Human-Centered Software Design Embedded Camera Project

Daniel Bahrami
Syddansk Universitet
Odense, Denmark
dabah20@student.sdu.dk

Stefan-Daniel Horvath
Syddansk Universitet
Odense, Denmark
sthor23@student.sdu.dk

Ahmadullah Naibi
Syddansk Universitet
Odense, Denmark
ahnai17@student.sdu.dk

***Abstract*—This paper presents a novel approach to enhance user interface (UI) testing by incorporating emotion detection through facial expression analysis using an embedded camera. Leveraging machine learning algorithms, the project aims to observe and record users' emotional responses during product interaction. This automated process is designed to identify moments of negative emotions or surprise, providing software engineers with valuable insights into user experiences and pinpointing areas in the UI that may trigger frustration or disengagement.**
The system comprises three primary components: a Python script for real-time emotion detection, a notebook for model training, and an analysis script for data visualization. The emotion detection component utilizes Convolutional Neural Networks (CNNs) optimized for accuracy and bias reduction, while ensuring adherence to ethical standards, particularly in data collection and user privacy under GDPR guidelines.
The project's methodology blends qualitative and quantitative data collection, enabling a holistic view of the user experience. Additionally, the research explores foundational theories in emotion recognition, such as Paul Ekman's theory of basic emotions and Robert Plutchik's wheel of emotions, to underpin the system's design and functionality.
Experimental results demonstrate the system's efficacy in providing detailed emotional insights, compared to traditional manual UI testing methods. Future enhancements include integrating advanced models like Yolov8 and PAtt-Lite for more accurate emotion detection and expanding the system's capabilities for broader application in UI testing and user experience optimization. The project's findings and methodologies offer significant implications for the future of user-centric software development and emotional analysis in human-computer interaction.

***Index Terms*—Machine vision, Human computer interaction, human centered software development, machine learning, embedded camera**

I. Introduction

A. Overview

The paper describes the development of the project regarding human interaction for an embedded camera, where the purpose is to detect people's emotions by their facial expressions during usage of the product to gain information about, what the user liked and disliked when they were testing the product. To identify faces different machine learning algorithms were tried, and tested to select the best performing one among them for detecting facial expressions, where the algorithm would run on a UI that users can use to record their facial expressions, which then get saved as a CSV file. Experiments were conducted using participants using the system, where afterward they were interviewed about their experience with it to provide feedback to the developers of the system for future improvements.

B. Problem and Requirement Definition

1) Problem Statement: The process of testing user interfaces (UI) can often be inefficient and time-consuming, particularly when it comes to understanding the user's emotional and cognitive responses to the interface. Traditional methods may not effectively capture subtle yet critical user reactions that are key to optimizing UI design.

2) Project Aim: Our project aims to address this challenge by streamlining the UI testing process. We propose a software solution that leverages machine learning algorithms to observe and analyze people's reactions as they navigate through a UI. This system will record these observations in a timeline format, making it easily interpret-able for software/UI engineers.

3) Requirements: The envisioned system is designed to provide insights into how users are affected by other software, detecting moments when the user's emotions become negative or indicate surprise, pinpointing timestamps in a recording where a developer should review to find what is causing these emotions.
The system should be able to pinpointing areas in the UI that may be causing frustration or disengagement. These insights can be instrumental in enhancing user experience by targeting areas for improvement.

C. System Components

The system comprises three main components:

- A Python script equipped with a machine learning model to detect user states such as concentration, boredom, surprise, anger, etc. This script runs in the background while the user is engaging with another system and saves relevant data, and generates a CSV file with emotion data, and recordings of the screen and webcam while the script was running.
- A notebook to train the model.
- A python script that can analyze generated data and allows the user to find points of interest in the saved recording.

D. Importance

This project aims to improve UI testing by using AI for emotion detection, making user testing faster and providing developers with clear data to enhance their project's user experience and usability.

II. Background and related work

A. Literature review

- **Reference 1 Lasso and Ridge regression:** Paper [1] is about the two regularization algorithms that we used for optimizing the deep neural network, which was aimed at dealing with over-fitting, however in our case it had limited effect since it started over-fitting 20 epochs later with only a slight increase in accuracy.
- **Reference 2 DDAMFN model trained with AffectNet:** Paper [2] is about the best model by using the dataset from AffectNet, which contains 400000 images manually labeled for eight types of facial expressions, and that is neutral, happy, sad, fear, surprise, disgust and contempt. The paper was included due to being similar to the classes that we trained with 7, it was trained using PyTorch similar to how we trained our Yolov8 models in the same format, and lastly it was a lightweight model so it can be used on small embedded devices with a camera.
- **Reference 3 PAtt-Lite model trained on FER+:** Paper [3] is the best model we found regarding performance for facial expression recognition, which is a lightweight patch and attention network that is based MobileNetV1. The dataset it trained was called facial expression recognition plus (FER+), which 8 label classes similar to the classes we trained on with only 7 labels, and contains the emotions neutral, happiness, surprise, sadness, anger, disgust, fear, and contempt.
- **Reference 4 PAtt-Lite: Lightweight Patch and Attention MobileNet for Challenging Facial Expression Recognition:** Paper is describing the model PAtt-Lite in detail, since we were curious on how it could have such a high performance for detecting facial expressions, and there was a lot we could learn from the techniques used for designing and training this neural network.

B. Existing solutions

There are multiple neural network models built to identify and locate facial expressions, where one of them is called the dual-direction attention mixed feature network, and was specifically designed for facial expression recognition. The model was found in a leaderboard for neural networks (see reference [2]), but the best model found on that site was PAtt-Lite (see reference [3]), which was trained on the FER+ data set, and was lightweight so it could be used on mobile devices without any issues with an accuracy of 95 percent making the best model found so far for facial expression recognition (for further documentation on the model see [4]).

C. Theoretical Framework

1) Emotion Recognition Using AI: **Foundational Theories:** Emotion recognition technology in AI is primarily grounded in psychological theories like Paul Ekman's theory of basic emotions, which categorizes emotions into six basic types - happiness, sadness, fear, disgust, anger, and surprise - each with distinct facial expressions. AI systems, particularly in human-centered design, leverage these theories to interpret and analyze facial expressions, translating them into recognizable emotional states.

Machine Learning Models: For emotion recognition, Convolutional Neural Networks (CNNs) are commonly employed due to their proficiency in image processing and pattern recognition. These models, adept at handling the nuances of facial expressions, are trained on large datasets to identify and categorize emotional states. Studies have shown CNNs' effectiveness in capturing subtle facial cues that convey emotions, making them a preferred choice for this task.

Model Optimization: Optimizing these models involves enhancing their accuracy and reducing biases, a crucial aspect given the diverse nature of human expressions. Techniques like augmenting training datasets, fine-tuning model parameters, and employing transfer learning are applied to improve performance and ensure the models are responsive and accurate in real-time applications.

2) Ethics in User Data Collection and Processing: **GDPR and Privacy Laws:** Our approach adheres to the General Data Protection Regulation (GDPR) and similar privacy laws, ensuring ethical handling of user data. These laws mandate user consent, data minimization, and the right to data erasure, shaping how we collect, store, and use user data, especially sensitive information like facial expressions.

Consent and Anonymity: Informed consent is a cornerstone of our ethical framework, particularly when handling facial expression data. Users are fully informed about what data is collected and how it is used, and their anonymity is strictly maintained to respect their privacy and ensure ethical compliance.

Balancing Data Collection and Privacy: The challenge lies in collecting sufficient data for effective analysis while respecting user privacy. Our methodology reflects a balance, following ethical guidelines and frameworks to ensure data collection is purposeful, minimal, and secure.

3) User-Centered Design Principles: **Incorporating User Feedback:** Our design process integrates iterative and participatory design approaches, where user feedback is continually solicited and incorporated. This approach ensures that the system evolves in response to real user needs and preferences, enhancing usability and satisfaction.

Impact on User Experience: Emotion recognition technology, informed by user feedback, plays a critical role in understanding user experiences. By analyzing emotional responses, we gain insights into user satisfaction and areas of improvement, thereby directly impacting the design and functionality of the system.

4) Data Analysis and Interpretation: **Analytical Methods:** We utilize statistical methods for quantitative data analysis and thematic analysis for qualitative insights. This combination allows us to systematically interpret complex emotional data, ensuring a comprehensive understanding of user interactions.

Interpreting Emotional Data: Our interpretation of emotional responses is guided by theories that link emotions to user behavior. By analyzing these responses in the context of user interactions, we gain valuable insights into the emotional impact of the software, informing future design improvements.

5) Emotional Analysis Framework: To deepen our understanding of emotions in evaluating them through our system, we explored Robert Plutchik's theory of emotions. Plutchik's wheel of emotions, a comprehensive model, delineates the interrelations among various primary and secondary emotions. This model categorizes emotions into eight principal sectors, including anticipation, disgust, anger, joy, surprise, fear, and sadness, offering a structured approach to understanding the complexities of emotional responses in the context of user interaction with software.

III. Study design

A. Requirement engineering

Requirement engineering is a crucial component in the design of our study, pivotal for the successful development and implementation of our emotion detection system. This process involves a series of steps to ensure the software effectively fulfills its intended purpose and meets user needs.

1) Creation of personas:: The initial step is the development of personas. These are fictional, yet realistic, representations of our potential users, encompassing different user types, their behaviors, needs, and likely interactions with our system. In this project, personas include UI engineers, end-users/test participants, project managers, and UX designers.

2) Development of user stories:: Following the creation of personas, user stories are developed. These stories are short, narrative descriptions that articulate software features from the user's viewpoint.
User stories serve as a guiding tool for the development process, ensuring alignment with user expectations and needs.

3) Developing functional and non-functional requirements:: This phase includes specifying the particular requirements of the system. Functional requirements detail what the system is expected to do.
Non-functional requirements, focus on how the system operates.

4) Validation and verification:: The concluding step in requirement engineering is to validate and verify that the documented requirements are feasible and within the project's scope. This involves stakeholder reviews of the requirements, necessary adjustments, and confirmation of their accuracy and clarity in documentation. It also includes testing the system against these requirements to ensure satisfactory fulfillment.

B. Testing methodology

Testing methodology refers to the systematic approach and procedures used to conduct an experiment or test a hypothesis. In the context of software development and user experience design, it encompasses the strategies and methods employed to evaluate the effectiveness and usability of a tool or interface. This involves defining clear objectives, selecting appropriate variables, designing the experiment setup, and determining how data will be collected and analyzed. The choice of testing methodology is crucial, as it directly impacts the reliability and validity of the findings. In our project, we have developed a specific testing methodology to assess the efficacy of our emotion detection and recording tool in identifying user interface issues, compared to traditional manual observation methods. This section outlines the objectives, design, procedures, and ethical considerations of our experimental approach.
Objective of the experiment: The experiment aims to compare the effectiveness of traditional manual observation versus the use of our emotion detection and recording tool in identifying issues within a user interface (UI).

Experiment design:

- **Participants:** The experiment involves two developers and one tester. The tester interacts with a specific UI, while the developers employ different methods to identify potential issues in the UI.
- **Roles and methods:**
  - *Developer 1 (control group):* This developer will engage in manual observation of the tester to identify UI issues. This method represents the traditional approach to user testing.
  - *Developer 2 (experimental group):* This developer will utilize the data generated by our software, along with a recording of the tester's interaction with the UI, to identify issues. This approach represents the use of our tool in the user testing process.
- **Variables:**
  - *Independent variables:*
    * The emotional responses of the tester, specifically focusing on negative emotions and surprise.
    * The data generated by our tool during the tester's interaction with the UI.
  - *Dependent variable:* The effectiveness of issue identification in the UI, comparing the use of our tool versus manual observation.

Procedure: Both developers will observe the same tester interacting with the same UI. However, their methods of observation and analysis will differ as outlined above. The experiment will be conducted in a controlled environment to ensure consistency.

Data collection and analysis: We will record the number and nature of issues identified by each developer. A comparison will be made to evaluate the effectiveness of our tool in identifying UI issues against traditional manual observation methods.

Ethical considerations: Consent will be obtained from all participants. The testers who get recorded will have to consent to having recordings of them made and distributed with this report.

C. Data collection

Our data collection approach combines both qualitative and quantitative methods to ensure a comprehensive understanding of user interactions and emotional responses. This mixed-methods approach aligns with our project's objective to deeply understand user experiences and behaviors.

Quantitative data: Includes timestamps and emotion data extracted from webcam recordings and screen captures. This data provides measurable insights into user interactions and emotional reactions.

Qualitative data: Comprises observational notes and manual reviews of webcam recordings to understand the context of user emotions and behaviors.

By integrating both quantitative and qualitative data, we gain a holistic view of the user experience, allowing us to draw more nuanced conclusions about the software's usability and emotional impact.

List of collected data:

- **Screen Recordings:** Captures user interaction with the software interface. Recorded on a guest PC to avoid capturing personal information.
- **Webcam Recordings:** Focuses on the user's facial expressions for emotion analysis. Used for both automated emotion analysis and manual review.
- **CSV File with Emotion Data:** Logs emotions detected from webcam recordings with corresponding timestamps, correlated with screen recordings for comprehensive analysis.

Data collection methodology:

- **Recording Procedure:** Initiate the recording script on a guest machine during user interaction with the target software.
- **Emotion Analysis:** Each webcam frame is processed through a machine learning model for emotion analysis. The output, along with a timestamp, is saved to a CSV file, and the video frame is saved separately.

Data storage and management:

- **Privacy and Security Protocols:** Strict protocols should be implemented where recordings and data are stored on non-personal, dedicated machines. This measure is crucial for maintaining data privacy and preventing unauthorized access or tampering.
- **Data Handling Post-Test:** The data should be securely transferred to protected storage after testing, employing encryption or password protection to safeguard sensitive information.

Ethical considerations:

- **Relevance and necessity:** We tried to adhere to the principle of data minimization, ensuring only essential data (like video without audio) is collected.
- **Informed consent:** When starting the recording script, the tester should be shown a consent form that they need to digitally agree to before the recording is started.
  Of course, they should be informed of what data will be recorded before they are sat in front of the computer, so this should only be a formality.

Quality control and accuracy:
Emotion detection accuracy in deep learning neural networks is the metric for evaluating how good our prediction is compared to the ground truth data, which is measured by the following formula below.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

The accuracy for the model can be affected by the quality of the data set used for training, validation and testing, where for example you need to have images testing similar to the ones you trained on, so you can't train on images of people with glasses or beards, and then test on faces without those attributes, because you will get poor results from it. The plot

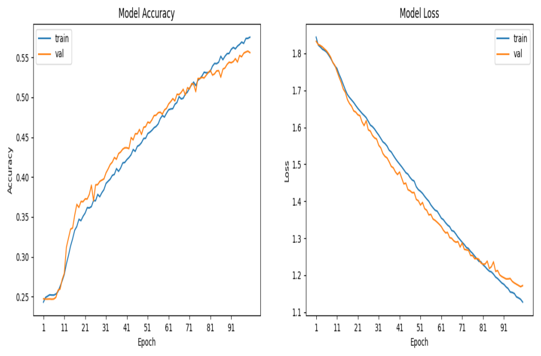


Fig. 1. Training neural network with SGD optimizer using TensorFlow for 100 epochs in Kaggle notebook

given in figure1 is the ideal way to train a neural network, where training and validation accuracy follows each other, and the same should go for the validation and training loss. So if validation loss is not getting lower, but training loss is that means the model is over-fitting, and if the validation loss gets lower but training loss isn't means it is under-fitting. For object detection you also have recall and precision, where recall is how many objects the model found in the image compared to the ground truth, precision is how accurate the placement of the bounding box is around the detected object compared to the ground truth labeled in the training and test set. The formulas for precision and recall can be found below

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

The values from the precision and recall can be used to calculate the F1 score (see equation 4), which is the harmonic mean of precision and recall scores that ranges from 0 to

100 percent, and gives the overall performance of the object detection algorithm.

$$F1 = \frac{TP}{TP + \frac{1}{2} * (FP + FN)} \tag{4}$$

D. *Emotion analysis methodology*

In order to enhance our comprehension of emotions and identify crucial factors to consider while assessing them using our system, we started investigating Robert Plutchik's theory of emotions. The links between different basic and secondary emotions are illustrated in detail by Plutchik's wheel of emotions, a comprehensive model. Created by psychologist Robert Plutchik, this wheel divides feelings into eight main categories: anticipation, disgust, anger, joy, surprise, fear, and sadness. [5]
The wheel shows how these main emotions may merge together to create secondary feelings. For example, pairing joy and trust could lead to love, and combining serenity and interest could lead to optimism. Plutchik's model also illustrates how emotions can change from one to another and their range in intensity. Speaking about Plutchik's wheel in relation to emotion analysis technique might be helpful because it provides an organized approach to comprehending and classifying emotions. It can be used as a framework for dissecting and evaluating emotional states in speech, writing, or any other kind of communication.

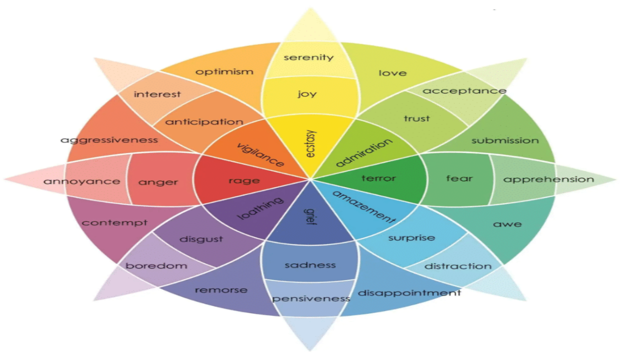


Fig. 2. Plutchik's Wheel of Emotions

IV. Case Description

Our group of developers brainstormed for ideas of a project that uses the embedded camera and does something with the gathered data.
We came up with the idea in this project of using the embedded camera to observe a user's emotions while they are using another piece of software, then we were looking for use cases and we found that it could be used for usability tests.
Usability tests are a user-centered design technique used to evaluate a product by testing it with representative users. Typically, during a usability test, participants will try to complete typical tasks while observers watch, listen, and take notes. The goal is to identify any usability problems, collect qualitative and quantitative data, and determine the participant's satisfaction with the product.
Our idea is that our system could be used to automate parts of the usability test, and remove the need for a observer to be present in person. Instead they would use the data and recordings our system generates to hopefully make the observation as if they were there in person, but while using way less time and effort.
To do this, our system should be able to at least record the screen, and be able to record the emotional state of the user based on their facial expressions. Full requirements, personas and user stories were developed for our system.

A. *Personas*

- **Software/UI Engineer (Primary User)**
  - *Role:* Uses the tool to analyze user interactions and emotional responses.
  - *Needs:* Quick access to insightful data, ease of use, and detailed reports to make informed decisions about UI improvements.
- **End User/Test Participant**
  - *Role:* Interacts with the UI being tested.
  - *Needs:* A seamless and unobtrusive testing experience, asked for consent to be recorded.
- **Project Manager/Product Owner**
  - *Role:* Oversees the UI testing process and uses data to guide project direction.
  - *Needs:* High-level insights into user satisfaction and areas requiring improvement.
- **UX/UI Designer**
  - *Role:* Utilizes the feedback to improve the UI design.
  - *Needs:* Detailed, actionable insights into user behavior and emotional response data and metadata.

B. *User stories*

Based on the developed personas, we came up with the following user stories.

1) *For the Software/UI Engineer:*

- As a Software/UI Engineer, I want to quickly access and visualize user interaction data to efficiently analyze user behaviors and emotions.
- As a Software/UI Engineer, I need to generate detailed reports from user data to make informed decisions about UI improvements.
- As a Software/UI Engineer, I desire an intuitive interface to manage and navigate through emotion analysis data without extensive training.

2) *For the End User/Test Participant:*

- As an End User participating in tests, I want a seamless interaction with the UI without disruptions from the recording software.
- As an End User, I need to be prompted for consent before any recording to ensure my privacy and comfort.

3) For the Project Manager/Product Owner:

- As a Project Manager, I want to oversee the UI testing process with access to high-level insights and analytics to direct product development effectively.
- As a Product Owner, I need a dashboard that highlights user satisfaction metrics and areas requiring improvement based on testing data.

4) For the UX/UI Designer:

- As a UX/UI Designer, I want to utilize user feedback and emotion data to refine and enhance the UI design.
- As a UX/UI Designer, I need detailed and actionable insights into user behavior patterns and emotional responses to optimize the user experience.

C. Functional requirements

- **User Observation:** The software must be able to observe and record user interactions with the UI in real-time.
- **Emotion and Reaction Analysis:** Implement machine learning algorithms capable of analyzing and interpreting user emotions and reactions.
- **Data Recording and Timeline Generation:** The system should record observations and map them onto a timeline for easy reference.
- **Data Interpretation and Insights:** Provide functionalities to interpret the recorded data, offering insights into user behavior, emotional responses, and interaction patterns.
- **User Interface for Engineers:** Develop a user-friendly interface for engineers to access, view, and analyze the recorded data.
- **Reporting and Exporting Data:** Include features for generating reports and exporting data for further analysis.

D. Non-functional requirements

- **Accuracy:** High accuracy in emotion detection and user behavior analysis.
- **Usability:** Intuitive and easy-to-use interfaces for both the testers and the engineers.
- **Performance:** Ensure real-time performance doesn't affect the experience of the software being tested.
- **Scalability:** Capability to handle different scales of UI testing, from small to large user groups.
- **Reliability:** Consistent and dependable operation of the software.
- **Security and Privacy:** Strong data security measures, especially considering the sensitive nature of emotional data, and adherence to privacy laws.

E. Tool functionality

How each part of the tool works, including the screen and webcam recording process and the data analysis part.

1) Recording script: This script runs the data collection part of our system. It will record the screen and the webcam and save the frames to respective video files, and it will also create a CSV file with the output of the model and a timestamp of when in the video that certain emotion was detected. This script will run continuously, and can be safely minimized. The only safe way to stop the script is to select the window it opens and press ”Q”.

2) Data visualization: To create a thorough visualization, a CSV file containing timestamped emotional records is used. The timestamped emotional dataset used in this visualization includes the intensity of six different emotions—happiness, sadness, anger, fear, surprise, and disgust—that were recorded over various time periods. The values in the dataset represent the strength or intensity of each emotion at that specific time, and each row corresponds to a distinct timestamp. The full list of columns is : video_time_readable, elapsed_time, Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral, DetectedString. In order to easily manipulate the CSV data, the Python script arranges it into a dataframe by utilizing the pandas package. To facilitate chronological understanding, the ’timestamp’ column is transformed into a datetime format.
With the help of the matplotlib package, the script creates a line graph that shows how emotions change over time. Plotting each emotion (happiness, sadness, anger, fear, surprise, and disgust) against the chronology allows the intensity differences at various intervals to be seen. The resulting visualization can be made more comprehensible by adding customization's such axes labels, a title that describes the image, a legend that differentiates between emotions, and grid lines for clarity. A clear narrative of emotional dynamics is provided by the resulting graphical representation, which makes it possible to quickly identify patterns, trends, and significant changes in emotional states throughout the course of the recording time. The Python script's visualization does a good job at capturing and conveying the subtle changes in emotional states over time. This visualization method has great potential in behavioral analysis. It makes it easier to comprehend human reactions by clarifying temporal emotional dynamics, which in turn makes it possible to create more improved user interfaces. Accepting these data-driven insights may open the door to more compassionate human-computer interactions and improved emotional intelligence applications.

3) Object detection of facial expressions using machine learning: There were 7 defined classes of emotions that were labeled regarding facial expressions, which contained anger, contempt, disgust, fear, happy, sadness and surprise. From the labels initially we had tried 4 different machine learning models regarding classification and object detection of facial expressions, where in the first case we used a classification model in TensorFlow with a simple convoluted neural network architecture using deep learning with OpenCV library and the given dataset from the GitHub repository. The model was trained using a Jupyter notebook using Kaggle, because it is an external GPU that could be used for training the model faster than on a personal laptop, but it depends on your computer hardware specifications. It was detected that there was an issue regarding overfitting after gaining around 60 percent in accuracy, so the next model the architecture, parameters and optimizer was changed along with adding L1 regularization to reduce overfitting and train on more epochs to attain higher accuracy. The model could not achieve more than 66 percent at best with the changes, so it was tried to train on a different dataset to see if it would make it perform better and it seemed only to be slightly better than the initial model at recognizing facial expressions. A face API was found afterwards to be used for detecting people's emotional state, since it could perform better than the earlier models we trained, but there were a lot of problems with integrating it with the rest of the system. It was decided to find a framework or an API that was open source, if it could help make better predictions on facial expressions, where it was initially tried with Yolov7 that used PyTorch since it was one of the best neural network architectures regarding object detection, but there was an issue during training since it could not train due to caching issue regarding the dataset. The dataset was found in Roboflow, where initially it only had 5 classes instead of the 7 we had originally, so another dataset was found that had the same number of classes that we could use for training the model. Due to problems using Yolov7 it was decided to try Yolov8 instead and it worked, since it could train the model using the imported dataset from Roboflow. The new model turned out to have an accuracy above 90 percent, which was a huge increase from the first model we used on this project. Since we moved from hd5f files to PyTorch files, we had to change the code for testing the detection of facial expressions and then save it to a CSV file, so the data could be analyzed for research purposes.

F. User interface

According to the requirements, we kept the user interface simple and clear.

1) Recording script: For the recording part of the system, we have a digital consent form that is shown to the tester (Fig:3) before any recording is done.
In this UI, the tester is able to review what they consent to, and can only start the recording if they checked the "I consent" checkbox, otherwise they can quit the program.
When the "Start recording" button is clicked, the actual recording script is ran, which show live video from the webcam to indicate that it is running. At this point, the tester should minimize the live camera view, and go on with their test. Script doesn't have any other visible UI except for the default windows actions on the top right of the corner, the idea is that once the user is done with their test, the recording can be stopped by an assistant that presses a hidden key while the webcam view is up. In our case this key is "Q". The window cannot be closed by pressing the "x" close button.

2) Analysis script: The user interface for the analysis script is currently very simple. When the analysis script is ran, the default operating system file selection dialogue box opens at the file path that should contain the generated CSV files.
Opening one of these CSV files will make the rest of the analysis script run, and the UI will show a line graph with the probabilities of each emotion over the time of the recording.(Fig:7)
In this interface, the user can zoom in the graph, and some interactions are possible such as zooming into the graph and panning the graph. The graph can also be exported as a png.

V. Results

The full source code can be found at https://github.com/HorvathStefanDaniel/HumanCenteredSoftwareDesign.

A. Produced scripts

This project consisted of a lot of experimentation and exploratory development. As such, a lot of scripts and machine learning models have been produced and are present in the source code to aid future development.
A video of the system starting, the generated data, and the data visualization is in the appendix.

1) Project file structure: A screenshot of the project files can be seen on fig:3

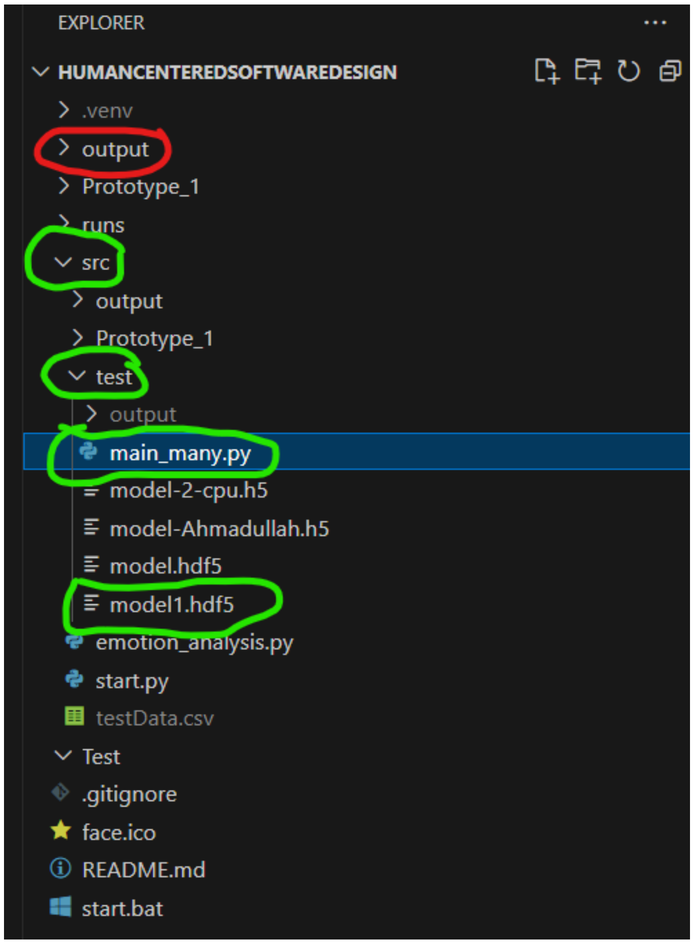
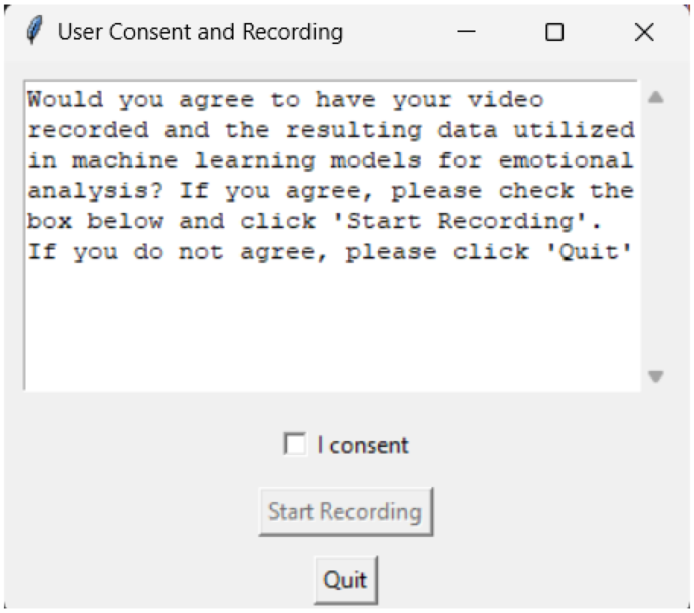


Fig. 3. Main files used.

The "main" script is the main_many.py script at src/test/main_many.py which will record the user and the screen, and generate a CSV file at output/CSV.It saves the video files at /output, with the format "screen_#date" for the screen recording, and "webcam_#date" for the webcam recording, where #date is replaced with the timestamp of

the recording with the format year-month-day-hour-minute-second.
This script uses the model1.hdfs model at src/test/model1.hdf5 to detect the user's emotions, the output of this model is what gets saved in the CSV files called ”emotions_data_#date.
The ”proper” way to run this script is to run the start.bat file (only works on windows) which acts as an executable that launches the start.py script, then the main_many.py script using .venv. This file is located at the root of the project. Full source code is provided to prove that these scripts are not malicious.
Data visualization is done by the emotion_analysis_MANY_UI.py script, located at output/CSV/emotion_analysis_MANY_UI.py.
To easily run these python scripts, two runners were created, which are simple .bat files that start the .venv environment, then run the appropriate script. Shortcuts were provided for easier accessability, and instruction on how the run the scripts are also in the readme file.

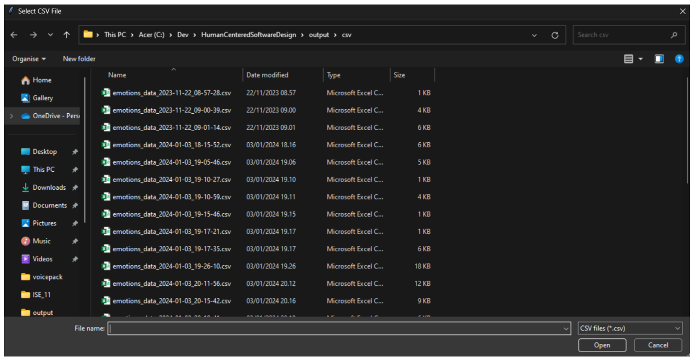
2) *Language and libraries used:* To implement the scripts, python was used due to the relative ease of use and ample documentation for working with machine learning models.
For recording the screen, pyautogui was used to take screenshots of the screen, which get added as frames in the generated video file.
For the prototype user interface which asks for consent before the recording script is started, tkinter was used. It allowed us to quickly make an acceptable user interface, with all the functionality we needed.



```
Would you agree to have your video
recorded and the resulting data utilized
in machine learning models for emotional
analysis? If you agree, please check the
box below and click 'Start Recording'.
If you do not agree, please click 'Quit'
```

Fig. 4. Main files used.

For data visualization, we used matplotlib and pandas to create and display an interactive graph. Tkinter was also used for the functionality of being able to open a .CSV file using the operating system default file opening functionality.



Name	Date modified	Type	Size
emotions_data_2023-11-22_08-57-28.csv	22/11/2023 08.57	Microsoft Excel C...	1 KB
emotions_data_2023-11-22_09-00-39.csv	22/11/2023 09.00	Microsoft Excel C...	4 KB
emotions_data_2023-11-22_09-01-14.csv	22/11/2023 09.01	Microsoft Excel C...	6 KB
emotions_data_2024-01-03_18-15-52.csv	03/01/2024 18.16	Microsoft Excel C...	6 KB
emotions_data_2024-01-03_19-05-46.csv	03/01/2024 19.06	Microsoft Excel C...	5 KB
emotions_data_2024-01-03_19-10-27.csv	03/01/2024 19.10	Microsoft Excel C...	1 KB
emotions_data_2024-01-03_19-10-59.csv	03/01/2024 19.11	Microsoft Excel C...	4 KB
emotions_data_2024-01-03_19-15-46.csv	03/01/2024 19.15	Microsoft Excel C...	1 KB
emotions_data_2024-01-03_19-17-21.csv	03/01/2024 19.17	Microsoft Excel C...	1 KB
emotions_data_2024-01-03_19-17-35.csv	03/01/2024 19.17	Microsoft Excel C...	6 KB
emotions_data_2024-01-03_19-26-10.csv	03/01/2024 19.26	Microsoft Excel C...	18 KB
emotions_data_2024-01-03_20-11-56.csv	03/01/2024 20.12	Microsoft Excel C...	12 KB
emotions_data_2024-01-03_20-15-42.csv	03/01/2024 20.16	Microsoft Excel C...	9 KB

Fig. 5. CSV file selection.

3) *Machine learning implementation details:* The technologies used for implementation were a Jupyter notebook for faster training times for the neural network by using an external GPU from Kaggle, Yolov8 is a framework for object detection from Ultralytics, and the smallest weight were used since that was the default configuration. Tensorflow library in Python was used for training a CNN algorithm for facial expression recognition. Roboflow was used to gather datasets for training the CNN, since it was a open source library that people can contribute to or use to train machine learning algorithms.
The kaggle source code for the yolov8 model can be found at https://www.kaggle.com/code/hakimsayad/yolov8-emotion-detection and the source code for training model1.hdf5 can be found at https://www.kaggle.com/code/hakimsayad/face-emotion-recognition.

B. Satisfying requirements

1) *Functional requirements:*

- **User observation:** The software observes the user through the webcam.
- **Emotion and Reaction Analysis:** The user's emotions are being recognized by the machine learning model used in the recording script.
- **Data recording and timeline generation:** The recognized emotions get recorded in a CSV file with timestamps of when after the start of the recording each emotion was detected. This timestamp defines the timeline.
- **Data interpretation and insights:** The analysis script generates points of interest in the plot where a shift in negative emotions or surprise is detected.
- **User interface for engineers:** The analysis of the timeline is provided in a standard pandas GUI which the engineer or other users can use to review the data.
- **Reporting and exporting data:** The analysis script is used to generate readable reports. It has functionality for exporting the generated graph built in.
  The recording script also generates standard CSV files, which can be exported and used in other analysis software if needed.

2) Non-functional requirements:

- **Accuracy:** High accuracy in the emotion recognition has been achieved by training our own machine learning models, and providing the testing results.
- **Usability:** The user interfaces of all system parts are very simple. All the scripts can be ran by executables which don't require any programming knowledge on the part of the users.
- **Performance:** The system is capable of running on laptops without any GPU to handle the machine learning model running in the background. More testing is necessary to determine the exact limits of what machines can run the recording script correctly.
  Since the scripts are written using python they should be able to run on any operating system, but currently it is only tested to work on Windows machines.
- **Scalability:** In the current form, the system is capable of recording and discerning the emotions of one user.
- **Reliability:** Emotion recognition is fairly reliable, testing has been done with static images and the system consistently detects the same emotion for the same image.
  In the current state of the recording script, it does not produce reliable recordings of the screen and webcam.
- **Security and Privacy:** In the current state of the system, produced files are stored in the root structure of the project, with no encryption of password protection.
  For development, the GitHub projects gitignore contains .avi and .CSV files, so no recordings are distributed online by accident.

C. Testing

Simply simulating a range of distinct feelings that were intended to be recorded allowed the emotion analyzer to be tested on its own. This was completed before the system as a whole was put into use. We tested the various models and API's that we had access to during this process. Selecting the most preferred model would be based on whether one produced the most accurate results in identifying the correct emotions.
A way we tested emotions was to search for images of "man smiling", or "picture of woman with neutral expression", and then show them in front of the camera, where the software would pick them up and analyze them. As seen in fig.6, showing a single image to the webcam will consistently result in a single emotions being dominant.
There is some deviation, but this most likely has to do with the imperfect way of holding a picture to the webcam, instead of digitally setting the picture as the input of the script. Because of manually holding up a picture to the webcam, some tilt and movement is added to the picture, which might make the neutral face appear to be showing a different emotion.

D. Known issues

In the current implementation, the timestamps from the generated CSV file and the generated video files don't match. This is because a frame doesn't actually get added to the video file in real time.

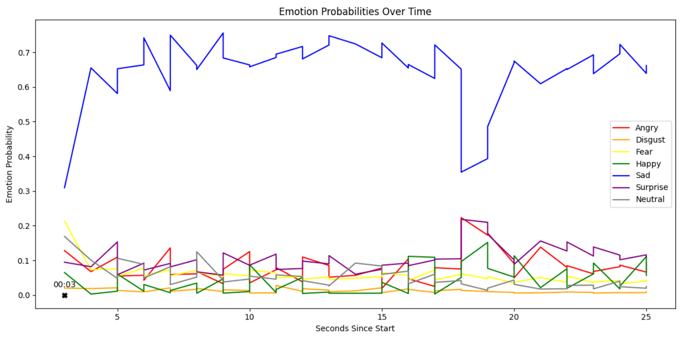


Fig. 6. Analysis on neutral face being shown.

This could be caused by the test environment not having enough processing power to save all the webcam and desktop recording frames, and analyze them at the same time, but it's most likely caused by the current implementation. Current recording software is much more complex than our script, and a way to buffer frames would need to be implemented, and minimum computer specifications need to be investigated to ensure smooth performance in real case scenarios.
In the case of our experiment, the CSV file time goes up to 04:31,271, but the videos generated are only 1 minute and 27 seconds long.
Another issue is with the models being used to detect emotions. The ones used in the experiment, namely model1.hdf5 seems to have a bias towards the "Sadness" emotion. A baseline neutral face was used with several subjects, and in every case sadness was the dominant emotion, instead of neutral. This can be observed in fig.6, where a picture of a person with a neutral facial expression was shown.

E. Experiment

1) Running the experiment: Our experiment was done on a "work in progress" simple game developed using python. The tester was given a set of actions they need to complete in the game.
The tester and their emotions were recorded using this system, while also being observed by one of the developers of the game which was taking notes on the tester. A different developer has been tasked with taking notes based on the data recorded and generated by our system.
The instructions, the notes taken by the in-person observer and the recordings and generated CSV file are available in the appendix.

2) Results: Using the recording and the generated graphs have allowed the second developer to find the same issues that the first developer found.
Below is a zoomed in section of the generated report, in which timestamps for emotions shifts to negative emotions can be seen. Due to the issue with the length of the video mentioned earlier, the videos generated for the experiment were manually edited to match the timestamps in the CSV file by slowing down the video. The amount of slow-down is not set, and

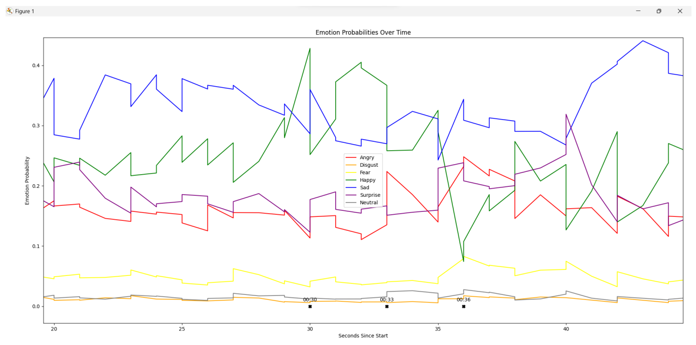


Fig. 7. Overview of data analysis in prototype.

can vary depending on the system, but in our case we set the play speed to around 33% and exported a video that matches the timestamps, and shows both the screen recording and the webcam recording in the bottom right corner.
This helped developer 2 of the experimental group to utilize the data generated by or software and complete the experiment.

VI. Discussion

A. General thoughts

Throughout the course of this project, our team has significantly deepened our understanding of human-computer interaction. This experience has been enlightening, providing us with valuable insights into the nuances of software design that leverages human data.

B. Machine model discussion

Using the first model based on a repository on GitHub worked without adding any modifications to the code, but the problem was the accuracy and bounding boxes of the detected facial expressions, which was it did not identify the correct facial expression as often as it was wished. The model built with the TensorFlow initially had good training, but overtime it would start to over fit after reaching around 60 percent in accuracy, which was far from optimal, and was the reason we tried to modify the parameters for the deep learning neural network, and we also tried to switch the optimizer algorithm from Adam to stochastic gradient descent (SGD) since in theory the algorithm would be better at generalizing the data-set.
The change to SGD gave better generalizations, when it was tested on real faces outside of the training and test data, but the difference between training and validation accuracy became larger over time after training for 100 epochs, and meant it was over fitting due to bias.
Regularization was tried to reduce or eliminate the over-fitting during training, where first Lasso regression (L1) was used since it could be unwanted features that could be the cause of the bias, and the algorithm could help eliminate the weakest features. It turned out to not be the case since Lasso regression increased the over-fitting problem instead of decreasing it, so we tried Ridge regression, where we would minimize the influence of the weakest features instead of fully eliminating them, and it turned out to help by increasing the accuracy to be above 60 percent to around 65-66 percent, but adding more epochs to 200 it started to over-fit again so there was limits to how much further it could be improved without changing the architecture of the neural network. So it was thought to instead of reinventing the wheel we looking into the best algorithms for object detection, where we found Yolov7 and Yolov8 to be the best around, but the decision to use Yolov8 instead of Yolov7 was due to not only problems with getting the model trained due to dataset caching issues from Roboflow, it was also because Yolov8 had faster inference with their smallest weight, had documentation of better performance compared to Yolov7 and their model were lighter in terms of memory, so it could be used on smaller devices with an embedded camera without worrying about memory usage. For the experiments we ended up using the second model from the Jupyter notebook in Kaggle, where we changed the optimizer algorithm from Adam to SGD. The reason the Yolov8 model wasn't used for the experiments was because it was easier to integrate it.

C. Experiment

The comparative analysis experiment yielded positive results, demonstrating that the remote developer could effectively draw the same conclusions as the one who was physically present.
However, it's important to highlight that during the usability test, the developer was available to provide guidance to the tester when challenges arose. Should our system be solely employed for future usability tests, it may be necessary to offer testers more comprehensive instructions to assist them in navigating any difficulties they encounter.
Something we could have improved on for our test, would be to temporarily remove ”fear” as a negative emotion, since our model is constantly detecting it, and it creates many false negatives in the generated report.

D. Decisions for the prototype

At the start of development we used an existing machine learning model [6] to recognize human emotions. We weren't happy with the accuracy of this so we trained our own model using TensorFlow, which was better at generalizing and reduced over fitting.
The model used in the experiment (From script main_many.py) was model1.hdf5 which still has some issues like the one mentioned before, where neutral expressions are detected as sadness. We did more research and trained a Yolov8 model with better performance, but we had issues integrating it into the existing script, so a new script, ”main_simple.py” was made which uses this new model, but it's still not fully functional. The performance for Yolov8 can be seen in figure 10. A weakness of the yolov8 model is that it needs larger input images, which can affect performance.

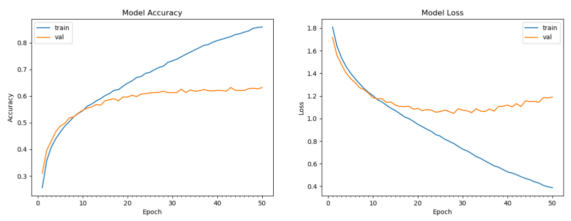


Fig. 8. First model (From saranshbht [6]) training accuracy and loss.

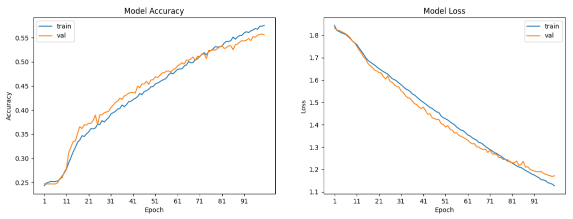


Fig. 9. First custom model (Model1.hdf5) accuracy and loss.

We also considered using face-api.js, but for the purpose of the prototype, we found it easier to use python, since we had to generate .csv files and do data analysis.

VII. Conclusion

This project developed a system for user interface (UI) testing that uses emotion detection technology to analyze facial expressions, providing real-time feedback on user emotions. By employing Python scripts and machine learning algorithms, notably Convolutional Neural Networks (CNNs), the system effectively captures and analyzes emotional responses, offering detailed insights for UI improvements.
Key achievements include adhering to ethical data collection standards, particularly GDPR compliance, and incorporating informed consent and data privacy practices. The system's potential extends beyond UI testing, with applications in market research, psychological studies, and enhancing AI's emotional intelligence.
The tool's full development could greatly impact UI testing efficiency and user experience design, offering time-saving automation and deeper user interaction insights. This project lays the foundation for advancing user-centered design and human-computer interaction, though further optimization and model refinement are necessary for broader applications.

VIII. Future works

The data collection for the experiments could be improved by using either the Yolov8 algorithm that were trained in Jupyter notebook in Kaggle, which could give us higher quality data if there wasn't problems regarding integrating the model to store different emotions over the timespan of the video recording. The other option could be to look deeper into the PAtt-Lite model by checking out the documentation and the repository to figure out how it works, because it can give very high quality data for facial expressions compared to

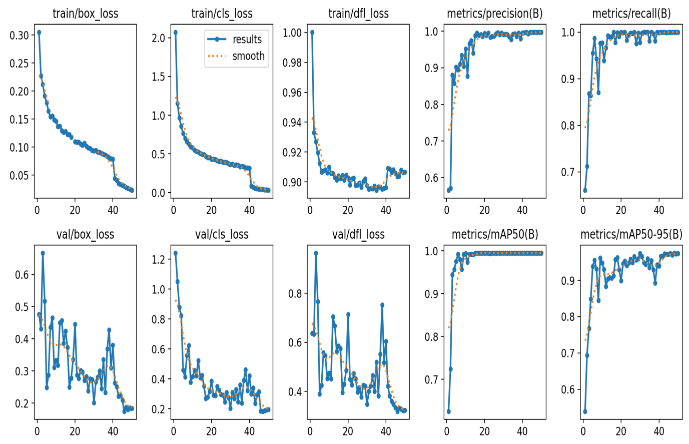


Fig. 10. Model trained using Yolov8 smallest weight with 416X416 image size

other algorithms that were found.
For optimizing the workflow efficiency in the project, it would help save time, increase productivity and help with finding bugs earlier by adding continuous integration and deployment pipeline for the project, because it will help the team identify bugs faster when they appear since the system will build, test and deploy the changes committed to the main branch, and if any errors appear during testing the person responsible would be able to make fixes needed to deploy the changes to production.
More development time could be used to fix the before mentioned bugs, and more testing should be done in regards to performance, to be able to establish minimum PC requirements for being able to run the recording script.
More features could be added to this project, we already have a working prototype of the emotion recognition recording and analysis, but more data and specifically human data could be recorded through the embedded camera. As a short list of things that could be added to help with the current use case, we should add

- Mouse pointer visibility in recordings. This is an obvious improvement, and can almost be considered a bug in our current prototype.
- Eye tracking functionality. This is another feature that could be implement though just code, where a different AI model can analyze the webcam footage to try and determine what on the screen exactly the user is looking at.
  However, adding more models to the recording script will impact performance, and it should be investigated if it would be possible to still do this analysis in real time.
- Multiple user tracking is another obvious enhancement to the current system.
  The implementation of multi-user tracking could significantly enhance the scope and utility of the project. This feature would enable the analysis of emotional responses from an entire audience during events like presentations

or movie previews. The primary objective of this enhancement would be to garner deeper understanding and more nuanced insights into the collective opinion of the audience regarding the content being presented.

- It would be a nice feature for the script to be able to produce a baseline at the beginning of a test, so that the bias is reduced. At the very least having a baseline for neutral expressions would be great, but having a more complex setup where the user must do various expressions can also be considered.

References

[1] Regularization: https://builtin.com/data-science/l2-regularization

[2] AffectNet models benchmark leaderboard: https://paperswithcode.com/sota/facial-expression-recognition-on-affectnet

[3] FER models benchmark leaderboard: https://paperswithcode.com/sota/facial-expression-recognition-on-raf-db

[4] Patt-Lite scientific research paper: https://arxiv.org/pdf/2306.09626.pdf

[5] The Emotion Wheel: Purpose, Definition, and Uses: https://www.berkeleywellbeing.com/emotion-wheel.html

[6] Model used for first iteration: https://github.com/saranshbht/Emotion-detection