with LSTM layers, as the weights can approach zero which will zero out multiple neurons, resulting in no output from said neurons. This is worst case outcome after a training session, as the model becomes 'dead' and can not be used.

The difference between ReLU and Leaky ReLU, see 33, is small but significant, a ReLU activation can only output values in the range of 0-infinity, meaning that if the input data consist of negative values as well as prior negative values from weight multiplications in the model, many values can be set to zero resulting in very sparse data and dead neurons. The reason for this is that during the backwards pass, the derivative of the slope to find the gradients is zero, meaning the accumulated weight update will be multiplied by zero, resulting in the weight update being zero resulting in no learning for this layers and as the output is of zero the following layers will not receive any meaningful data.

A Leaky ReLU, allows for small fractions for negative values, meaning that there is a slope for which the gradients can flow even if for negative values, that means that the derivative of the slope can never be zero allowing for data flow at all times. This can solve the problem of dead neurons during training, but this activation is also more costly in form of compute time compared to regular ReLU.

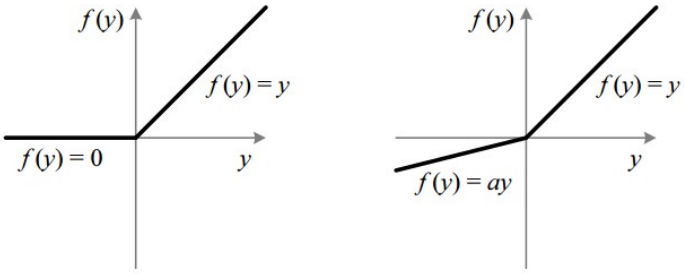


Figure 33: Figure showing two plots of different activation functions, namely ReLU (Left) and Leaky ReLU (Right)

The last layer will use a softmax activation to get class probabilities of each label used during training. This will in the end allow to get an output of probabilities for each label given a sample, by this it is also measurable how confident the algorithm is on the current prediction.

The following implementation section will focus on developing the designed system in this design section.

7.2 Implementation

The implementation will again be developed in Python[32] 3.6, using a range of libraries for data wrangling and neural network implementation (Numpy[33], Pandas[34]), as mentioned Keras[35] will be used on top of Tensorflow[36] for implementation of the algorithm. Same compute is used for training as mentioned in the previous implementation section.

7.2.1 Dataset

The data set used for training the continuation desire algorithm is the data gathered from the test session in the first iteration. As mentioned in the result and design section, the data consist of 12 participants over six test sessions, resulting in 128 intervals each containing five minutes of video data at 60 frames per second. That means that there is 2,597,174 samples in total. How the data was extracted and what kind of values exists in the data, is discussed in the design chapter, and the extraction is developed in this implementation.

It was noticed during preparation of the data, that there is a large class imbalance in the data set, which will make it difficult to infer specific continuation desire levels. Therefore two different methods will be tested during training, one trying to infer the seven different levels of continuation desire, and one trying to infer two classes, high and low continuation desire. The reason to the class imbalance, is that the reported continuation desire levels from the test session did not create an uniform distribution, but more a negative (right) skewed normal distribution, meaning that not all levels have the same amount of reported samples. The distribution can be seen on figure 28 in the results section 6.5.

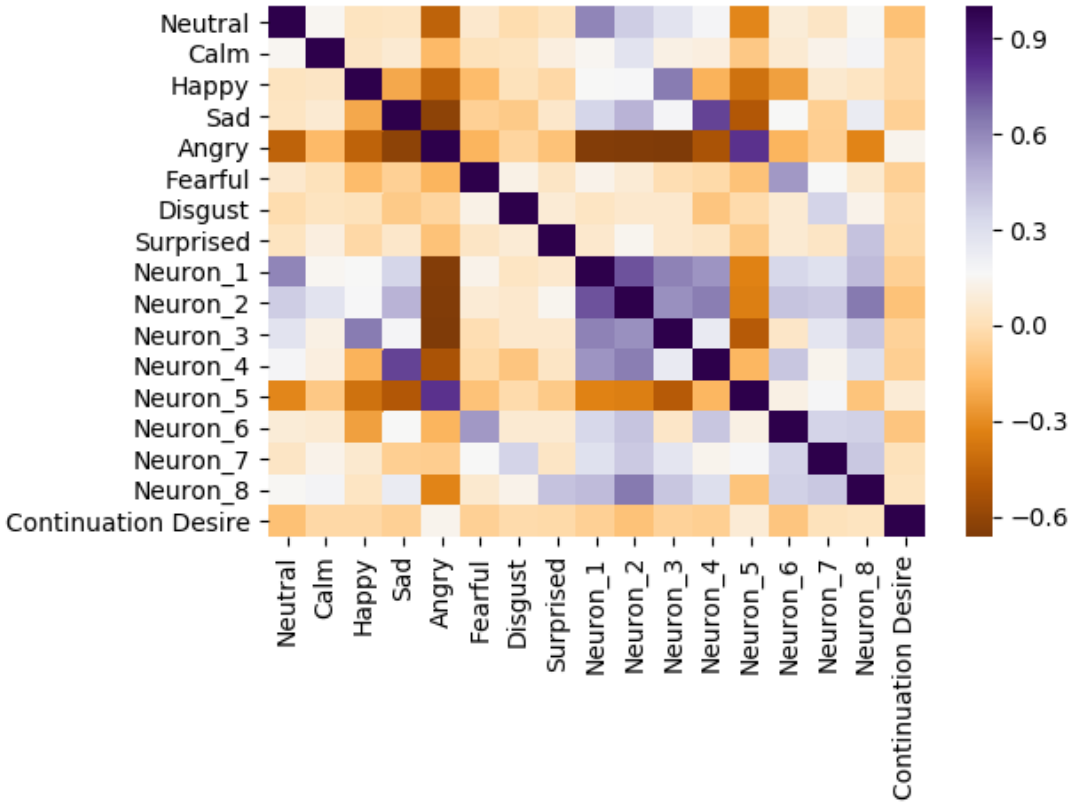


Figure 34: Heat map showing the correlation between each feature in the data set and the targets. 1 or deep blue is equal to high correlation and brown or -1 is equal to no correlation. The correlation is calculated with pairwise Pearson correlation

The heat map correlation figure 34 shows the correlation between each feature in the data set and the corresponding targets. A correlation score of +1 means that the data linearly correlates positively with each other, meaning that they are following the same pattern, such as when one variable increases the other increases as well. This can best be illustrated with the diagonal where

when the same data is on both axis the correlation is 1. A correlation of 0 means no linear correlation and is white on the figure. A correlation score of -1 means that the data linearly correlates negatively with each other, this means that they are following a opposite pattern. This can be shown with as one variable increases the other decreases.

As expected all emotional probabilities and neurons have almost equal correlation with continuation desire, but the angry emotions does have the lowest correlation. It can be assumed that the reason for the decreased correlation is the amount of angry emotional responses in the underlying data set, as the angry emotion is present more than the other emotions. It can also be seen that the angry emotion have little correlation with other emotions, while multiple of the other emotions like calm, happy, neutral and sad correlates with each other. The correlations indicate the relationship between the emotions, and which emotions typically happens in parallel or in the same situations. The angry emotion happens mostly in a singular expression, where happy and calm could intertwine and co exist as a facial expression.

One might question the low correlation between the data and continuation desire as seen with the Pearson Correlation. The variables could have a non linear relationship, which would give low linear relationships, as Pearson correlation measure the variance from the best fit linear line to the data. Moreover, the correlation is per data set basis. What happens when this data is analysed in relation to past values in their right order? Neural networks is extraordinarily at fitting non linear and linear relationships in a function, this means that regular statistical correlation models might fall short of the relationship which consist between the data samples.

The linearity test did not show much relationship between continuation desire and emotions, but it is still believed that there exists relations between the data points. The data needs the last step of preprocessing which will be presented in the next section.

7.2.2 Data Preparing

To extract the features for the new data set, the emotion recognition model have to process every image from the test session. On code snippet 7 the code can be seen for the extraction. On line 1-2 the image width and height, and input shape to the emotion recognition model is set. On line 5-6 the residual emotion recognition model is loaded from the saved weights from the training session, the weights is values from each neuron in the algorithm. The neurons and its weights is the values being tuned during training, these weights can therefore be loaded when the algorithm is needed.

As mentioned in the design, the features for this data set is more than the eight outputs produced from the emotion recognition algorithm. An altered model had to be programmed to get more outputs, the altered model design can be seen in the appendix 11.5. The main difference is that the output of the model is set to output two layers instead of one, namely the output layer and the second to last layer. This results in 16 values per input image.

Listing 7: Python code showing model definition (Batch_predict_conation.py)

```
1  img_width, img_height = 48, 48
2  input_shape = (img_width, img_height, 1)
3
4  # Load Model and data
5  model = models.residual_model_altered(input_shape, 8)
6  model.load_weights('models/Residual_512_final_model_512.hdf5')
7
8  with open('data/conation_images/conation_images_0.pkl', 'rb') as f:
9      x = pickle.load(f)
10
11 #Preprocess input
12 def preprocess_input(x, v2=True):
13     x = x.astype('float32')
14     x = x / 255.0
15     x = x - 0.5
16     x = x * 2.0
17     return x
18
19 predictions = []
20 all_samples = 0
21
22 # Loop through all image sequences and predict on each of them
23 for sequence in x:
24     x_preprocessed = preprocess_input(np.asarray(sequence))
25     predicted = model.predict(x_preprocessed)
```

```
26      all_samples += predicted[0].shape[0]
27      print(all_samples)
28      predictions.append(predicted)
29
30  # Save all predictions
31  print("all samples: " + str(all_samples))
32  with open('data/predicted_conation_vid/predicted_conation.pkl', 'wb') as f:
33      pickle.dump(predictions, f)
```

On line 8-9 the images extracted from the test session videos is loaded, to see how the images is extracted see appendix 11.7.

On line 13-18 a preprocess method to scale the pixel values to be between 0-1 as used previously during training. It is important to have the same scaling as used during training, as if the values have a significantly different mean, the algorithm does not work. Next on line 26-31 the emotion recognition is performed, first a for loop runs through all the sample sequences (images) in the extracted data. The image is then rescaled with the preprocess method, then the emotion recognition algorithm infers the emotion from the image, and appends the results to the predictions list. The predictions is then saved on line 36-37 to a .pkl file, which will hold all emotion probabilities and layer values for each sample in each sequence of data.

7.2.3 Model Definition

The model for the algorithm to infer continuation desire can be seen on code snippet 8. Compared to the emotion recognition algorithm, this model is significantly smaller. The model is smaller as the input space it significantly less than of the emotion recognition model, as this model have 16 features, the emotion model have 2,304 features (48*48), this means that the algorithm have to model less complex data, which can be done by a smaller amount of neurons.

The model is first defined with an input on line 3, setting that the features space is of shape 16. The None keyword corresponds to the batch size, which is variable, therefore specified as None. The input data will come in batch sizes corresponding to the size of a sequence which will hold samples corresponding to five minutes of play.

Next is a Cuda LSTM layer, which is a GPU implementation of a regular LSTM layer, normally LSTM layers have to trained on the CPU because of nature of how they work, that cannot be parallelised on the GPU, but a little simplification of the implementation makes it parallelisable, which greatly improves compute time. 150 neurons is used which is sufficient for the problem space. The return_sequences parameter is set to True, which means that the whole input sequence is returned to the next layer in the model. This is important if the model is to learn temporal correlations between data samples, especially if the sequences are long. The CuDNNLSTM is followed by a LeakyReLU activation, as mentioned in the design section, ReLU activation usually have a problem of exploding or vanishing gradients, making them hard to train as it is bound to occur in long training sessions. A reason for this happens with a ReLU activation is that negative values will be activated to 0, making the gradients in the model behave extremely to encompass the negative values.

Listing 8: Python code showing model initiation (train_conation_class.py)

```
1   def lstm_model():
2
3       input = Input(shape=(None, 16))
4
5       x = CuDNNLSTM(150, return_sequences=True)(input)
6       x = LeakyReLU()(x)
7       x = CuDNNLSTM(150, return_sequences=True)(x)
8       x = LeakyReLU()(x)
9       x = CuDNNLSTM(150, return_sequences=False)(x)
10      x = LeakyReLU()(x)
11      x = Dense(150)(x)
12      x = LeakyReLU()(x)
13
14      out = Dense(2, activation='softmax')(x)
15
16      return Model(input=input, output=out)
```

A LeaklyReLU can encompass this problem, by being leaky, which means that if the output of a neuron is negative, a LeakyReLU will output a small fraction instead of zero, giving the gradients during back propagation values to float on, making it easier to optimise.

The next is again a CuDNNLSTM layer with 150 neurons and return_sequences set to true, followed by a LeakyReLU activation, following that is a CuDNNLSTM layer with 150 neurons, with return_sequences set to false, meaning that it only returns the last value of the of the sequence, this is then again activated by a LeakyReLU. The last sample of the sequence is then input into a dense layer with 150 neurons, also activated by a LeakyReLU. Lastly the output layer is a dense layer with two neurons activated by a softmax layer, which will output probability for the two classes.

7.2.4 Data Loading

For the training of the model a special data generator is used to feed the correct data at each batch. First a DataGenerator class is created on line 2, and it starts by opening the data and the corresponding labels on line 4-8. As two different models were trained, one multi labels classification and one binary, therefore the variable is_binary. On line 12-15 the shortest sequence is found to trim all sequences to the same length, as sequences cannot vary in size between batches in the same epoch.

Then the data needs to be arranged in a list sequence by sequence and trimmed, this is done on line 19-27, this yields a list containing 128 samples all trimmed to the same length.

For the binary model, the labels needs to be converted to high/low continuation desire, this is done on line 29-39 which loops through the labels to the data, first trimming the label sequence length to the shortest length as done to the samples before. Label 1,2,3,4 is converted to 0, and 5, 6, 7 to 1, the new labels is then appended to a labels list.

On line 44, the labels is converted to one hot encoding meaning a label of 0 meaning low continuation desire will be transformed to [1, 0] and a label of 1 would be [0, 1].

Then two step variables is set, current step to 0, which is the sample the training will start at, and valid_step to 125 which is where the sample for the validation will start.

Listing 9: Python code showing the data generator used in the training session (train_conation_class.py)

```
1
2  class DataGenerator(keras.utils.Sequence):
3      def __init__(self):
4          with open('data/predicted_conation_vid/predicted_conation.pkl', 'rb') as f:
5              self.x = pickle.load(f)
6
7          with open('data/conation_images/conation_images_labels_0.pkl', 'rb') as f:
8              self.y = pickle.load(f)
9
10         self.is_binary = True
11
12         self.shortest_seq = 18000
13         for sample in self.x:
14             if len(sample[0]) < self.shortest_seq:
15                 self.shortest_seq = len(sample[0])
16
17         print(self.shortest_seq)
18         self.concat_samples = []
19         for sample in self.x:
20             concat_seq_samples = []
21             for num in range(0, self.shortest_seq - 1):
22                 try:
23                     concat_seq_samples.append(np.concatenate((sample[0][num],
                           sample[1][num])))
24                 except:
25
26                     pass
27             self.concat_samples.append(concat_seq_samples)
28
29         self.labels = []
30         for label_seq in self.y:
31             tmp_label = np.asarray(label_seq[0:self.shortest_seq - 1])
32             tmp_label = np.where(tmp_label == 1, 0, tmp_label)
33             tmp_label = np.where(tmp_label == 2, 0, tmp_label)
34             tmp_label = np.where(tmp_label == 3, 0, tmp_label)
35             tmp_label = np.where(tmp_label == 4, 0, tmp_label)
36             tmp_label = np.where(tmp_label == 5, 1, tmp_label)
37             tmp_label = np.where(tmp_label == 6, 1, tmp_label)
```

```
38              tmp_label = np.where(tmp_label == 7, 1, tmp_label)
39              self.labels.append(tmp_label)
40
41          self.labels = np.asarray(self.labels)
42          self.concat_samples = np.asarray(self.concat_samples)
43
44          self.labels = to_categorical(self.labels)
45          print(self.labels.shape)
46          print(self.concat_samples.shape)
47          self.current_step = 0
48          self.current_valid_step = 125
49
50      def __len__(self):
51          pass
52
53      def __getitem__(self, index):
54          pass
55
56      def data_generation(self):
57          while True:
58              if self.current_step == 126:
59                  self.current_step = 0
60                  self.labels = self.labels.shuffle(random_state = 42)
61                  self.concat_samples = self.concat_samples.shuffle(random_state =
                        42)
62              X = self.concat_samples[self.current_step]
63              Y = self.labels[self.current_step]
64
65              self.current_step +=1
66              yield (np.expand_dims(X, axis=0), np.expand_dims(Y[0], axis=0))
67
68      def valid_data_gen(self):
69          while True:
70              if self.current_valid_step == 128:
71                  self.current_valid_step = 125
72              X = self.concat_samples[self.current_valid_step]
73              Y = self.labels[self.current_valid_step]
74
75              self.current_valid_step +=1
76              yield (np.expand_dims(X, axis=0), np.expand_dims(Y[0], axis=0))
```

Both the data_generation and valid_data_gen is based on the same logic, the first is for feeding batches during training and the latter is for feeding batches in the validation step. The method is a while loop, which always is true, first an if statement check what sample the training is currently at, if the step is 126, it will reset the step to 0 and begin from the start, moreover the sample sequences will be shuffled so the order of which the samples is presented is not the same. Then a sample sequence and corresponding labels is taken on line 62-63 and lastly those values is yielded as a return.

That means if the X sequence is 18,000 samples long, the corresponding Y sequence is 18,000 samples as well.

7.2.5 Training

The training of the model can be seen on code snippet 10, first on line 1 the model is initialised, then compiled with Adam as optimiser, binary_crossentropy as the loss function and accuracy as the metric. Then a DataGenerator object is instantiated, and that object is passed to the fit_generator method on line 5. Then step per epoch is set to the amount of training samples per epoch, and epochs set to 100. The same parameters is set on the validation data generator.

lastly the model weights is saved at line 7.

Listing 10: Python code showing training script for the recurrent model (train_conation_class.py)

```
1 model = lstm_model()
2 model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
3
4 generator = DataGenerator()
5 hist = model.fit_generator(generator.data_generation(), steps_per_epoch=126, epochs
      =100, validation_data=generator.valid_data_gen(), validation_steps=3)
6
7 model.save('models/conation_model/conation_cdnnLSTM_binary_v2.hdf5')
```

The training samples per epoch is the amount of samples the algorithm should look at before end an epoch, the value should correspond to the amount of samples in the data set for training, such as 126 for this data set.

This end the implementation of the continuation desire predictor, the next section will present the results from the tests conducted on the system specified in the methodology section.

7.3 Results

This section will present the results from the training of the algorithm, results on the interview regarding usage of the system in a real environment and lastly a validation test of the algorithms.

7.3.1 Training

The training of the continuation desire algorithm achieved a high accuracy of 95.1 % on the validation set and 95.2 % on the in sample training set. This speaks for a well generalised algorithm, where no over fitting issues are observed as the training accuracy does not exceed the validation accuracy by a significant amount.

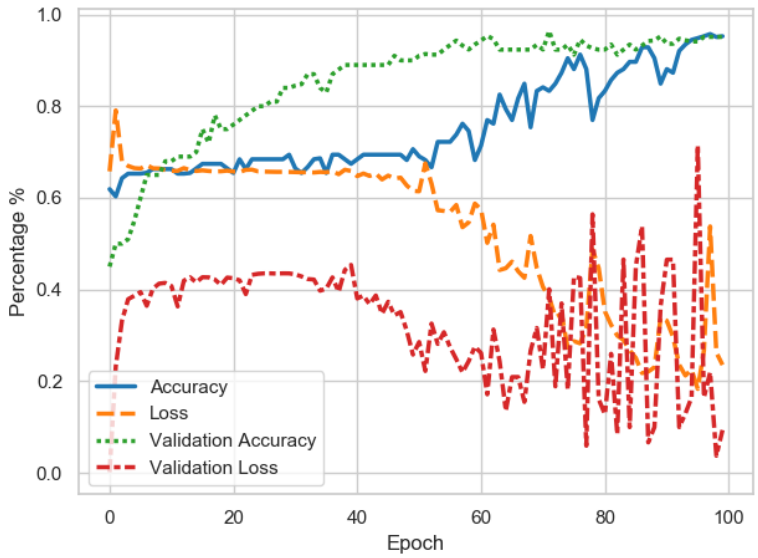


Figure 35: Figure showing training accuracy, training loss, validation accuracy and validation loss over epochs for the continuation desire model. Y-axis showing percentage for accuracies and continues numerical values for losses, X-axis is numerical for epochs.

Training time took around 14 hours over 100 epochs with 126 samples in each epoch, one batch consisting of one sample which then again consisted of 18000 sample points, corresponding to five minutes data sequences. It was observed that the loss ended with a significant amount of noise when the training went past 70 epochs, it seems that the network tried to over tune the weights in the network to achieve a higher accuracy, but the loss terrain is not a smooth curve and thus can end up noisy. This can indicate a problem in the underlying data, it could be an insufficient amount of data, and more data would solve that generalise-ability of the algorithm in high accuracy terrains. Training the algorithm for more than 100 epochs would not change the overall result, but a plateau would form where no additional improvements would be made. However overfitting could occur meaning that the model would fit the training data instead of making a generalised solution.

7.3.2 Validation

The validation tests goal was to validate both the machine learning algorithms developed in this thesis, in regards to their performance in a realistic scenario and to investigate if the algorithms could be used on a single player game as well. This will be done by using a game called "This is the only level" , which is a lightweight 2D graphics game where the player have to complete stages of the same level where small mechanics change from each stage. The test was conducted in accordance with the methodology section 5, under test 3.

The get an indication of the speed of the algorithms, the inference time for the emotion recognition system, a single frame takes 0.002 seconds to process making it indeed quick enough to run along side multiple other high load systems, such as a game. For the continuation desire predictor, it processes 18000 samples in 0.1 second, as this algorithm runs every five minutes, the slightly

larger overhead is not significant, and would still be able to run a long side a game. This means that both algorithm is able to run in real time, even along side a heavy graphical game.

The validation test was conducted over multiple days (22-23-24 of May), all in all 18 participants were tested (n=18), resulting in a mean play length of 8.2 minutes and a variance of 1.43 ($\bar{x} = 8.2, \sigma = 1.43$). This aggregates to 147 minutes of game play video. All participants completed all 15 stages of the game.

On figure 36 the age distribution of all participant can be seen, showing an age range from 22-30 ($\bar{x} = 25.61, \sigma = 1.94$). The sampling for the test, is, as said in the methodology section, convenience sampling as used in the continuation desire test. This means that the distribution of ages is much the same as in the continuation desire test session. It can be seen that most participants is about 25-26 years of age, while a couple of participants is of the start twenties, and late twenties. This can be seen by the spread of the variance, and outlier points on the graph, see 36.

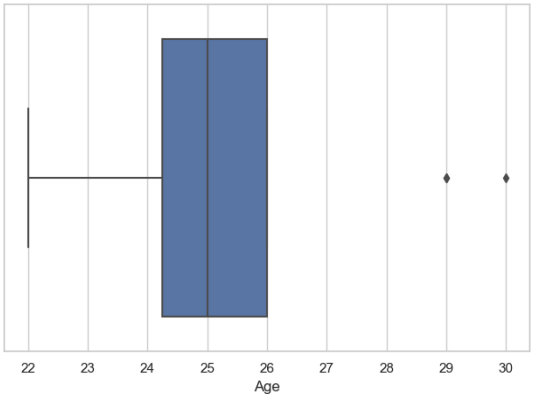


Figure 36: Graph showing box plot of the ages reported from the validation test on the game "This is the only level"

On figure 37, the participants reported weekly hours spent on gaming can be seen. It is seen that the general skill level of the participants is higher than the average, as it can be expected that most participants will complete all stages in a quickly manner.

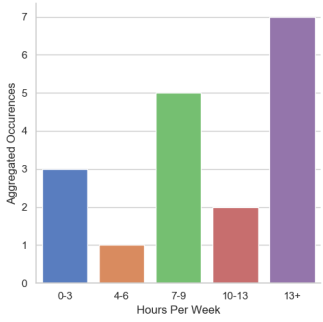


Figure 37: Gameplay time distribution from the validation test on the game "This is the only level"

On figure 38, the aggregated occurrences of the reported continuation desire levels can be seen. As seen in the reported continuation desire levels from the continuation desire test sessions (test 1), the levels in this test shows that the continuation desire is in the high area above four, this creates a negative skewed distribution.

On figure 39 a box plot show the mean continuation desire values reported from each participant. It shows all 15 stages at the x axis, and there is a tendency for the continuation desire to decrease over stages. This is assumed as the game is quite repetitive, and can become boring after several levels. It is also seen that in the first seven stages the reported continuation desire is stable, and does not variate much. Towards to end stages, the variance becomes larger, and here the players

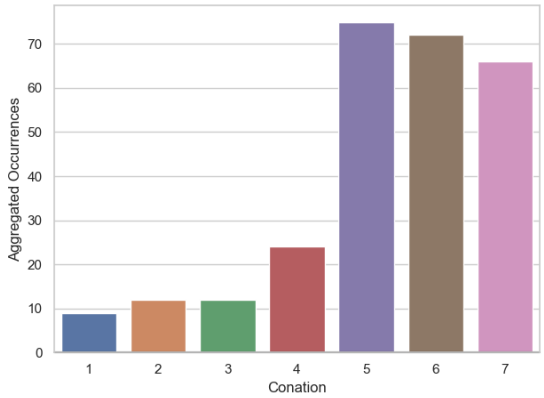


Figure 38: Graph showing histogram plot of continuation desire reported from the validation test on the game "This is the only level"

preferred game genre might come in to play. Some players do like the repetitive game play, where others do not, and if you are not too fond of the repetitive game style, the game will become boring, and thus feel disengaging. It is seen that the level of continuation desire decreases over time.

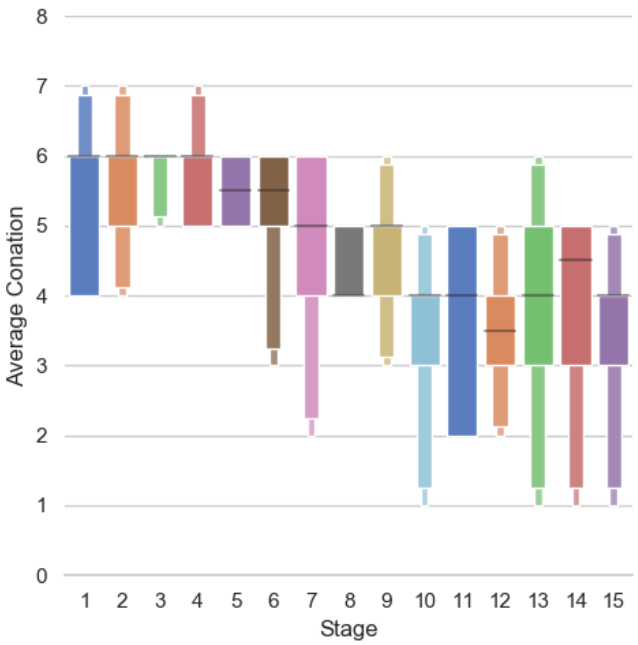


Figure 39: Graph showing box plot of continuation desire at each stage reported from the validation test on the game "This is the only level"

On figure 40, all the recognised emotions is seen as a bar chart. Three emotions was not spotted, namely calm, disgust and surprised. It is not surprising that these emotions were not captured, as the game play does not actively stimuli those emotions. Happy and sad were the most captured emotions. The sad response is surprising as the game does not create situations where it a sad response makes sense. The distribution however is different from the emotion distribution from the first test on Little Big Planet, where the most captured emotion was angry.

As mentioned in the methodology section 5, the emotion recognition system was validated as well. This was conducted by taking every 30th frame from the video taken of each participant and manually tagging the frame with the emotion from which the validator believes. The whole test have 132,387 frames with faces, every 30th of that is 4,412 frames which have been analysed, to

see if the recognition algorithm predicted correctly. The investigation revealed that the emotion recognition system predicted correctly 91.28% of the time, making the algorithm well generalised from the training. It was also noticed that, as mentioned in the 1st iteration result section, that the background, light and shadows was producing significant noise in the predicting, indicating that the algorithm is sensitive to the environment in which it is operating.

Moreover as sad is also seen almost as much as happy, it was noticed that the facial expressions when sad was predicted looked more like it was supposed to be neutral or frustration over the current stage.

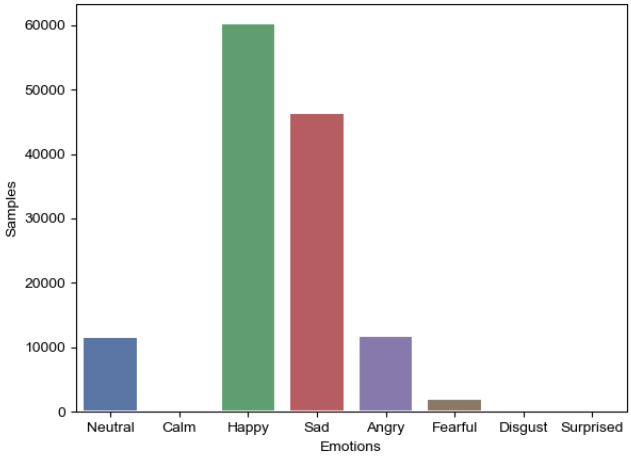


Figure 40: Graph showing the predicted emotions from the validation test on the game "This is the only level"

On figure 41, a graph showing the play through analysis of 15 stages of "This Is The Only Level". In this particular session, the continuation desire predictor have low continuation desire through out the session, but the participant had high continuation desire the first seven stages and thereon after low continuation desire. It also shows two strong periods of happiness, in between the two spikes, sadness rises. In this sadness spike, the participant had problems with completing a level. The participant looked frustrated which at times can look like a sad or angry facial expression, this frustration is something that the emotion recognition algorithm does not account for, and therefore subjects can be classified wrongly in certain situations.

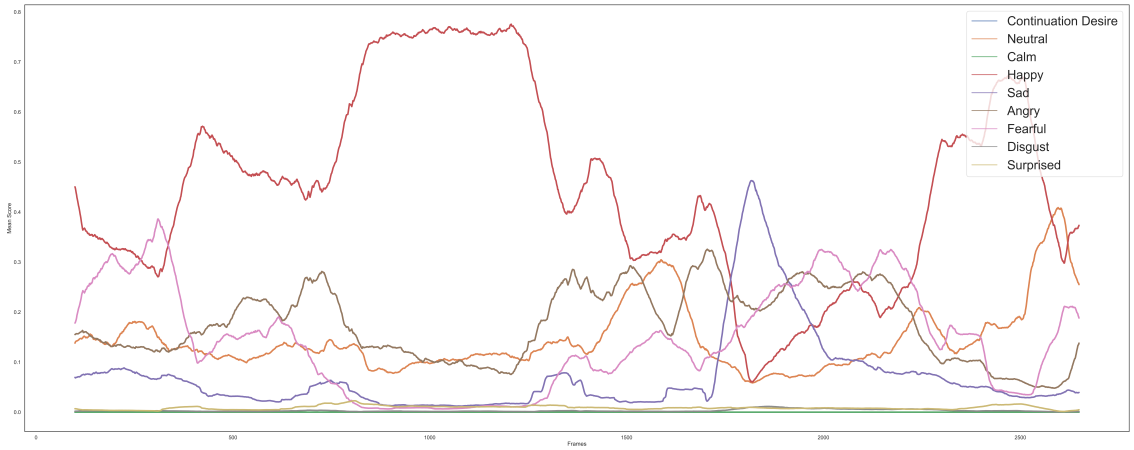


Figure 41: Graph showing a play through analysis for all 15 stages from the validation test on the game "This is the only level", the y-axis show the aggregated probability of the emotions

To analysis how well the continuation desire predictor performed on the test, the reported values were compared to the predicted values. A comparison of the percentages of low/high continuation desire from the algorithm and the reported was conducted, to see the percentages from

the algorithm see figure 42 and for the reported see figure 43. The figures were made by calculating the the percentages on high/low continuation desire of both the reported and predicted levels. The reported levels was converted from seven levels to high/low by binning them. Continuation desire level 1, 2, 3, 4 is low, while 5, 6, 7 is high. In the comparison analysis the algorithm predicted correctly 78.48% of the time, which is lower than observed during training on the test set. One reason why the accuracy is lower is that the domain in which the algorithm was used is not where it was meant to be used, as this is not a co operational game or multiplayer game. This means that there is no interaction between players which does increase a subjects will to express emotional responses. Taken that into account the algorithm does perform quite well for a real scenario even if the applied environment is out of context.

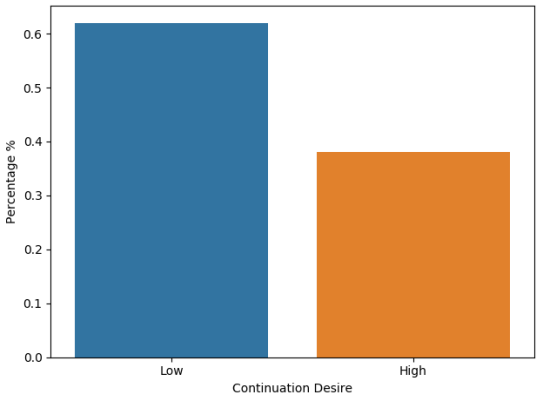


Figure 42: Barplot showing the predicted continuation desire percentages from the validation test on the game "This is the only level". The plot is made by how many of the predicted levels where high/low in percentages of all the data.

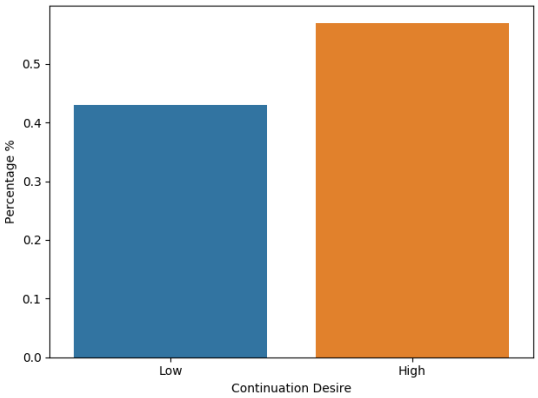


Figure 43: Barplot showing the true continuation desire percentages from the validation test on the game "This is the only level". The reported levels was converted from seven levels to high/low by binning them. Continuation desire level 1, 2, 3, 4 is low, while 5, 6, 7 is high. The plot is made by how many of the predicted levels where high/low in percentages of all the data.

It can be seen that the algorithm does predict less 'high' level of continuation desire relative to the true reported level from the subjects (38% high predicted versus 58% high reported), this can be explained with that the subjects had more periods with neutral or sad, which the continuation desire predictor takes as low continuation desire. This supports the hypothesis about that the subjects does not express as many reactions when playing single player games. This is also discussed in the analysis, where it is said that when a person is alone, emotional expression is far more uncommon than when interacting with other persons. Moreover the game is also a puzzle game, which can introduce high cognitive load, which does suppress facial expressions as well.

7.3.3 Case Study - Interview

The following section will present results regarding the case study about game testing at Invisible Walls. The interview was conducted in accordance with the methodology section 5, and as stated, the reason for conducting this interview was to see if there is a need for a deeper insight into the player in relation to emotions and continuation desire measured unobtrusively. The goal is to either confirm or not the final problem statement relating to this subject.

The interview with Invisible Walls Creative Director Sebastian Hurup Bevensee was conducted on the 16th May. The whole transcribed interview can be found in the appendix 11.9. In the following section, differnt parts from the interview will be quoted and highlighted, and the answers discussed in relation to the goal of the interview.

The first part of the interview questions was about how Invisible Walls use game testing in their production to validate the different mechanics and use these results to further develop their game. Their current testing methodology is simple, as their production is still in early stages, where they are still testing mechanics in the game, and seeing how a player interact with other players. As written in the next quote, their focus right now is to see reactions of the players when

playing the game, both reactions concerning the mechanics and level design of the game, but also if they use the key mechanic in their game, communication. As mentioned the production is still in early stages, therefore the testing session up until now have been focusing on how the players are playing the game, and if they understand what to do. Also seeing that the players have fun and are happy, as this is a good indicator for their game.

> **Question - "What is your current test methodology"**
> "It was that method* we used to see their reactions, and listen to the communication that they made, and see if they did what they had to do, and if they could make sense of how to play the game. So we tested if the players did what we wanted them to do, because we are quite early in the process of the production, and the only think that we need right now is an indication about if the mechanics works or not. So we not need data from large questionnaires yet" - Appendix 11.9. *Using game capture and facial capture to see reactions afterwards.

The results from their test sessions is mainly used for indications of what works and what needs to be changed in order to the game to be understood better, but also how the flow through the game is, and when and how information is presented to the players. Their main indicator is happiness throughout the play session, but they also use a variation of continuation desire to see if the players would return and play again, the quote below refers to this.

> **Question - "What do you use your results from a testing session for?"**
> "That the participants is having fun while playing the game, and actually a similar method to continuation desire, that we asked them afterwards if they want to continue playing, or if they want to play the game again. And usually people really want to continue playing, and actually after that test session, the participants asked to play more games even though we had all the data we needed. And that was a really good indication that the game was actually very fun and engaging to play." - Appendix 11.9.

As seen, their testing session often went well, and players wanted to play more even after the test sessions were over. They afterwards also asked the participants if they wanted to play the game again, a similar methodology to continuation desire, which usually indicates how well replay-ability and retention is in the game.

> **Question - "Up until now you have tested the same way, with letting people play and recording them and then observing. What are your plans for future testing, do you plan to have a deeper testing like you did at Aporia?"**
> "Yes, we are going to create a deeper test, when the production moves forward. And then we will use the same kind of questionnaires as with Aporia*. When its a social game, then its actually just our job to ensure that its fun socially in-game, and that is pretty easy to measure, as its easy to observe. What is really hard is to measure how fun is it after you have played the game 30-40-50 times" - Appendix 11.9. *Reference to self developed questionnaires about the experience and narrative, and the understanding thereof.

Mentioned in the last quote, their testing scheme is of a non complex nature, but as the production moves forward, the tests will get more and more complex. And when the time is right, a deeper dive into how the players are feeling will be done with questionnaires, very similar to how they tested their previous released game Aporia. With Aporia, they had several testing sessions where players would play for up to four hours. The testing would end in questionnaires with regards regarding mechanics, narratives, game behaviour, visuals. One of the things Sebastian Bevensee wants to test is how fun it is to play the game over and over. It is just as fun when you have played the game 50 times? This problem requires a long testing method, and a controlled setup, with a data pipeline which can measure this over time.

> **Question - "We have touched upon it a little bit, but do you think that there is a need for deeper insights into that player other than just video and in-game metrics when testing your game? Especially in relation to what I am developing?"**

> "Totally, totally, we have not tracked emotional responses per say, we have just looked at the overall response. As long as they laughed and had fun, that was enough for us. Also because if you see some negative expressions, then you could pinpoint if its something in the game in terms of mechanics or if its something with the communication" - Appendix 11.9.

According to Sebastian, they could use some software to track emotion trough a testing session, as they right know do it themselves from the video data but their method is just to observe the overall responses. It was mentioned that they both want to see the overall response to the game, but would also want to dive deeper into specific situations, to pin point what cause the specific emotional response. This is to locate which areas, props or game mechanics are working in the game and which are not, and at their current development stage, this is very important.

At this stage of the interview the data for three different participants in the testing session was visualised and discussed, one of the figures can be seen at figure 44, the other two is in the appendix 11.10.

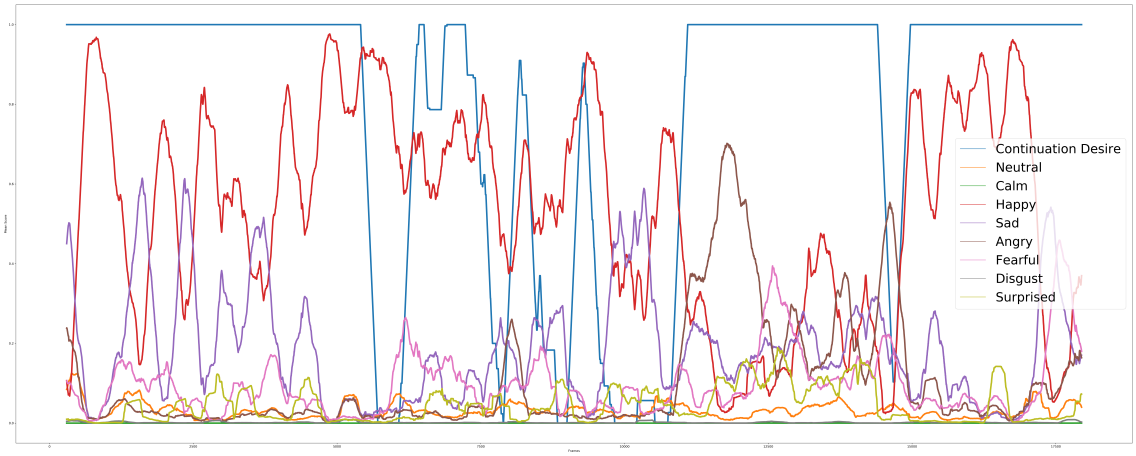


Figure 44: Graph showing the analysis of a play through from one participant from the test session. Eight emotions is tracked and continuation desire as well.

The graph is showing the participants emotional responses over the course of a play session being 13 minutes long. At around 11,000 frames the participants dies within the game, which shows a strong reaction and a flip between angry and happy, and also a spike in surprise, see 45. The overall play session shows a player which is very engaged with the game, and is happy throughout the whole session up until the death.

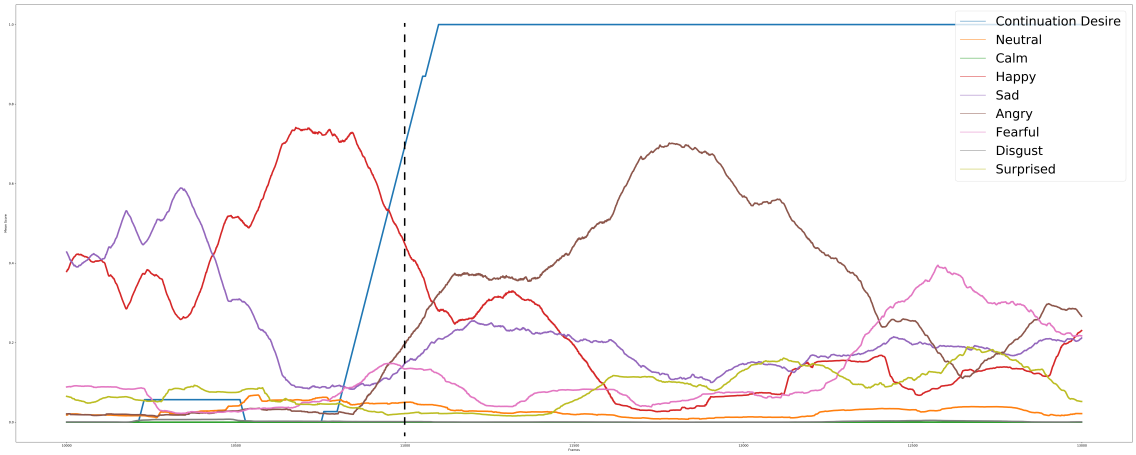


Figure 45: Graph showing a zoomed in version of figure 44. The area is around where the participant was voted off (dotted line).

In response to seeing this graph, Sebastian Bevensee says.

"Yeah, she is very happy about the game, and really likes to play it, it can clearly be seen that she is very engaged. I notice some angry spikes, I can not help but notice that she looks a bit angry when she thinks or focuses*. Angry is just a hard emotion,as it can look like others as well, such as puzzled. Its very interesting that you can see when she was voted off that strongly, the data clearly shows the shift in emotions." - Appendix 11.9. *Based on video footage and being present at the actual test.

Which a good observation, that when a human is puzzled a reaction can many times be seen as angry, as there is a high cognitive load, as mentioned in the analysis, under these conditions, emotional expressions tend to mean to a more neutral state, but a soft puzzled response, can results in an angry classification. But Sebastian is also surprised that spikes and parts of the graph relates closely the the video, and of what he sees in the video. And follows with

Interviewer response - "And as you can see her continuation desire is full on the whole experience, and that is because she is very happy at all times. And the primary metric for continuation desire is happiness." "Yes that makes sense, a little note is that I believe that when you have different emotions spiking at all times, means that the player is engaged, as his emotions is being activated." - Appendix 11.9.

As seen in the graph as well, there is primary emotional responses, from especially happy, but other emotions is activated as well, and as Sebastian says, being able to spike many different emotions, could indicate high engagement, as the game makes the play react.

To get a better indication of how this type of software could be used in a game production, a question about the use of such software was asked.

Question - "Does this kind of data and deeper analysis make sense in terms of game testing, and especially your production?" "Yes I think that we could use something like this in the production ... also to test specific mechanics, we could definitely use something like this. And we would like to see that happiness and continuation desire would rise over time spend in the game. It would able us to pinpoint more accurately what kind of mechanics works, but also to control the tempo of the game, to see if somethings should be done slower or faster. It could help create a form of drama manager, that could control the flow of the game. But also over a period of games, to see how their responses changes from the first game to their 40th game." - Appendix 11.9.

Sebastian believes that a software product which is able to output data like presented could be of high value in a production, especially to conduct tests on level design, prop placement and flow in terms of emotions responses, and on a higher level, use continuation desire to evaluate longer game sessions in respect to the overall game play, and how the game unfolds. As he mentions, it is also important to investigate the replay ability or retention of players, so tracking their continuation desire, and emotional responses over time is equally important.

In summation, the response received from the prototype demo and the data it created of a test session is good, and the usage of such a system is possible in a game company such as Invisible Walls, but it depends on the nature of the game and what kind of mechanics is present, and how emotionally the game binds the players. In the discussion chapter the results will be reflected on.

This concludes the results sections on the case study with Invisible Walls and the validation test on "This Is The Only Level", the results look promising, and generally support the final problem statement and the list of requirements. The next section will discuss and reflect the results and the thesis as a whole.

8 Discussion

In the following section a discussion of the thesis as a whole, what results and new knowledge have been created, and what could have been done to enhance the method, development and tests. The goal of the thesis is to investigate to which extend it is possible to infer a players continuation desire and emotions during an interactive experience. This had to be done by only using a video feed, and do this analysis in real time as well. Lastly it was also in scope to investigate whether this data driven approach into the players of game is use-able when evaluating an interactive experience.

The developed emotion recognition system achieved a high accuracy of 98% which is, compared to other algorithms in the research [Moore and Bowden, 2011, Tawari and Trivedi, 2013, Shan et al., 2009, Arriaga et al., 2017, Goodfellow et al., 2013, Rashid et al., 2013], a significant increase to a point where it is significantly better than humans. Humans can recognise micro-expressions 47% of the time [Matsumoto and Willingham, 2009, mat, 2013], and macro-expression to a degree of 84% [Matsumoto and Willingham, 2009, mat, 2013] in real life scenarios. The algorithm was validated in the 2nd iteration, and ended at a 91.28% accuracy in a real world test, which is excellent performance compared to 80.17% [Moore and Bowden, 2011], 86.9% for [Shan et al., 2009], 78% for [Tawari and Trivedi, 2013], 66% for [Arriaga et al., 2017], 71.16% for [Goodfellow et al., 2013], 88.8% [Albanie et al., 2018] and 74.15% for [Rashid et al., 2013]. For the mentioned papers, they all have data quality and sufficient data set in common, as mentioned in the analysis many papers use data set which consist of less than 1000 samples, and with even fewer different persons, this means that the algorithms learns the data, and the persons in the data set. This is a significant bias, which is common for older papers. This introduced a bias the the accuracies, meaning that the proposed solutions only worked on internal data sets, and not in real scenarios. This thesis have used 2,000,000 images on 500,000 + different persons, which makes the final results more valid than tests on smaller data sets. But the results between papers is not directly comparable as no benchmark data set have been made for emotion recognition as of yet, and this is a problem for comparing different approaches and algorithms. This is a common practise in other fields within machine learning, such as Natural Language processing, object detection and landmark detection.

It is noted that in the first test to gather data about players emotional response while playing, that a large portion of the emotional responses were angry, which is interesting, as seeing the video data post test, shows that the players where not angry in large portions of the time. This can indicate an error in the test setup, where the light and background during the play test were disturbing the algorithms internal systems to such a degree that some emotions were more prone to occur than others, and emotions being misclassified.

This problem is a general problem in machine learning, as algorithms have assumptions about the data which comes from the training data, if key aspects of the data changes a little it can have large implication of the output of the algorithm. To account for this, more work needs to be done with the preprocessing of the images, so light and shadows do not bias the image to such a degree that facial expressions is being misclassified. A thorough investigation should have been conducted into the data of the data gathering test to see the actually cause of the errors. But from the case study and validation test, the results were much more accurate, one reason for this could be the resolution and placement of the camera. In both the case study and validation test from the 2nd iteration have cameras placed much closer to the participants thus giving a more detailed facial expression. Both these setups the participants were no longer than 50 centimetres away, while at the data gathering test from the 1st iteration participants were around 150 centimetres away from the camera, which could introduce bias in the data. An interesting approach could be to use another form of camera to capture the facial expressions, such as a depth camera or using infrared light to accommodate the problems of shadow and irregular light on subjects faces.

The data used for the emotion recognition system, is of high quality, but observed from the data, many of the expression is of high intensity, meaning that expression is over exaggerated that would not occur in a real environment, this was mostly from the FER data set, which had 35,000 samples. A data set formed from playing persons emotional responses would be of greater value and possibly create a more specialised algorithm for interactive experience emotion recognition.

With these errors the network overall performed well, and within the specified accuracy from the methodology section as it is possible to recognise emotions from real time video data to a high degree, beating humans on eight emotions.

The continuation desire algorithm achieved high accuracy as well, 95.2% was the final accuracy on the test set from training, which is higher than previous work done in this area. Because the

algorithm was trained on data from the data gathering test from the 1st iteration, where bias in the test environment have made the data biased in terms of predicted emotions, the algorithm may have learned wrong underlying relations between emotions and continuation desire. This error in the data could implicate the continuation desire algorithm to an extend were the output is wrong. But the validation test shows that it still perform relatively well in a out of sample environment to an extend where it is acceptable, but it is not as close to the original test set accuracy as seen with the emotion recognition algorithm. But an accuracy of 78.48% is still within acceptable, but not as close as specified in the methodology section. But further testing needs to be done to verify the in-sample environment accuracy of the continuation desire algorithm, as it is predicted that it will perform better on a multiplayer or interactive game.

The continuation desire algorithm is trained on only 12 subjects data, which is to a degree sufficient, but with more data, the algorithm would be able to accommodate more player types and expressional styles. More data would also make the algorithm more robust to different emotional changes which happens in games, as games have different flows and tempo which affects the emotional response. Capturing this data is time consuming but it is needed to create an even better performing algorithm. Using one type of game also comes with risk, as each game produces different emotional curves, testing on different games should also be done to minimise the the classification bias which is introduced by only using one game.

The eight emotions that have been used as outputs from the emotion recognition model, was chosen because of the basic emotions, happiness, fear, sad, surprise, anger and disgust, these emotions is the most occurring ones, but humans are not expressing emotions at all times, therefore neutral and calm was added to accommodate this behaviour. But from hours of video data from game playing, it becomes more apparent that those expressions might not be enough, as frustration and focus is facial expressions, that is very important to game play, especially in single player games, as this typically means that the player is engaged. But these emotions is very close to other emotions as well, such as frustration to sadness and anger, and focus to calm and neutral, distinguishing between these expressions is difficult. This could possibly be done by another approach such as outputting arousal/valence values instead, so that each emotion have a numerical value instead, which could create a more fine incremental steps between emotions. An intensity index on each emotions could also enhance the interpret-ability of the predicted emotion, to see if the person is very angry or little angry.

Gathering these two algorithms and using them to infer emotions and continuation desire from video data was done in the case study with Invisible Walls in the 2nd iteration. The overall points from the interview is that they are currently still in early stages of their development, but they could see that a data driven approach like this could give great value in their tests, even in the short usability tests. Because of the only requirement for the system is a video camera, no additional equipment for a expensive and complex setup is needed, it becomes attractive to smaller studios as well. The use case is only based on one interview with one game studio, but the information gathered leads to that larger game studios could need a system as this as well. This approach is largely seen in the marketing sector where emotion recognition is used extensively to see responses from test group regarding a commercial or product.

In summation, the developed algorithms and evaluation framework proves that it is indeed possible to predict emotions and continuation desire in real time using only video, and that the data formed can be of use in a game studio. Meaning that the final problem statement is proven to an extend where a system is use-able in its core, and with different tweaks and additional features it could work as a product for testing and evaluation of interactive experiences.

9 Conclusion

The purpose of this project was to investigate the possibility to measure and classify emotions and continuation desire from video in real time by using state of the art and novel machine learning approaches. The proposed solution is a emotion recognition system which perform better than state of the art algorithms and humans, and is trained on the largest publicly known emotion recognition dataset.

The results of the thesis is an emotion recognition algorithm which performs in static test sets 98% accuracy and in a real scenario 91.28%, which is quite high, even with the problems in the emotion recognition data. A continuation desire algorithm which performed 95.2% of accuracy and 78.48% on an out of environment test, suggesting an algorithm which is well generalised and use-able on different game genres and styles. As the case study shows that such a system is also use-able in a company suggests that there is a gap in both the research and market, which now have performed research, with a high performing solution.

It can therefore be concluded that the thesis successfully created an emotion recognition and continuation desire predictor algorithm which is use-able in real time and use-able in game testing and create value for the testing team during development. With smaller changes the algorithms could perform better and achieve higher accuracies, while making them more robust to changes in the environments surrounding the participants.

10 Future Work

The prototype developed in this project, can perform what the project set out to solve. But alterations can be made to make the system, and algorithms perform better. Even though the six basic emotions is said to be universal, there is still a large variance in the shape and placement of eyes, mouth, eye brows and nose on faces that can bias the algorithm into classifying incorrectly. One way to solve this problem, is to create a baseline calibration at the start of using the system, this could be done in multiple ways, but one way could be to ask users to do the eight different emotions sequentially on video beforehand, this could allow the algorithm to adapt to the user and be more confident on the classifications that it make. This can also diversify the algorithm, to skin tones differences and cultural differences as the algorithm right now is trained on mostly Caucasian individuals.

One features which was proposed in the case study interview, was to be able to navigate the game play video to pinpoint specific areas of interest while seeing an indication on the emotion and continuation desire response graph to more easily locate spikes. In relation to this, the graphs is complex and there is a lot of information pressed into a confined space, one way to bypass this is to make a zooming function which would allow the user to zoom into the graph, thus making it more granular the further you zoomed. This would make the graph of the whole play session smooth (through rolling mean and tangent normalisation) and easier to interpret, and when looking at specific parts of the play through, you could zoom in and get a more complex graph and insights.

Another feature which would create great value would be automatic identification of events in the game, such as killed another player, or died, or solves this task or quest. A machine learning algorithm could potentially solve this by analysing the game play and tagging sequences with labels of what happened. This can make the navigation and analysis of the play session more access-able and use-able by testers. This meta data extraction could as well be used for continuation desire prediction, as events in the game could give strong indications of a players engagement.

As the algorithms can run in real time, the data produced from the algorithms could be used in adaptive game play and narratives structures. A use case of this, is that the environment that you play in will change depending on the responses you produce to different areas of the environment, such as items or atmospheres that you have a good reaction to. Another approach could be to have the narrative adaptive change to your reactions to different story elements within the game, so that the story and behaviour of NPC's will change depending on your emotional response.

As for continuation desire, an interactive experience could adapt so that the player/viewer is more engaged as the content will change to keep continuation desire high throughout the experience. Creating more personalised experiences and stories which is heavily depended on the user.

Acknowledgements

Thanks to Henrik Schønau Fog for supervising this project. Thanks to the Samsung Media Innovation Lab for Education (SMILE) for providing computer hardware, screens and smartphones. Thanks to Invisible Walls for using time to interview about their testing methods.

Notes

[1] http://www.human-memory.net/processes_recall.html
[2] https://www.statista.com/statistics/862278/global-video-game-revenues-worldwide/
[3] https://www.statista.com/topics/964/film/
[4] https://imotions.com/
[5] http://modl.ai/
[6] https://www.crowdemotion.co.uk/
[7] https://sensum.co/
[8] https://www.affectiva.com
[9] http://sspnet.eu/avec2014/
[10] https://mionix.io/products/naos-qg
[11] http://jalammar.github.io/visual-interactive-guide-basics-neural-networks/
[12] https://medium.com/@julian.harris/stochastic-gradient-descent-in-plain-english-9e6c10cdba97
[13] http://cs231n.github.io/convolutional-networks/
[14] https://store.myndplay.com/products.php?prod=28
[15] https://mionix.net/naos-qg
[16] https://tobiigaming.com/eye-tracker-4c/
[17] https://tobiigaming.com/games/
[18] https://www.playstation.com/en-us/games/littlebigplanet-3-ps4/
[19] https://www.logitech.com/en-us/product/hd-webcam-c525?crid=34
[20] https://obsproject.com/
[21] https://www.invisiblewalls.co/
[22] https://www.invisiblewalls.co/project-cainwood
[23] https://store.steampowered.com/app/573130/Aporia_Beyond_The_Valley/
[24] https://www.invisiblewalls.co/project-cainwood
[25] https://www.invisiblewalls.co/project-cainwood
[26] https://obsproject.com/
[27] http://www.onemorelevel.com/game/there_is_only_one_level
[28] https://www.python.org/
[29] https://keras.io/
[30] https://www.tensorflow.org/
[31] https://www.limelight.com/resources/white-paper/state-of-online-gaming-2018/
[32] https://www.python.org/
[33] https://www.numpy.org/
[34] https://pandas.pydata.org/
[35] https://keras.io/
[36] https://www.tensorflow.org/

References

Foundations for modelling emotions in game characters: Modelling emotion effects on cognition, 2009. ISBN 978-1-4244-4800-5. doi: 10.1109/ACII.2009.5349473.

Usage of emotion recognition in military health care, 2011. ISBN 978-1-4244-9276-3. doi: 10.1109/DSR.2011.6026823.

Audio visual cues in driver affect characterization: Issues and challenges in developing robust approaches, 2011. ISBN 978-1-4244-9635-8. doi: 10.1109/IJCNN.2011.6033615.

Nonverbal Communication: Science and Applications AU - Matsumoto, David AU - Hwang, Hyi Sung. SAGE Publications, Inc., Thousand Oaks, 2013. doi: 10.4135/9781452244037. URL `http://sk.sagepub.com/books/nonverbal-communication`.

Emotion recognition from audio signals using Support Vector Machine, 2015. ISBN 978-1-4673-6670-0. doi: 10.1109/RAICS.2015.7488403.

Pose invariant method for emotion recognition from 3D images, 2015. ISBN 978-1-4673-7399-9. doi: 10.1109/INDICON.2015.7443288.

Samuel Albanie, Arsha Nagrani, Andrea Vedaldi, and Andrew Zisserman. Emotion recognition in speech using cross-modal transfer in the wild. *arXiv preprint arXiv:1808.05561*, 2018.

Ding Hooi Ting Amir Zaib Abbasi and Helmut Hlavacs. Engagement in games: Developing an instrument to measure consumer videogame engagement and its validation. *International Journal of Computer Games Technology*, 2017, 2017.

Octavio Arriaga, Matias Valdenegro-Toro, and Paul Ploger. Real-time convolutional neural networks for emotion and gender classification. *CoRR*, abs/1710.07557, 2017.

Richard Bartle. Hearts, clubs, diamonds, spades: Players who suit muds. *Journal of MUD research*, 1(1):19, 1996.

Dmitri Bitouk, Ragini Verma, and Ani Nenkova. Class-level spectral features for emotion recognition. *Speech Communication*, 52(7):613 – 625, 2010. ISSN 0167-6393.

Alexander Bogner, Beate Littig, and Wolfgang Menz. *Introduction: Expert Interviews — An Introduction to a New Methodological Debate*, pages 1–13. Palgrave Macmillan UK, London, 2009. ISBN 978-0-230-24427-6. doi: 10.1057/9780230244276_1. URL `https://doi.org/10.1057/9780230244276_1`.

Margaret M Bradley, Laura Miccoli, Miguel A Escrig, and Peter J Lang. The pupil as a measure of emotional arousal and autonomic activation. *Psychophysiology*, 45(4):602–607, 2008.

Amy Brett, Melissa Smith, Edward Price, and William Huitt. Overview of the affective domain. *Educational Psychology Interactive*, pages 1–21, 2003.

S.L. Brown and C.C. Vaughan. *Play: How it Shapes the Brain, Opens the Imagination, and Invigorates the Soul*. Avery, 2009. ISBN 9781583333334. URL `https://books.google.dk/books?id=ESQDsgqfgusC`.

Daniel Bruneau, M Angela Sasse, and JD McCarthy. The eyes never lie: The use of eye tracking data in hci research. In *Proceedings of the CHI*, volume 2, page 25. Citeseer, 2002.

Buchanan, Richard Buchanan, and Mihaly Csikszentmihalyi. Flow: The psychology of optimal experience. 8:80–, 1991. ISSN 0747-9360.

John T Cacioppo, Louis G Tassinary, and Gary Berntson. *Handbook of psychophysiology*. Cambridge University Press, 2007.

A John Camm, Marek Malik, JT Bigger, Sergio Breithardt, Cerutti, Richard J Cohen, Philippe Coumel, Ernest L Fallen, Harold L Kennedy, RE Kleiger, et al. Heart rate variability. standards of measurement, physiological interpretation, and clinical use. *European heart journal*, 17(3):354–381, 1996.

C. Campbell and Y. Ying. *Learning with Support Vector Machines*. Synthesis lectures on artificial intelligence and machine learning. Morgan & Claypool, 2011. ISBN 9781608456161.

Linlin Chao, Jianhua Tao, Minghao Yang, Ya Li, and Zhengqi Wen. Long short term memory recurrent neural network based multimodal dimensional emotion recognition. In *Proceedings of the 5th International Workshop on Audio/Visual Emotion Challenge*, AVEC '15. ACM, 2015.

Jenova Chen. Flow in games (and everything else). 50:31–34, 2007. ISSN 1557-7317.

Wai Ki Rebecca Cheng. *Relationship between visual attention and flow experience in a serious educational game: An eye tracking analysis*. PhD thesis, George Mason University, 2014.

Thomas Christy and Ludmila I. Kuncheva. Technological advancements in affective gaming: A historical survey. *GSTF International Journal on Computing (JoC) Vol.3 No.4*, 2013.

Roddy Cowie, Naomi Sussman, and Aaron Ben-Zeev. Emotion: Concepts and definitions. In *Emotion-oriented systems*, pages 9–30. Springer, 2011.

Charles Darwin. *The expression of the emotions in man and animals*. New York ;D. Appleton and Co.,, 1872.

Thomas Debeauvais. Challenge and retention in games. 2016.

Andrew Dekker and Erik Champion. Please biofeed the zombies: Enhancing the gameplay and display of a horror game using biofeedback. In *DiGRA Conference*, 2007.

Craig DeLancey. *Handbook of Research on Synthetic Emotions and Sociable Robotics*. 2009. ISBN 1-60566-354-9. doi: 10.4018/978-1-60566-354-8.

Anders Drachen, Lennart E Nacke, Georgios Yannakakis, and Anja Lee Pedersen. Correlation between heart rate, electrodermal activity and player experience in first-person shooter games. In *Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games*, pages 49–54. ACM, 2010.

Andrew T Duchowski. Eye tracking methodology. *Theory and practice*, 328, 2007.

M. Easterby-Smith, R. Thorpe, and A. Lowe. *Management Research: An Introduction*. SAGE series in Management Research. SAGE Publications, 2002. ISBN 9780761972853. URL `https://books.google.dk/books?id=EczlVa2192gC`.

Darragh Egan, Sean Brennan, John Barrett, Yuansong Qiao, Christian Timmerer, and Niall Murray. An evaluation of heart rate and electrodermal activity as an objective qoe evaluation method for immersive virtual reality environments. In *Quality of Multimedia Experience (QoMEX), 2016 Eighth International Conference on*, pages 1–6. IEEE, 2016.

Paul Ekman. Body position, facial expression, and verbal behavior during interviews. 68:295, 1964. ISSN 0096-851X. doi: 10.1037/h0040225.

Paul Ekman. An argument for basic emotions. *Cognition and Emotion*, 6(3-4):169–200, 1992.

Paul Ekman. Emotions revealed. 12, 2004. ISSN 0966-6494. doi: 10.1136/sbmj.0405184.

Nico H Frijda. *The emotions*. Cambridge University Press, 1986.

Chris Frith. Role of facial expressions in social interactions. *Philos Trans R Soc Lond B Biol Sci*, 364(1535):3453–3458, Dec 2009.

T. Fullerton. *Game Design Workshop: A Playcentric Approach to Creating Innovative Games, Third Edition*. CRC Press, 2014. ISBN 9781498785877.

Ian J. Goodfellow, Dumitru Erhan, Pierre Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, Yingbo Zhou, Chetan Ramaiah, Fangxiang Feng, Ruifan Li, Xiaojie Wang, Dimitris Athanasakis, John Shawe-Taylor, Maxim Milakov, John Park, Radu Ionescu, Marius Popescu, Cristian Grozea, James Bergstra, Jingjing Xie, Lukasz Romaszko, Bing Xu, Zhang Chuang, and Yoshua Bengio. Challenges in representation learning: A report on three machine learning contests. In Minho Lee, Akira Hirose, Zeng-Guang Hou, and Rhee Man Kil, editors, *Neural Information Processing*, pages 117–124, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

Alex Graves, Abdel rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013.

Dan Witzner Hansen and Qiang Ji. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE transactions on pattern analysis and machine intelligence*, 32(3):478–500, 2010.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

Ernest R Hilgard. The trilogy of mind: Cognition, affection, and conation. *Journal of the History of the Behavioral Sciences*, 16(2):107–117, 1980.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. 9:1735–80, 12 1997.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL `http://arxiv.org/abs/1608.06993`.

Jian Huang, Ya Li, Jianhua Tao, Zheng Lian, and Jiangyan Yi. End-to-end continuous emotion recognition from video using 3d convlstm networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6837–6841. IEEE, 2018.

Eva Hudlicka. To feel or not to feel: The role of affect in human–computer interaction. *International Journal of Human-Computer Studies*, 59:1 – 32, 2003.

Eva Hudlicka. Affective game engines: Motivation and requirements. *Proceedings of the 4th International Conference on Foundations of Digital Games*, 2009.

William Huitt and S Cain. An overview of the conative domain. *Educational psychology interactive*, pages 1–20, 2005.

Wijnand Ijsselsteijn, Wouter van den Hoogen, Christoph Klimmt, Yvonne De Kort, Craig Lindley, Klaus Mathiak, Karolien Poels, Niklas Ravaja, Marko Turpeinen, and Peter Vorderer. Measuring the experience of digital game enjoyment. *Proceedings of Measuring Behavior*, 2008.

David E Irwin. Visual memory within and across fixations. In *Eye movements and visual cognition*, pages 146–165. Springer, 1992.

E. H. Jang, B. J. Park, S. H. Kim, Y. Eum, and J. H. Sohn. Identification of the optimal emotion recognition algorithm using physiological signals. pages 1–6, Nov 2011.

Charlene Jennett, Anna L. Cox, Paul Cairns, Samira Dhoparee, Andrew Epps, Tim Tijs, and Alison Walton. Measuring and defining the experience of immersion in games. *Int. J. Hum.-Comput. Stud.*, 66(9):641–661, September 2008. ISSN 1071-5819.

Quan Huynh-Thu Julien Fleureau, Philippe Guillotel. Physiological-based affect event detector for entertainment video applications. *IEEE Transactions on Affective Computing*, 2012.

Marcel A Just and Patricia A Carpenter. A theory of reading: From eye fixations to comprehension. *Psychological review*, 87(4):329, 1980.

Kostas Karpouzis and Georgios N. Yannakakis. *Emotion In Games*. Springer, 2015.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

Peter J Lang, Mark K Greenwald, Margaret M Bradley, and Alfons O Hamm. Looking at pictures: Affective, facial, visceral, and behavioral reactions. *Psychophysiology*, 30(3):261–273, 1993.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

Chungk Lee, SK Yoo, Yoonj Park, Namhyun Kim, Keesam Jeong, and Byungchae Lee. Using neural network to recognize human emotions from heart rate variability and skin resistance. *Engineering in Medicine and Biology 27th Annual Conference*, 2005.

Matthew Lombard and Theresa Ditton. At the heart of it all: The concept of presence. *Journal of Computer-Mediated Communication*, 3(2):0–0, 1997.

Regan Mandryk and Margaret Atkins. A fuzzy physiological approach for continuously modeling emotion during interaction with play technologies. 65:329–347, 2007a.

Regan L Mandryk and M Stella Atkins. A fuzzy physiological approach for continuously modeling emotion during interaction with play technologies. *International journal of human-computer studies*, 65(4):329–347, 2007b.

Qi-rong Mao, Xin-yu Pan, Yong-zhao Zhan, and Xiang-jun Shen. Using kinect for real-time emotion recognition via facial expressions. *Frontiers of Information Technology & Electronic Engineering*, 16(4):272–282, 2015. doi: 10.1631/FITEE.1400209.

Gilleade Kiel Mark, Dix Alan, and Allanson Jen. Affective videogames and modes of affective gaming: Assist me, challenge me, emote me. In *DiGRA - Proceedings of the 2005 DiGRA International Conference: Changing Views: Worlds in Play*, 2005. ISBN ISSN 2342-9666.

David Matsumoto and Bob Willingham. Spontaneous facial expressions of emotion of congenitally and noncongenitally blind individuals. *Journal of personality and social psychology*, 96:1–10, 02 2009. doi: 10.1037/a0014037.

Rollin McCraty, Mike Atkinson, William A Tiller, Glen Rein, and Alan D Watkins. The effects of emotions on short-term power spectrum analysis of heart rate variability. *The American journal of cardiology*, 76(14):1089–1093, 1995.

Daniel Mestre. Immersion and presence. 04 2019.

Bernard Hoscheke Modic. Serious games and affective gaming, 2017.

Ali Mollahosseini, Behzad Hassani, and Mohammad H. Mahoor. Affectnet: A database for facial expression, valence, and arousal computing in the wild. *CoRR*, abs/1708.03985, 2017.

Nicola Montano, Alberto Porta, Chiara Cogliati, Giorgio Costantino, Eleonora Tobaldini, Karina Rabello Casali, and Ferdinando Iellamo. Heart rate variability explored in the frequency domain: a tool to investigate the link between heart and behavior. *Neuroscience & Biobehavioral Reviews*, 33(2):71–80, 2009.

S. Moore and R. Bowden. Local binary patterns for multi-view facial expression recognition. *Computer Vision and Image Understanding*, 115(4):541 – 558, 2011.

Nagisa Munekata, Teruhisa Nakamura, Rei Tanaka, Yusuke Domon, Fumihiko Nakamura, and Hitoshi Matsubara. *A Biofeedback Game with Physical Actions.* Entertainment Computing - ICEC 2010. 2012.

LE Nacke, Anders Drachen, and Stefan Gobel. Methods for evaluating gameplay experience in a serious gaming context. *International Journal of Computer Science in Sport*, 9(2):1–12, 2010.

Lennart Erik Nacke, Michael Kalyn, Calvin Lough, and Regan Lee Mandryk. Biofeedback game design: using direct and indirect physiological control to enhance game interaction. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 103–112. ACM, 2011.

Y.Y. Ng, C.W. Khong, and H. Thwaites. A review of affective design towards video games. *Procedia - Social and Behavioral Sciences*, 51:687 – 691, 2012. ISSN 1877-0428.

Heather L. O'Brien and Elaine G. Toms. What is user engagement? a conceptual framework for defining user engagement with technology. *Journal of the American Society for Information Science and Technology*, 59(6):938–955, 2008. doi: 10.1002/asi.20801.

Sebastien Ouellet. Real-time emotion recognition for gaming using deep convolutional network features. *CoRR*, abs/1408.3750, 2014.

Michael Quinn Patton. *Qualitative Research & Evaluation Methods.* Sage Publications, 2001.

Rosalind W. Picard. *Affective Computing.* MIT Press, Cambridge, MA, USA, 1997. ISBN 0-262-16170-2.

Rosalind W. Picard. Emotion research by the people, for the people. *Emotion Review*, 2010.

D. Pollreisz and N. TaheriNejad. A simple algorithm for emotion recognition, using physiological signals of a smart watch. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2353–2356, 2017.

Posner, Jonathan, Russell, James A, Peterson, and Bradley S. The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology. *Development and psychopathology*, 17(3):715–734, 2005.

Rahul Banerjee Rajiv Ranjan Singh, Sailesh Conjeti. A comparative evaluation of neural network classifiers for stress level analysis of automotive drivers using physiological signals. *Biomedical Signal Processing and Control*, 8(6):740 – 754, 2013. ISSN 1746-8094.

Munaf Rashid, S. A. R. Abu-Bakar, and Musa Mokji. Human emotion recognition from videos using spatio-temporal and audio features. *The Visual Computer*, 29(12):1269–1275, Dec 2013. ISSN 1432-2315. doi: 10.1007/s00371-012-0768-y. URL `https://doi.org/10.1007/s00371-012-0768-y`.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

Dennis W Rowe, John Sibert, and Don Irwin. Heart rate variability: Indicator of user state as an aid to human-computer interaction. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 480–487. ACM Press/Addison-Wesley Publishing Co., 1998.

James A. Russell. A circumplex model of affect. 39:1161, 1980. ISSN 0022-3514.

James A. Russell and Lisa Feldman Barrett. Core affect, prototypical emotional episodes, and other things called emotion: Dissecting the elephant. 76:805, 1999.

Ryerson. The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english. *PLoS ONE*, 2018.

C. Saarni. *The Development of Emotional Competence*. Guilford series on social and emotional development. Guilford Publications, 1999. ISBN 9781572304345.

Bhargava Rohit Sagi and Rachel Silvestrini. Application of combinatorial tests in video game testing. 29:745–759, 2017. ISSN 0898-2112. doi: 10.1080/08982112.2017.1300919.

Henrik Schoenau-Fog. Hooked! – evaluating engagement as continuation desire in interactive narratives. pages 219–230, Berlin, Heidelberg, 2011a. Springer Berlin Heidelberg. ISBN 978-3-642-25289-1.

Henrik Schoenau-Fog. The player engagement process - an exploration of continuation desire in digital games. In *DiGRA - Proceedings of the 2011 DiGRA International Conference: Think Design Play*. DiGRA/Utrecht School of the Arts, January 2011b. ISBN ISSN 2342-9666.

Henrik Schoenau-Fog. *At the Core of Player Experience: Continuation Desire in Digital Games*, pages 388–410. John Wiley & Sons Inc., 2014a. ISBN 9781118796443. doi: 10.1002/9781118796443.ch14.

Henrik Schoenau-Fog. Designing and evaluating conative game-based learning scenarios. In Carsten Busch, editor, *Proceedings of The 8th European Conference on Games Based Learning - ECGBL 2014*, pages 512–518. Academic Conferences and Publishing International, 10 2014b. ISBN 978-1-910309-55-1.

Henrik Schoenau-Fog, Alexander Birke, and Lars Reng. Evaluation of continuation desire as an iterative game development method. In *Proceeding of the 16th International Academic MindTrek Conference*, MindTrek '12, pages 241–243, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1637-8. doi: 10.1145/2393132.2393185.

Henrik Schoenau-Fog, Sandy Louchart, Theodore Lim, and Soto-Sanfiel. Narrative engagement in games-a continuation desire perspective. In *FDG*, pages 384–387, 2013.

Schultz, Charles P. Schultz, Koenig Boespflug, Robert Bryant, and Tim Langdell. *Game Testing All in One*. 2005. ISBN 1-59200-373-7.

Dines Selvig, Nikolaj Stovring, Andreas Sjoblom, and Jeppe Korsholm. Continuation desire and its physiological underpinnings. 2018.

K. P. Seng, L. Ang, and C. S. Ooi. A combined rule-based amp; machine learning audio-visual emotion recognition approach. *IEEE Transactions on Affective Computing*, 9(1):3–13, Jan 2018. ISSN 1949-3045.

Caifeng Shan, Shaogang Gong, and Peter W. McOwan. Facial expression recognition based on local binary patterns: A comprehensive study. *Image and Vision Computing*, 27(6):803 – 816, 2009. ISSN 0262-8856.

Jose L Soler-Dominguez, Jorge D Camba, Manuel Contero, and Mariano Alcañiz. A proposal for the selection of eye-tracking metrics for the implementation of adaptive gameplay in virtual reality based games. In *International Conference on Virtual, Augmented and Mixed Reality*, pages 369–380. Springer, 2017.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2377–2385. Curran Associates, Inc., 2015. URL `http://papers.nips.cc/paper/5850-training-very-deep-networks.pdf`.

Stanney, Kelly S. Hale, and Kay M. Stanney. *Handbook of Virtual Environments*. 2002. ISBN 978-0-8058-3270-9.

Harald M Stauss. Heart rate variability. *American Journal of Physiology-Regulatory, Integrative and Comparative Physiology*, 285(5):927–931, 2003.

Robert Morris Stern, William J Ray, and Karen S Quigley. *Psychophysiological recording*. Oxford University Press, USA, 2001.

Bas R Steunebrink, Mehdi Dastani, and John-Jules Ch Meyer. The occ model revisited. Department of Information and Computing Sciences, Utrecht University.

Jason E. Warnick Vinessa K. Jones Gary L. Yarbrough Tenea M. Russell Chris M. Borecky Reagan McGahhey John L. Powell III Jamie Beavers Emmanuelle Monte Steven J. Haggbloom, Renee Warnick. The 100 most eminent psychologists of the 20th century. 2002.

Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016.

A. Tawari and M. M. Trivedi. Face expression recognition by cross modal data association. *IEEE Transactions on Multimedia*, 15(7):1543–1552, Nov 2013. ISSN 1520-9210.

Julian F Thayer, Anita L Hansen, Evelyn Saus-Rose, and Bjorn Helge Johnsen. Heart rate variability, prefrontal neural function, and cognitive performance: the neurovisceral integration perspective on self-regulation, adaptation, and health. *Annals of Behavioral Medicine*, 37(2): 141–153, 2009.

Sabine Trepte and Leonard Reinecke. The pleasures of success: Game-related efficacy experiences as a mediator between player performance and game enjoyment. *Cyberpsychology, Behavior, and Social Networking*, 14(9):555–557, 2011.

P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, Dec 2001.

Tine Vyvey, Elena Nunez Castellar, and Jan Van Looy. Loaded with fun? the impact of enjoyment and cognitive load on brand retention in digital games. 18:72–82, 2018. ISSN 1525-2019. doi: 10.1080/15252019.2018.1446370.

Chen Wang, Thierry Pun, and Guillaume Chanel. A comparative survey of methods for remote heart rate detection from frontal face videos. *Frontiers in bioengineering and biotechnology*, 6, 2018.

Andreas Wulff-Jensen. Communication framework for the mionix naos qg mouse: For online and offline usage, 2017.

S. N. Yu and S. F. Chen. Emotion state identification based on heart rate variability and genetic algorithm. pages 538–541, Aug 2015. doi: 10.1109/EMBC.2015.7318418.

Ke Zhang, Rocky Peng Chen, and Sara Kim. Anthropomorphized Helpers Undermine Autonomy and Enjoyment in Computer Games. *Journal of Consumer Research*, 43(2):282–302, 2016.

T. Zhang, W. Zheng, Z. Cui, Y. Zong, and Y. Li. Spatial-temporal recurrent neural network for emotion recognition. *IEEE Transactions on Cybernetics*, 2018.

11 Appendix

11.1 OCC - Emotional Responses

- Joy: (pleased about) a desirable event
- Distress: (displeased about) an undesirable event
- Happy-for: (pleased about) an event presumed to be desirable for someone else
- Pity: (displeased about) an event presumed to be undesirable for someone else
- Gloating: (pleased about) an event presumed to be undesirable for someone else
- Resentment: (displeased about) an event presumed to be desirable for someone else
- Hope: (pleased about) the prospect of a desirable event
- Fear: (displeased about) the prospect of an undesirable event
- Satisfaction: (pleased about) the confirmation of the prospect of a desirable event
- Fears-confirmed: (displeased about) the confirmation of the prospect of an undesirable event
- Relief: (pleased about) the disconfirmation of the prospect of an undesirable event
- Disappointment: (displeased about) the disconfirmation of the prospect of a desirable event
- Pride: (approving of) one's own praiseworthy action
- Shame: (disapproving of) one's own blameworthy action
- Admiration: (approving of) someone else's praiseworthy action
- Reproach: (disapproving of) someone else's blameworthy action
- Gratification: (approving of) one's own praiseworthy action and (being pleased about) the related desirable event
- Remorse: (disapproving of) one's own blameworthy action and (being displeased about) the related undesirable event
- Gratitude: (approving of) someone else's praiseworthy action and (being pleased about) the related desirable event
- Anger: (disapproving of) someone else's blameworthy action and (being displeased about) the related undesirable event
- Love: (liking) an appealing object
- Hate: (disliking) an unappealing object

11.2 Code: Frame Extractor

Listing 11: Python code showing the frame extractor to extract images from the RAWDES video database (video_to_images.py)

```
1
2  import cv2
3  import os
4  import utils.inference as utils
5  import pickle
6  import  numpy as np
7
8  global_frame_count = 0
9
10 detection_model_path = 'models/detection_models/haarcascade_frontalface_default.xml
       '
11 face_detection = utils.load_detection_model(detection_model_path)
12 emotion_offsets = (20, 40)
13
14
15
16 def capture_frames(path_to_file, path_to_save, image_name_specifier, label):
17     local_frames = []
18     local_labels = []
19     vidcap = cv2.VideoCapture(path_to_file)
20
21     success,image = vidcap.read()
22     image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
23     count = 0
24     success = True
25
26     while success:
27         try:
28             face = utils.detect_faces(face_detection, image)
29             x1, x2, y1, y2 = utils.apply_offsets(face[0], emotion_offsets)
30             image = image[y1:y2, x1:x2]
31         except:
32             pass
33         image = cv2.resize(image, (224, 224))
34
35         image = np.expand_dims(image, -1)
36         local_frames.append(image)
37         local_labels.append(label)
38         success,image = vidcap.read()
39         if success:
40             image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
41         count += 1
42     return local_frames, local_labels
43
44 video_path = r'D:/MasterData/raw_data/'
45
46 dirs = os.listdir(video_path)
47
48 dir_count = 0
49 for dir in dirs:
50     images = []
51     labels = []
52     cur_dir = os.listdir(video_path + dir)
53     video_files_path = os.listdir(video_path + dir + "/" + cur_dir[0])
54
55     for video in video_files_path:
56         segment_name = video.split('-')[2]
57
58         if segment_name == '01':
59             path_to_file = video_path + dir + "/" + cur_dir[0] + "/" + video
60             path_to_save = "D:/MasterData/image_data/neutral"
61             frame_list, seq_labels = capture_frames(path_to_file, path_to_save,
                   cur_dir[0]+"_"+ video[0:-4], 1)
62             images.extend(frame_list)
63             labels.extend(seq_labels)
64
65         elif segment_name == '02':
66             path_to_file = video_path + dir + "/" + cur_dir[0] + "/" + video
67             path_to_save = "D:/MasterData/image_data/calm"
```

```
68              frame_list, seq_labels = capture_frames(path_to_file, path_to_save,
                    cur_dir[0]+"_"+ video[0:-4], 2)
69              images.extend(frame_list)
70              labels.extend(seq_labels)
71
72          elif segment_name == '03':
73              path_to_file = video_path + dir + "/" + cur_dir[0] + "/" + video
74              path_to_save = "D:/MasterData/image_data/happy"
75              frame_list, seq_labels = capture_frames(path_to_file, path_to_save,
                    cur_dir[0]+"_"+ video[0:-4], 3)
76              images.extend(frame_list)
77              labels.extend(seq_labels)
78
79          elif segment_name == '04':
80              path_to_file = video_path + dir + "/" + cur_dir[0] + "/" + video
81              path_to_save = "D:/MasterData/image_data/sad"
82              frame_list, seq_labels = capture_frames(path_to_file, path_to_save,
                    cur_dir[0]+"_"+ video[0:-4], 4)
83              images.extend(frame_list)
84              labels.extend(seq_labels)
85
86          elif segment_name == '05':
87              path_to_file = video_path + dir + "/" + cur_dir[0] + "/" + video
88              path_to_save = "D:/MasterData/image_data/angry"
89              frame_list, seq_labels = capture_frames(path_to_file, path_to_save,
                    cur_dir[0]+"_"+ video[0:-4], 5)
90              images.extend(frame_list)
91              labels.extend(seq_labels)
92
93          elif segment_name == '06':
94              path_to_file = video_path + dir + "/" + cur_dir[0] + "/" + video
95              path_to_save = "D:/MasterData/image_data/fearful"
96              frame_list, seq_labels = capture_frames(path_to_file, path_to_save,
                    cur_dir[0]+"_"+ video[0:-4], 6)
97              images.extend(frame_list)
98              labels.extend(seq_labels)
99
100         elif segment_name == '07':
101             path_to_file = video_path + dir + "/" + cur_dir[0] + "/" + video
102             path_to_save = "D:/MasterData/image_data/disgust"
103             frame_list, seq_labels = capture_frames(path_to_file, path_to_save,
                    cur_dir[0]+"_"+ video[0:-4],7 )
104             images.extend(frame_list)
105             labels.extend(seq_labels)
106
107         elif segment_name == '08':
108             path_to_file = video_path + dir + "/" + cur_dir[0] + "/" + video
109             path_to_save = "D:/MasterData/image_data/surprised"
110             frame_list, seq_labels = capture_frames(path_to_file, path_to_save,
                    cur_dir[0]+"_"+ video[0:-4], 8)
111             images.extend(frame_list)
112             labels.extend(seq_labels)
113
114         else:
115             print('No Match')
116
117     final_faces = np.asarray(images)
118     final_labels = np.asarray(labels)
119
120     with open('data/ryer_data_csv_224/face_images_large_' + str(dir_count) + '.pkl
        ', 'wb') as f:
121         pickle.dump(final_faces, f)
122
123     with open('data/ryer_data_csv_224/face_images_labels_large_' + str(dir_count) +
         '.pkl', 'wb') as f:
124         pickle.dump(final_labels, f)
125
126     dir_count += 1
```

11.3 Code: Residual Model Code

Listing 12: Python code showing the machine learning model implemented with Keras (models.py)

```
def Residual_Model(input_shape, num_classes, l2_regularization=0.01):
    regularization = l2(l2_regularization)

    # base
    img_input = Input(input_shape)
    x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
               use_bias=False)(img_input)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
               use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    # module 1
    residual = Conv2D(16, (1, 1), strides=(2, 2),
                      padding='same', use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = SeparableConv2D(16, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = SeparableConv2D(16, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])

    # module 2
    residual = Conv2D(32, (1, 1), strides=(2, 2),
                      padding='same', use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = SeparableConv2D(32, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = SeparableConv2D(32, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])

    # module 3
    residual = Conv2D(64, (1, 1), strides=(2, 2),
                      padding='same', use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = SeparableConv2D(64, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = SeparableConv2D(64, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])

    # module 4
    residual = Conv2D(128, (1, 1), strides=(2, 2),
```

```
                      padding='same', use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = SeparableConv2D(128, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = SeparableConv2D(128, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])

    # module 5
    residual = Conv2D(256, (1, 1), strides=(2, 2),
                      padding='same', use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = SeparableConv2D(256, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = SeparableConv2D(256, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])

    # module 6
    residual = Conv2D(512, (1, 1), strides=(2, 2), padding='same', use_bias=False)(
        x)
    residual = BatchNormalization()(residual)

    x = SeparableConv2D(512, (3, 3), padding='same', kernel_regularizer=
        regularization, use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    x = SeparableConv2D(512, (3, 3), padding='same', kernel_regularizer=
        regularization, use_bias=False)(x)
    x = BatchNormalization()(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])

    x = Conv2D(num_classes, (3, 3),
               padding='same')(x)
    x = GlobalAveragePooling2D()(x)
    output = Activation('softmax', name='predictions')(x)

    model = Model(img_input, output)
    return model
    }
```

11.4 Residual Model Summary

Listing 13: Summary of model

```
1  _________________________________________________________________
2  Layer (type)                 Output Shape              Param #   Connected to
3  =================================================================
4  input_1 (InputLayer)         (None, 48, 48, 1)         0
5  _________________________________________________________________
6  conv2d_1 (Conv2D)            (None, 46, 46, 8)         72        input_1[0][0]
7  _________________________________________________________________
8  batch_normalization_1 (BatchNor (None, 46, 46, 8)      32        conv2d_1[0][0]
9  _________________________________________________________________
10 activation_1 (Activation)    (None, 46, 46, 8)         0
       batch_normalization_1[0][0]
11 _________________________________________________________________
12 conv2d_2 (Conv2D)            (None, 44, 44, 8)         576       activation_1[0][0]
13 _________________________________________________________________
14 batch_normalization_2 (BatchNor (None, 44, 44, 8)      32        conv2d_2[0][0]
15 _________________________________________________________________
16 activation_2 (Activation)    (None, 44, 44, 8)         0
       batch_normalization_2[0][0]
17 _________________________________________________________________
18 separable_conv2d_1 (SeparableCo (None, 44, 44, 16)     200       activation_2[0][0]
19 _________________________________________________________________
20 batch_normalization_4 (BatchNor (None, 44, 44, 16)     64        separable_conv2d_1
       [0][0]
21 _________________________________________________________________
22 activation_3 (Activation)    (None, 44, 44, 16)        0
       batch_normalization_4[0][0]
23 _________________________________________________________________
24 separable_conv2d_2 (SeparableCo (None, 44, 44, 16)     400       activation_3[0][0]
25 _________________________________________________________________
26 batch_normalization_5 (BatchNor (None, 44, 44, 16)     64        separable_conv2d_2
       [0][0]
27 _________________________________________________________________
28 conv2d_3 (Conv2D)            (None, 22, 22, 16)        128       activation_2[0][0]
29 _________________________________________________________________
30 max_pooling2d_1 (MaxPooling2D)  (None, 22, 22, 16)     0
       batch_normalization_5[0][0]
31 _________________________________________________________________
32 batch_normalization_3 (BatchNor (None, 22, 22, 16)     64        conv2d_3[0][0]
33 _________________________________________________________________
34 add_1 (Add)                  (None, 22, 22, 16)        0         max_pooling2d_1
       [0][0]
35                                                                 batch_normalization_3
                                                                       [0][0]
36 _________________________________________________________________
37 separable_conv2d_3 (SeparableCo (None, 22, 22, 32)     656       add_1[0][0]
38 _________________________________________________________________
39 batch_normalization_7 (BatchNor (None, 22, 22, 32)     128       separable_conv2d_3
       [0][0]
40 _________________________________________________________________
41 activation_4 (Activation)    (None, 22, 22, 32)        0
```

```
    batch_normalization_7[0][0]
42 ____________________________________________________________________________________________________
43 separable_conv2d_4 (SeparableCo (None, 22, 22, 32)   1312        activation_4[0][0]
44 ____________________________________________________________________________________________________
45 batch_normalization_8 (BatchNor (None, 22, 22, 32)   128         separable_conv2d_4
       [0][0]
46 ____________________________________________________________________________________________________
47 conv2d_4 (Conv2D)              (None, 11, 11, 32)   512         add_1[0][0]
48 ____________________________________________________________________________________________________
49 max_pooling2d_2 (MaxPooling2D)  (None, 11, 11, 32)   0
       batch_normalization_8[0][0]
50 ____________________________________________________________________________________________________
51 batch_normalization_6 (BatchNor (None, 11, 11, 32)   128         conv2d_4[0][0]
52 ____________________________________________________________________________________________________
53 add_2 (Add)                    (None, 11, 11, 32)   0           max_pooling2d_2
       [0][0]
54                                                                 batch_normalization_6
                                                                       [0][0]
55 ____________________________________________________________________________________________________
56 separable_conv2d_5 (SeparableCo (None, 11, 11, 64)   2336        add_2[0][0]
57 ____________________________________________________________________________________________________
58 batch_normalization_10 (BatchNo (None, 11, 11, 64)   256         separable_conv2d_5
       [0][0]
59 ____________________________________________________________________________________________________
60 activation_5 (Activation)       (None, 11, 11, 64)   0
       batch_normalization_10[0][0]
61 ____________________________________________________________________________________________________
62 separable_conv2d_6 (SeparableCo (None, 11, 11, 64)   4672        activation_5[0][0]
63 ____________________________________________________________________________________________________
64 batch_normalization_11 (BatchNo (None, 11, 11, 64)   256         separable_conv2d_6
       [0][0]
65 ____________________________________________________________________________________________________
66 conv2d_5 (Conv2D)              (None, 6, 6, 64)     2048        add_2[0][0]
67 ____________________________________________________________________________________________________
68 max_pooling2d_3 (MaxPooling2D)  (None, 6, 6, 64)     0
       batch_normalization_11[0][0]
69 ____________________________________________________________________________________________________
70 batch_normalization_9 (BatchNor (None, 6, 6, 64)     256         conv2d_5[0][0]
71 ____________________________________________________________________________________________________
72 add_3 (Add)                    (None, 6, 6, 64)     0           max_pooling2d_3
       [0][0]
73                                                                 batch_normalization_9
                                                                       [0][0]
74 ____________________________________________________________________________________________________
75 separable_conv2d_7 (SeparableCo (None, 6, 6, 128)    8768        add_3[0][0]
76 ____________________________________________________________________________________________________
77 batch_normalization_13 (BatchNo (None, 6, 6, 128)    512         separable_conv2d_7
       [0][0]
78 ____________________________________________________________________________________________________
79 activation_6 (Activation)       (None, 6, 6, 128)    0
       batch_normalization_13[0][0]
80 ____________________________________________________________________________________________________
81 separable_conv2d_8 (SeparableCo (None, 6, 6, 128)    17536       activation_6[0][0]
82 ____________________________________________________________________________________________________
```

batch_normalization_14 (BatchNo [0][0]	(None, 6, 6, 128)	512	separable_conv2d_8		
conv2d_6 (Conv2D)	(None, 3, 3, 128)	8192	add_3[0][0]		
max_pooling2d_4 (MaxPooling2D) batch_normalization_14[0][0]	(None, 3, 3, 128)	0			
batch_normalization_12 (BatchNo	(None, 3, 3, 128)	512	conv2d_6[0][0]		
add_4 (Add) [0][0]	(None, 3, 3, 128)	0	max_pooling2d_4 batch_normalization_12 [0][0]		
separable_conv2d_9 (SeparableCo	(None, 3, 3, 256)	33920	add_4[0][0]		
batch_normalization_16 (BatchNo [0][0]	(None, 3, 3, 256)	1024	separable_conv2d_9		
activation_7 (Activation) batch_normalization_16[0][0]	(None, 3, 3, 256)	0			
separable_conv2d_10 (SeparableC	(None, 3, 3, 256)	67840	activation_7[0][0]		
batch_normalization_17 (BatchNo separable_conv2d_10[0][0]	(None, 3, 3, 256)	1024			
conv2d_7 (Conv2D)	(None, 2, 2, 256)	32768	add_4[0][0]		
max_pooling2d_5 (MaxPooling2D) batch_normalization_17[0][0]	(None, 2, 2, 256)	0			
batch_normalization_15 (BatchNo	(None, 2, 2, 256)	1024	conv2d_7[0][0]		
add_5 (Add) [0][0]	(None, 2, 2, 256)	0	max_pooling2d_5 batch_normalization_15 [0][0]		
separable_conv2d_11 (SeparableC	(None, 2, 2, 512)	133376	add_5[0][0]		
batch_normalization_19 (BatchNo separable_conv2d_11[0][0]	(None, 2, 2, 512)	2048			
activation_8 (Activation) batch_normalization_19[0][0]	(None, 2, 2, 512)	0			
separable_conv2d_12 (SeparableC	(None, 2, 2, 512)	266752	activation_8[0][0]		
batch_normalization_20 (BatchNo separable_conv2d_12[0][0]	(None, 2, 2, 512)	2048			
conv2d_8 (Conv2D)	(None, 1, 1, 512)	131072	add_5[0][0]		

```
125 max_pooling2d_6 (MaxPooling2D)  (None, 1, 1, 512)     0
        batch_normalization_20[0][0]
126 ____________________________________________________________________________________________
127 batch_normalization_18 (BatchNo (None, 1, 1, 512)     2048        conv2d_8[0][0]
128 ____________________________________________________________________________________________
129 add_6 (Add)                     (None, 1, 1, 512)     0           max_pooling2d_6
        [0][0]
130                                                                   batch_normalization_18
                                                                          [0][0]
131 ____________________________________________________________________________________________
132 conv2d_9 (Conv2D)               (None, 1, 1, 8)       36872       add_6[0][0]
133 ____________________________________________________________________________________________
134 global_average_pooling2d_1 (Glo (None, 8)             0           conv2d_9[0][0]
135 ____________________________________________________________________________________________
136 predictions (Activation)        (None, 8)             0
        global_average_pooling2d_1[0][0]
137 ============================================================================================

138 Total params: 762,168
139 Trainable params: 756,088
140 Non-trainable params: 6,080
141 ____________________________________________________________________________________________
```

11.5 Code: Altered Residual Model

Listing 14: Python code showing the altered machine learning model implemented with Keras (models.py)

```
def residual_model_altered(input_shape, num_classes, l2_regularization=0.01):
    regularization = l2(l2_regularization)

    # base
    img_input = Input(input_shape)
    x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
               use_bias=False)(img_input)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Conv2D(8, (3, 3), strides=(1, 1), kernel_regularizer=regularization,
               use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    # module 1
    residual = Conv2D(16, (1, 1), strides=(2, 2),
                      padding='same', use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = SeparableConv2D(16, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = SeparableConv2D(16, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])

    # module 2
    residual = Conv2D(32, (1, 1), strides=(2, 2),
                      padding='same', use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = SeparableConv2D(32, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = SeparableConv2D(32, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])

    # module 3
    residual = Conv2D(64, (1, 1), strides=(2, 2),
                      padding='same', use_bias=False)(x)
    residual = BatchNormalization()(residual)

    x = SeparableConv2D(64, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = SeparableConv2D(64, (3, 3), padding='same',
                        kernel_regularizer=regularization,
                        use_bias=False)(x)
    x = BatchNormalization()(x)

    x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
    x = layers.add([x, residual])

    # module 4
```

```
70      residual = Conv2D(128, (1, 1), strides=(2, 2),
71                        padding='same', use_bias=False)(x)
72      residual = BatchNormalization()(residual)
73
74      x = SeparableConv2D(128, (3, 3), padding='same',
75                          kernel_regularizer=regularization,
76                          use_bias=False)(x)
77      x = BatchNormalization()(x)
78      x = Activation('relu')(x)
79      x = SeparableConv2D(128, (3, 3), padding='same',
80                          kernel_regularizer=regularization,
81                          use_bias=False)(x)
82      x = BatchNormalization()(x)
83
84      x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
85      x = layers.add([x, residual])
86
87      # module 5
88      residual = Conv2D(256, (1, 1), strides=(2, 2),
89                        padding='same', use_bias=False)(x)
90      residual = BatchNormalization()(residual)
91
92      x = SeparableConv2D(256, (3, 3), padding='same',
93                          kernel_regularizer=regularization,
94                          use_bias=False)(x)
95      x = BatchNormalization()(x)
96      x = Activation('relu')(x)
97      x = SeparableConv2D(256, (3, 3), padding='same',
98                          kernel_regularizer=regularization,
99                          use_bias=False)(x)
100     x = BatchNormalization()(x)
101     x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
102     x = layers.add([x, residual])
103
104     # module 6
105     residual = Conv2D(512, (1, 1), strides=(2, 2), padding='same', use_bias=False)(
            x)
106     residual = BatchNormalization()(residual)
107
108     x = SeparableConv2D(512, (3, 3), padding='same', kernel_regularizer=
            regularization, use_bias=False)(x)
109     x = BatchNormalization()(x)
110     x = Activation('relu')(x)
111
112     x = SeparableConv2D(512, (3, 3), padding='same', kernel_regularizer=
            regularization, use_bias=False)(x)
113     x = BatchNormalization()(x)
114
115     x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)
116     x = layers.add([x, residual])
117
118     x = Conv2D(num_classes, (3, 3),
119                padding='same')(x)
120     x = GlobalAveragePooling2D()(x)
121     output = Activation('softmax', name='predictions')(x)
122
123     model = Model(img_input, [output, x])
124     return model
```

11.6 Code: Densely Connection Model

Listing 15: Python code showing the densely machine learning model implemented with Keras (models.py)

```
def densly_model(input_shape, num_classes, l2_regularization=0.01):
    regularization = l2(l2_regularization)
    img_input = Input(input_shape)
    layer_mul = 1
    dropout = 0
    use_bias = True

    # Base
    x = Conv2D(8*layer_mul, (3, 3), strides=(1, 1), kernel_regularizer=
        regularization,
               use_bias=use_bias)(img_input)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Dropout(dropout)(x)
    x = Conv2D(8*layer_mul, (3, 3), strides=(1, 1), kernel_regularizer=
        regularization,
               use_bias=use_bias)(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = Dropout(dropout)(x)

    # module 1
    x_1 = SeparableConv2D(16*layer_mul, (3, 3), padding='same',
                          kernel_regularizer=regularization,
                          use_bias=use_bias)(x)
    x_1 = BatchNormalization()(x_1)
    x_1 = Activation('relu')(x_1)
    x_1 = Dropout(dropout)(x_1)
    x_1 = SeparableConv2D(16*layer_mul, (3, 3), padding='same',
                          kernel_regularizer=regularization,
                          use_bias=use_bias)(x_1)
    x_1 = BatchNormalization()(x_1)

    dense_con = Concatenate()([x, x_1])

    # module 2
    x_2 = SeparableConv2D(32*layer_mul, (3, 3), padding='same',
                          kernel_regularizer=regularization,
                          use_bias=use_bias)(dense_con)
    x_2 = BatchNormalization()(x_2)
    x_2 = Activation('relu')(x_2)
    x_2 = Dropout(dropout)(x_2)
    x_2 = SeparableConv2D(32*layer_mul, (3, 3), padding='same',
                          kernel_regularizer=regularization,
                          use_bias=use_bias)(x_2)
    x_2 = BatchNormalization()(x_2)

    dense_con = Concatenate()([x, x_1, x_2])

    # module 3
    x_3 = SeparableConv2D(64*layer_mul, (3, 3), padding='same',
                          kernel_regularizer=regularization,
                          use_bias=use_bias)(dense_con)
    x_3 = BatchNormalization()(x_3)
    x_3 = Activation('relu')(x_3)
    x_3 = Dropout(dropout)(x_3)
    x_3 = SeparableConv2D(64*layer_mul, (3, 3), padding='same',
                          kernel_regularizer=regularization,
                          use_bias=use_bias)(x_3)
    x_3 = BatchNormalization()(x_3)

    dense_con = Concatenate()([x, x_1, x_2, x_3])

    # module 4
    x_4 = SeparableConv2D(128*layer_mul, (3, 3), padding='same',
                          kernel_regularizer=regularization,
                          use_bias=use_bias)(dense_con)
    x_4 = BatchNormalization()(x_4)
    x_4 = Activation('relu')(x_4)
```

```
68      x_4 = Dropout(dropout)(x_4)
69      x_4 = SeparableConv2D(128*layer_mul, (3, 3), padding='same',
70                            kernel_regularizer=regularization,
71                            use_bias=use_bias)(x_4)
72      x_4 = BatchNormalization()(x_4)
73
74      x_4 = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x_4)
75      dense_con = Concatenate()([x,x_1,x_2,x_3, x_4])
76
77      # Output
78      x = Conv2D(num_classes, (3, 3), padding='same')(dense_con)
79      x = GlobalMaxPool2D()(x)
80      output = Activation('softmax', name='predictions')(x)
81
82      model = Model(img_input, output)
83      return model
```

11.7 Code: Image Extractor Conation

Listing 16: Python code showing the densely machine learning model implemented with Keras (video_to_image_conation.py)

```
import cv2
import os
import utils.inference as utils
import pickle
import  numpy as np

global_frame_count = 0

detection_model_path = 'models/detection_models/haarcascade_frontalface_default.xml'
face_detection = utils.load_detection_model(detection_model_path)
emotion_offsets = (20, 40)



def capture_frames(path_to_file, label):
    local_frames = []
    local_labels = []
    vidcap = cv2.VideoCapture(path_to_file)

    success,image = vidcap.read()
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    count = 0
    success = True

    while success:
        try:
            face = utils.detect_faces(face_detection, image)
            x1, x2, y1, y2 = utils.apply_offsets(face[0], emotion_offsets)
            image = image[y1:y2, x1:x2]
        except:
            pass
        image = cv2.resize(image, (48, 48))

        image = np.expand_dims(image, -1)
        local_frames.append(image)
        local_labels.append(label)
        success, image = vidcap.read()
        if success:
            image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        count += 1
    return local_frames, local_labels

video_path = r'D:/conation_video/split/'

dirs = os.listdir(video_path)
images = []
labels = []
dir_count = 0
for dir in dirs:

    cur_dir = os.listdir(video_path + dir)

    for video in cur_dir:

        segment_name = video.split('-')[-1][0]
        if segment_name == '1':
            path_to_file = video_path + dir + "/" + video
            frame_list, seq_labels = capture_frames(path_to_file, 1)
            images.append(frame_list)
            labels.append(seq_labels)
        elif segment_name == '2':
            path_to_file = video_path + dir + "/" + video
            frame_list, seq_labels = capture_frames(path_to_file, 2)
            images.append(frame_list)
            labels.append(seq_labels)
        elif segment_name == '3':
            path_to_file = video_path + dir + "/" + video
            frame_list, seq_labels = capture_frames(path_to_file, 3)
```

```
69              images.append(frame_list)
70              labels.append(seq_labels)
71          elif segment_name == '4':
72              path_to_file = video_path + dir + "/" + video
73              frame_list, seq_labels = capture_frames(path_to_file, 4)
74              images.append(frame_list)
75              labels.append(seq_labels)
76          elif segment_name == '5':
77              path_to_file = video_path + dir + "/" + video
78              frame_list, seq_labels = capture_frames(path_to_file, 5)
79              images.append(frame_list)
80              labels.append(seq_labels)
81          elif segment_name == '6':
82              path_to_file = video_path + dir + "/" + video
83              frame_list, seq_labels = capture_frames(path_to_file, 6)
84              images.append(frame_list)
85              labels.append(seq_labels)
86          elif segment_name == '7':
87              path_to_file = video_path + dir + "/" + video
88              frame_list, seq_labels = capture_frames(path_to_file, 7)
89              images.append(frame_list)
90              labels.append(seq_labels)
91          else:
92              print('No Match')
93          print()
94
95  with open('data/conation_images/conation_images_' + str(dir_count) + '.pkl', 'wb')
        as f:
96      pickle.dump(images, f)
97  with open('data/conation_images/conation_images_labels_' + str(dir_count) + '.pkl',
        'wb') as f:
98      pickle.dump(labels, f)
```

11.8 Consent Form

AALBORG UNIVERSITET

Informed Consent form

Purpose:
The purpose of this test is to gather data about a player's emotional expressions during game play, and use this information to correlate continuation desire.

Test Procedure:
The test will be conducted today _____. You will be asked to participate in a test, where you will first answer a couple of questions and next you will play a game for minimum 30 minutes. During your play session I will ask you how much you want to continue playing right now.

Confidentiality:
All information gathered during the test will be kept confidential and will only be used in context of a thesis paper. The information gathered from the test will be analyzed by the team and kept until the thesis is complete, but kept no longer than till 1st of September.

The test will be fully anonymous meaning that no names will be matched to the recorded data or questions.

During the test you will be free to change your mind about the data storage and we will end the test.

Please sign below if you agree with these terms and conditions.

_____ _____

Date Participant Signature

_____ _____

Date Test conductor Signature

11.9 Interview Transcript - Invisible Walls

The interview were conducted the 16-06-2019 at Invisible Walls HQ by Dines Selvig. The interviewee is Sebastian Hurup Bevensee. The interview was in danish, and is therefore translated to english.

Question: What is your current test methodology

> We have not changed the wat that we test, in term of setup so we are testing more or less the same. At the moment its a but weird with our test. I have in the past worked as a test designer, and i have made a lot tests there and also at the university when we made aporia where i thought of many different creative ways of how to test.
>
> Right now in the project we are not at a stage where we will go into depth with test, and make a large test where you have a large questionnaire with EEG and EMG sensors. We simply do not have that, and its far to expensive and complex, even though it can be used for many different things. Because we are so deep into the game development, the only thing that we needed was to capture the game play and a webcam of the user, so you can see their behaviour.
>
> It was that method we used to see their reactions, and listen to the communication that they made, and see if they did what they had to do, and if they could make sense of how to play the game. So we tested if the players did what we wanted them to do, because we are quite early in the process of the production, and the only think that we need right now is an indication about if the mechanics works or not. So we not need data from large quesitonnaires yet
>
> In our latest game, Aporia we did really large tests, where we tested for many things, both mechanics and narrative, like if they understood the narrative, or if the puzzles were to hard or to easy. These questionnaires were after the exprience, which either lasted 2 or 4 hours to play. So it was a good mix of questions about the experience.

Question: What do you use your results from a testing session for?

> That the participants is having fun while playing the game, and actually a similar method to continuation desire, that we was them afterwards if they want to continue playing, or if the want to play the game again. And usually people really want to continue playing, and actually after that test session, the participants asked to play more games even though we had all the data we needed. And that was a really good indication that the game was actually very fun and engaging to play. They also talked a lot with each other after the game has ended, which was also positive.
>
> The video capture has smart, because as a level designer I can go back and see where people navigates in the level, where are the participants a lot, and where do they not go. Also what kind of items to they see and which do they not. So we are doing analysis of the video, but only observing. We are also looking for facial expression to how they react to the game. But you have to remember that the idea of filming and testing the game was also to make a video to be showed to the public of the game. That was actually the main objective. But at the same time getting a lot of data.
>
> Its also a complex game to test, and we are now at a point that we can not test it ourselves anymore. We can only have an idea of what is going to happen.
>
> We are simply to deep into making the game, that we can not foresee how players react.

Question: It was also observed that people who talk a lot in-game also shows a lot of emotions, as the research also indicates.

> I guess that you could also see that when people are running around alone in the game, they do not show a lot of emotional responses.

Question: Up until now you have tested the same way, with letting people play and recording them and then observing. What are your plans for future testing, do you plan to have a deeper testing like you did at Aporia?

> Yes, we are going to create a deeper test, when the production moves forward. And then we will use the same kind of questionnaires as with Aporia. When its a social game, then its actually just our job to ensure that its fun socially in-game, and that is pretty easy to measure, as its easy to observe. What is really hard is to measure how fun is it after you have played the game 30-40-50 times, and on that note out method is to release an open alpha test to players in October. The game will be 100% standalone on steam to selected players, and we do not have to do anything, it just works. Its of course only playable when enough players are online. And the idea with this session is that we will track a lot of in-game metrics such as positions. I do not think we can track their communication, because that is not legal. Find out how many times to they interact with objects, can they find the elements they have to, all these sort of things related to player behaviour, but we have not set these things up yet. Also track how many times to players keep on player and so on.
>
> It could also be very nice to do a short demographics on the player base, before playing, to see what kind of player types that are playing. Because there is definitely some people who this game is not for. And its not them that we are interested in getting data on.

Question: we have touched upon it a little bit, but do you think that there is a need for deeper insights into that player other than just video and in-game metrics when testing your game? Especially in relation to what i am developing?

> Totally, totally, we have not tracked emotional responses per say, we have just looked at the overall response. As long as they laughed and had fun, that was enough for us. Also because if you see some negative expressions, then you could pinpoint if its something in the game in terms of mechanics or if its something with the communication. Do find the problem if there is a problem. It can also be a problem that people do not talk enough, and we already know that, that is a problem. Because those who dont talk that much. does not talk that much either in-game. And in those situations we have to do something to enforce communication. Where the players start in a room and they can talk to another player in another room, and they have to describe a sigil to the other person to get out of the room. This enforces communication from the beginning, and works as an ice breaker. So we try to get as many mechanics into the game that enforces this.
>
> On a side note, we are also trying to get the game working without the use of a microphone, as its a bit dangerous that if you want to make money on a game that relies on people to have and want to use a microphone.

Lets now take a look at the data, and the first graph is from the player Dengi. And on the x axis we have the frame count from the whole sequence of 18 minutes. There is 9 different lines, 8 different emotions, and continuation desire.

He was very quite, and did not talk too much, but still showed expressions when he interacted with the other players. We can see here that around 16000 frames that he killed another player, but there was not too much of a response from his side. At around 25000 frames he is voted off, and you can clearly see a spike of happiness when they talk about voting him off, as he is about to get caught, and when that happens, he returns to a completely neutral state, as he is no longer part of the game. Overall his reactions was very subtle and he is showing many different emotions all the time, and no many spikes.

> Yeah, that is very interesting, he is also a very laid back guy, that does not play these kind of games regularly. But its very fun and insight full to see these reactions on an actual graph.

Now lets take a look on another player, Mille, which is completely different in terms of how she plays the game and interacts with other people. She has very big and large spikes of happiness, that is present almost the entire time. So you can really see that she is having fun. We also see some surprise spikes as well as angry spikes.

When she dies, she is first surprised as she was a good guy, and then angry, as she tells the others that she was actually good!

> Yeah, she is very happy about the game, and really likes to play it, it can clearly be seen that she is very engaged. I notice some angry spikes, I can not help but notice that she looks a bit angry when she thinks or focuses. Angry is just a hard emotion, as it can look like others as well, such as puzzled. Its very interesting that you can see when she was voted off that strongly, the data clearly shows the shift in emotions.

And as you can see her continuation desire is full on the whole experience, and that is because she is very happy at all times. And the primary metric for continuation desire is happiness.

> Yes that makes sense, a little note is that I believe that when you have different emotions spiking at all times, means that the player is engaged, as his emotions is being activated.

Then the last graph is from a play session where you played the game yourself, and it is a shorter session of 10 minutes. I noticed that when looking how the algorithm classified your expression, it was very prone to classify you as angry when you were in fact concentrated or neutral. And that is why you see these spikes in your data.

> Laughter. Yes, i think i look at bit angry when i concentrate.. so there is probably some corrections to be made to the algorithm regarding that.
>
> So I do not see the dark green line a lot, where is it?

The dark green is calm, and in all test session that is basically non existing as not many people are calm when playing, especially not this game.

Right now i use a rolling mean on the data, to smooth it, as it would be much more spiky than it is now.

> Yeah, I was about the ask about that, maybe you could also tangent normalise to get a smoother line. And of course you would loose some of these spikes, but it would be much easier to read from a testers purpose. I think as this as a software where you could get some nice curvy lines with good colours, and when i then zoomed in on the graph it would get more granular so i could dive deeper into the data.

Question: When looking at this data, is there something that surprises you, or interests you?

> I am actually surprised to see that there is not more surprise emotions present in the data, because im certain the people felt more surprised that the data shows. But I know that you need to show a lot of surprise in your face for that to be detected. Especially the first times they played the game, they were surprised a lot by the mechanics, and then they died and so on. But i think they did not show it that much. But that is a bit tricky to measure. I think that as a software which a large team had been doing, i would like to choose the emotions that i wanted to track, as some emotions is not useful it some games, as calm is not something relevant for this game. but happiness is. If the genre was horror i would like to look more at disgust and fearful.

Question: Does this kind of data and deeper analysis make sense in terms of game testing, and especially your production?

> Yes i think that we could use something like this in the production, but it is a but tricky with our game. As what we test at the moment and also in the future is how to get people to talk and have fun. And we also have to do this very good. And for these things also to test specific mechanics, we could definitely use something like this. And we would like to see that happiness and continuation desire would rise over time spend in the game. It would able us to pinpoint more accurately what kind of mechanics works, but also to control the tempo of the game, to see if somethings should be done slower or faster. It could help create a form of drama manager, that could control the flow of the game. But also over a period of games, to see how their responses changes from the first game to their 40th game.
>
> Do your algorithm adapt to the user?

Right now it does not, but i am working on doing a baseline at the start of a play session, either by making the participants go through each emotion and show that emotion, or watch some content that should produce an emotion. This data should then be learned by that algorithm. so that it understands your expressions better.

> Exactly! That was what i was thinking about, as emotions can be a bit different between people. If you had more time to to a baseline you could make the participants rate their own responses to content they experienced, in a form of a self reported measure. It is very tricky working with emotions, as many times people do not show any expression.
>
> I think its also hard to push continuation desire on emotions, it works in our game, but what is its a single player game, and you do not really show any emotional expressions.
>
> Its very nice that its very accessible, that i only need a webcam and then able to produce all this data, that before required big setups of complex and expensive gear.

Question: How much do you use of your time on testing?

> As we are an indie game company, its very hard to find the money and time to actually conduct these tests. Because the money that is being invested in the game is usually just enough to get the game done, and not so much testing of the actual game.
>
> Also a testing session takes a lot of time away from the production, which means its a big loss every time a game session is held. So they are being down prioritised a lot.
>
> I my world, i think you should really have a lot of tests, and focus groups and so on, but its just not possible. When we made Aporia, I think that we were the ones that tested the most compared to other companies. We tested many times over a year at least couple of times a month, some of the with 30 participants with big questionnaires, and we even outsourced some testing to an external company in the UK, which made detailed reports and so on. And even after these tests, I felt that i could use three times as much data as i have.
>
> For us right now, we get a lot from just observing, but not so much of how they feel.
>
> But at out current state, we just need an indication of what works and what does not. I think that when we come closer to the release the tests will also become bigger and more complex, and then i think we could use something like your tool.
>
> And its very nice that if we had 1000 people testing, the algorithm could analyse that very fast, but if we had to observe the same amount of that, it would take weeks, and then it becomes too expensive.

11.10 Test Analysis Graphs

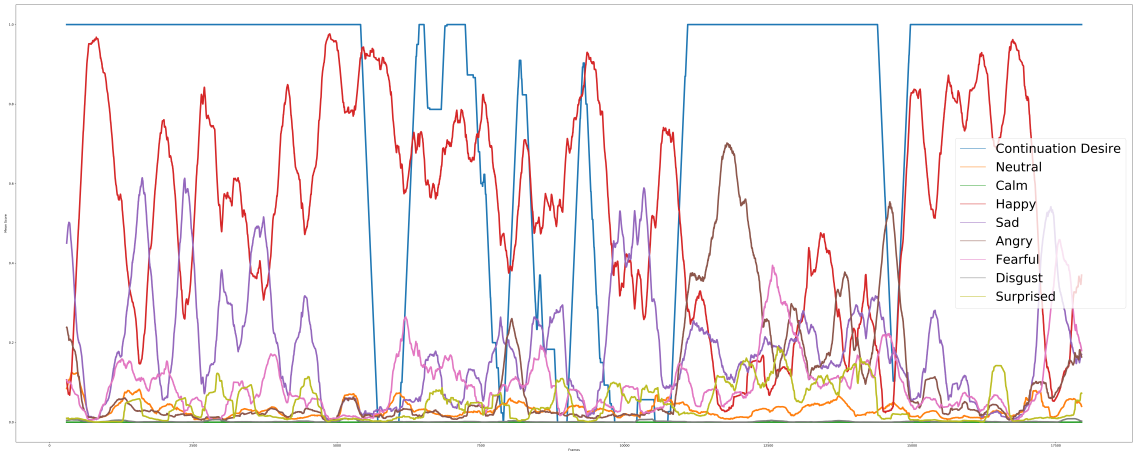


Figure 46: Graph showing the analysis of a play through from participant 1 from the test session. Eight emotions is tracked and continuation desire as well.

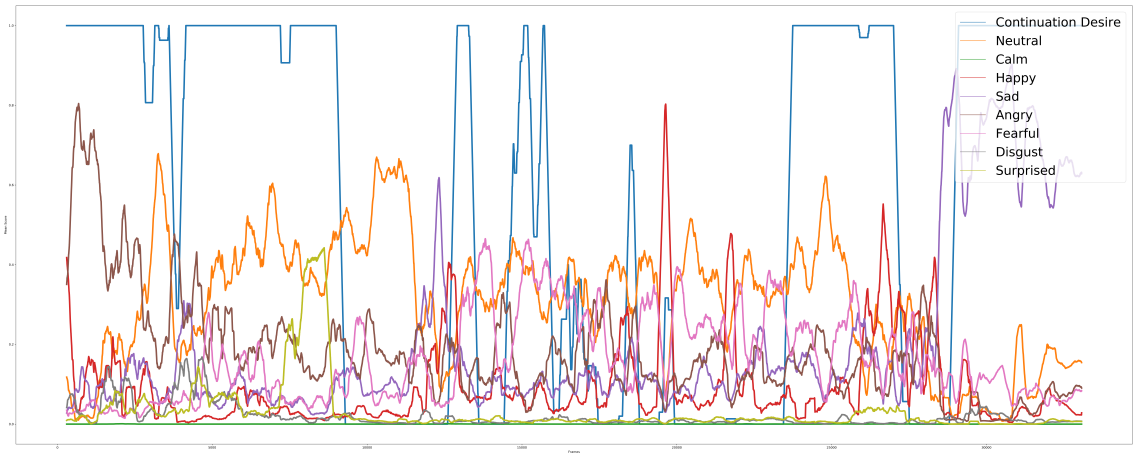


Figure 47: Graph showing the analysis of a play through from participant 2 from the test session. Eight emotions is tracked and continuation desire as well.

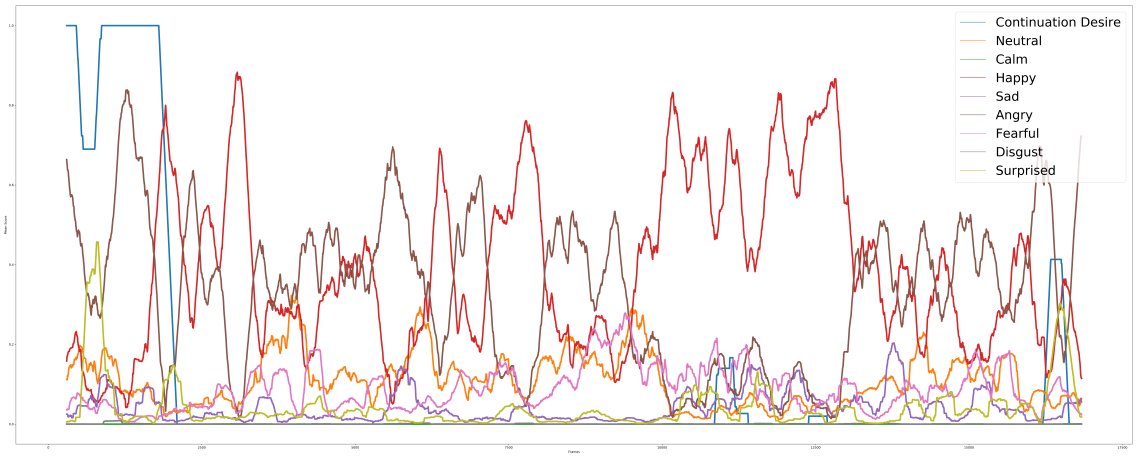


Figure 48: Graph showing the analysis of a play through from participant 3 from the test session. Eight emotions is tracked and continuation desire as well.

11.11 Visualisation of Network Layers

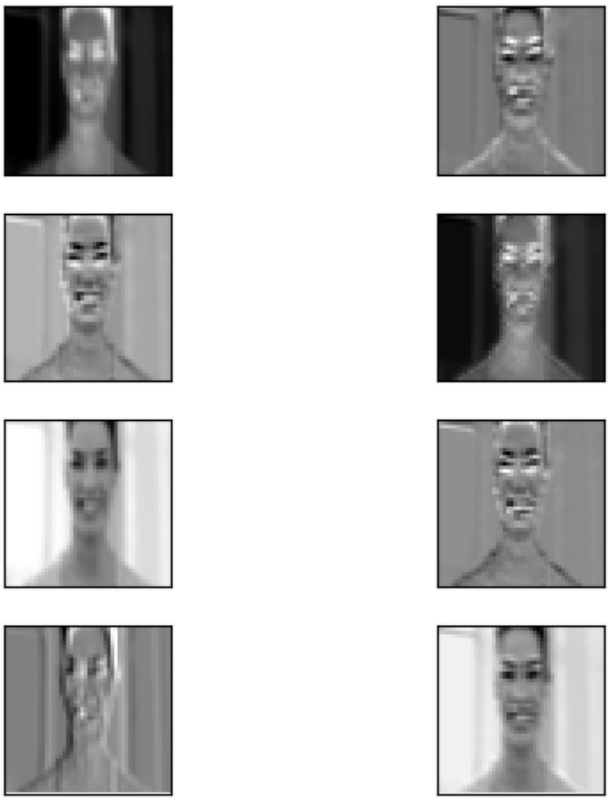


Figure 49: Figure showing eight feature maps from a convolution layer (1st layer), from inference of an image showing a happy emotion

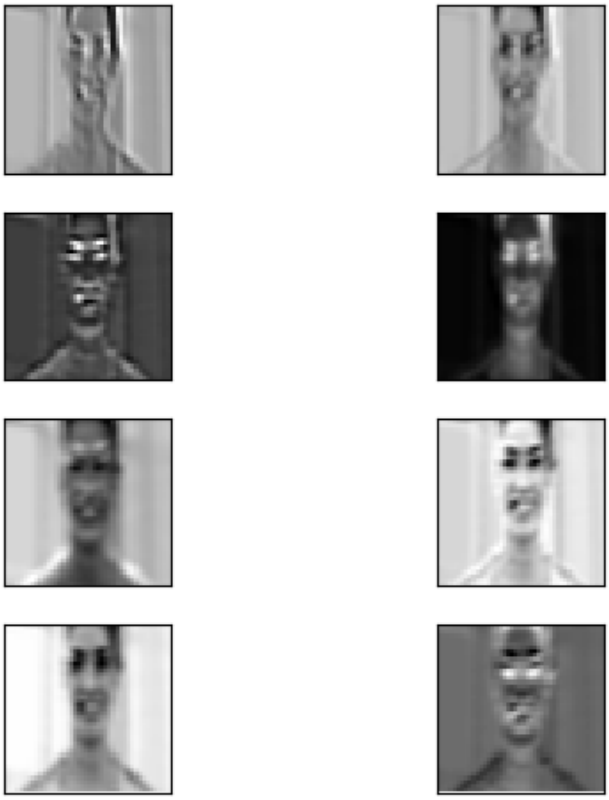


Figure 50: Figure showing eight feature maps from a convolution layer (4th layer), from inference of an image showing a happy emotion

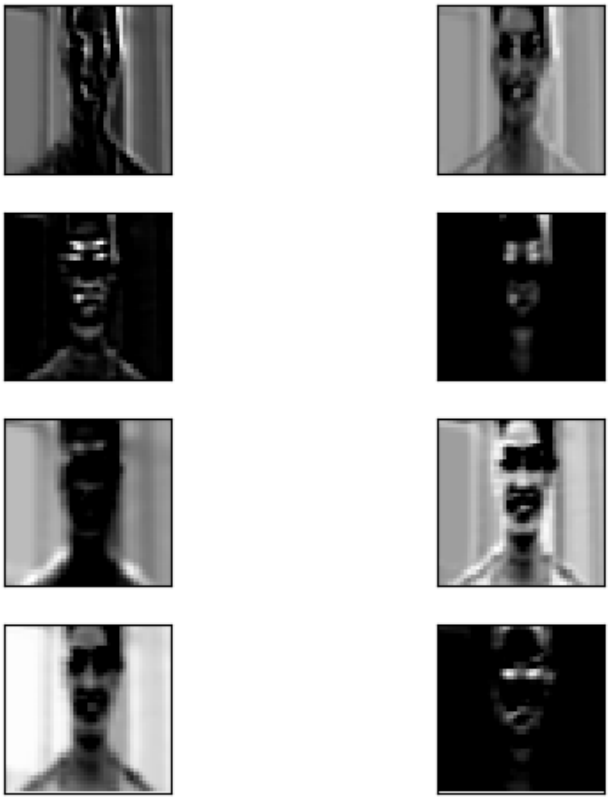


Figure 51: Figure showing eight feature maps from a convolution layer (7th layer), from inference of an image showing a happy emotion

Figure 52: Figure showing eight feature maps from a convolution layer (10th layer), from inference of an image showing a happy emotion

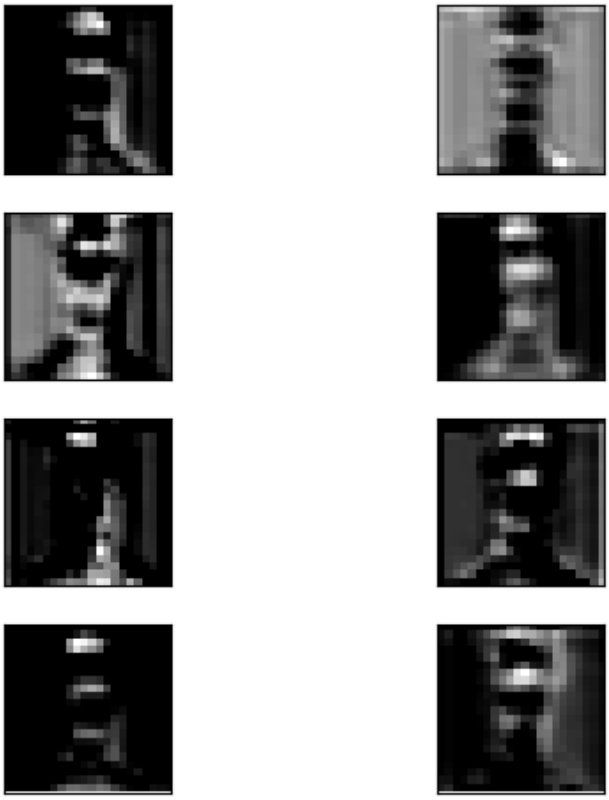


Figure 53: Figure showing eight feature maps from a convolution layer (19th layer), from inference of an image showing a happy emotion

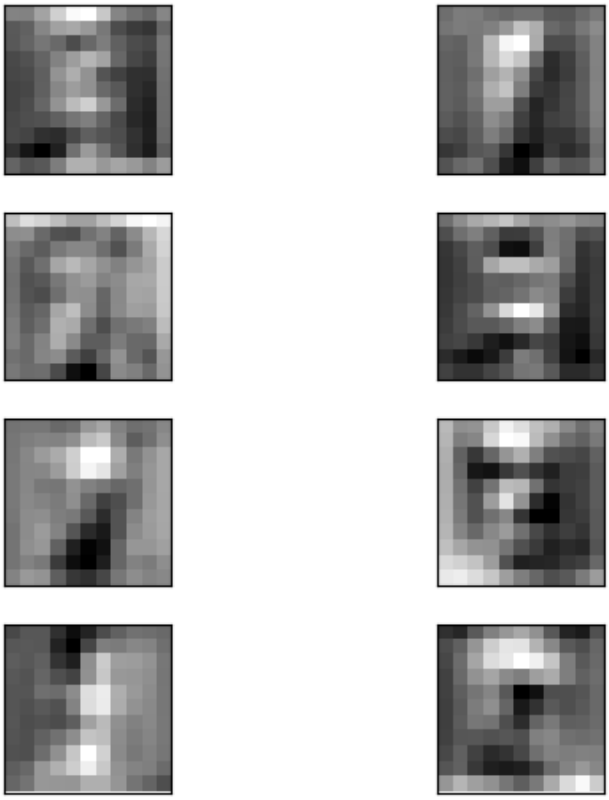


Figure 54: Figure showing eight feature maps from a convolution layer (29th layer), from inference of an image showing a happy emotion

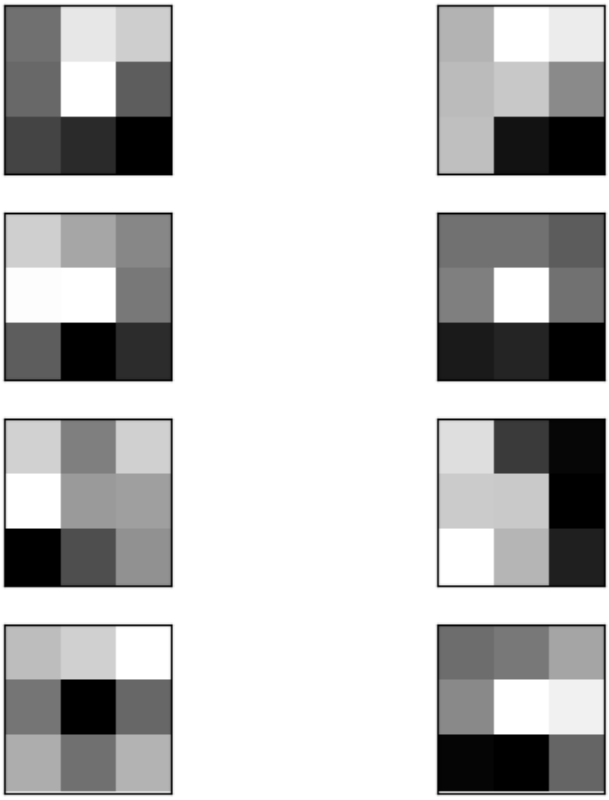


Figure 55: Figure showing eight kernels maps from a convolution layer (48th layer), from inference of a image showing a happy emotion