Carreon et al. *Research and Practice in Technology Enhanced Learning* (2027) 22:17
https://doi.org/10.58459/rptel.2027.22017

Research and Practice in Technology Enhanced Learning

RESEARCH Free and Open Access

Using face, gaze, and code features detected with off-the-shelf equipment to derive student emotion in an introduction to programming context

Mario Carreon[12]*, Ko Watanabe[3], Tomokazu Matsui[1], Yuki Matsuda[4], Shoya Ishimaru[5], Hirohiko Suwa[1], and Keiichi Yasumoto[1]

*Correspondence:
mtcarreon@up.edu.ph
Department of Computer Science, University of the Philippines Diliman, Quezon City, Philippines
Full list of author information is available at the end of the article

Abstract

Students in an online Introduction to Programming (CS 1) class suffer a severe disadvantage. In a face-to-face learning environment, teachers can see if students struggle with the assigned programming task through their body language or overall demeanor. In online learning, all that the teachers see is a small image of their student, if at all, and are thus unable to provide the needed intervention to help the student overcome their stumbling block. Our research investigates extrapolating student emotion through a web camera and code logs gathered while students work on programming exercises, with a hardware-based gaze tracker to serve as ground truth for software-based gaze tracking. Code, face, and gaze data, together with annotated emotion data gathered from our experiments, were used to train learning models via XGBoost. Our best model can predict a student's emotional state at a precision, recall, and F1-score of 72.7%, 77.3%, and 74.9%, respectively. We achieved these results from analyzing 8 hours and 20 minutes of experimental data from 26 participants using only software-based gaze tracking, with statistically similar results to hardware-based gaze tracking to within a ±8% equivalence margin.

Keywords: introduction to programming, emotion detection, machine learning

Introduction

A typical Introduction to Programming (CS 1) class involves students working on programming exercises in a computer laboratory. A teacher or teaching assistant is available to answer students' questions about their exercises.

A teacher may intervene even if the student does not ask for help. A struggling student can be detected through the student's body language or overall demeanor, and an experienced teacher watches out for these tell-tale signs of student difficulty.

In an online class setting, the teacher only gets a minimal view of the student, if the view is even available at all. Even in face-to-face settings, a teacher is unable to watch all the students for signs of struggle with the lesson, especially in large classrooms. As such, opportunities for beneficial teacher intervention can be missed. A system that can automatically detect student emotion would greatly assist a teacher to provide the needed student assistance both in an online and face-to-face paradigm.

This research aims to develop an automated mechanism that can effectively detect signs of learning difficulties in the context of a CS 1 class. While a substantial body of work in the literature attacks multiple aspects of this problem (Luxton-Reilly et al., 2018), our study focuses on detecting student emotions to facilitate beneficial teacher intervention.

Specialized equipment like electrodermal activity sensors (Gorson et al., 2022) or electroencephalograms (Mangaroska et al., 2022) can accurately measure learner emotion through biometrics. However, the cost and expertise required to operate such equipment often pose barriers. Even affordable, straightforward equipment like wristbands (Girardi et al., 2020) can be challenging for some low-income students or school districts. While the school can purchase equipment, the students may hesitate to use it due to the fear of breaking it (Sipitakiat & Blikstein, 2010).

This research takes a practical approach by utilizing sensing equipment readily and cheaply available to a typical student: a web camera. We gather three pieces of information from a student working on their programming problem. Facial expressions, gaze traces, and code logging.

Intuitively, one can detect emotions from facial expressions. On gaze traces, a student staring outside of the screen for long stretches of time can be considered as expressing boredom. Lastly, coding coming in rapid bursts may indicate that the student is engaged with solving the programming problem. Running these data sets through a learning algorithm may find correlations that can link face, gaze, and code features to emotions.

In our experiments, our participants used a custom Integrated Development Environment (IDE) that integrates gaze tracking and code logging. We use a separate module to detect facial expressions, but we are in the process of integrating this into our IDE.

These three information streams are then run through a model trained from the experiment participant data, with the goal of extrapolating the student's current emotion. The goal of emotion prediction is to determine if the student is consistently showcasing negative emotions, which can prompt the teacher to provide the needed intervention.

Our core research questions are as follows:

1. To what degree does using only off-the-shelf sensing equipment detect student programmer emotions? (**RQ1**)
2. How does the code, face, and gaze dataset contribute to the overall performance of the emotion detection model when multimodally combined? (**RQ2**)

Related Literature

CS 1 and emotions

Computer Science is a challenging course (Robins, 2019). Worldwide, around one-third of all CS 1 students fail their first take (Bennedsen & Caspersen, 2007, 2019; Watson, 2014). This may be due to the innate difficulty of Computer Science and Engineering courses as they require critical thinking to solve technical problems (Robins et al., 2003; Jonassen, et al., 2006). While this issue has had substantial focus over many years (Luxton-Reilly et al., 2018), our approach focuses on the effect of emotions in CS 1 (Nolan & Bergin, 2016; Batra & Atiq, 2025).

Emotions play a key role in student learning. Positive emotions while learning can improve reasoning and memory, while negative emotions make it difficult to remember important information (Frasson & Chalfoun, 2010; Pekrun, 2006). From a more formal standpoint provided by Control-Value Theory (Pekrun, 2006), emotions may be linked to the perceived control and the subjective value that a student finds in an academic task. A student that feels a high control over a highly valued task can feel emotions such as enjoyment or pride. However, a student that is doing the same highly valued task but feels a lack of control or understanding of the said task may instead experience anxiety or frustration. As CS 1 courses require problem solving, continuous failure in creating solutions may lead to boredom, anxiety, and frustration (Luxton-Reilly et al., 2018), and, without receiving adequate intervention, may end up giving up on the task (D'Mello et al., 2008), or dropping out of the course entirely (Bosch et al., 2013).

Bosch et al. (2013) directly ask the question, "What emotions do novices experience during their first computer programming session?" They measured the presence of 13 emotions grouped into two broad categories: basic emotions (anger, disgust, fear, sadness, surprise, happiness) and learning-centered emotions (anxiety, boredom, frustration, engagement, curiosity, confusion), plus a neutral state, with these emotions as defined in Pekrun's taxonomy (Pekrun & Stephens, 2012). Their work shows that 86% of all expressed emotions are confusion, frustration, boredom, neutral, and engagement. As such, our work solely focuses on these emotions.

To add more weight to this design choice, we follow the recommendations by Lapierre et al. (2024). In their work, they used electrodermal activity, automated facial emotion recognition (using FaceReader), and pupil diameter in a multiple regression analysis to

determine relationships between cognitive and emotional states and learning performance in beginner and novice programmers. They chose to use Ekman's taxonomy of emotions, which consists of specific emotions of anger, disgust, fear, happiness, sadness, and surprise (Ekman et al., 1999). The researchers note that these particular emotions lacked pedagogical specificity and recommended that future work use more learning centered emotion models such as Pekrun's.

While this research primarily focuses on facial expression, gaze tracking, and code analysis to detect emotions, one can refer to Harley (2016) for a good summary of additional techniques that can be used to detect emotions in the programming context. For example, body posture can be measured by sensors such as Kinect depth cameras.

All in all, the goal of emotion detection in the programming environment is to create a system that can actively respond to the negative emotions of confusion, frustration, and boredom, in order to facilitate the learning process (Craig et al., 2008). However, our research adds the additional constraint of using only off-the-shelf equipment for detection.

Improving established methods

The following work in the same and adjacent fields to our domain space provided inspiration to our research. Craig et al. (2008) had students emote aloud while working in an interactive learning environment, while two experts recorded their Facial Action Units (AU). Their work strongly linked particular AU's with emotions of confusion, boredom, and frustration via statistical analysis. Their work provides a statistical precedent of detecting emotions via AUs. Sharma et al. (2021) used OpenFace (Baltrušaitis et al., 2016) to automatically extract the AUs but also used statistical tools to associate these with the emotions of boredom, frustration, confusion, and delight.

Jacob et al. (2018) focused on interest detection in newspaper articles via gaze tracking. Their experiment setup and subsequent analysis provided guidelines on how to conduct our own data gathering, as described later in the methodology section.

In the programming domain, Jbara & Feitelson (2017) extensively used eye-tracking to see how programmers read programs and determine gaze patterns where programmers struggle. Another example of gaze tracking in the programming domain is the work of Villamor & Rodrigo (2018), who used gaze tracking to determine the quality of a pair programming collaboration. They assert that a pair looking at the same code at the same time may be an indicator of quality interaction and better comprehension. These and similar literature inspired us to explore gaze tracking in detecting programmer emotion.

In another work, Rodrigo & Baker (2009) used code compilation patterns to detect student frustration, examining indicators such as the number of pairs of consecutive compilations with the same edit location, pairs of consecutive compilations with the same error, the average time between compilations and the total number of errors.

A solid way forward would be the evaluation of multiple data streams in multimodal analysis. Using multiple data sources allows for the examination of events from multiple perspectives and thus gain a "multilayered understanding of a phenomenon" (Villanueva Alarcón et al., 2023). As mentioned earlier, Lapierre et al. (2024) multimodally combined electrodermal activity, automated facial emotion recognition (using FaceReader), and pupil diameter for their work. In another example, the comprehensive work by Mangaroska et al. (2020), while not directly measuring emotion, combined gaze tracking, student data, IDE events, biometrics (heart rate, blood volume, pulse, skin temperature), and facial expressions to predict participant debugging performance.

We build upon these previous works by combining these disparate strategies into a single unified whole. Our work combines processing face information (Craig et al., 2008; Sharma et al., 2021), gaze information (Jacob et al., 2018; Villamor & Rodrigo, 2018), and code information (Rodrigo & Baker, 2009) within a multimodal analysis (Mangaroska et al., 2020; Lapierre et al., 2024) to detect emotions in the specific domain of CS 1 without the need for specialized equipment.

To our knowledge, this combination of approaches is closely approximated only by the work of Tiam-Lee & Sumi (2018, 2019). The authors predicted student frustration, confusion, boredom, and engagement from facial features, code compilations, and keypresses on their IDE. Using various machine learning algorithms, they achieved 82%, 72%, 95%, and 76% accuracy for frustration, confusion, boredom, and engagement detection.

With respect to the machine learning methodology, our work is differentiated from theirs by considering gaze data and specifically using XGBoost as our learning algorithm. Unfortunately, a direct comparison between the accuracy achieved by their models and our research is not possible as we have no access to the actual emotional distributions of their dataset or to the precision, recall, and F1-scores of their model predictions.

In addition, instead of creating four separate learning models for predicting frustration, confusion, boredom, and engagement, we made a single model that predicts negative (frustration, confusion, boredom) and positive (neutral, engagement) emotions. Having one model instead of four reduces the complexity of the final system, decreases the resources needed to make predictions, and provides more straightforward guidance to the CS 1 teacher (e.g., The student is feeling negative emotions at this time). We also made a 5-emotion model that directly predicts the emotion (e.g., The student is feeling confused).

Lastly, Tiam-Lee & Sumi's IDE was purpose built to solve the specific problems covered by their experiment. In contrast, our IDE extends Online Python Tutor (Guo, 2013). As such, our IDE can be used to solve any general simple problem-solving exercise, and provides learning avenues to beginner students by showing the values of their variables as they run their code line by line. Our IDE can be readily incorporated into a typical CS 1

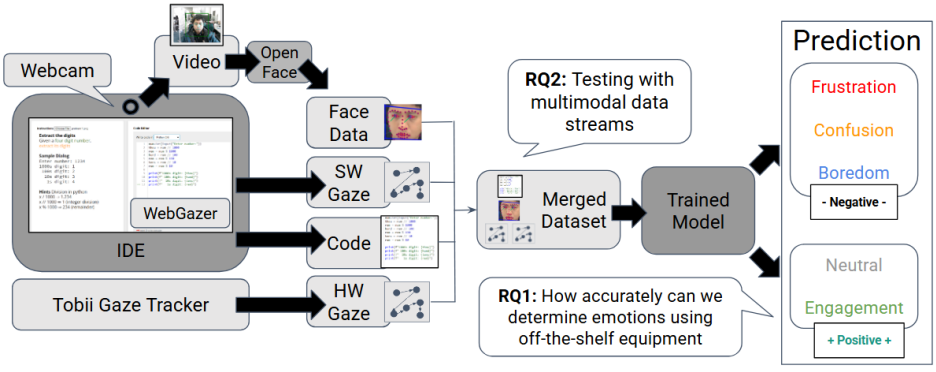
laboratory class with the eventual goal of not only providing mechanisms for teacher intervention, but also collaboration between students.

To summarize, our work is differentiated from other work in general through the specific prediction of programmer emotion using only off-the-shelf equipment (face, code, and gaze data) bundled in a general-purpose IDE that can be used directly in a CS 1 context.

Methodology

System overview

Figure 1
System modules and data flow



Our research can be described as a system that provides automated feedback in an online learning environment (as discussed in Cavalcanti et al. (2021)). Following the paper's major classifications, our work primarily consists of a dashboard, sensors, and machine learning modules working in combination to provide feedback in the form of a prediction of student emotion. Figure 1 provides an overview of the modules of our system and the data flow from input (student video and code) to output (emotion prediction).

As a dashboard, our IDE extends Online Python Tutor, a web-based program visualizer developed by Philip Guo and accessible at https://pythontutor.com/ (Guo, 2013). This online code editor allows users to program their Python, Java, C, C++, JavaScript, and Ruby applications while allowing them to visualize their variables. As of 2021, over 10 million people have used Online Python Tutor (Guo, 2021). In addition, 55 research publications have used or extended Online Python Tutor as of 2021.

Regarding sensors, our IDE extends the functionality of Online Python Tutor by providing gaze tracking through the WebGazer.js eye-tracking software library. (Papoutsaki et al., 2016). Developed entirely in JavaScript, it can determine the location where the user gazes on any web page where the WebGazer libraries are imported. WebGazer does not require specialized equipment, allowing for its use on any computer if

a web camera is present. Once integrated, the location of where the user is looking can be quickly accessed within any additional scripts integrated into the webpage in real time.

While not directly integrated with our IDE, our research uses the logger code developed by Jacob et al. (2018) to gather gaze traces through Tobii Pro Nano, an external hardware-based gaze-tracking device. Hardware-based gaze-tracking provides a form of ground truth to WebGazer. In the main WebGazer paper itself (Papoutsaki et al., 2016), the developers noted a mean error rate of 169 pixels comparing WebGazer to Tobii EyeX, an earlier version of the Tobii Pro. We discuss our own internal testing with our newer hardware in the Results section.

All in all, while software-based gaze-tracking through the web camera can generally achieve a generally acceptable level of accuracy, some data fidelity is lost to noise and the low sampling rate as compared to dedicated gaze-tracking hardware (Madsen et al., 2021; Steffan et al., 2024). As such, a core tenet of our research is to determine if the loss of accuracy in emotion detection when using software-based gaze tracking is worth the potential to deploy gaze tracking at scale.

Our experiment setup uses OpenFace to provide facial landmark detection (Baltrušaitis et al., 2016). This work primarily used the Facial Action Unit detection module (Baltrušaitis et al., 2013; Zadeh et al., 2017) to extract the presence and strength of facial action units. Examples of facial action units are the Inner Brow Raiser (AU1), Nose Wrinkler (AU9), and Lips Stretched (AU20). OpenFace is extensively used in the literature with over 3000 citations in extracting facial information.

Figure 2

Experiment setup

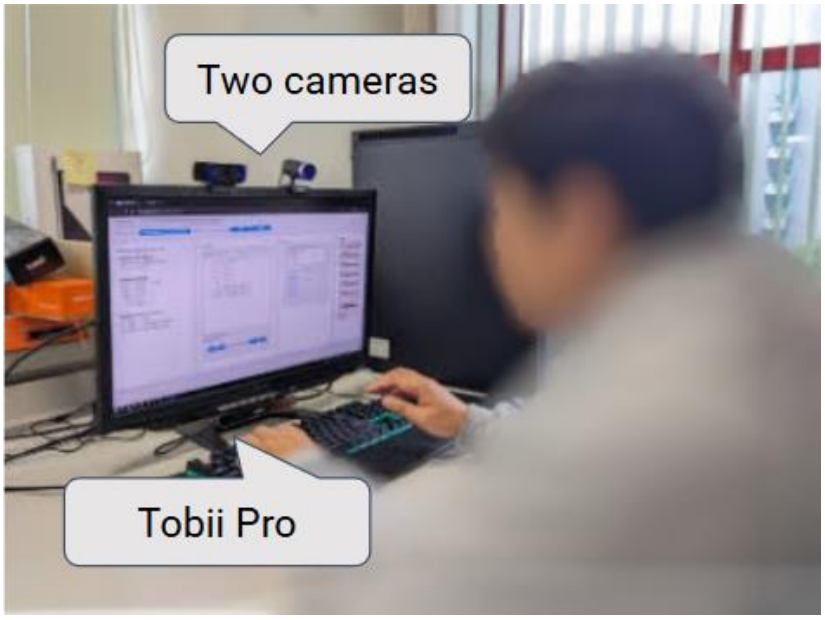


Figure 2 illustrates these modules in action. Our experiment setup uses two cameras. The software-based WebGazer, integrated with the IDE, uses one of the web cameras. A second web camera records the video for participant annotation purposes and for processing with OpenFace. Eventually, our IDE with a fully integrated OpenFace module would only need one camera.

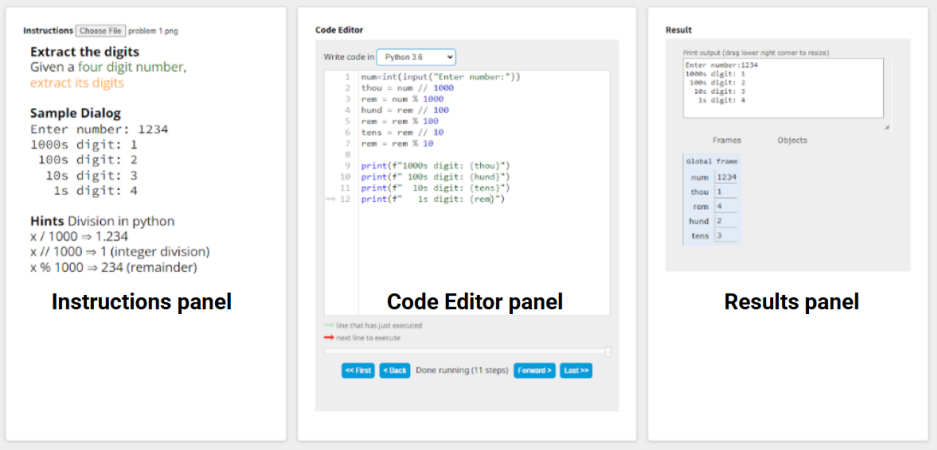
To test the reliability of the software-gaze tracking via the web camera, a hardware-based Tobii Pro gaze tracking device is located below the monitor. Tobii Pro provides more accurate and reliable gaze data to which the software-based gaze tracks can be compared to. Our full system will not need a hardware gaze-tracker to work.

Finally, the IDE itself keeps track of the student code as it changes over time. All in all, the system collects face data, code data, software gaze data (via WebGazer), and hardware gaze data (via Tobii).

These four datastreams are merged then fed into an XGBoost trained model which tries to predict the student emotion based on previously trained data. XGBoost (Chen & Guestrin, 2016) is an ensemble learning method where the predictions of multiple decision trees are combined using a gradient-boosting framework to create a much stronger prediction outcome. XGBoost maps the information gathered from code, face, and gaze data to predict the student's emotions based on a model trained from previously learned information (with XGBoost hyperparameters optimized using Optuna (Akiba et al., 2019)). The predicted emotion can then be used by a teacher or student assistant to determine if an intervention is necessary to help the student with the learning process. This paper primarily discusses the training of XGBoost models and the performance in predicting the actual emotions displayed by experiment participants.

IDE and experiment setup

Figure 3
Experiment IDE



As mentioned earlier, our IDE integrates the functionality of Online Python Tutor (Guo, 2013) and WebGazer (Papoutsaki et al., 2016) into a single IDE that can show the Programming problem, execute the code line by line, visualize program variables, track the user's gaze, and log both gaze and code information.

Prior to the experiment, participants are given an overview of the system. The crucial step of obtaining informed consent is also initiated, as the experiment involves recording a video of the participants during their work. Only after informed consent is granted does the experiment proceed. Our experiment procedure was also approved by our institution's Ethics Review Board (Approval Number: 2022-I-53-1, valid until Sept 2026).

Overall, participants tackle three programming problems using the IDE, one as a warmup exercise and for the participant to get to know the system, and two for the actual experiment where logging is active. These problems are at the level that is given to CS 1 students (see Table 1).

Table 1

Programming problems for the experiment

Problem #	Problem Description
Warmup	You are given the scores of four exams. Output their average Sample dialog: Input exam 1 score: 100 Input exam 2 score: 90 Input exam 3 score: 94 Input exam 4 score: 83 Average: 91.75
Problem 1	Given a four-digit number, extract its digits Sample Dialog: Enter number: 1234 1000s digit: 1 100s digit: 2 10s digit: 3 1s digit: 4
Problem 2	The Japan Meteorological Agency classifies storms according to their wind speed. • Violent Typhoon - 54m/s and higher • Very Strong Typhoon - 44m/s to 53m/s • Typhoon - 33m/s to 43m/s • Severe Tropical Storm - 25m/s to 32m/s • Tropical Storm - 17m/s to 24m/s • Tropical Depression - Lower than 17m/s Write a program, given the wind speed, outputs the storm classification Sample Dialog: Enter wind speed: 30 Severe Tropical Storm

The participants are given adequate time to solve a programming problem, though a maximum duration of 30 minutes was enforced to provide a definite end-time to the programming session. After the experiment is over, the participants are next asked to view videos of their entire programming session divided into 20-second blocks. The video shows their faces as well as their code development. The participants are asked to identify the emotion they felt in that 20-second video given the following emotion descriptions:

- Frustration: You have strong feelings of anger or disappointment.
- Confusion: You have feelings of uncertainty on how to proceed.
- Boredom: You have a lack of interest in continuing with the activity.
- Neutral: You have no positive or negative feelings at this time.
- Engagement: You are immersed in the activity and know what to do next.

These emotion descriptions are defined in the same way as in Tiam-Lee & Sumi (2018, 2019) as was formalized by Bosch et al. (2013). This strategy of participants reviewing their performance video is a well used technique in eliciting emotion data (Harley, 2016) and was used in previous work such as D'Mello & Graesser (2012) and Tiam-Lee & Sumi (2018, 2019) but does come with some drawbacks as discussed in the Limitations section of this paper.

Participants were rewarded with a 2000-yen Amazon voucher for their work with the project.

Raw data preprocessing

The experiment gathers three data streams: a video recording of the participant's face, gaze tracking information through hardware (Tobii) and software (WebGazer), and code logs. In this section, we provide a short discussion on the preprocessing needed to convert the raw data logs into a dataset that will be fed to the learning model.

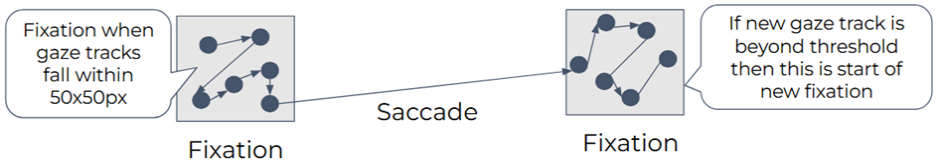
As mentioned previously, the participant's Facial Action Units were extracted through OpenFace at the rate of 30 frames for each second of video. The AU data is appended with the labeled emotion to form the Face dataset.

Second, WebGazer and Tobii provide the estimated location of where the user is looking at the screen in terms of x and y coordinate values. Leveraging the additional sensitivity of hardware based Tobii gaze tracking, the participant's pupil diameter was also recorded. As pupil diameter is different per participant, we normalize a participant's pupil diameter over the entire programming session.

These raw x and y values are converted into two basic metrics used in gaze track analysis: fixations and saccades. A fixation occurs when the gaze focuses on a specific point of interest on the screen, while saccades are the rapid movement between fixations. Buscher et al.'s (2008) strategy was used to convert the raw x and y gaze tracks into fixations and saccades (see Figure 4).

Figure 4

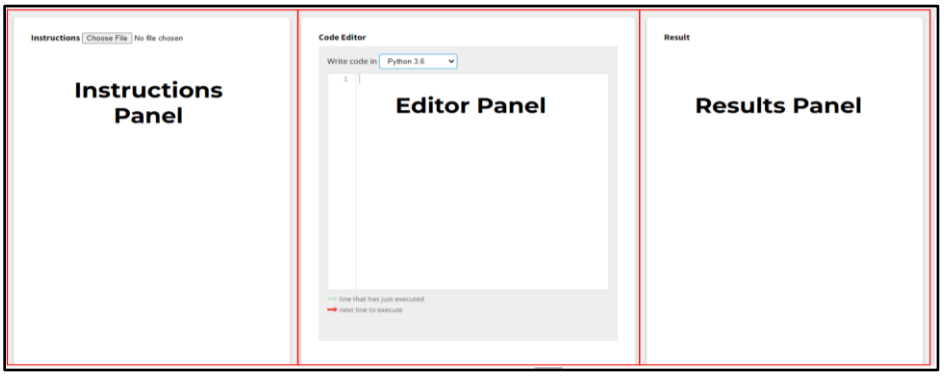
Fixations and saccades



In addition to fixation and saccades, we divided the IDE into Areas of Interest (AOI), a typical strategy used in gaze-tracking research (see Jacob et al., 2018; Lagmay & Rodrigo, 2022; and Mangaroska et al., 2022 for some examples). Raw gaze tracks were considered to fall into one of four AOIs: the Instructions Panel, Code Editor Panel, Results Panel, and Outside (for all other x and y values not falling within the different panes). We will discuss how AOIs are handled in the next section. Figure 5 shows the IDE's Areas of Interest.

Figure 5

Areas of interest



Next, the IDE logs user code as it changes over time. The code, together with the associated emotion data, makes up the Code dataset. The processing of the Code dataset will be discussed in the next section.

Finally, all four datasets are merged into one giant dataset, aligning to a single unified timestamp, simplifying the next step of our data processing.

Time blocks

We decided to divide the data into 60-second time blocks. Time blocking allows for the IDE to gather all information within a set time interval for processing with the learning model, allowing for a live prediction of the user's current emotion.

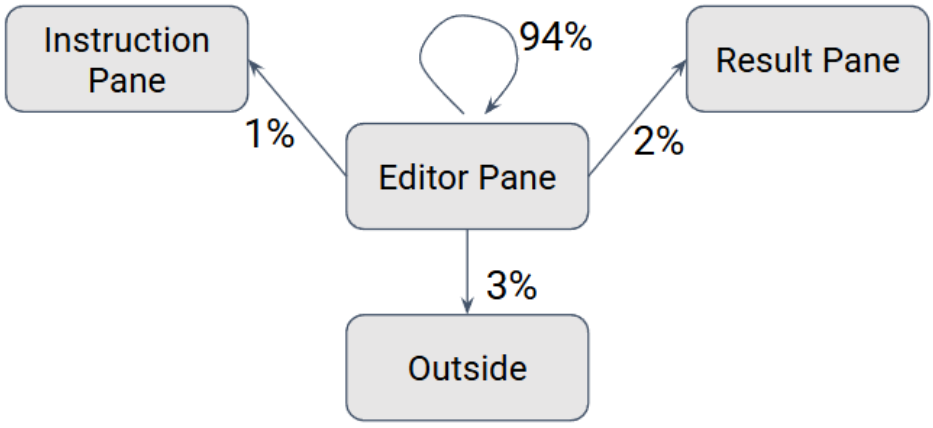
We performed multiple strategies to consolidate the information inside a block, a summary of which can be found in Table 2. In general, we computed the descriptive statistics for code duration, strength and presence of facial action features, fixation duration,

saccade duration, and saccade velocity. In addition, Tobii data also has pupil diameter, so we computed for its mean and standard deviation per block.

We also considered the percentage of time spent within each AOI per block and the two-way transition probability between the different AOIs, a variation of the technique used in Mangaroska et al. (2022). Figure 6 provides an example of the transition probability from one of the AOIs.

Figure 6

Transition probability



All in all, the consolidated information for the Code, Face, and Gaze datasets consists of 295 input features, not counting the target emotion label.

Table 2

Input features

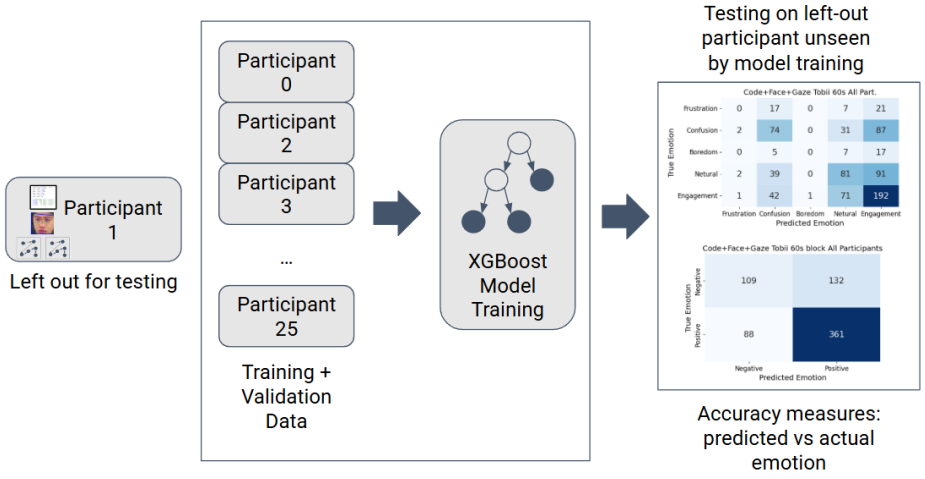
Dataset	Description	Descriptive Statistics
Code	Duration between code changes	Mean, Stddev
Face	Presence and Strength of the following Facial Action Units: • AU1 - Inner Brow Raiser • AU2 - Outer Brow Raiser • AU4 - Brow Lowerer • AU5 - Upper Lid Raiser • AU6 - Cheek Raiser • AU7 - Lid Tightener • AU9 - Nose Wrinkler • AU10 - Upper Lip Raiser • AU12 - Lip Corner Puller • AU14 - Dimpler • AU15 - Lip Corner Depressor • AU17 - Chin Raiser • AU20 - Lip Stretched • AU23 - Lip Tightener • AU25 - Lips Part • AU26 - Jaw Drop • AU45 – Blink	Mean Stddev Min Quartile 1 Median Quartile 3 Max
Tobii Gaze (hardware) WebGazer (software)	Fixation Duration Saccade Duration, Distance, Velocity Normalized Pupil Diameter (Tobii only) Area of Interest percent duration Area of Interest Transition Probability	Mean Stddev Min Quartile 1 Median Quartile 3 Max

Machine Learning

Once the data streams have been gathered, merged, and consolidated, we can now begin the XGBoost model training on different combinations of the datasets.

Figure 7

Leave-one-participant-out testing



We used leave-one-participant-out cross validation for all of our tests (see Figure 7). To follow our illustration, we leave out Participant 1 for testing. The remaining 25 other participants are used as training data to create an XGBoost model. We then check for the accuracy of this model on the left-out participant. We do this 26 times, leaving out each of the participants in turn. We showcase the aggregate accuracy measures later in the Results section.

Analysis and Results

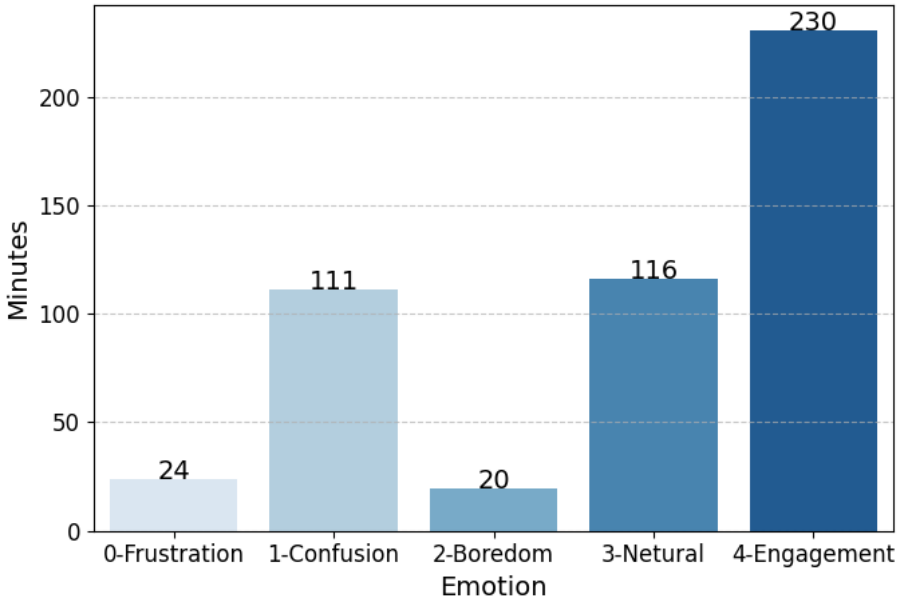
This section discusses the results of model training and testing. To facilitate an easier discussion, some of the tables are placed in the Appendix section of this paper.

Dataset Description

Our experiments involved 26 participants taking up graduate studies in a technical course. About 8 hours and 20 minutes of video data was collected, with an emotion distribution shown in Figure 8.

Figure 8

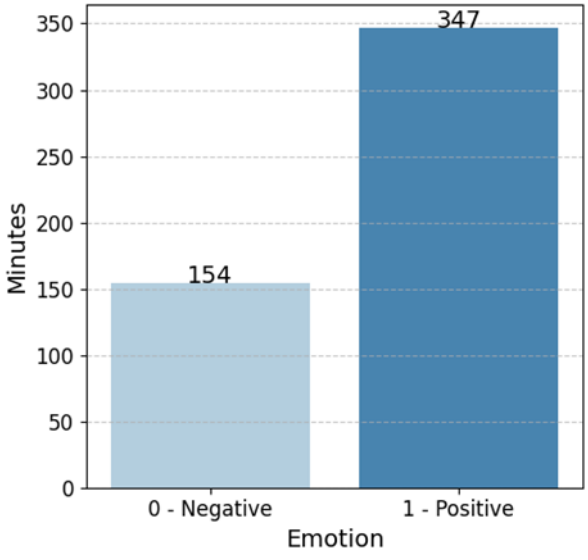
5-Emotion distribution showing length of time of each emotion



To balance the training data, we consolidated the emotions into negative emotions (frustration, confusion, boredom) and positive emotions (neutral, engagement). Figure 9 shows the distribution between these two emotions.

Figure 9

2-emotion distribution showing length of time for each emotion. Negative emotions: frustration, boredom, confusion. Positive emotions: neutral, engagement



Tests Overview

For each of our tests, we showcase the spread of accuracies obtained from the 26 learning models produced by leave-one-participant-out cross validation. Both the full 5-emotion and the consolidated 2-emotion outcomes are explored.

Our core analyses address RQ2, focusing on the impact of increasing levels of multimodality in model performance. We begin by evaluating models trained on individual datasets (code only, face only, gaze only). We next assess combinations of two datasets (code + face, code + gaze, face + gaze), followed by a fully multimodal dataset incorporating all three (code + face + gaze). We provide statistical tests that attempt to show the gains in accuracy by increasing the multimodality level.

To address RQ1, we examine the two gaze tracking methods: hardware-based Tobii and software-based WebGazer alone and in combination with the other datasets. We then provide a comparison using statistical analyses to assess their similarities.

Here is the shorthand version that we will use to refer to the datasets in combination:

- code - code only
- face - face only
- g_tobi - gaze Tobii
- g_web - gaze Webgazer
- code+face - code and face
- code+g_tobi - code and gaze Tobii
- code+g_web - code and gaze Webgazer
- face+g_tobi - face and gaze Tobii
- face+g_web - face and gaze Webgazer
- all_tobi - code + face + gaze Tobii
- all_web - code + face + gaze Webgazer

2-emotion testing

First, we discuss 2-emotion testing results. Figure 10 shows the accuracy spreads, while Table 3 shows the mean and standard deviation.

Figure 10
Testing out the different 2-emotion datasets in combination

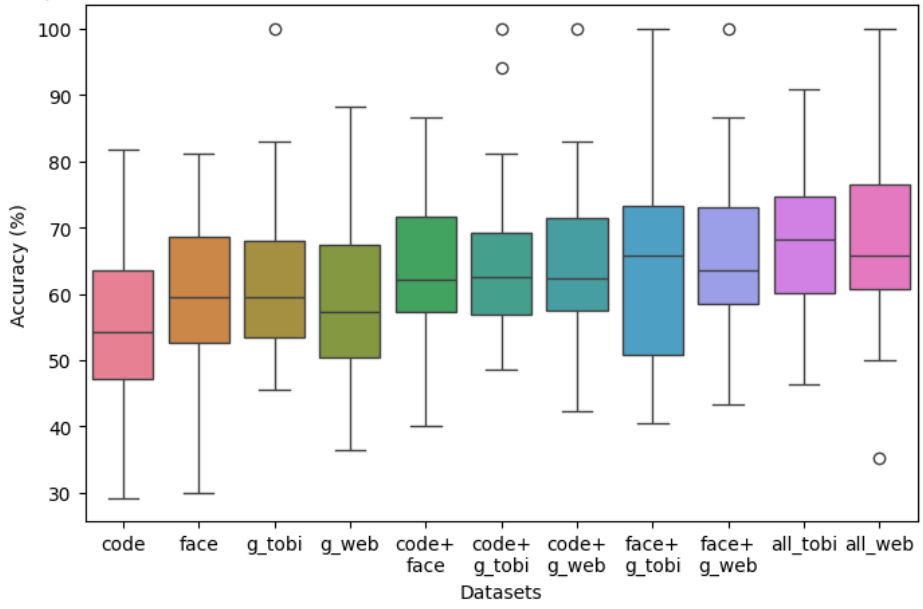


Table 3
Mean accuracies and standard deviations of the 2-emotion dataset combinations

2emo	code	face	gtobi	gweb	code face	code gtobi	code gweb	face gtobi	face gweb	all tobi	all web
mean	56.02	60.06	62.52	59.32	64.21	65.00	64.30	63.62	65.33	68.77	66.88
stdev	11.71	12.21	12.61	12.83	11.13	12.52	12.99	14.27	12.80	12.57	12.95

There is a considerable variation in the accuracies given by leave-one-participant-out testing, with median accuracies mainly falling between 55% and 65%. The highest mean accuracies are achieved by the dataset that contains a combination of all three input streams (i.e., all_tobi and all_web).

Although not universally observed across all comparisons, increasing the level of multimodality generally leads to a statistically significant improvement in accuracy scores (t-test, $p < 0.05$). In certain cases, such as comparing the code-only dataset with the code+face+gaze datasets (using both Tobii and WebGazer), the p-values fall below 0.01, providing strong evidence of significant gains in accuracy. Combinations with statistically significant results are presented in Table A1.

We also performed a t-test and Two One-Sided Test (TOST) analysis between paired Tobii and WebGazer datasets. There is no statistical difference between each pair (t-test, $p > 0.05$), and each pair is statistically equivalent to within ±10% ($p < 0.05$). To highlight this specific result, the datasets consisting solely of gaze data show statistically equivalent results within ±10% between Tobii gaze data and WebGazer gaze data. In addition, for

code+g_tobi vs code+g_web, and all_tobi vs all_gweb, each pair is statistically equivalent to an even stricter ±8% equivalence margin.

See Table A2 for the full statistical results.

5-emotion testing

Figure 11
Testing out the different 5-emotion datasets in combination

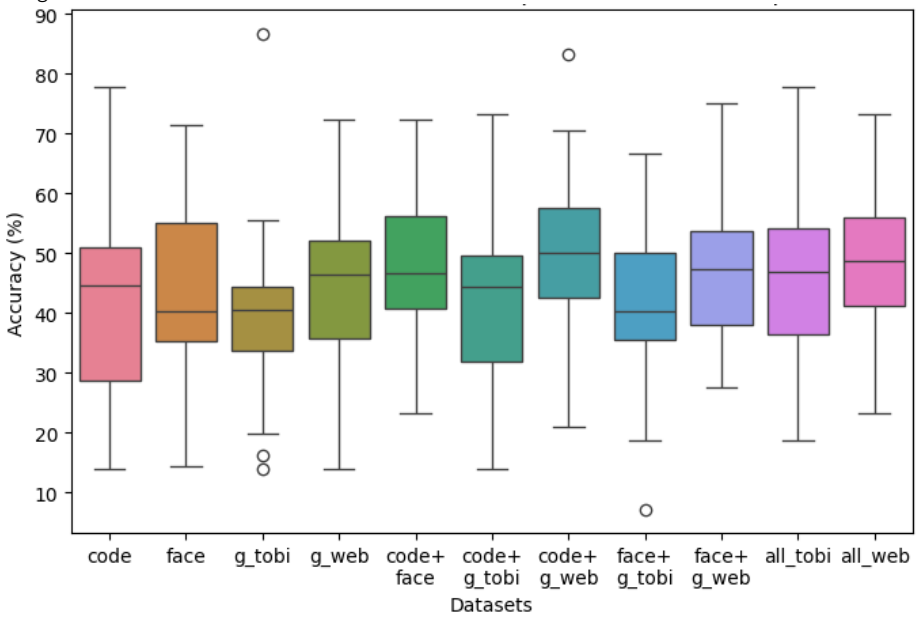


Table 4
Mean accuracies and standard deviations of the 2-emotion dataset combinations

2emo	code	face	gtobi	gweb	code face	code gtobi	code gweb	face gtobi	face gweb	all tobi	all web
mean	41.92	43.43	39.28	44.52	48.17	42.10	50.14	42.23	47.20	46.83	50.01
stdev	15.85	14.86	14.12	13.21	11.89	14.48	14.71	14.59	13.12	13.19	13.64

Figure 11 and Table 4 above show the results of the 5-emotion tests. As before, accuracy is generally lower than 2-emotion testing and has a much wider accuracy spread. Interestingly, the highest accuracy for all datasets belongs to software-based gaze tracking code+face+gaze WebGazer dataset at 50%.

The effect of multimodality on the 5-emotion model is not as strong as statistically significant increases in accuracy measures can only be found when comparing the gaze Tobii dataset against multimodal dataset combinations (see Table A3). In addition, Tobii and WebGazer dataset equivalence can only be seen in comparing code+face+gaze Tobii against code+face+gaze WebGazer (t-test, $p > 0.05$, TOTS, $p < 0.05$, equivalence margin ±10%). See Table A4 for full details.

Emotion Analysis

In this section, we showcase the per-emotion prediction of the model as consolidated over all participants in the leave-one-participant-out testing. Figure 12 and Table 5 compares Tobii and WebGazer 2-emotion outcomes. To provide a brief explanation, the model in the left side of Figure 14 correctly predicted 109 cases as showcasing negative emotions (true negative), and 361 cases were correctly identified as showcasing positive emotions (true positive). Each case corresponds to a 60s block of data.

Figure 12

2-emotion confusion matrices Negative emotions: frustration, confusion, boredom. Positive emotions: neutral, engagement

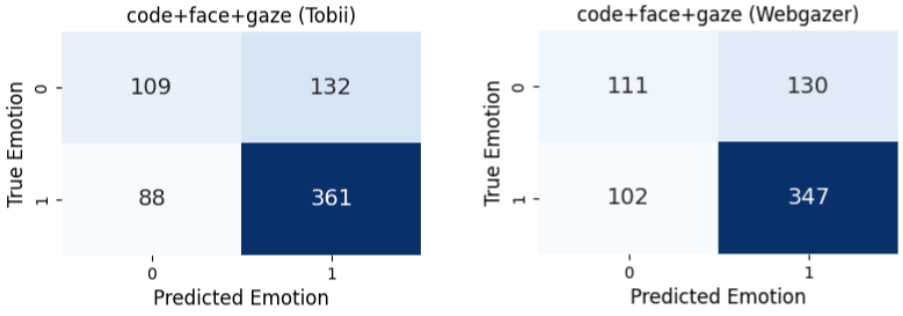


Table 5

Accuracy results 2-emotions

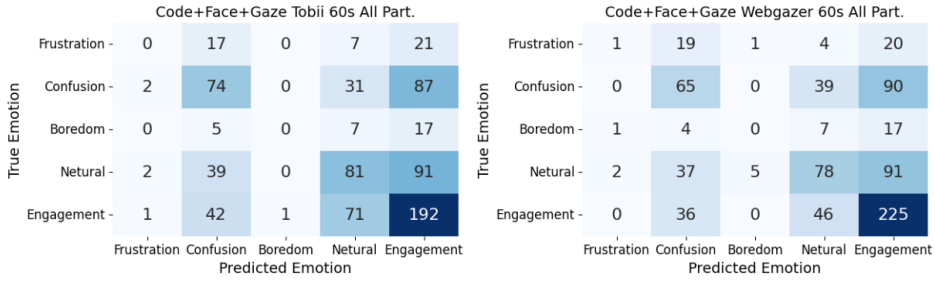
	2-emotions Tobii	2-emotions WebGazer
Accuracy	68.1%	66.4%
Precision	73.2%	72.7%
Recall	80.4%	77.3%
F1 score	76.6%	74.9%

Overall, hardware-based Tobii gaze tracking provides a higher level of accuracy than software-based WebGazer gaze tracking, even achieving 80.4% recall in getting all expressions of positive emotions.

We take a more nuanced analysis with 5-emotion outcomes. Figure 13 shows the 5-emotion confusion matrix, while Table 6 shows the multiple class accuracy metrics. We will discuss these separately.

Figure 13

5-emotion confusion matrices



Our models have the most difficulty in detecting frustration and boredom, no less in part due to the lack of training data available for these two emotions. The model commonly mistakes frustration and boredom for engagement. However, frustration is also falsely recognized as confusion. Compared to these two, the model can capture feelings of confusion but is also mistaken for engagement. The same case also applies to the neutral emotion. Most engagement emotions are correctly identified, with WebGazer predicting more engagement cases than Tobii, though their performance is nearly identical to those of the other emotions.

Table 6

Accuracy results 5-emotions

	5-emotions Tobii	5-emotions WebGazer
Macro Precision	0.26	0.32
Macro Recall	0.28	0.29
Macro F1	0.27	0.28
Micro Precision	0.44	0.47
Micro Recall	0.44	0.47
Micro F1	0.44	0.47
Weighted Precision	0.40	0.43
Weighted Recall	0.44	0.47
Weighted F1	0.41	0.44

In 5-emotion and in contrast to the 2-emotion models, WebGazer outperforms Tobii across all metrics, though these metrics do have room for improvement. The noticeably better Macro scores for WebGazer indicate that WebGazer is more confident in its predictions for all classes while Tobii struggles with less frequent cases.

Next, we consider the effect of increasing modality with the correct detection of specific emotion cases. Let us start with 2-emotion.

Table 7

True negative and true positive emotion detection vs actual emotion count

emo	code	face	gtobi	gweb	code face	code gtobi	code gweb	face gtobi	face gweb	all tobi	all web	actual
neg	112	93	96	82	106	120	114	106	114	109	111	241
pos	266	327	327	291	388	317	332	338	332	361	347	449

Considering individual modalities, face features provide the highest accuracy in detecting positive emotions, detecting 327 out of 449 instances (72.83%), at par with Gaze Tobii. Interestingly, the code-only database is able to detect the most number of true negative emotion cases with 112 out of 241 (or 46.47%) correctly negative emotions, one case more than the all WebGazer dataset.

The effect of increased modality appears best in the detection of true positive emotion cases. The highest level of true positive emotion detection appears with the code+face dataset, with 388 out of 449 cases (86.41%) detected. The all Tobii captured 361 true positive emotions (80.40%) while the all WebGazer dataset captured 347 cases or 77.28%.

Table 8

Correct emotion detection count vs actual number of emotions

emo	code	face	gtobi	gweb	code face	code gtobi	code gweb	face gtobi	face gweb	all tobi	all web	actual
Frus	2	0	1	0	1	1	0	0	0	0	1	45
Conf	59	43	31	30	64	57	69	46	38	74	65	194
Bore	2	0	0	0	0	0	0	0	0	0	0	29
Neut	38	72	81	63	76	77	73	72	77	81	78	213
Eng	218	216	169	229	203	169	219	190	222	192	225	307

With 5-emotions, there is not enough information to show which individual modality is best for detecting Frustration and Boredom. Confusion is best detected by Code analysis at 59 out of 194 cases (30.41%). The Neutral emotion is best detected using Gaze Tobii at 81 out of 213 cases (38.03%). Finally, Engagement is best detected using Gaze Webgazer at 229 out of 307 cases (74.59%), the highest of the group.

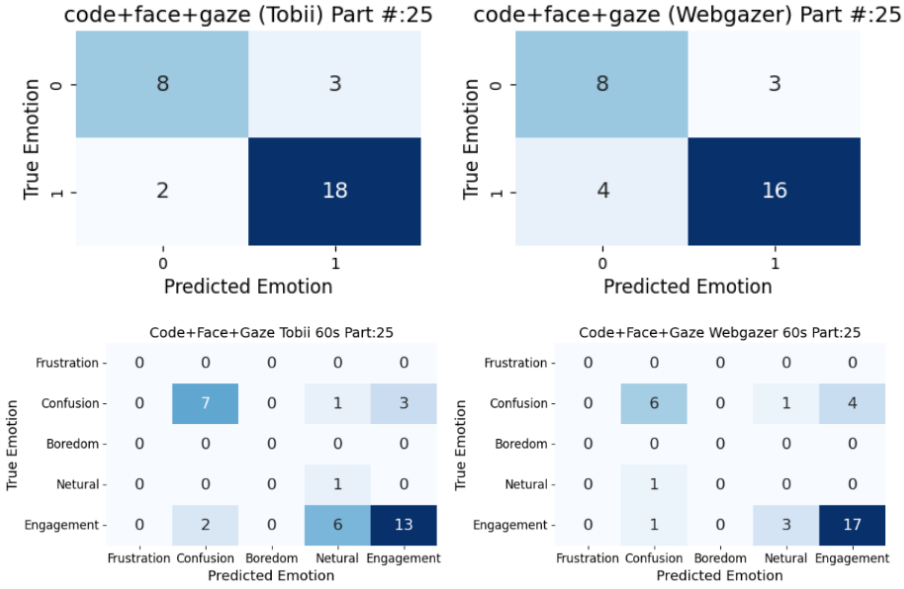
Unfortunately, the effect of increased modality is not as clear. The code+face+gaze datasets were not able to detect a single confusion case, and just one frustration instance in the case of the all WebGazer dataset.

That being said, the code+face+gaze Tobii and code+face+gaze Webgazer have better across-the-board coverage in correct case detection. Interestingly, the code dataset contributes the most towards the correct detection of Confusion cases, a matter that would be good to explore in the future.

To conclude this section, let us consider a specific participant. In this case, participant 25. Figure 14 shows the confusion matrices for 2-emotion vs 5-emotion, and Tobii vs WebGazer.

Figure 14

Confusion matrices of participant #25



Considering that we are using 60-second time blocks, we can consider our WebGazer based model to correctly detect 8 minutes of negative emotions and more than 16 minutes of positive emotions that participant #25 showcased during their experiment. For 5-emotion detection, the model identified 6 minutes in which the participant was confused and more than 13 minutes in which the participant was engaged in the problem-solving experiment.

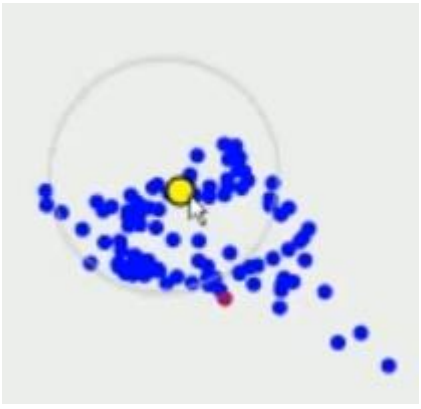
It may be noted that while there are some false detections (such as the 1 minute of Engagement incorrectly identified as Confusion for the 5-emotion WebGazer model), these false detections can still be overcome by having the system signal the teacher whenever these negative emotions are detected. The teacher will be able to pay attention to students that were flagged and their expert judgement can be leveraged to determine the student's emotion and, more importantly, if an intervention is warranted.

In-depth gaze track analysis

In this section, we provide more information on the difference between hardware-based Tobii and software-based WebGazer. Taking the raw gaze tracks over the entire experiment run we found that Tobii and WebGazer vary on average 610±446 pixels. Figure 15 provides an illustration of the uncertainty inherent in WebGazer as was gathered in one of our earlier experiments.

Figure 15

Screen capture of separate experiment showing participant gaze location (yellow circle), Tobii gaze detection (grey circle), WebGazer gaze detection (red circle), previous WebGazer gaze tracks (blue circles)



Next, we conducted a minute-by-minute statistical analysis of the engineered features of fixation duration, saccade duration, saccade distance, and Area of Interest durations derived after aggregation into blocks. In total, 510 60s blocks of Tobii gaze data were statistically compared to their corresponding WebGazer data blocks using independent samples t-tests.

We found only a small number of blocks with no statistically significant differences when analyzing saccade distance (8.84%), saccade duration (16.78%), and fixation duration (22.60%). However, we found that nearly half of detected Area of Interest duration blocks (47.53%) have no statistically significant differences. This suggests that AOI duration reduces the discrepancy between Tobii and WebGazer data as they provide statistical similarity even with the wide variation of the raw x, y values.

As we are comparing the raw information provided by Tobii and WebGazer, we take this opportunity to delve a bit into feature analysis. A per-feature analysis of the training models reveals that pupil_diameter, a feature detectable only by Tobii, does not appear in the top 20 most important features in determining emotion. As such, one can reasonably do away with this additional hardware-only feature with the model training.

Discussion

We now review our results in the context of our research questions.

RQ1: To what degree does using only off-the-shelf sensing equipment detect student programmer emotions?

Ans: Our models, trained from data from only off-the-shelf equipment, provide accuracy, precision, recall, and F1-scores of 66.4%, 72.7%, 77.3%, and 74.9% for 2-emotion

detection. In 5-emotion detection, accuracy metrics provided by off-the-shelf sensing equipment provide less than 50% accuracy but are marginally better than results provided by specialized gaze-tracking hardware.

Regarding the comparison between software-based and hardware-based gaze tracking, both the 2-emotion and 5-emotion models, using all available datasets (code, face, and gaze) segmented into 60s time blocks, showed no statistically significant difference (t-test, $p > 0.05$) and demonstrated statistical equivalence within ±10% margins (TOST, $p < 0.05$). While the software-based gaze tracks have a high degree of uncertainty as compared to the hardware-based gaze tracks, grouping these into Areas of Interest greatly increased the similarities between the corresponding 60s time blocks of the hardware-based Tobii gaze tracking and software-based WebGazer.

RQ2: How does the code, face, and gaze dataset contribute to the overall performance of the emotion detection model when multimodally combined?

Ans: The code dataset provides increased accuracy in detecting true negative cases (in the 2-emotion model), and in detecting Confusion (in the 5-emotion model). Face data contributes to increased cases of correctly detected positive emotions (in the 2-emotion model). Interestingly enough, the less accurate WebGazer data provides a good basis for Engagement detection in the 5-emotion model, detecting a good number of cases on its own, and improving the detection when combined with more data from the other datasets.

Taking into account the aggregated accuracy numbers, in 2-emotion accuracy there is a statistically significant increase (t-test, $p < 0.05$) in accuracy spreads from singleton datasets (code, face, or gaze WebGazer) to the full dataset (code+face+gaze Tobii and WebGazer). This signal is not as strong in the 5-emotion model as the statistically significant increase (t-test, $p < 0.05$) is only seen from the gaze Tobii and code+gaze Tobii dataset to the full dataset.

To summarize, the best results in accuracy in 2-emotion detection is the multimodally combined code+face dataset in looking at individual case numbers, while the code+face+gaze dataset has significantly higher accuracy ranges as compared to the datasets taken alone. This assertion is more difficult to do in 5-emotion detection as singleton datasets provide good performance when taking particular emotions. The full dataset, however, can detect 5-emotions reasonably well across-the-board.

Limitations

In this section we discuss the limitations of our work and provide potential strategies for improvement.

One key limitation of this study is our small sample size (N = 26), which constrains the strength of our conclusions. We, however, believe that the 8 hours and 20 minutes of data that was analysed by our learning models provides a general estimate of the performance

of our methodology in actual use. At the least, our answers for RQ1 allow us to confidently perform additional experiments without needing hardware-based gaze tracking.

Using self-reported emotional annotations by the participants comes with its own inherent flaws. For example, the user may not accurately remember the emotion at that time, or their definition of the emotion is different than the researcher's, or the participant may not be willing to report negative emotions. In addition, watching oneself again may cause additional emotions that weren't in the video. (Harley, 2016)

One way to address this is by involving expert annotators, such as experienced CS1 instructors, to review the participant videos and provide standardized annotations with inter-rater reliability metrics, such as Cohen's Kappa, employed to ensure consistency across annotators. Nevertheless, the weakness of self-annotation also presents an opportunity as it enables the development of personalized models trained on each user's own data, reducing the need for external observers and preserving participant privacy.

While the best accuracy of 68.1% and an F1-score of 74.9% using the 2-emotion dataset may seem lacking in some instances, the 5-emotion dataset presents greater challenges. The lack of instances of boredom and frustration causes their frequent misclassification into the other majority classes of Engagement, Neutral, and Confusion, resulting in a low Macro-F1 score of 27%. However, the Micro-F1 score of 44% shows that the model does provide a degree of recognition on the majority classes of Engagement, Neutral, and Confusion.

To improve performance, future experiments could elicit feelings of frustration and boredom at predefined intervals to better balance the dataset. For example, specifically defining moments where the IDE becomes unresponsive or having it output syntax errors where none exist can induce these emotions.

Given these findings, the 2-emotion dataset does provide more robust results in emotion detection. However, the 5-emotion dataset still has some utility. While frustration and boredom are harder to detect consistently, when they are correctly identified, the system can at least alert the teacher to take a closer look. Calling in a human expert can help compensate for the current system's limitations and still provide the needed intervention to help struggling students.

Lastly, our use of 60s blocking was only made from an engineering standpoint. It would benefit future work on choosing blocking durations that are grounded in physiological emotion research.

Conclusion

This paper discussed our research on detecting student emotions through an online IDE via equipment readily accessible to typical students of a CS 1 class. Our results showed that software-based gaze tracking provides lower but reasonably close results to hardware-

based gaze tracking. With this result, our future data-gathering experiments can be safely done without hardware-based gaze tracking, allowing us to deploy our work at scale.

Considering individual modalities, we found that face features provide the highest accuracy in detecting positive emotions, while code features are best used to detect true negative emotions. Again, considering individual modalities, code features can be used to detect Confusion, hardware gaze tracking for Neutral, and software gaze tracking for Engagement. However, training models on multiple data streams provide statistically higher accuracies, and thus multimodality is a solid way forward in improving the reliability of both 2-emotion and 5-emotion detection.

Our model currently uses a simple way of evaluating code logs by only noting the duration between code changes. More detailed measures can be explored, such as in the work by Pullar-Strecker et al. (2023), where they noted features such as when the program was run, text insertions and deletions, time from first and last text edit events, average latency between code events, among others. In addition, we can also explore Hidden Markov Model (HMM) code modeling (Blikstein et al., 2014) and similarity scores to known solutions (Dong et al., 2021). Using other machine learning strategies is also a worthwhile avenue to explore.

In its present form, our IDE only logs gaze and code information to gather data for training learning models. Future versions of the IDE will use these models to provide automated feedback on the user's emotional state. One way would be through the Players panel. The Player's health bar is planned to react based on the user's emotions. It will decrease the more negative emotions that are detected. The Players panel also shows the health bars of the other students in the class and the overall health of the entire class, which can serve as a prompt for teacher or classmate intervention.

The ultimate goal of this research is to develop a system that allows CS 1 students to passively ask for help as they work on their programming exercises. While this system can never replace a teacher's expertise, a tool like this could be a valuable resource for teachers, enabling them to provide timely intervention to struggling students and enhance the learning experience for all.

Appendix

Table A1

2-emotion datasets with statistically significant difference

Dataset1	Dataset2	Statistical Measure	Description
code	code+face	t_statistic = -2.5844, p_value = 0.0127	code statistically less accurate than code+face (p-value < 0.05)
code	face+gaze tobii	t_statistic = -2.0986, p_value = 0.0409	code statistically less accurate than face+gaze tobii (p-value < 0.05)
code	face+gaze webgazer	t_statistic = -2.7358, p_value = 0.0086	code statistically less accurate than face+gaze webgazer (p-value < 0.01)
code	code+gaze tobii	t_statistic = -2.6713, p_value = 0.0102	code statistically less accurate than code+gaze tobii (p-value < 0.05)
code	all tobii	t_statistic = -3.7823, p_value = 0.0004	code statistically less accurate than all tobii (p-value < 0.01)
code	all webgazer	t_statistic = -3.1701, p_value = 0.0026	code statistically less accurate than all webgazer (p-value < 0.01)
face	all webgazer	t_statistic = -2.5328, p_value = 0.0145	face statistically less accurate than all webgazer (p-value < 0.05)
gaze webgazer	all tobii	t_statistic = -2.6829, p_value = 0.0099	Gaze webgazer statistically less accurate than all tobii (p-value < 0.01)
gaze webgazer	all webgazer	t_statistic = -2.1150, p_value = 0.0394	gaze webgazer statistically less accurate than all webgazer (p-value < 0.05)

Table A2

Comparing 2-emotion Tobii and WebGazer datasets

Dataset	Statistical Measure (t-test/TOST)	Description
gaze tobii vs gaze webgazer	t_statistic = 0.9078, p_value = 0.3683 Lower: t = 3.7417 p = 0.0002 df = 49.98 Upper: t = -1.9261 p = 0.0299 df = 49.98	Two groups not statistically different (p-value > 0.05). Two groups statistically equivalent to within ±10%
face+gaze tobii vs face+gaze webgazer	t_statistic = -0.4549, p_value = 0.6511 Lower: t = 2.2050 p = 0.0161 df = 49.42 Upper: t = -3.1149 p = 0.0015 df = 49.42	Two groups not statistically different (p-value > 0.05). Two groups statistically equivalent to within ±10%
code+gaze tobii vs code+gaze webgazer	t_statistic = 0.1985, p_value = 0.8435 Lower: t = 2.4591 p = 0.0087 df = 49.93 Upper: t = -2.0622 p = 0.0222 df = 49.93	Two groups not statistically different (p-value > 0.05). Two groups statistically equivalent to within ±8%.
all tobii vs all webgazer	t_statistic = 0.5345, p_value = 0.5953 Lower: t = 2.7948 p = 0.0037 df = 49.96 Upper: t = -1.7257 p = 0.0453 df = 49.96	Two groups not statistically different (p-value > 0.05). Two groups statistically equivalent to within ±8%

Table A3

5-emotion datasets with statistically significant difference

Dataset1	Dataset2	Statistical Measure	Description
gaze tobii	code+face	t_statistic = -2.4529, p_value = 0.0177	gaze tobii significantly less (p-value < 0.05)
gaze tobii	code+gaze webgazer	t_statistic = -2.7135, p_value = 0.0091	gaze tobii significantly less (p-value < 0.05)
gaze tobii	face+gaze webgazer	t_statistic = -2.0951, p_value = 0.0412	gaze tobii significantly less (p-value < 0.05)
gaze tobii	all webgazer	t_statistic = -2.7848, p_value = 0.0075	gaze tobii significantly less (p-value < 0.05)
code+gaze tobii	all webgazer	t_statistic = -2.0263, p_value = 0.0481	code+gaze tobii significantly less (p-value < 0.05)

Table A4

Comparing 5-emotion Tobii and WebGazer datasets

Dataset	Statistical Measure (t-test/TOST)	Description
gaze tobii vs gaze webgazer	t_statistic = -1.3813, p_value = 0.1733 Lower: t = 1.2554 p = 0.1076 df = 49.78 Upper: t = -4.0180 p = 0.0001 df = 49.78	Two groups not statistically different (p-value > 0.05). Two groups are **not** statistically equivalent to within ±10%
code+gaze tobii vs code+gaze webgazer	t_statistic = -1.9847, p_value = 0.0527 Lower: t = 0.4850 p = 0.3149 df = 49.99 Upper: t = -4.4545 p = 0.0000 df = 49.99	Two groups not statistically different (p-value > 0.05). Two groups are **not** statistically equivalent to within ±10%
face+gaze tobii vs face+gaze webgazer	t_statistic = -1.2919, p_value = 0.2023 Lower: t = 1.3071 p = 0.0986 df = 49.44 Upper: t = -3.8910 p = 0.0001 df = 49.44	Two groups not statistically different (p-value > 0.05). Two groups are **not** statistically equivalent to within ±10%
all tobii vs all webgazer	t_statistic = -0.8541, p_value = 0.3971 Lower: t = 1.8331 p = 0.0364 df = 49.95 Upper: t = -3.5414 p = 0.0004 df = 49.95	Two groups not statistically different (p-value > 0.05). Two groups are statistically equivalent to within ±10%

Abbreviations

AOI: Areas of Interest; AU: Facial Action Unit; CS 1: Introduction to Programming class; HMM: Hidden Markov Model; IDE: Integrated Development Environment; TOST: Two One-Sided Test

Acknowledgements

The authors would like to thank all the participants of the research experiments.

Authors' contributions

This research is the output of Mario Carreon's Doctoral Studies. His research supervisors, Hirohiko Suwa, Yuki Matsuda, and Keiichi Yasumoto, were primary responsible for the rigorous and critical review of the research. Direct contributions to the experimental setup, data analysis, and machine learning strategies were provided by authors Ko Watanabe, Tomokazu Matsui, and Shoya Ishimaru. All authors read and approved the final manuscript.

Authors' information
Mario Carreon is an Assistant Professor at the Department of Computer Science, University of the Philippines Diliman and formerly a PhD student at the Nara Institute of Science and Technology (NAIST) in Japan at the Ubiquitous Computing Systems Laboratory. He is primarily focused in improving how computer science is taught at the undergraduate level.

Ko Watanabe is a post-doc researcher at the German Research Center for Artificial Intelligence GmbH (DFKI). He is interested in human cognitive augmentation through multimodal sensing, but his current focus is on Trustworthy AI (fairness) in the medical domain.

Tomokazu Matsui is an Assistant Professor of the Ubiquitous Computing Systems Laboratory at NAIST. His main interest is activity recognition of residents in a home environment. Currently, he is working on the development and operation of a smart home kit, and the analysis and evaluation of the collected information.

Yuki Matsuda is a Lecturer at Faculty of Environmental, Life, Natural Science and Technology, Okayama University, Japan where he established the Convivial Computing Laboratory. His research interests are human-in-the-loop systems based on the cooperation of information technology and people toward advanced society using IoT & AI.

Shoya Ishimaru is a Professor at Osaka Metropolitan University and the CEO of Affectify Inc. He received his PhD in Engineering at the University of Kaiserslautern and the title of MITOU Super Creator by the Ministry of Economy, Trade, and Industry in Japan. His research interest is to invent new technologies that augment human intellect.

Hirohiko Suwa is an Associate Professor of the Ubiquitous Computing Systems Laboratory at NAIST. His research interests are in Data Mining, AI Marketing, Disaster Information, Social Information Systems, Ubiquitous Computing Systems, Recommendation Systems, and Social Media.

Keiichi Yasumoto is a Professor and Head of the Ubiquitous Computing Systems Laboratory at NAIST. His research interests include Distributed Computing Systems, Mobile Computing, and Ubiquitous Computing.

Funding
Mario Carreon's Doctoral studies was supported by the Philippine Government's Department of Science and Technology - Engineering Research and Development for Technology (DOST-ERDT) Faculty Development Program.

Availability of data and materials
The dataset used for this research may be shared upon communicating with Mario Carreon with all identifying information linking the data to experiment participants removed as adhering to data privacy laws.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Ubiquitous Computing Systems Lab, Nara Institute of Science and Technology, Japan
[2]Department of Computer Science, University of the Philippines Diliman, Quezon City, Philippines
[3]German Research Center for Artificial Intelligence GmbH (DFKI), Germany
[4]Faculty of Environmental, Life, Natural Science and Technology, Okayama University, Japan
[5]Department of Computer Science, Osaka Metropolitan University, Japan

Received: 21 April 2025 Accepted: 10 December 2025
Published online: 1 January 2027 (Online First: 27 May 2026)

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019, July). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2623–2631). https://doi.org/10.1145/3292500.3330701

Baltrusaitis, T., Robinson, P., & Morency, L. P. (2013). Constrained local neural fields for robust facial landmark detection in the wild. In *Proceedings of the IEEE international conference on computer vision workshops* (pp. 354–361). https://doi.org/10.1109/ICCVW.2013.54

Baltrušaitis, T., Robinson, P., & Morency, L. P. (2016, March). Openface: an open source facial behavior analysis toolkit. In *2016 IEEE winter conference on applications of computer vision (WACV)* (pp. 1–10). IEEE. https://doi.org/10.1109/FG.2018.00019

Batra, R., & Atiq, Z. (2025, February). Emotions and Self-Efficacy of Undergraduate Computing and Engineering Students: A Systematic Literature Review. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1* (pp. 88–94). https://doi.org/10.1145/3641554.3701904

Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. ACM SIGCSE Bulletin, 39(2), 32–36. https://doi.org/10.1145/1272848.1272879

Bennedsen, J., & Caspersen, M. E. (2019). Failure rates in introductory programming: 12 years later. *ACM inroads*, *10*(2), 30-36. https://dx.doi.org/10.1145/3324888

Blikstein, P., Worsley, M., Piech, C., Sahami, M., Cooper, S., & Koller, D. (2014). Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming. *Journal of the Learning Sciences*, 23(4), 561–599. https://doi.org/10.1080/10508406.2014.954750

Bosch, N., D'Mello, S., & Mills, C. (2013, July). What emotions do novices experience during their first computer programming learning session? In *International Conference on Artificial Intelligence in Education* (pp. 11–20). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-39112-5_2

Buscher, G., Dengel, A., & van Elst, L. (2008). Eye movements as implicit relevance feedback. In *CHI'08 extended abstracts on Human factors in computing systems* (pp. 2991–2996). https://doi.org/10.1145/1358628.1358796

Cavalcanti, A. P., Barbosa, A., Carvalho, R., Freitas, F., Tsai, Y. S., Gašević, D., & Mello, R. F. (2021). Automatic feedback in online learning environments: A systematic literature review. *Computers and Education: Artificial Intelligence*, 2, 100027. https://doi.org/10.1016/j.caeai.2021.100027

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the *22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794). https://doi.org/10.1145/2939672.2939785

Craig, S. D., D'Mello, S., Witherspoon, A., & Graesser, A. (2008). Emote aloud during learning with AutoTutor: Applying the Facial Action Coding System to cognitive–affective states during learning. *Cognition and emotion*, 22(5), 777–788. https://doi.org/10.1080/02699930701516759

D'Mello, S., Jackson, T., Craig, S., Morgan, B., Chipman, P., White, H., ... & Graesser, A. (2008, June). AutoTutor detects and responds to learners affective and cognitive states. In *Workshop on emotional and cognitive issues at the international conference on intelligent tutoring systems* (pp. 306-308).

D'Mello, S. K., & Graesser, A. (2012). Language and discourse are powerful signals of student emotions during tutoring. *IEEE Transactions on Learning Technologies*, 5(4), 304–317. https://doi.org/10.1109/TLT.2012.10

Dong, Y., Marwan, S., Shabrina, P., Price, T., & Barnes, T. (2021). Using Student Trace Logs to Determine Meaningful Progress and Struggle during Programming Problem Solving. *International Educational Data Mining Society*.

Ekman, P., Dalgleish, T., & Power, M. (1999). Basic emotions. *San Francisco, USA*, *1*.

Frasson, C., & Chalfoun, P. (2010). Managing learner's affective states in intelligent tutoring systems. In *Advances in intelligent tutoring systems* (pp. 339–358). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-14363-2_17

Girardi, D., Novielli, N., Fucci, D., & Lanubile, F. (2020, June). Recognizing developers' emotions while programming. In *Proceedings of the ACM/IEEE 42nd international conference on software engineering* (pp. 666–677). https://doi.org/10.1145/3377811.3380374

Gorson, J., Cunningham, K., Worsley, M., & O'Rourke, E. (2022, August). Using electrodermal activity measurements to understand student emotions while programming. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1* (pp. 105–119). https://doi.org/10.1145/3501385.3543981

Guo, P. J. (2013, March). Online python tutor: embeddable web-based program visualization for cs education. In *Proceedings of the 44th ACM technical symposium on Computer science education* (pp. 579–584). https://doi.org/10.1145/2445196.2445368

Guo, P. (2021, October). Ten million users and ten years later: Python tutor's design guidelines for building scalable and sustainable research software in academia. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (pp. 1235–1251). https://doi.org/10.1145/3472749.3474819

Harley, J. M. (2016). Measuring emotions: A survey of cutting edge methodologies used in computer-based learning environment research. *Emotions, technology, design, and learning*, 89–114. https://doi.org/10.1016/B978-0-12-801856-9.00005-0

Jacob, S., Ishimaru, S., Bukhari, S. S., & Dengel, A. (2018, June). Gaze-based interest detection on newspaper articles. In Proceedings of the 7th Workshop on Pervasive Eye Tracking and Mobile Eye-Based Interaction (pp. 1–7). https://doi.org/10.1145/3208031.3208034

Jbara, A., & Feitelson, D. G. (2017). How programmers read regular code: a controlled experiment using eye tracking. *Empirical software engineering*, 22, 1440–1477. https://doi.org/10.1007/s10664-016-9477-x

Jonassen, D., Strobel, J., & Lee, C. B. (2006). Everyday problem solving in engineering: Lessons for engineering educators. *Journal of engineering education*, 95(2), 139–151. https://doi.org/10.1002/j.2168-9830.2006.tb00885.x

Lagmay, E. A. D., & Rodrigo, M. M. T. (2022). Enhanced automatic areas of interest (aoi) bounding boxes estimation algorithm for dynamic eye-tracking stimuli. *APSIPA Transactions on Signal and Information Processing*, 11(1).

Lapierre, H. G., Charland, P., & Léger, P. M. (2024). Looking "under the hood" of learning computer programming: the emotional and cognitive differences between novices and beginners. *Computer Science Education*, 34(3), 331–352. https://doi.org/10.1080/08993408.2023.2214033

Luxton-Reilly, A., Simon, Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., Ott, L., Paterson, J., Scott, M. J., Sheard, J., & Szabo, C. (2018). Introductory programming: a systematic literature review. In *Proceedings companion of the 23rd annual ACM conference on innovation and technology in computer science education* (pp. 55–106). https://doi.org/10.1145/3293881.3295779

Madsen, J., Júlio, S. U., Gucik, P. J., Steinberg, R., & Parra, L. C. (2021). Synchronized eye movements predict test scores in online video education. *Proceedings of the National Academy of Sciences*, 118(5), e2016980118. https://doi.org/10.1073/pnas.2016980118

Mangaroska, K., Sharma, K., Gaševic, D., & Giannakos, M. (2020). Multimodal Learning Analytics to Inform Learning Design: Lessons Learned from Computing Education. *Journal of Learning Analytics*, 7(3), 79–97.

Mangaroska, K., Sharma, K., Gašević, D., & Giannakos, M. (2022). Exploring students' cognitive and affective states during problem solving through multimodal data: Lessons learned from a programming activity. *Journal of Computer Assisted Learning*, 38(1), 40–59. https://doi.org/10.1111/jcal.12590

Nolan, K., & Bergin, S. (2016, November). The role of anxiety when learning to program: A systematic review of the literature. In *Proceedings of the 16th koli calling international conference on computing education research* (pp. 61-70). https://doi.org/10.1145/2999541.2999557

Papoutsaki, A., Sangkloy, P., Laskey, J., Daskalova, N., Huang, J., Hays, J. (2016): WebGazer: Scalable webcam eye tracking using user interactions. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence* (IJCAI-16). pp. 3839–3845.

Pekrun, R. (2006). The control-value theory of achievement emotions: Assumptions, corollaries, and implications for educational research and practice. *Educational psychology review*, 18, 315–341. https://doi.org/10.1007/s10648-006-9029-9

Pekrun, R., & Stephens, E. J. (2012). Academic emotions. https://psycnet.apa.org/doi/10.1037/13274-001

Pullar-Strecker, Z., Pereira, F. D., Denny, P., Luxton-Reilly, A., & Leinonen, J. (2023, March). G is for Generalisation: Predicting Student Success from Keystrokes. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (pp. 1028–1034). https://doi.org/10.1145/3545945.3569824

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2), 137–172. https://doi.org/10.1076/csed.13.2.137.14200

Robins, A. V. (2019). 12 novice programmers and introductory programming. *The Cambridge handbook of computing education research*, 327–376.

Rodrigo, M. M. T., & Baker, R. S. (2009, August). Coarse-grained detection of student frustration in an introductory programming course. In *Proceedings of the fifth international workshop on Computing education research workshop* (pp. 75–80). https://doi.org/10.1145/1584322.1584332

Sharma, K., Papavlasopoulou, S., & Giannakos, M. (2021, June). Faces don't lie: Analysis of children's facial expressions during collaborative coding. *In FabLearn Europe/MakeEd 2021-An International Conference on Computing, Design and Making in Education* (pp. 1–10). https://doi.org/10.1145/3466725.3466757

Sipitakiat, A., & Blikstein, P. (2010, June). Robotics and environmental sensing for low-income populations: Design principles, impact, technology, and results. In Proceedings of the 9th International Conference of the Learning Sciences (ICLS 2010), Volume 2: Learning in the Disciplines (pp. 447–448). International Society of the Learning Sciences. https://repository.isls.org/bitstream/1/2899/1/447-448.pdf

Steffan, A., Zimmer, L., Arias-Trejo, N., Bohn, M., Dal Ben, R., Flores-Coronado, M. A., ... & Schuwerk, T. (2024). Validation of an open source, remote web-based eye-tracking method (WebGazer) for research in early childhood. *Infancy*, 29(1), 31–55. https://doi.org/10.1111/infa.12564

Tiam-Lee, T. J., & Sumi, K. (2018, May). Adaptive feedback based on student emotion in a system for programming practice. In *International Conference on Intelligent Tutoring Systems* (pp. 243–255). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-91464-0_24

Tiam-Lee, T. J., & Sumi, K. (2019, May). Analysis and prediction of student emotions while doing programming exercises. In *International Conference on Intelligent Tutoring Systems* (pp. 24–33). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-22244-4_4

Watson, C., & Li, F. W. (2014, June). Failure rates in introductory programming revisited. In *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education* (pp. 39–44). https://doi.org/10.1145/2591708.2591749

Villanueva Alarcón, I., Anwar, S., & Atiq, Z. (2023). How multi-modal approaches support engineering and computing education research. *Australasian Journal of Engineering Education*, 28(2), 124–139. https://doi.org/10.1080/22054952.2023.2274513

Villamor, M., & Rodrigo, M. M. (2018, July). Predicting successful collaboration in a pair programming eye tracking experiment. In *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization* (pp. 263–268). https://doi.org/10.1145/3213586.3225234

Zadeh, A., Chong Lim, Y., Baltrusaitis, T., & Morency, L. P. (2017). Convolutional experts constrained local model for 3d facial landmark detection. In *Proceedings of the IEEE international conference on computer vision workshops* (pp. 2519–2528).