Universiteit Leiden
The Netherlands

# Opleiding Informatica

Automatically detecting when participants want help from their facial expressions in an online quiz

Jelle Keulemans (s2357682)

Supervisors:
Joost Broekens
Fons Verbeek

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

August 27, 2022

**Abstract**

Making predictions about when students want help can provide teachers and Intelligent Systems with possibly useful information, as they can use this data to help students without them needing to ask for it. This could prove to be especially useful in cases where, for example, students are too 'shy' to ask for help. In this paper we propose a system that encodes facial expressions with FACS and uses this to predict when a student will click on an 'ask for help'-button in a learning environment. The system can predict when someone will ask for help with up to 60% accuracy from a data set that has a 50% chance of students asking for help. This data set was collected from experiments with 17 students. The methods used to make these predictions are based on frustration and confusion detection techniques. Looking at the accuracy of the predictions, the method used may prove relevant for predicting when a student is about to ask for help in intelligent tutoring systems.

# Contents


**1 Introduction** **1**
- 1.1 Thesis goal and overview . . . 1

**2 Related Work** **2**
- 2.1 Classifying facial expressions . . . 2
- 2.2 Automatic detection of facial expressions . . . 2
- 2.3 Making predictions from facial expressions . . . 4
- 2.4 Overview of intelligent and affective tutoring systems . . . 6

**3 Methods** **6**
- 3.1 Dataset creation . . . 7
  - 3.1.1 Website creation . . . 7
  - 3.1.2 Steps of the experiment . . . 7
  - 3.1.3 Data collection . . . 9
  - 3.1.4 Security . . . 11
  - 3.1.5 Pilot experiment . . . 11
  - 3.1.6 Demographic information . . . 11
  - 3.1.7 Constructing the dataset . . . 12
- 3.2 Classification . . . 16
  - 3.2.1 Neural network structure . . . 17
  - 3.2.2 Implementation . . . 18
  - 3.2.3 hyper parameters . . . 18
  - 3.2.4 Proposed experiments . . . 20

**4 Results** **23**
- 4.1 Hyper parameter optimization . . . 23
- 4.2 Window picking optimisation . . . 25
- 4.3 Relevant Features . . . 27
- 4.4 Training on individual participants . . . 27
- 4.5 Model evaluation . . . 28

**5 Discussion** **29**

**6 Conclusions** **32**

**References** **34**

**A Puzzles in the data collection experiment** **35**

# 1 Introduction

Improving education with automated tutoring systems has been a large area of interest over last several decades. Often, the goal is to design intelligent tutoring systems (ITS) that are able to provide one-one-one tutoring and track progress of students [18], which comes down to providing personalized education. Examples are Snappet which is able to teach basic courses on elementary schools while showing teachers a detailed view of the progress of students [5, 6] or Duolingo which teaches foreign languages [2].

Most of these ITS are also able to give on-demand help, just like human teachers. To give on-demand help, some kind of assessment must be made to see if a student wants or needs help [30]. ITS generally make these assessments with algorithms like fuzzy logic [7] that can for example look at progress of students and make decisions accordingly. So ITS can for example show a help message when a student makes some mistakes repeatedly. Often, it also is possible for students to ask for help with a simple help button. Human teachers may assess this by monitoring facial expressions, like frowning, eye squinting or generally looking confused.

Research in affective computing has shown that ITS can, similarly to human tutors, also use predictions about affective data like facial expressions to improve assessments they make [22]. ITS in the past have, for example, already been able to respond appropriately upon recognizing emotions that are relevant for learning [18]. Some of these relevant learning emotions are boredom, hope and enjoyment [26].

However, these affective tutoring systems (ATS) have not been able to specifically recognize if students *want* help by looking at their facial expressions. This application of facial expression recognition could provide teachers and intelligent systems with possibly useful information, as they can use this data to make better informed decisions about when to provide on-demand help to students. This additional information could prove to be especially useful in situations where, for example, students are too 'shy' or embarrassed to seek help.

Students could non-verbally ask for help by expressing emotions like confusion and frustration, as psychological research found [13]. Moreover, these emotions have proven to play a critical role in the learning process [26, 22]. These expressions of emotions can already be automatically detected from facial expressions [34, 9, 19, 25, 16]. This means that it is likely also possible to automatically recognize if people want help from facial expressions.

## 1.1 Thesis goal and overview

The ultimate goal of this research is to make predictions about if people want help by looking at their facial expressions.

The structure of the paper is as follows. Section 2 shows related work for facial expressions classification, predictions using facial expressions and intelligent/affective tutoring systems. Section 3 describes how a dataset is created that contains data about people wanting help, and how this is used to classify when people want help. Section 4 list all results. Section 5 and 6 interpret, discuss

and make conclusions about these results, and conclude if the goal of this research is reached.

# 2 Related Work

## 2.1 Classifying facial expressions


**Figure 1:** An illustration of Ekman's basic emotions, from [20].

Humans can express many intentions and emotions with their face. There are many techniques to recognize these expressions. A basic technique is to label certain expressions with Ekman's 5 basic emotions, illustrated in 1. In the literature, there are also other (additional) labels that can be used to describe emotions and facial expressions [10]. However, labeling of expressions could introduce biases. People could be influenced by sounds, the environment and their own mood. And more importantly, labels of expressions can be interpreted differently in different cultures [28].

The Facial Action Coding System (FACS) prevents this by encoding facial expressions in Action Units (AU) [28]. AUs define contractions of one or more facial muscle(s) that are responsible for changes in facial expressions. Current FACS parameterize expressions with 28 main AUs, as summarized in figure 2. A recent blog describes many more AUs related to head pose and movements [15].

Other techniques that can encode facial expressions in the literature include: Face Animation Parameters (FAP), Maximally Discriminative Facial Movement Coding System (MAX) and Monadic Phases Coding System (MP) [34]. However, FACS has become the dominant technique for parameterizing facial expressions [28, 34].

## 2.2 Automatic detection of facial expressions

The process of automatic facial expression analysis (AFEA) generally consists of the following steps [28]:

1. Face Acquisition: locate and/or track a face in a video or image;
2. Facial Expression Extraction: extract facial expression data from the face, which can for example be achieved with FACS encoding;

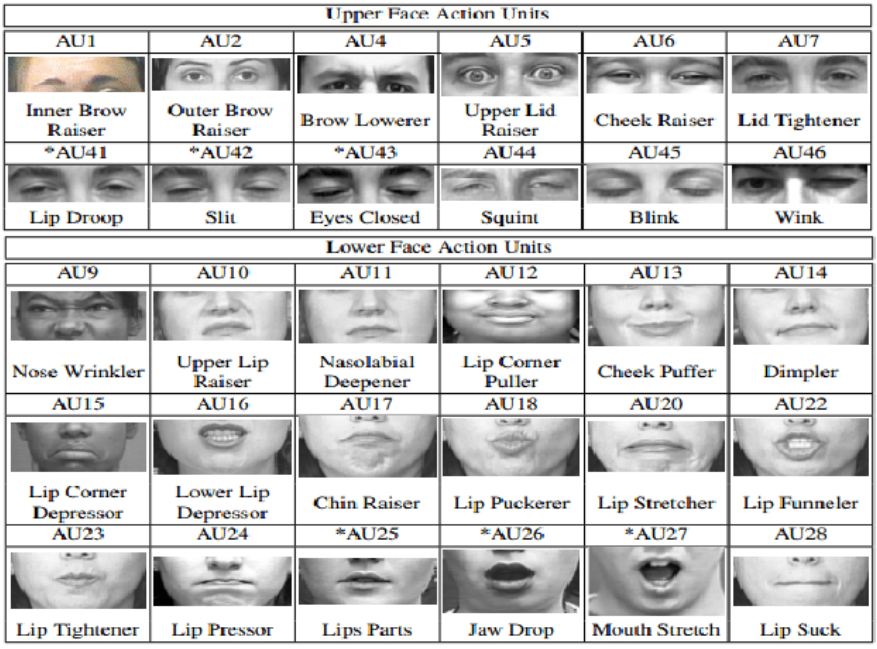



**Figure 2:** A visual representation of the first action units [27].

3. Expression Recognition: identify expressions like anger or happiness using the extracted features.

The literature includes ready-to-use implementations of AFEA that are able to reliably extract AU from facial expressions from video feed, some with academic background. The ones considered for this research are Open Face [8], Automated Facial Affect Recognition (AFAR) [14], Face Reader and mocap4face [4].

AFAR, Open Face and Face Reader were compared in [24]. AFAR and Open Face were both able to recognize AUs well beyond the 'chance level'. However, Open Face is able to detect more AUs from images than AFAR, possibly making it better AU detection software. Face Reader underperformed compared to both AFAR and Open Face. Table 1 summarizes the predictions of the three considered AFEA solutions. The researchers tested only the first 26 main action units, also listed in figure 2.

| | **In the wild** | **In conversation** | **Posed** |
|---|---|---|---|
| **AFAR** | 0.649 | 0.730 | 0.831 |
| **Open Face** | 0.723 | 0.717 | 0.891 |
| **Face Reader** | 0.667 | 0.581 | 0.550 |

**Table 1:** A summary of the mean prediction scores of 26 different AUs [24]. The best mean result is picked when multiple configurations were tested.

For mocap4face there are no clear methodologies or metrics made available that describe the accuracy of the predictions. However, a large 'selling-point' of this library is that it is able to recognize facial expressions based on FACS in browser-based environments *out of the box*.

## 2.3 Making predictions from facial expressions

AFEA is commonly used to make predictions about emotions or other cognitive states. FaceReader 7.1 can for example predict Ekman's emotions using Action Units. Much more recently, much research has also been targeting prediction of non-classical emotions like confusion, frustration or other 'non-primary' affective states, like cognitive engagement or depression, from facial expressions, often using deep learning techniques. Cognitive processes are very complex and intermingled, but it's clear that Ekman's basic emotions are rarely present in learning [26]. So especially techniques for predicting emotions like confusion and frustration can prove to be useful for this research as argued in the introduction.

A paper published in 2019 reports an average accuracy of 83.71% for classifying confused, happy, negative and neutral facial expressions, using AUs extracted with FaceReader [9]. The classifier considers AUs over a range of time (a time series) of 2 seconds instead of one moment/frame. The researchers implement this with a Long-Short-Term-Memory (LSTM) layer in a neural network, which excels at making predictions about series of dependent data like time series [23]. When comparing this model to a standard model that's incorparated in FaceReader 7.1, they found that the misclassification of confusion as 'neutral' was reduced. Their model was able to recognize the more subtle, almost neutral, confused expressions because it also considered the temporal context, which is especially useful for confused expressions; A confused expression starts with an almost neutral expression, but does change over time. They also found that filtering action units was crucial, because many introduced noise in the dataset, introducing misclassifications. However, their dataset was not large enough to accurately see what AUs could be dropped as their dataset only includes 34 sequences of facial expressions in total.

Recently, a more specific study about confusion detection was published where, again, sequences of images were analysed [19]. Their best classifier was able to achieve an accuracy of 96.3415% (using Quadratic Discriminant Analysis (QDA)). In this study, the researchers used a much larger dataset of 490 sequences to train their classifiers. The researchers determine which AUs can be dropped (AU 1, 2, 9, 14, 15, 17, 20, 25, 26, 28), and which are essential for confusion detection (AU 4, 5, 6, 7, 10, 12 and 23).

In the field of frustration detection, early research already identified significant action units in identifying frustrated expressions [17]. Automatic frustration detection from AUs requires reliable AFEA, which did not exist until toolboxes like the Computer Expression Recognition Toolbox (CERT) were created, that use FACS. A paper published in 2013 used CERT with the significant AUs 1, 2, 7 and 14 to show that frustrated expressions are able to be classified with linear regression [16].

One of the most recent publications about a classifier that can also classify frustration follows the deep learning trend, similar to the mentioned classifier that uses a LSTM [25]. This classifier is able to classify sleepy, yawning, boredom, frustration, confusion and focus from single images using a convolutional neural network (CNN), with an average accuracy of 73.68%.

Table 2 contains a summary of the AFEA processes that are used and their results of the mentioned papers, and additional ones.

| Reference | Facial expression extraction | Expression recognition | Results |
|---|---|---|---|
| [19] | Extracts AU 1, 2, 4, 5, 6, 7, 9, 10, 12, 14, 15, 17, 20, 23 and 25 uses 'Constrained Local Neural Field' (CLNF) fitting on detected face. | Recognizes confused expressions (with QDA) from 490 clips of 10-30 images that each are encoded with 15 AUs. | A detection accuracy of 96.34%, only with AUs 4, 5, 6, 7, 10, 12 and 23. |
| [9] | Using FaceReader 7.1 that extracts 20 AUs: 1, 2, 4, 5, 6, 7, 9, 10, 12, 14, 15, 17, 18, 20, 23, 24, 25, 26, 27 and 43. | Recognizes confused, happy, negative and neutral expressions (using an LSTM layer in a neural network) from clips of 2 seconds with in total 30 frames with each 20 AUs. | An average detection accuracy of 87.07%. |
| [16] | Using CERT that extracts 20 AUs: 1, 2, 4, 5, 6, 7, 9, 10, 12, 14,15, 17, 18, 20, 23, 24, 25, 26, 28 and 45. | Recognizes frustrated expressions using linear regression on single frames of AUs. | A root mean squared error (RMSE) of 8.5%. |
| [25] | No feature extraction, only face aquicition. | Recognizes sleepy, yawning, boredom, frustration, confusion and focus from static images of extracted faces using a convolutional neural network (CNN). | An average detection accuracy of 73.68% |

**Table 2:** An overview of notable AFEA approaches with their methodology and results.

### 2.4 Overview of intelligent and affective tutoring systems

There are many papers that discuss implementations of affective computing in ITS (which are called Affective Tutoring Systems). Table 3 summarizes how they collect and use affective data. This related work is considered and mentioned because the research in ATS aims to, similarly to this research, apply automatic recognition of affect in a learning-environment.

| Reference | Measured data | Affective data | Goal(s) |
|---|---|---|---|
| [29] | Heart rate | Positive/negative emotional state | Send motivational messages to maintain positive emotional state. |
| [12] | Posture, facial expression | Boredom, confusion, frustration, surprise | Send motivational messages to help improve learning gains. |
| [21] | Emotional feedback text, facial expression | Positive/negative emotional state | Show virtual tutor with supporting emotions and adjust difficulty of the material to help maintain postive learning emotions. |
| [31] | Emotional feedback text, facial expressions, learning progress | Ekman's basic 5 emotions | Show virtual tutor with supporting emotions and inform teacher of the measured emotional state and progress of students. |
| [32] | Answers to emotional questions | Positive/negative emotional state | Guide students towards positive emotions by asking questions that are associated with positive emotions. |

**Table 3:** An overview of how ATS collect and use affective data. Positive and negative emotional states refer to emotions or general affective states that influence the learning process respectively positively and negatively.

## 3 Methods

This research consists of two parts: (1) creating a dataset and (2) making predictions from this dataset. We opted to make predictions from **time series of facial expression data**, because including the temporal component proved to be fruitful in earlier research [9, 19], as also described in section 2.

## 3.1 Dataset creation

To collect necessary data about facial expressions and students 'wanting help', we designed an experiment in which participants needed to solve puzzles (also referred to as answering questions), and could ask for help if they got stuck. Participants could participate in this experiment by visiting a website (https://www.jellekeulemans.nl) on a laptop or computer. In total, 17 people participated in this experiment. 3 Participants were observed while participating in a pilot version to test and improve the experiment.

### 3.1.1 Website creation

The website was built using the Jspsych framework [11], which is able to show survey-like questions, and handle user input out-of-the-box. We chose to use this framework because it facilitates a clear user interface (UI), and because there are many plugins and extensions available for the framework that can make the application capture interesting data like mouse movements or eye gazes. Additionally, Jspsych captures response times, can shuffle the puzzles and shows a progress bar.

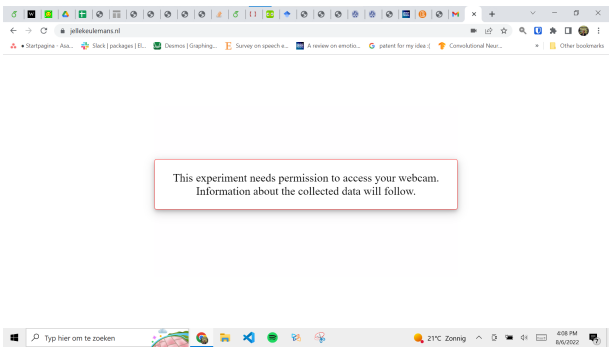
To capture help seeking of participants, we created a custom plugin for Jspsych that is able to show puzzles in a survey-like format with a help-button, and show hints when pressing this button. The plugin records when participants press this help button (the **help time**). After clicking, this button changes to an 'I don't know'-button to prevent users from guessing, and provide 'a way out'.

To record facial expression data during this experiment, we used the Mocap4face toolbox. This toolbox is able to extract a set of AUs (summarized in section 3.1.3) from video feed of the camera of the participants in the browser. Mocap4face has no scientific background and has not been evaluated by any scientists as mentioned in section 2.2, but has the advantage that no video data needs to be recorded and sent because it can already be encoded in the browser of the participant, meaning that only facial expression encoding is collected, sent and stored. Additionally, it is really easy to setup, because there are many examples available of setting up mocap4face that could be consulted when programming the data collection experiment. The frame rate at which the toolbox can extract AUs from the video depends on the processing speed of the computer and environmental factors like lighting.

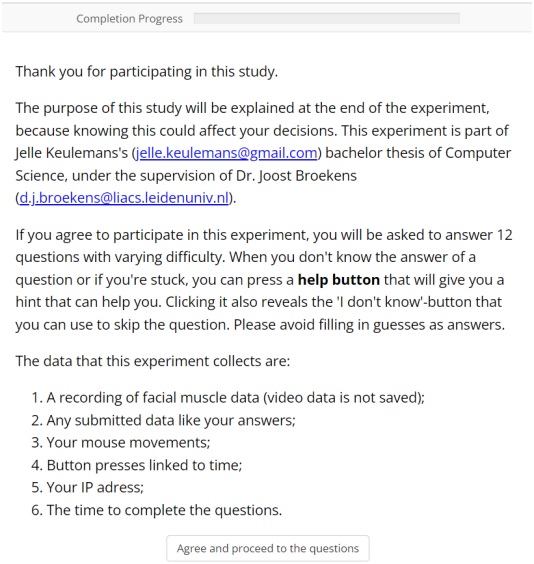
### 3.1.2 Steps of the experiment

Before the main part of the experiment starts where the participant solves puzzles, three pages are shown as listed in figure 3. The first page asks for camera permission and starts tracking the participants face, and extracting the features. After this step, the participant will be notified if their face cannot be tracked. The features are not recorded until the puzzle-solving part of the experiment begins.

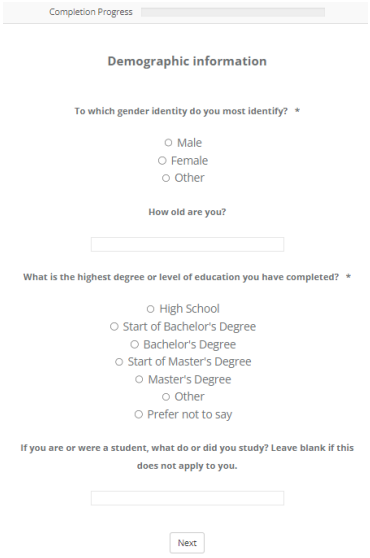
The next page informs about the data collected and when to use the help button. Participants were instructed to use then help button only when they got stuck solving the puzzle, and to not guess




**(a)** First page that is shown that asks permission for camera access.




**(b)** The next page informs the participant about the experiment and asks for consent.




**(c)** Here the participant is asked several demographic questions.

**Figure 3:** Screenshots of the pages that are shown before starting the experiment.

answers.

Next, participants are asked to answer several demographic questions. Demographic information might be crucial as different types of people could respond with different facial expressions, as was also discussed in [19]. Table 5 lists the submitted demographic information of the participants.

The main experiment consists of 12 questions, which are all listed in table 12 in Appendix A. Figure 4 shows a screenshot of a question and the interface. The user can navigate to the next question by entering an answer and clicking 'next', or by clicking the 'help' and 'I don't know' buttons sequentially. It is not possible for users to go back and change submitted answers. The order of the questions is randomized to reduce biases, which could be introduced when, for example, the focus of a participant reduces towards the end.

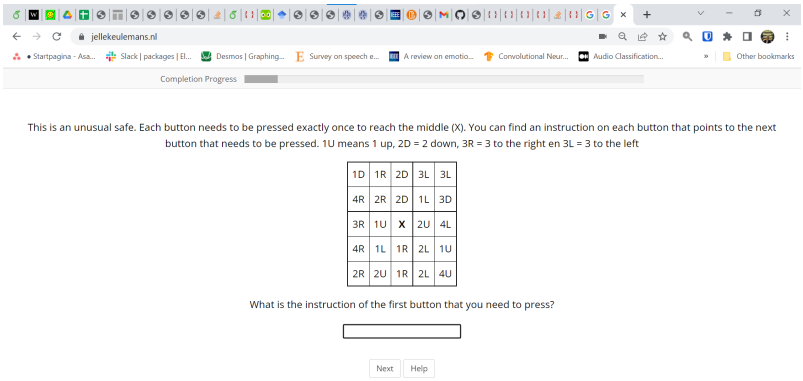



**Figure 4:** An screenshot of one of the questions that is presented in the actual experiment.

### 3.1.3 Data collection

First of all, it is not clear from the documentation of Mocap4face which action units it extracts, as also noted in section 2. From a test where we personally tried how Mocap4face would respond to our facial expressions, we concluded that mocap4face uses a modified version of FACS encoding. In table 4, the standard FACS paramaterization is compared and mapped to the mocap4face parameterization. These parameters are also called features.

When a participant finished a puzzle, the recorded data for that puzzle was sent to a server that saved it. There are 12 puzzles in total. The recorded data includes:

- A list of captured facial expressions with Mocap4face features (which are mapped to AUs in table 4), of which each captured facial expression is linked to the time at which it was captured. Each participant's computer attempts to capture as many facial expressions as possible;
- The time at which a participants pressed the help button;
- An ID for every participant;
- The IP address of the participant;
- The corresponding question;
- The participant's answer and correct answer to the corresponding question;
- Mouse movements;
- Demographic information (this same data is sent with each submitted question).

The server saves this data in a Javascript serialized object notation (JSON) file (the raw data). Every JSON file contains the recorded data of a participant solving one puzzle. So the server saves 12 files with captured data for a participant who submitted all 12 puzzles.

| Action unit | description | mocap4face parameterization |
|---|---|---|
| 1 | inner brow raiser | browInnerUp_L, browInnerUp_R |
| 2 | outer brow raiser | browOuterUp_L, browOuterUp_R |
| 4 | brow lowerer | browdown_L, browdown_R |
| 9 | nose wrinkler | noseSneer_L, noseSneer_R |
| 10 | upper lip raiser | mouthUpperUp_L, mouthUpperUp_R |
| 12 | lip corner puller | mouthSmile_L, mouthSmile_R |
| 15 | lip corner depressor | mouthFown_L, mouthFrown_R |
| 16 | lower lip depressor | mouthLowerDown_L, mouthLowerDown_R |
| 17 | chin raiser | mouthShrugUpper |
| 18 | lip puckerer | mouthPucker |
| 20 | lip stretcher | mouthLeft, mouthRight |
| 22 | lip funneler | mouthFunnel |
| 25, 26, 27 | lips part, jaw drop, mouth stretch | jawOpen |
| 28 | lip suck | mouthRollLower, mouthRollUpper |
| 44 | eye squint | eyeSquint_L, eyeSquint_R |
| 45 | blink | blink_L, blink_R |
| 51 | head turn left | headLeft |
| 52 | head turn right | headRight |
| 53 | head up | headUp |
| 54 | head down | headDown |
| 55 | head tilt left | headRollLeft |
| 56 | head tilt right | headRollRight |
| 63 | eyes up | eyeLookUp_L, eyeLookUp_R |
| 64 | eyes down | eyeLookDown_L, eyeLookDown_R |
| Other | | tongueOut, cheekPuff, jawLeft, jawRight, eyeLookIn_L, eyeLookIn_R, eyeWide_L, eyeWide_R |

**Table 4:** The action units from standard FACS paramaterization with their descriptions mapped to the slightly modified paramaterization of mocap4face. All AUs that are mapped in the table are AUs 1, 2, 4, 9, 10, 12, 15, 16, 17, 18, 20, 22, 25, 26, 27, 28, 44, 45, 51, 52, 53, 54, 55, 56, 63, 64.

The combination of the IP address and unique ID ensures that, even though the participants are anonymous, participants cannot participate multiple times, unless they are actively trying to sabotage this experiment by intentionally changing their IP address.

In case an issue occurs with sending the data, a file containing the data of the failed puzzle submission is automatically downloaded onto the computer of the participant. The participant is then asked to send this file over e-mail.

### 3.1.4 Security

To ensure that the website was secure, and no data could be leaked, it uses several techniques:

- Both the website and the server that collects data have a valid and registered (with https://www.letsencrypt.org) SSL certificate, which enables them to send and receive data over HTTPS protocols. This means that the data that is sent between the server and participant is encrypted and unreadable for anyone eavesdropping.
- All communication and traffic from the server is routed through a Cloudflare proxy [1], which hides its actual address. This also hides possible entry points for hackers like the ssh port of the server, making it much harder for attackers to break into the system. They will first need to find the address of the server. Additionally, this proxy also protects against simple denial of service (DOS) spam attacks, that could overload the server.
- The server ensures that all sent files originate from https://jellekeulemans.nl, ensuring that no other sites are sending (untrustworthy) data to the servers.

### 3.1.5 Pilot experiment

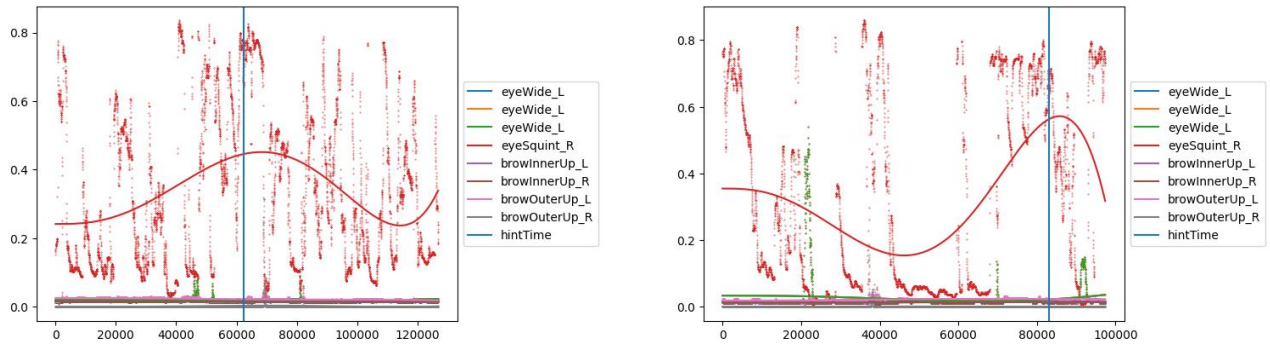
To ensure that the experiment runs smoothly, and to gain insight in how participants behave during the experiment, we conducted a pilot experiment with three students. During these experiments, we monitored their facial expressions, and noted any problems with the experiment that they had. These are the conclusions of the pilot:

- One of the participants seemed so stubborn to the extend that he/she would rather give an incorrect answer than to use a hint;
- Some questions just took too long, making the participants ask for help, just to save time;
- One of the participants clearly squinted with his/her eyes before asking for a hint, which looked like a confused expression, which we could also see in the captured facial expression as shown in figure 5: there clearly are many high measurements of the eye squinting AU around the time when help is asked. More concretely, Figure 5a shows high eye squinting values roughly from 58000 to 65000 milliseconds and Figure 5b shows this from 62000 to 83000 milliseconds.

We removed the puzzles that took too long for the participants to answer, and clarified the instructions by adding that it's better to use the help button than to guess answers.

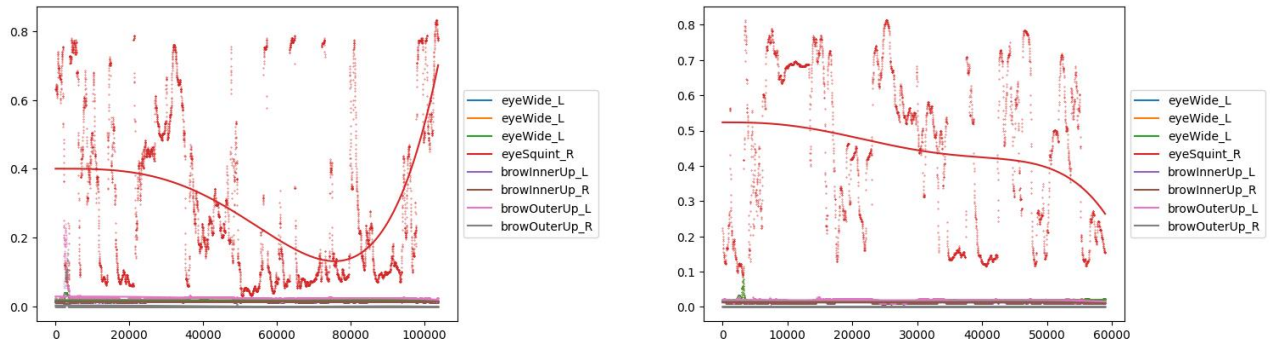
### 3.1.6 Demographic information

Mostly students participated in the experiment. They were asked about their gender, age, education level and latest study they they were educated in if any. In Table 5 all demographic information of




**(a)** Answered incorrectly and asked help.

**(b)** Answered correctly and asked help.




**(c)** Answered correctly without help.

**(d)** Answered incorrectly without help.

**Figure 5:** Plots of a set of action units (AUs 1, 2 and 44 looking at table 4) captured from a participant in the pilot experiment. The x-axis shows time, and y-axis the activation of the parameters. This figure includes two out of three plots of puzzles where help is needed, and two random ones of puzzles where no help is needed. The blue lines represent when the participant asks for help. Each caption of each plot describes if they were answered correctly. From these plots, we can mainly see eye squinting, as you can for example see in Figure 5a around time 60000. Almost all nonzero data points in all graphs are red, which corresponds to eye squinting of the right eye.

the participants is listed, including the puzzles they completed, how often many times they asked for help and how many puzzles they answered correctly. In total, 17 people participated with the main experiment.

### 3.1.7 Constructing the dataset

In total 16 participants answered all 12 questions, and one participant only answered 11. This resulted in 213 files of captured data, including 77 files that recorded a participant asking for help.

| Gender | Age | Education | Study | Puzzles made | Help asked | Correct answers |
|---|---|---|---|---|---|---|
| Female | 21 | Start of Master's Degree | Computer Science: Data Science | 12 | 3 | 9 |
| Male | 23 | Start of Master's Degree | Computer Science: AI | 12 | 3 | 7 |
| Male | 22 | Bachelor's Degree | Computer Science | 12 | 5 | 7 |
| Other | 23 | Start of Bachelor's Degree | Cultural Anthropology and Development Sociology | 12 | 4 | 10 |
| Male | 21 | Bachelor's Degree | Computer Science | 12 | 1 | 5 |
| Female | 61 | Bachelor's Degree | Pabo | 12 | 7 | 5 |
| Male | 21 | Start of Bachelor's Degree | Economie en recht | 12 | 6 | 8 |
| Male | 22 | Start of Bachelor's Degree | Bio Pharmaceutical Sciences | 12 | 3 | 6 |
| Male | 22 | Start of Bachelor's Degree | Public Administration (Bestuurskunde) | 12 | 5 | 6 |
| Female | 20 | Start of Bachelor's Degree | Psychology | 12 | 5 | 8 |
| Male | 22 | Start of Master's Degree | Biology | 11 | 5 | 8 |
| Female | 22 | Start of Bachelor's Degree | Psychology | 12 | 4 | 6 |
| Female | 30 | High School | Industrial design engineering | 12 | 11 | 7 |
| Female | 24 | Bachelor's Degree | International business and languages | 12 | 5 | 3 |
| Female | 21 | Start of Bachelor's Degree | Criminology | 12 | 2 | 9 |
| Male | 25 | Start of Master's Degree | Computer Science | 12 | 6 | 8 |
| Male | 18 | Start of Bachelor's Degree | Artificial Intelligence | 12 | 3 | 6 |

**Table 5:** Demographic information about the participants of the main experiment. This also includes the amount of times they asked for help and answered correctly.

The received data includes data that the classifier does not need like mouse movements, or is formatted incorrectly for the classifier like the help time. So, first, each raw data file (which represents captured data of a participant making *one* puzzle) is parsed to form a comma separated values (CSV) file with the following column names:

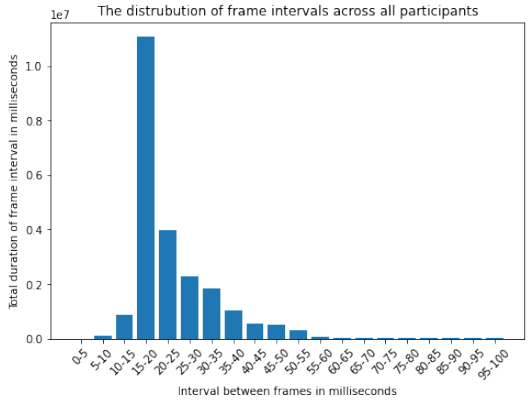
`t (in milliseconds), used hint (0 or 1), ...all mocap4face parameters`

Each row in this CSV file represents a 'frame' (which basically is one extracted and encoded facial expression) of captured data, which is described using these columns. Captured data of one puzzle of one participant can for example include thousands of frames of captures facial expressions. `t` represents the time elapsed in milliseconds since starting the corresponding puzzle. The `used hint` column is filled with 0's, except in the row with a `t`-value that's closest to the help time, which is then labeled with a value of 1. And finally, all the mocap4face features are spread out over the remaining 46 columns. Each of these features has values ranging from 0 to 1.

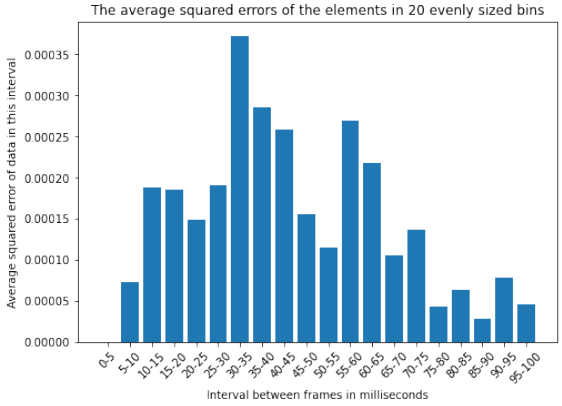
Now, the data still is not ready for the classifier. For the classifier, we needed to sample series of data (time series) that each contain the same number of items, and have a constant interval between the items. Each of these time series also needs to be labeled with 'wants help' or 'does not

want help', depending if the participant asked for help just after the time series.

For making the frame rate constant, a constant sampling frame rate is picked that is high enough to maintain the same variance in the data when sampling at this frame rate, but still as low as possible to form a time series with only the nontrivial and *existing* data. A metric that can describe the variance between the data points is the Mean Squared Error (MSE), because it averages out the squared distance (which is the 'error') between the points. Variance in data increases when the data for example includes more extreme values, or outliers. It would for example also increase if there are many changes of features (which are the facial expressions) over time.




**(a)** Distibution of the intervals between consecutive frames in the entire dataset. Every bar represents a bin that contains all intervals that fall within the corresponding range. Each interval is counted *interval* times.




**(b)** Distribution of the mean squared errors (MSE) of consecutive frames, per interval. Each bar shows the average MSE of all MSE that are placed into that bin.

**Figure 6:** Illustrations of the frame rate distribution and mean squared errors that belong to those frame rates of the entire dataset.

To find the desired frame rate, we looked at the MSE of the mocap4face features between consecutive rows, and the interval between these two measurements. These MSE are then added to one of 20 'bins', depending on the interval. When all MSE are calculated and added to the appropriate bins, the MSE are averaged out per bin. Figure 6b shows these average MSE per bin. Additionally, we counted the distribution of the frame intervals for the same bins, which is done by counting each interval *interval* times in each bin, resulting in Figure 6a. This figure describes the time span of all data in the corresponding bin.

From these results, a resampling frame interval of 35 milliseconds seems optimal.

Figure 6a shows that almost all captured data has a higher frame interval than 35 milliseconds. Resampling with a lower frame interval than the data essentially means that it has to create new data to match this higher frame rate. So when resampling with a frame interval of 35 milliseconds, the resampling method won't need to create much data that does not exist (also called hallucinating data), or sample the same data multiple times. Hallucinated data or repeated samples could reduce the accuracy of a classifier if it for example doesn't follow a pattern that's present in all other data, which could make it harder for the classifier to find these now 'hidden' patterns.

From the distribution of the MSE in Figure 6b it's also clear that this interval of 35 milliseconds is sufficient for sampling. The MSE significantly drops before and after the peak of bin 30-35. When sampling just above this bin, it is possible to keep a similar variance in the data when sampling, because the average distance between the data in this bin is roughly the same. To clarify: if we would for example sample at a much higher frame interval, the sample would not contain as much *changing* data because it would often sample points that have roughly the same values, and ignore the points *in between* that have significant values. With the sampling interval of 35 milliseconds, some measurements have to be skipped, but a good random sample could still accurately represent the entirety of the data, because random missing data won't affect the MSE (or standard deviation) significantly because there are a significant amount of data points in each file.

The drop of MSE in bins with higher intervals than the 35-40 bin could be partly caused by randomness because the amount of items in these bins only represent roughly half an hour of captured facial expressions, whereas the other data accounts for roughly 5 hours. These are rough estimated values derived from Figure 6a to give an idea of the significance. Also, slower capture rates can indicate that some measurements are missing, which can for example happen in poor lighting conditions that makes the face harder to track. We noticed that this was the case when testing the experiment in a poorly lit room.

Finally, to make the data usable for our classifier, the data is resampled using the found optimal frame interval of 35 milliseconds. The goal with this is to create time series with constant sizes and frame intervals of facial expression data that accurately represents the data is was picked from, and label them with yes or no depending on if a hint was used after the time series.

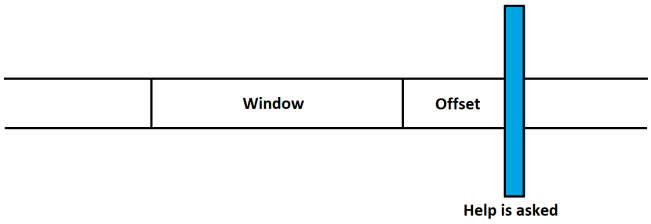



**Figure 7:** An illustration of the offset constraint used with window picking.

Each time series (also called **window**), is picked from the previously constructed CSV files. However, before filling them with data, they are first defined as ranges (that store the start and end times of the windows) with a set of constraints defined by `positive_offset`, `negative_offset`, `negative_ratio`, and `window_size`. In figure 7 the process of defining these windows is illustrated. For positively labeled windows, the distance between the end of the time series and help seeking has to be exactly `positive_offset` milliseconds. Windows that are labeled negatively, have to have a minimal distance of `negative_offset` milliseconds from any help-seeking. The size of these windows is defined with `window_size`, and the ratio of negative to positively labeled windows is defined by `negative_ratio`. The window picking algorithm also enforces the windows to not overlap, and that as many as possible windows are created (meaning that every help asking moment is used).

After the windows are defined, they have to be filled with data from the CSV files using the found

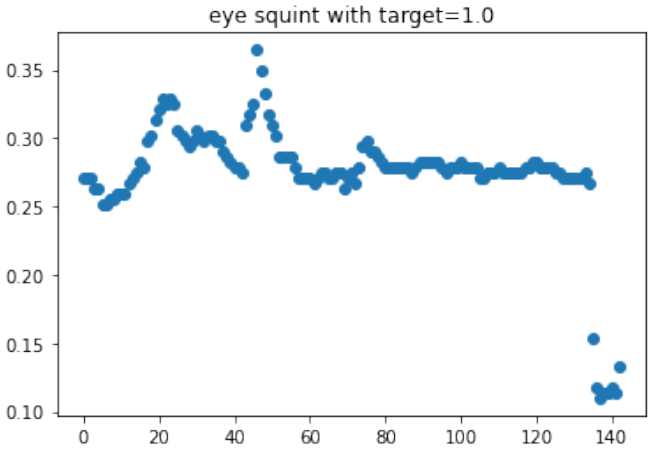



**Figure 8:** An example of the values (on the y-axis) of the eye squint parameter of a random window that was sampled. The x-axis shows the index of the parameter. The size of this window is $143 \cdot 0.035 = 4$ seconds. It is labeled with help asking because the positive_offset for this window is 500 milliseconds, and exactly 500 milliseconds after the window, help was asked.

optimal frame interval. Each window is filled with $\texttt{floor}(\frac{\texttt{window\_length}-1}{\texttt{frame\_interval}})$ items that are spaced evenly. The CSV file contains items without a consistent frame interval, and with different time intervals than the window. Sampling data from this is done by picking the closest real data point to every frame in the window. So, for example, if a window would be defined with $t_{start} = 1$, $t_{end} = 5$ and a frame interval of 2, it would sample data points that are closest to $t = 1$, $t = 3$ and $t = 5$.

The resulting data structure of each window is an array of an arrays of mocap4face-type Action Units. For example, a window that spans 3.5 seconds, will contain $\frac{3.5}{0.035} = 100$ arrays of action units arrays. This window is then paired with a 1 or 0, indicating if the it represents facial expressions of when help was asked or not. Figure 8 shows another example of the values of a feature in the data that a window contains.

## 3.2 Classification

The goal of the classifier is to predict if a participant will ask for help shortly after a time series of facial expressions (also called a window). More specifically, the classifier has to predict which windows are labelled with a 1, and which are labeled with a 0. This was implemented with a deep neural network that uses (a) convolutional layer(s), dense layers and/or a Long-Short-Term Memory (LSTM) layer. In creating this classifier, several experiments have to be conducted to see which of these layers to use, which data should be used and which 'hyper parameters' (explained later on) should be chosen.

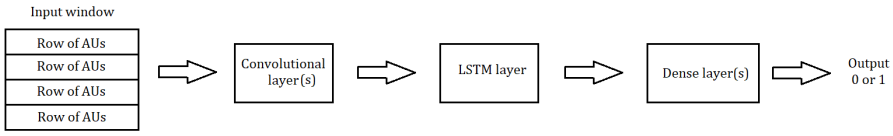



**Figure 9:** A summary of the architecture of the proposed neural network.

### 3.2.1 Neural network structure

A layer in a neural network transforms a set of input values to a (often smaller) set of output values using some calculations that are based on 'weights' and input values. A combination of such layers, like the proposed structure illustrated in Figure 9 can transform a window into a single value, like in this case a 1 or 0, which is the prediction of the network.

This proposed architecture of the neural network is based on [9].

This neural network can learn to make accurate predictions by optimizing the weights. This is done by continuously trying to fit a set of labeled training data, and by using an optimizer algorithm that tries to minimise the error that it makes (the loss) in the predictions. This optimizer can be configured with several so-called hyper-parameters that are mentioned in section 3.2.3.

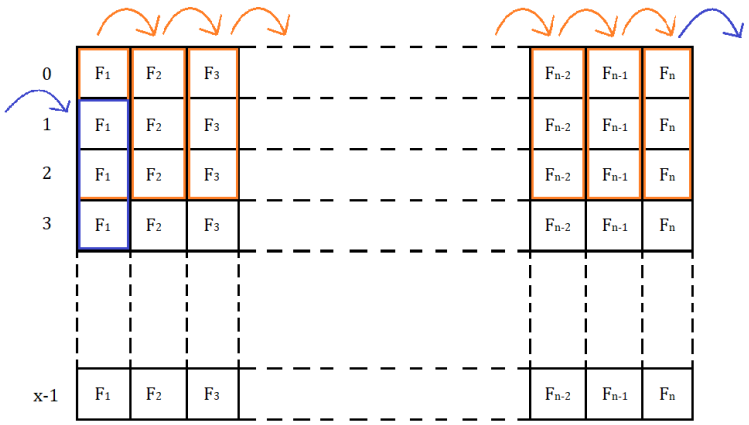



**Figure 10:** An illustration of how a kernel of a convolutional layer would 'walk' (also called convolve) through a window with a step size (strade) of 1, and a kernel size of $3 \times 1$. $F_1 \cdots F_n$ represent all AUs and $x$ is the amount of items in the window. The orange arrows show the kernel moving over the first three rows, and the blue arrows show where the kernel starts to convolve over the second to fourth rows.

The proposed convolutional layer is special because it can 'highlight' patterns with filters comprised of weights that walk over a window. The way it walks over the data is defined by a specified size and strade of the filter(s). The reason for using a convolutional layer to highlight patterns of feature activations over time. To do this, the layer has filters (also called kernels) of width 1 and a height that can be specified and experimented with. The amount of kernels can also be specified. We made it walk over the window one step at a time, as illustrated in figure 10. The goal of this method is

for example to highlight short frowning expressions which have been linked to frustration [16].

These described filter(s) convolute over all features because the purpose of the filters is to highlight simple patterns that could be present in all features, like sudden high activation values, or short activations. When the filters were trained for each feature individually, they would be able to highlight more specific patterns, which is not necessary because the LSTM layer is intended for this.

Similarly to the convolutional layer, the LSTM is also 'special'. The LSTM layer does not only pass its output to the next layer, but also feeds it back into itself. Because of this, it is a type of recurrent neural networks (RNN), which are good at finding relationships between multiple rows of dependent data like time series [23], because it can consider previous data points. A LSTM-layer takes this even further by also keeping a state. This layer can therefore also be used for classifying the sampled windows. An important (hyper)parameter that this layer uses is the filter count.

The other layers in the neural network are dropout and dense layers. A dropout layer will act as an intermediate layer that will pass the data that it receives to the next layer, but will randomly leave out some items by setting them to 0, depending on the specified dropout rate. This helps to prevent 'overfitting'.

A dense layer is a classical fully-connected layer that consists of an input and hidden layer. The size of the hidden layer is defined by another hyper parameter.

### 3.2.2 Implementation

This proposed neural network is implemented with the Tensorflow `2.8.0` library and the Keras package. Keras contains ready-to-use convolutional, LSTM, dense and dropout layers and optimizer algorithms. Additionally, it contains a specialized LSTM layer (called CudnnLSTM) that runs faster on graphics cards using CUDA.

The neural network is trained on a computer with an `intel i7 6700` processor, `16GB DDR4` memory and a `nvidia geforce gtx 1060 6gb` graphics card.

### 3.2.3 hyper parameters

hyper parameters are the set of values that influence the learning process of the neural network [33]. Some are already briefly discussed in the description of the architecture of the proposed neural network, like the size of the kernel of the convolutional layer, or the filter count of the LSTM layer. For the experiments, we assigned default values to all hyper parameters, which can then be seperately tested and optimized. Table 6 lists all the hyper parameters and default values.

| Hyper parameter | Default value | Description |
|---|---|---|
| train, test, validation split | 70:10:20 | How the sampled windows are split to train, validate and test the network. |
| epochs | 100 | How many times the neural network fits the training data. |
| optimizer function | Adam | The algorithm that attempts to change the weights to fit the data better. |
| learning_rate | 5e-4 | Specifier of step size at which the model learns to fit the data. |
| loss function | binary_crossentropy | The function that calculates how the model fits the data |
| dropout rate | 0.3 | The percentage of values that are randomly discarded between each pair of layers. |
| conv_count | 1 | The amount of convolutional + pooling layers at the start of the network. |
| conv_filters | 20 | The number of filters that each convolutional layer trains. |
| conv_height | 3 | The count of rows that the kernel of the convolutional layers spans. |
| lstm_filters | 64 | The amount of hidden nodes that the LSTM layer uses. TODO... |
| dense_count | 2 | The amount of dense layers at the end of the network. |
| activation | relu | The function that almost each node uses to decide what value to output, given the input values. |
| final_activation | sigmoid | The function that outputs a 1 or 0 depending on the values it receives. |
| batch_size | 32 | The number of items from the training set that are passed through the neural network simultanuously. |
| window size | 5000 | The size of the windows in milliseconds. |
| window positive_offset | 500 | The difference in time between the end of a positively labeled window and help asking. |
| window negative_offset | 5000 | The difference in time between the end of a negatively labeled window and help asking. |

**Table 6:** An overview of all hyper parameters and assigned default values.

### 3.2.4 Proposed experiments

For experimentation with the neural network, we split the dataset into a training set, validation set and testing set according to the 'train, test, validation split' ratio's described in table 6. This split is done randomly, but balanced, meaning that each of the splits has a 50/50 ratio of positive versus negative cases. The model is trained with the training set, meaning that the weights will be optimised by only looking at the training set. To see how well it performs after each round of optimisation, the model is evaluated with the validation set. Finally, to see if any optimisations of the model had no bias towards randomly learning to also fit the validation set, but not any other left out data, the model is evaluated with the testing set.

**Hyper parameters**

We conducted several experiments that monitor the training process, with the goal of obtaining useful information to be able to optimize it by fine-tuning hyper parameters. We monitor the training process by visualizing the accuracy and loss (of the training and validation sets) over the epoch index (the training 'round' index). The accuracy shows how accurate the predictions of the NN are, and the loss represents how sure it is of the predictions.

With monitoring the 'training process', we identified and visualized several characteristics:

- Increasing accuracy and/or a decreasing loss show that the network is still learning. When this change stagnates, the optimal amount of epochs (for the given set of hyper parameters) have been reached.
- When the training accuracy increases over the amount of epochs, while the validation accuracy decreases or stagnates, the model is overfitting. This means that it is learning to fit patterns that are present in the training data, but not in any other data like the validation set.
- A high variation (more than 0.1) in the validation accuracy over a small amount of epochs indicates that the model is sometimes randomly fitting the validation set because of good random initialization. This model therefor cannot classify data that it has not seen before with the same accuracy.

First of all, when the training process shows a high variation in validation accuracy, we need to monitor the averages of multiple training processes instead of one. For this experiment, we will take averages of 30.

When the model is overfitting, it could learn to ignore general patterns that are also present in the validation and test sets, because it could find patterns that are more reliable for getting a good *training* accuracy. The goal of this research is to make a model that's useful for all data, also for data that the NN has not seen before. Therefore we experimented with reducing overfitting. Measures against overfitting that we experimented with are to increase the dropout rate from 0.3 to 0.5 and to reduce the amount of trainable weights of the model by halving the `conv_filters` and `lstm_filters`, one at a time. We monitored the training process for each change individually.

In case the validation accuracy decreased when taking these measures, we concluded that these measures countered our goal to optimise the model, because in the process of overfitting, the NN

was also able to learn patterns that are present in the validation set, which are therefore possibly also useful for classification of data that it has not seen before. We always prioritised maximising evaluation accuracy over preventing overfitting. If this is not the case, and if the difference between the training and validation accuracy and/or loss decreased, we concluded that these changes were effective.

When it was clear which overfitting measures were effective, and what effects the changes of those parameters had on the amount of epochs required, we proceeded to do a grid search to find the best combination of considered parameters for validation accuracy. For this we used the amount of epochs that showed optimal losses and accuracies. The grid search includes the following values of hyper parameters:

- dropout (if it was an effective overfit measure): `[0.4, 0.5, 0.7]`;
- dropout: `[0.3, 0.5]`;
- conv_filters (if it was an effective overfit measure): `[1, 5, 10, 20, 40]`;
- conv_filters: `[10, 20, 40, 80, 160]`;
- lstm_filters (if it was an effective overfit measure): `[8, 16, 32, 64]`;
- lstm_filters: `[16, 32, 64, 128]`;

From this grid search, we recorded the combinations of parameter values that gave the best validation accuracy, and used those for the next steps.

**Specific data selection**

Next, we experimented with fine tuning the window size and window offset of the window selection. We used the found hyper parameters in the previous step as a base line for plotting validation accuracies over different window sizes and window offsets. We did this because this can filter out noise, and allows for a more specific answer to our research question: if we find good window parameters, we can exactly say when and for what duration participants show expressions before asking for help.

We do this in two steps:

1. Plot the validation and training accuracy of trained models on shifting windows of length 1000 over a sampled window with length 20000. This means that the model's accuracy is plotted for 20 different offsets.
2. Perform a grid search on different combinations of offsets and window sizes. This step should include combinations of offsets and window sizes that reach the found good offsets that are found in the previous step. So, for example, if an offset of 6000 had a good accuracy in step 1, we know that the windows contain important information from $t = 6000$ to $t = 7000$. This data is then included in some or multiple combinations of the grid search like `{offset: 2000, length: 7000}`.

For step two, we created the following set of values of the window lengths and offsets, of which all combinations were considered for the grid search. This was tweaked during experimentation when a good offset needed to be 'reached'.

```
window lengths = [1000, 2000, 5000, 7000, 10000]
window offsets = [500, 1000, 5000, 6000]
```

The best offset and window length combination was used for the next experiments.

Next, we looked at which features were used in the classification. This was analyzed by leasing one feature at a time when training the model. If the validation accuracy did not decrease significantly when ignoring a feature, the feature was considered useless for the classifier. We did not lease the trivial features in the next steps of experimentation, because it is not necessary for model optimisation, as the model should be able to 'learn' which features to ignore.

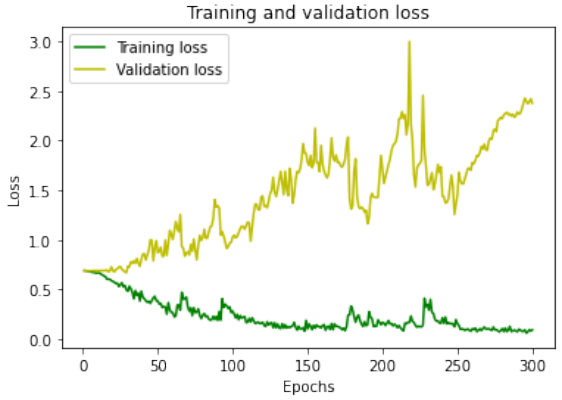
Similarly to looking at how features influence the validation accuracy of trained models, we looked how participants influenced the training process. To do this, the model was trained individually for each participant. So, it was trained 17 times. Again, the (average) validation accuracy is reported for each participant.

Finally, we evaluated the model with confusion matrices that show the predictions against the ground truths. This allows us to gain an insight in the precision (true positives to false positives ratio) and the recall (true positives to false negatives ratio).

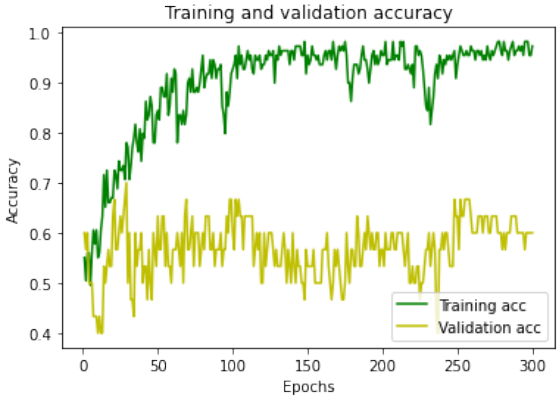
# 4 Results

There are two types of experiments conducted to train and evaluate the model: (i) optimizing hyper parameters and (ii) specific data selection.

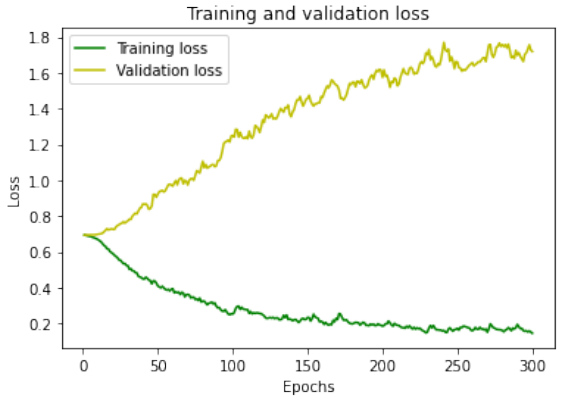
## 4.1 Hyper parameter optimization



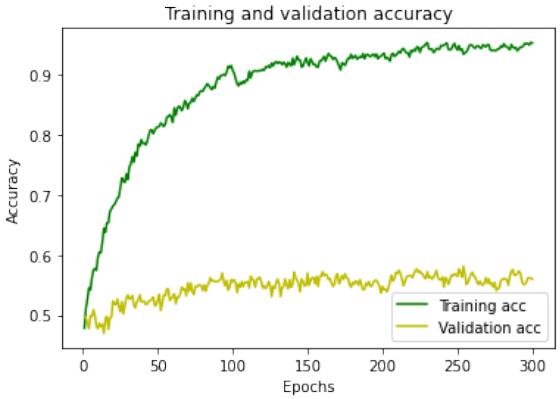

**(a)** The recorded (training and validation) binary cross entropy loss of one fitted model plotted over epochs.




**(b)** The recorded (training and validation) accuracies of one fitted model plotted over epochs.



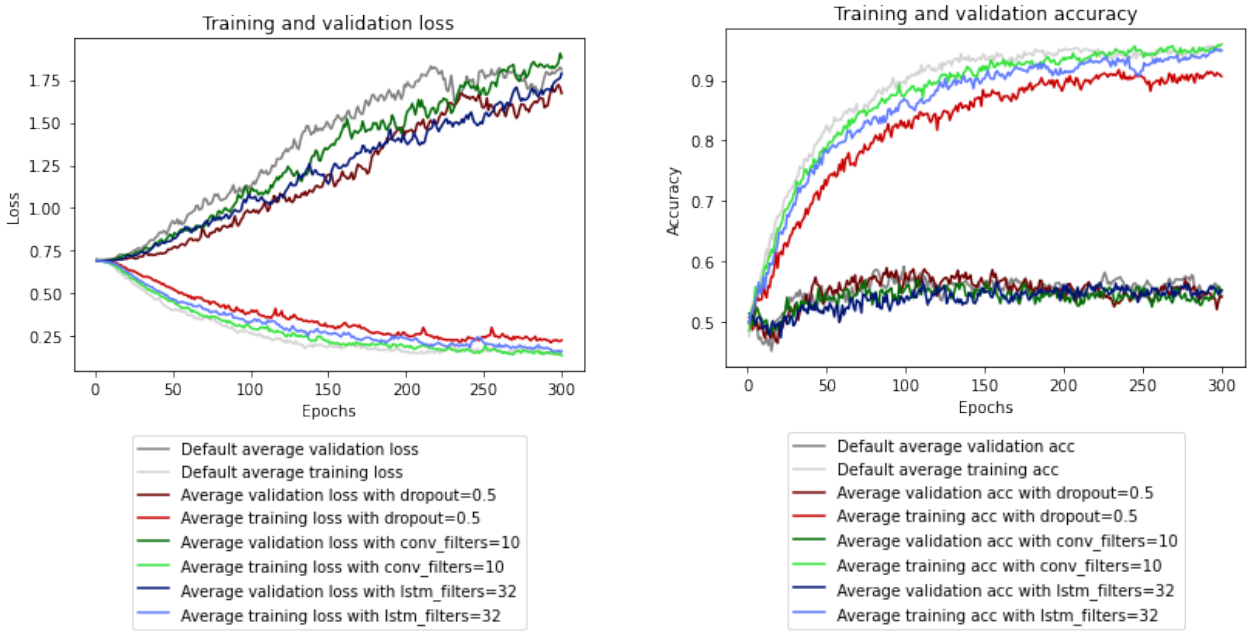

**(c)** The average recorded (training and validation) binary cross entropy losses of 30 fitted models plotted over epochs.




**(d)** The recorded (training and validation) accuracies of 30 fitted models plotted over epochs.

**Figure 11:** Graphs of the average training and validation losses and accuracies over the number of epochs.

Figure 11 shows the progressing accuracies and losses when a neural network is trained with 300 epochs. It also includes averages of these values of 30 trained neural networks, because there was a high degree of randomness in 11b and 11a. The model is overfitting because the training accuracy is much higher than the validation accuracy, and because the training loss reduces, while the validation loss increases. This means that we next need to look at halved filter counts and increase the dropout rate to 0.5 to see if these measures can be effective.




**(a)** Mean square loss over epochs.

**(b)** Accuracy over epochs.

**Figure 12:** Graphs of average (from 30 runs) training and validation losses (12a) and accuracies (12b) over the number of epochs for a set of hyper parameter changes.

Figure 12 shows the effect on the training process of changing the hyper parameters `dropout`, `conv_filters`, `lstm_filters`. It shows that the difference between the training and validation results decreases, but that the accuracy actually decreases when doing so. The overfitting model seems to be able to best fit the validation set.

It is also clear that the model needs around 100 epochs for the best accuracy, which holds true for every hyper parameter change. After 100 epochs, the accuracy does not increase, and seems to even slowly decrease.

So, to more specifically see the effect of changing hyper parameter values, only 100 epochs need to be used. The next step described in the methodology is to conduct a grid search of filter counts and dropout rates. The described values that we use in this grid search are:

- dropout: `[0.3, 0.5]`;
- conv_filters: `[10, 20, 40, 80, 160]`;
- lstm_filters: `[16, 32, 64, 128]`;

Table 7 shows the results of the grid search. The parameter values of the top result are used in the next experiments, which are dropout = 0.3, conv_filters = 10 and lstm_filters = 128.

| avg validation accuracy | dropout | conv_filters | lstm_filters |
|---|---|---|---|
| 0.588889 | 0.3 | 10 | 128 |
| 0.581111 | 0.5 | 10 | 128 |
| 0.574444 | 0.3 | 40 | 64 |
| 0.574444 | 0.3 | 20 | 64 |
| 0.5711111 | 0.5 | 20 | 128 |
| 0.567778 | 0.5 | 40 | 128 |
| 0.565556 | 0.5 | 20 | 64 |
| 0.564444 | 0.5 | 80 | 64 |
| 0.564444 | 0.5 | 40 | 64 |

**Table 7:** The best results of the grid search for the dropout, conv_filters and lstm_filters hyper parameters.

## 4.2 Window picking optimisation

In the next couple of experiments, the influence of the window length and offset are analysed.

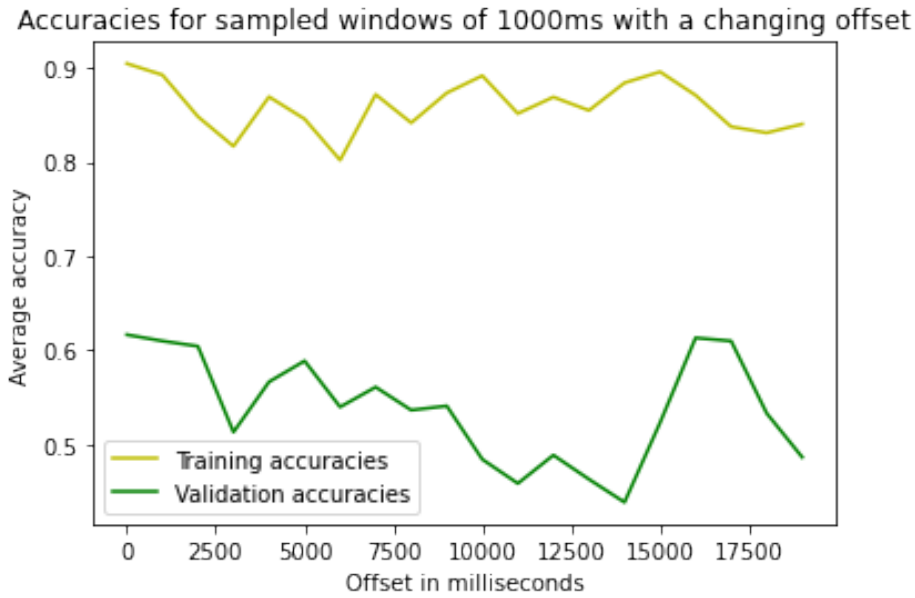



**Figure 13:** A figure that shows the accuracies of a window that is 1 second long with different offsets from when the hint was used.

Figure 13 displays the average validation and training accuracy of the model that's trained on the 20 segments that each span 1 second of a window that spans 20 seconds. For each segment, the average results are taken from 30 models that were trained on that segment.

In the next experiment we did a grid search of a set of window lengths and window offsets. The described set of values are:

```
window lengths = [1000, 2000, 5000, 7000, 10000]
window offsets = [500, 1000, 5000, 6000]
```

This set of values needs to be tweaked to reach good segments shown in figure 13. Good segments are 1000-2000, 5000-6000, 16000-17000 and 17000-18000. The segments spanning 16000 to 18000 are not yet included in the grid search. We include them by changing the offset 6000 to 7000. Table 8 shows the 10 combinations of parameters with the best average validation accuracy when training the model with those parameters.

| Average validation acc | window length | window offset |
|---|---|---|
| 0.671111 | 11000 | 500 |
| 0.645556 | 11000 | 1000 |
| 0.611111 | 5000 | 500 |
| 0.601111 | 7000 | 1000 |
| 0.565556 | 7000 | 500 |
| 0.558889 | 7000 | 6000 |
| 0.556667 | 1000 | 500 |
| 0.545556 | 11000 | 5000 |
| 0.544445 | 1000 | 1000 |
| 0.541111 | 2000 | 500 |

**Table 8:** The results of a grid search of combinations of window length and window offset values.

It's clear from table 8 that a window length of 11000 milliseconds with an offset of 500 milliseconds gave the best results. We used this result in the next experiments.

## 4.3 Relevant Features

This experiment shows the effect on the validation accuracy of each feature when 'ignoring' it in training and validation by setting all values of the feature in the dataset to 0. Table 9 lists the results.

| Ignored feature | Avg. val. acc. | Ignored feature | Avg. val. acc. | Ignored feature | Avg. val. acc. |
|---|---|---|---|---|---|
| None | 0.6722 | jawLeft | 0.6522 | eyeLookDown_R | 0.6944 |
| browOuterUp_L | 0.6756 | cheekPuff | 0.6700 | eyeLookIn_R | 0.6822 |
| browInnerUp_L | 0.6700 | mouthShugUpper | 0.6833 | eyeLookOut_R | 0.6911 |
| browDown_L | 0.6767 | mouthFunnel | 0.6856 | eyeLookUp_R | 0.6911 |
| eyeBlink_L | 0.6611 | mouthRollLower | 0.6889 | eyeWide_R | 0.6700 |
| eyeSquint_L | 0.6733 | jawOpen | 0.6811 | eyeSquint_R | 0.6800 |
| eyeWide_L | 0.6689 | tongueOut | 0.6833 | eyeBlink_R | 0.6467 |
| eyeLookUp_L | 0.6856 | mouthPucker | 0.6811 | browDown_R | 0.6667 |
| eyeLookOut_L | 0.6722 | mouthRollUpper | 0.6822 | browInnerUp_R | 0.6856 |
| eyeLookIn_L | 0.6967 | jawRight | 0.6800 | browOuterUp_R | 0.6778 |
| eyeLookDown_L | 0.6689 | mouthLowerDown_R | 0.6978 | headLeft | 0.6756 |
| noseSneer_L | 0.6978 | mouthFrown_R | 0.6711 | headRight | 0.6667 |
| mouthUpperUp_L | 0.6833 | mouthRight | 0.6956 | headUp | 0.6689 |
| mouthSmile_L | 0.6933 | mouthSmile_R | 0.6667 | headDown | 0.6556 |
| mouthLeft | 0.6744 | mouthUpperUp_R | 0.6833 | headRollLeft | 0.6756 |
| mouthFrown_L | 0.6778 | noseSneer_R | 0.6833 | headRollRight | 0.6700 |
| mouthLowerDown_L | 0.6822 | | | | |

**Table 9:** The average validation accuracies when training the model on a dataset 30 times where a feature is 'ignored'.

## 4.4 Training on individual participants

Next, the model was trained on 17 datasets, each containing the data of one participant. Some datasets did not contain enough items to train a model on. The results are listed in Table 10.

| Training set size | Validation set size | Average validation accuracy |
|---|---|---|
| 4 | 2 | 83.3% |
| 4 | 2 | 55.0% |
| 6 | 4 | 32.5% |
| 2 | 0 | - |
| 9 | 5 | 60.0% |
| 8 | 4 | 55.8% |
| 4 | 2 | 43.33% |
| 6 | 4 | 72.5% |
| 6 | 4 | 50.0% |
| 6 | 4 | 40.0% |
| 4 | 2 | 10.0% |
| 14 | 8 | 50.8% |
| 6 | 4 | 46.7% |
| 3 | 1 | - |
| 5 | 3 | 65.6% |
| 8 | 4 | 34.2% |
| 4 | 2 | 40.0% |
| 0 | 0 | - |

**Table 10:** The average (from 30 trained models) validation accuracy where the model is trained on datasets from individual participants.

## 4.5 Model evaluation

| | Actual yes | Actual no | |
|---|---|---|---|
| Predicted yes | 9 | 2 | precision 81.8% |
| Predicted no | 6 | 13 | |
| | recall 60.0% | | accuracy 73.3% |

**(a)** Validation confusion matrix.

| | Actual yes | Actual no | |
|---|---|---|---|
| Predicted yes | 4 | 3 | precision 57.1% |
| Predicted no | 3 | 5 | |
| | recall 57.1% | | accuracy 60.0% |

**(b)** Test confusion matrix.

**Table 11:** Confusion matrices of a trained model on the validation and the test sets, that include the precision, recall and accuracy.

Finally, the neural network was evaluated. The results are shown in confusion matrices in Tables 11. Table 11a shows the predictions of the model on the validation set, and table 11b shows the predictions on the left out test set.

# 5 Discussion

In this study we created a quiz where participants could ask for help by pressing a button, captured their facial expressions through their webcam, and designed and tested a neural network that was able to make predictions about help asking.

The goal of this study was to predict when students want help, which is not exactly the same as a student asking for help, which we captured in the experiment. In this context, the relationship between wanting help and actually seeking help by pressing a button is somewhat unclear. Although research points out that students may ask for help non-verbally, through expressions (so without asking for it) [13], it is unclear if they would also ask for help non-verbally if there is an easy-to-use help button. Further research could experiment with a similar system, but also review when students *look* like they want help to gain more insight into this relationship.

We were able to run the experiment on a website, enabling everyone to participate from home. This is very similar to popular ITS applications like Snappet [5] or Duolingo [2]. Moreover, the same data capturing methods could actually be added to both ITS. The data collection of this research is (intentionally) very practical, and highly anonymous because no video data is collected or sent.

However, we noticed a few caveats with this approach. The frame rates at which data was captured was not constant, and could even contain gaps of data when someone's face could not be tracked for a while. A poor camera or poor lighting conditions could cause this. This meant that we had to (re)sample the data to be able to use it in the classifier. Also, participants' facial expressions could be *not* related to seeking help, and making the quiz, if they were, for example, talking with family members or listening to music while participating in the experiment, introducing noise into the dataset. Further research could enforce better, more constant, conditions or monitor participant's camera feed.

The library Mocap4face [4] that was used to capture facial expressions, was able to capture facial expressions directly in the browser of participants. However, it uses non-conventional FACS encoding, as shown in Table 4. This meant that we could not focus on the selected AUs of frustration and confusion detection for classification. Further research could use another toolbox like Open Face, in order to compare which AUs are needed for the classification.

We took samples of sequential data, which we call windows or time series, from all captured data, which have a constant time interval between each item and a constant length. We succeeded in analysing which frame interval to pick to be able to sample data without biases. In this process, the same number of windows were sampled with (positively labeled) and without (negatively labeled) the window ending in the participant asking for help, to form a balanced dataset.

Our classifier was able to predict if those windows ended with the participant asking for help or not. This classifier used a convolutional layer and a Long-Short-Term Memory (LSTM) layer to be able to remember patterns of facial expressions over time (temporal context), similar to [9]. The final testing accuracy was 60%, as shown in Table 11b.

However, for this final evaluation, a very small testing set was used that included 15 items. This uneven amount also means that the dataset could not be balanced: 7 items were labeled with yes,

while 8 are labeled with no. The confusion matrix shows that the model is able to predict negatively labeled windows better than positively labeled windows, implying that the accuracy on a balanced test dataset might be lower.

Also, it is clear from the validation accuracy in Table 11a, that there is a high degree of randomness in the accuracy. Earlier results in Table 9 show an average validation accuracy of 67.22%, which means that the validation accuracy can fluctuate by at least $73.3 - 67.2 \approx 6.1\%$.

Another thing that we found when 'monitoring' the training process in Figure 11, is that the model was overfitting, because it got a 90% accuracy on the training set while only achieving 60% or lower accuracy on the validation set. When reducing the number of trainable weights, and increasing the dropout rate in Figure 12, the training accuracy reduced, but so did the validation accuracy. Therefore we did not take any measures against overfitting. We think that the model was overfitting because there might be *much noise in the captured data* and/or too little training validation and testing data. The variation (or randomness) in accuracy confirms this.

This noise could be caused by the earlier described caveats in the data collection of the experiment, and the yet unknown relationship between expressing 'wanting help' while not explicitly asking for help. It could be impossible for the classifier to distinguish between expressions that participants make when 'wanting help while not asking for it' and 'asking for help', because the expressions could be the same.

After monitoring the training process, we were able to find a good combination of conv_filters, lstm_filters and the dropout hyper parameters as listed in Table 7. We specifically chose to optimise the conv_filters and lstm_filters because they were most directly responsible for the amount of trainable weights in the model, and therefore also for the complexity of the model and its ability to overfit. In hindsight, it was not necessary to experiment with the dropout rate, because the top 2 results of the grid search include the 2 different dropout values, meaning that it was most-likely trivial.

In this grid search, there are no real significant changes in validation accuracy, especially if we consider that the model's accuracy is very random, and could easily have a confidence interval of $\pm 0.1$, even when taking an average of 30. However, from the grid search, it seems that there still is a pattern: the more convolutional filters we use, the less LSTM filters we need, and vica versa. But again, it is very difficult to find such patterns in data with a very low variation and a large confidence interval. Further research could circumvent this by using a larger dataset, with less noise.

The results in specifying the exact start time and end time of considered data show a clear pattern. Figure 13 shows that when a model trained on any windows that span the last 0-1, 1-2, 2-3, 3-4, 4-5, 5-6, 6-7 and 8-9 seconds of the window (so before the point where help is asked), that there was enough data for the model to perform better than random, and also from 16-17 and 17-18 seconds. However, it is noteworthy that the average accuracy drops substantially after 18 seconds and before 16 seconds. We suspect that this happened because the validation set was randomly selected, and did not contain any data with patterns before 16 seconds and after 18 seconds by chance. Further research could circumvent this by using a larger validation set.

The grid search of a good window length with offset, summarized in Table 8, shows a combination that includes the first 11.5 seconds, which is in line with the findings of the previous experiment, apart from not including the peak of information from 16 to 18 seconds. This confirms that this peak could be caused by chance, making the model overfit to the validation set. We could also have looked at the accuracy on the test set, but this could introduce a bias to also overfit on the test set when optimising the model's hyper parameters with the best results of the grid search.

The best window length is 11 seconds, with an offset of 500 milliseconds, which is the minimal offset that we included in the grid search to prevent biases. It seems that longer window sizes with small offsets are substantially better for the model to classify wanting help. This leads us to believe that the relevant data is at the end of the window, towards the point where help is asked. It could also be the case that it is easier for the model to learn from longer time series which windows are labeled negatively.

These large window sizes indicate that the architecture of the model is sufficient to analyse (longer) time series. We suspect that temporal context also plays a role in the classification of seeking help, like [9] also concluded for confusion detection. Further research could analyse the role of temporal context in this classification problem.

The results about which features are relevant for the training process seem random, and are not in line with earlier observations of for example eye squinting being relevant. When lesioning some features like eyeLookIn˙L, the validation accuracy actually increased from 0.6722 to 0.6967, indicating that it might be worth researching which features should be excluded. However, no features seemed essential to be included in classification. The most sustantial ones are eye blinking (left: 0.6611, right: 0.6467), head down (0.6556) and jaw left (0.6522). These features weren't linked to confusion or frustration in earlier research, but they could be important indicators of wanting help. Further research is needed to prove that these values are not caused by chance.

Training on data from individual participants seems to be very fruitful, as listed in Table 10. Even though the data sets are small, the validation accuracy of the model seemed to fluctuate substantially when trained on different participants. This indicates that individuals may express wanting help differently. However, because of the small data sets, it is not clear if these results are random. Further research could focus on individual differences.

After the optimisation of all hyper parameters with the described grid searches, the model performs better on the validation set than on the testing set that was left out. This indicates that the model overfitted to the validation set in the process of optimising the hyper parameters. Especially optimising the window-picking parameters could have caused this, because it could have selected window lengths and offsets to form windows that contain patterns that are both present in the training and validation sets, but not in the testing set.

Despite this, the model is able to perform slightly better than chance (60% versus 53.33%), meaning that it is likely able to predict when people are about to ask for help, in a balanced dataset. Further research could experiment with improving the accuracy, with unbalanced datasets and applying this in an ITS.

# 6 Conclusions

The goal of this research was to predict if students want help by looking at their facial expressions. To capture if students want help, we set up an experiment where they were able to ask for help while their facial expressions were captured. We then constructed a classifier that was able to predict with up to 60% accuracy if people were about to press the help button. We also saw indications that the model performed better when trained on some participants than others. From previous work it's clear that this could be a useful prediction to make in intelligent tutoring systems (ITS). However, in the results there are many signs that the dataset was insufficient in size. These signs include (i) random results, (ii) unclear significance of individual FACS parameters and (iii) very 'extreme' accuracies on individuals ranging from 10% to 83.3%.

# References


[1] Cloudflare. https://www.cloudflare.com/. Accessed: 2022-03-05.

[2] Duolingo. https://www.duolingo.com/. Accessed: 2022-05-20.

[3] Mensa puzzles. https://www.mensa.org.uk/puzzles. Accessed: 2022-01-10.

[4] mocap4face. https://alter.xyz/mocap4face/. Accessed: 2022-01-20.

[5] Snappet nl. https://nl.snappet.org/. Accessed: 2022-05-14.

[6] Snappet us research. https://us.snappet.org/research/. Accessed: 2022-05-16.

[7] Ali Alkhatlan and Jugal Kalita. Intelligent tutoring systems: A comprehensive historical survey with recent developments. *arXiv preprint arXiv:1812.09628*, 2018.

[8] Tadas Baltrusaitis, Amir Zadeh, Yao Chong Lim, and Louis-Philippe Morency. Openface 2.0: Facial behavior analysis toolkit. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 59–66. IEEE, 2018.

[9] Niklas Borges, Ludvig Lindblom, Ben Clarke, Anna Gander, and Robert Lowe. Classifying confusion: autodetection of communicative misunderstandings using facial action units. In *2019 8th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*, pages 401–406. IEEE, 2019.

[10] Joost Broekens. Emotion. In *The Handbook on Socially Interactive Agents: 20 years of Research on Embodied Conversational Agents, Intelligent Virtual Agents, and Social Robotics Volume 1: Methods, Behavior, Cognition*, pages 349–384. 2021.

[11] Joshua R De Leeuw. jspsych: A javascript library for creating behavioral experiments in a web browser. *Behavior research methods*, 47(1):1–12, 2015.

[12] Jeanine A DeFalco, Jonathan P Rowe, Luc Paquette, Vasiliki Georgoulas-Sherry, Keith Brawner, Bradford W Mott, Ryan S Baker, and James C Lester. Detecting and addressing frustration in a serious game for military training. *International Journal of Artificial Intelligence in Education*, 28(2):152–193, 2018.

[13] Bella M DePaulo and Jeffrey D Fisher. Too tuned-out to take: The role of nonverbal sensitivity in help-seeking. *Personality and Social Psychology Bulletin*, 7(2):201–205, 1981.

[14] Itir Onal Ertugrul, László A Jeni, Wanqiao Ding, and Jeffrey F Cohn. Afar: A deep learning based tool for automated facial affect recognition. In *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, pages 1–1. IEEE, 2019.

[15] BrynBryn Farnsworth Farnsworth. Facial action coding system (facs) – a visual guidebook. https://imotions.com/blog/facial-action-coding-system/, Aug 2019.

[16] Joseph Grafsgaard, Joseph B Wiggins, Kristy Elizabeth Boyer, Eric N Wiebe, and James Lester. Automatically recognizing facial expression: Predicting engagement and frustration. In *Educational data mining 2013*, 2013.

[17] Joseph F Grafsgaard, Kristy Elizabeth Boyer, and James C Lester. Toward a machine learning framework for understanding affective tutorial interaction. In *International Conference on Intelligent Tutoring Systems*, pages 52–58. Springer, 2012.

[18] Muhammad Asif Hasan, Nurul Fazmidar Mohd Noor, Siti Soraya Abdul Rahman, and Mohammad Mustaneer Rahman. The transition from intelligent to affective tutoring system: A review and open issues. *IEEE Access*, 2020.

[19] Fatima I Yasser, Bassam H Abd, and Saad M Abbas. Detection of confusion behavior using a facial expression based on different classification algorithms. *Engineering and Technology Journal*, 39(2):316–325, 2021.

[20] Agata Kołakowska, Agnieszka Landowska, Mariusz Szwoch, Wioleta Szwoch, and Michał Wróbel. *Modeling emotions for affect-aware applications*, pages 55–67. 01 2015.

[21] Hao-Chiang Koong Lin, Ching-Ju Chao, and Tsu-Ching Huang. From a perspective on foreign language learning anxiety to develop an affective tutoring system. *Educational Technology Research and Development*, 63(5):727–747, 2015.

[22] Lisa Linnenbrink-Garcia and Reinhard Pekrun. Students' emotions and academic engagement: Introduction to the special issue. *Contemporary Educational Psychology*, 36(1):1–3, 2011.

[23] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[24] Shushi Namba, Wataru Sato, Masaki Osumi, and Koh Shimokawa. Assessing automated facial action unit detection systems for analyzing cross-domain facial expression databases. *Sensors*, 21(12):4222, 2021.

[25] Chakradhar Pabba and Praveen Kumar. An intelligent system for monitoring students' engagement in large classroom teaching through facial expression recognition. *Expert Systems*, 39(1):e12839, 2022.

[26] Reinhard Pekrun, Thomas Goetz, Wolfram Titz, and Raymond P Perry. Positive emotions in education, 2002.

[27] Ömer M Soysal, Shahrzad Shirzad, and Kazim Sekeroglu. Facial action unit recognition using data mining integrated deep learning. In *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 437–443. IEEE, 2017.

[28] CP Sumathi, T Santhanam, and M Mahadevi. Automatic facial expression analysis a survey. *International Journal of Computer Science and Engineering Survey*, 3(6):47, 2012.

[29] Nik Thompson and Tanya Jane McGill. Genetics with jean: the design, development and evaluation of an affective tutoring system. *Educational Technology Research and Development*, 65(2):279–299, 2017.

[30] Kurt VanLehn. The behavior of tutoring systems. *International journal of artificial intelligence in education*, 16(3):227–265, 2006.

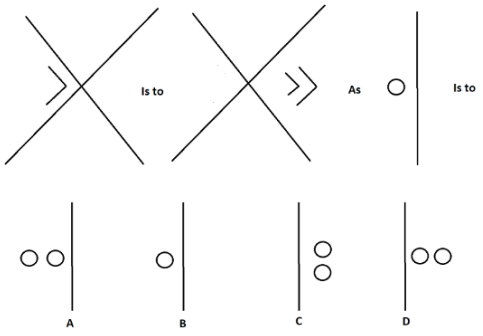
[31] Cheng-Hung Wang and Hao-Chiang Koong Lin. Emotional design tutoring system based on multimodal affective computing techniques. *International Journal of Distance Education Technologies (IJDET)*, 16(1):103–117, 2018.

[32] Tao-Hua Wang, Hao-Chiang Koong Lin, Hong-Ren Chen, Yueh-Min Huang, Wei-Ting Yeh, and Cheng-Tsung Li. Usability of an affective emotional learning tutoring system for mobile devices. *Sustainability*, 13(14):7890, 2021.

[33] Tong Yu and Hong Zhu. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*, 2020.

[34] Ruicong Zhi, Mengyi Liu, and Dezheng Zhang. A comprehensive survey on automatic facial action unit analysis. *The Visual Computer*, 36(5):1067–1093, 2020.

# A Puzzles in the data collection experiment

| | Question | Hint |
|---|---|---|
| Q1 | In this puzzle you have to look what the ouput is of a program written with JavaScript. It does not matter if you don't have prior knowledge of JavaScript. You can find the output between the brackets of the *console.log()* statement. Here's the script:<br>`const dictionary = new dict();`<br>`for(let i = 0; i < 4; i++) {`<br>`dictionary[String.fromCharCode(65 + i)] = i*2;`<br>`}`<br>`console.log(Object.keys(dictionary).join(", "));`<br>**What is the output of this script?** | Hint: the output of the script:<br>`const dictionary = new dict();`<br>`for(let i = 0; i < 5; i++) {`<br>`dictionary[String.fromCharCode(65 + i)] = i*2;`<br>`}`<br>`console.log(Object.keys(dictionary).join(", "));`<br>Is *A, B, C, D, E* |
| Q2 | This question is about a concept that basically all programming languages share: variables. In programming languages, you can store values in *variables*. These values could be numbers, text, or other more complicated data.<br>Lets start with storing a number. We can assign a value of 5 to the variable named *test* by writing (in JavaScript again)<br>`let test = 5;`<br>To assign a text value to a variable, you need to use quotes.<br>**How do we create a variable named *test2* with the text-value *hello world*?** | We can give the variable named *hint* a text value of *hint tekst*, by writing:<br>`let hint = "hint tekst";` |
| Q3 | Which number is missed out?<br>46 34 24 16 **?** 6 4 | Look by how much the numbers count back each time. Do you see a pattern? |
| Q4 | In a jewellers, Alfie is buying Ophelia an Opal ring.<br>Byron is buying a ring for Ruth, which precious stone does it contain? | Try to combine some letters of the names to form the stone name. |
| Q5 |  | Look at the number of elements before and after the lines. |
| Q6 | What letter comes next?<br>A Y C W E U G ? | The letters Y, W, U count back... |

<table>
<tr><td>Q7</td><td>Which numbers could replace the question marks? Write them seperated with a komma.<br><br>12 17 85<br>15 19 76<br>18 21 63<br>21 ? ?</td><td>In the first row, add 5 and multiply by 5 to get 85. Do this with 4 with the second row and so on..</td></tr>
<tr><td>Q8</td><td>This is an unusual safe. Each button needs to be pressed exactly once to reach the middle (X). You can find an instruction on each button that points to the next button that needs to be pressed. 1U means 1 up, 2D = 2 down, 3R = 3 to the right en 3L = 3 to the left<br><br><table><tr><td>1D</td><td>1R</td><td>2D</td><td>3L</td><td>3L</td></tr><tr><td>4R</td><td>2R</td><td>2D</td><td>1L</td><td>3D</td></tr><tr><td>3R</td><td>1U</td><td>X</td><td>2U</td><td>4L</td></tr><tr><td>4R</td><td>1L</td><td>1R</td><td>2L</td><td>1U</td></tr><tr><td>2R</td><td>2U</td><td>1R</td><td>2L</td><td>4U</td></tr></table><br>What is the instruction of the first button that you need to press?</td><td>Look which button cannot be reached by following an instruction on another button. That must be the first button.</td></tr>
<tr><td>Q9</td><td>Which number can replace the question mark?<br><br>25---14<br>67---26<br>19---20<br>61---14<br>95---?</td><td>Add the two digits of the first number and try to multiply this by something to get the number on the right.</td></tr>
<tr><td>Q10</td><td>Which number should replace the question mark?<br><br><table><tr><td>5</td><td>8</td><td>1</td><td>7</td><td>4</td></tr><tr><td>3</td><td>4</td><td>7</td><td>5</td><td>4</td></tr><tr><td>9</td><td>4</td><td>5</td><td>3</td><td>8</td></tr><tr><td>1</td><td>6</td><td>5</td><td>7</td><td>2</td></tr><tr><td>6</td><td>7</td><td>2</td><td>6</td><td>?</td></tr></table></td><td>Try to make the number 20.</td></tr>
<tr><td>Q11</td><td>Find words to fit the clues, all the words end in the same three letters. What are the words? Seperate them with a space in order of the question.<br><br>_ _ _ _ _ _ Eat quickly<br><br>_ _ _ _ _ _ Unverified story<br><br>_ _ _ _ _ An outline</td><td>The first one starts with d.....</td></tr>
<tr><td>Q12</td><td>Find words to fit the clues. Each group of crosses should be replaced with the same three letter word. What are the words? Seperate them with a space in order of the question.<br><br>_ _ X X X _ Put in money or effort<br>_ _ _ X X X _ Gather crops<br>_ _ X X X _ _ _ _ Secretly listen</td><td>The first X is a V</td></tr>
</table>

**Table 12:** All the questions and hints used in the data collection experiment. Q3 to Q12 are from [3].