

A Survey on Emotion-Aware Programming Assistance Using Large Language Models, Eye-Gaze Analysis and Keystroke Dynamics

K. Mohan, Suneeta, D A. Swathi, Varsha Annappa Madival

Department of Computer Science and Engineering, Vivekananda College of Engineering and Technology, Puttur, Karnataka, India

Published October 2026

Abstract

Large language models (LLMs) are entering programming education quickly, where they are used to produce hints, feedback, exercises, and code explanations. Most of these tools respond only to the code or error message in front of them and remain unaware of whether the learner is confused, frustrated, bored, or overloaded. This survey reviews twenty studies that bear on this problem, covering LLM-based programming assistance, eye tracking and gaze-aware assistants, facial-expression recognition of engagement and frustration, keystroke dynamics, and behavioral indicators of cognitive load. The literature shows that LLM feedback is generally well received and becomes more useful when it is tied to the student's own code, that gaze reflects attention and comprehension effort, and that typing rhythm changes with emotional state and can be captured without a camera. However, most studies examine a single signal, rely on controlled or non-programming tasks, depend on self-reports, or target authentication rather than learning. From this synthesis we identify a gap: we found very little work that links learner-state sensing to LLM feedback in programming, and none that tests whether doing so improves learning. We outline a conceptual emotion-aware programming tutor as a research direction, kept separate from the reviewed evidence, and discuss privacy, bias, and evaluation challenges.

Keywords: adaptive feedback; cognitive load; emotion-aware tutoring; eye tracking; keystroke dynamics; large language models; programming education

1 Introduction

Learning to program is mentally demanding, especially in the first year of study. Beginners must handle syntax, program logic, tools, and abstraction at the same time, and small mistakes often surface as compiler messages that are hard for them to interpret [1]. When the same error keeps returning, confusion can turn into frustration and then into disengagement, and a learner's affective state has been linked to how much is learned during tutoring [11]. Debugging also loads working memory, and learners who are overloaded tend to show it in the way they interact with a learning system [19].

Large language models (LLMs) have quickly become part of the response to these difficulties. Researchers have used GPT-4 to generate hints for compiler errors in an introductory course [1], used ChatGPT to give feedback on programming assignments [2], collected students' opinions of ChatGPT as a programming helper [4], and tested whether LLMs can produce new exercises and code explanations [5]. Broader reviews describe both the promise and the risks of such tools in education [3].

Even so, a correct hint or a fluent explanation is not the whole of good teaching. A human tutor notices when a student has stared at the same line for a minute, when typing becomes slow and hesitant, or when corrections pile up, and then adjusts the level, tone, and timing of help. The LLM-based assistants reviewed here do not have this awareness. The hints in [1], for example, were triggered by a compiler error and not by anything about the learner, and the students in [2] disagreed about the tone they wanted, which suggests that one fixed style cannot suit everyone. A frustrated beginner and a confident learner who made a slip receive the same kind of reply, although their needs differ.

This gap motivates emotion-aware and adaptive programming assistance. Information about the learner's state is available from several directions: eye gaze reflects attention and comprehension effort [6]–[9], facial expressions have been linked to engagement and frustration [11], typing rhythm changes with emotion [12], [14], [15], and ordinary programming activity such as compile-and-run cycles carries information about struggle [16], [18], [19]. Each of these signals has, however, been studied largely on its own and rarely together with an LLM.

The objectives of this survey are (i) to review how LLMs are used in programming education, (ii) to examine what gaze, facial, keystroke, and interaction signals can and cannot reveal about learner state, (iii) to compare the reviewed studies and identify research gaps, and (iv) to outline, as a clearly labelled research direction, an emotion-aware programming tutor. The scope is limited to the twenty papers cited in this article. It is not a systematic review of the whole field, and no new experiments are reported. Sections 2 to VI review the literature by theme, Section 7 compares the studies and lists gaps, Section 8 presents the proposed direction and its challenges, and Section 9 concludes.

2 LLM-based programming assistance

2.1 Hints, feedback, and explanations

Pankiewicz and Baker [1] connected GPT-4 to an automated assessment platform used in an introductory programming course. When a submission produced a compiler error, the platform generated a personalized hint; the experimental group received these hints for part of each module, while a control group did not. Students rated the hints as highly useful, although the outcomes differed across the error categories examined. Zhang et al. [2] took a different route. ChatGPT wrote feedback on four Java lab assignments in a CS1 class, and students judged it against formative-feedback guidelines. They perceived the feedback as formative and clearly preferred versions generated with their own code included in the prompt, mainly because such feedback was more specific, clearer, and corrective. They also wanted enough code examples, but their views on the tone of the feedback were divided [2].

Sarsa et al. [5] approached the instructor's side of the problem and tested whether LLMs can create programming exercises and explain code. They report that such material can be generated automatically and at scale with considerable promise for teaching, although generated content still needs to be checked by instructors.

2.2 Student perceptions and educational context

Yilmaz and Karaoglan Yilmaz [4] gathered students' views on using ChatGPT while learning to program. Learners felt supported in areas such as confidence, debugging help, and conceptual understanding. These results are encouraging, but they rest mainly on self-reported perceptions and not on

objective measures of behavior or emotion. Kasneci et al. [3] supply the wider frame. They discuss how LLMs can support personalized learning and reduce teachers' workload, but they also stress challenges such as the reliability of generated content, bias, data privacy, and the risk that learners depend on the tool too heavily, and they call for careful human oversight.

2.3 What the studies show together

Read together, these studies support three conclusions. First, context matters: feedback built on the student's own code and error was preferred in [2] and is the basis of the hint approach in [1]. Second, the evidence is mostly short-term and perception-based. Ratings and opinions [1], [2], [4] are more common than measured learning gains, and the review [3] and the generation study [5] do not test learners directly. Third, none of these systems senses the learner. Their trigger is a compiler error [1], an assignment submission [2], or a prompt written by a student or instructor [4], [5]. Personalization therefore means adapting to the code, not to the person. The divided opinions on tone in [2] and the varying results across error types in [1] hint that adapting to the person could matter, but these studies do not test it. The risks raised in [3], especially unreliable output and over-reliance, remain open and become more important if a tutor starts making decisions about when and how to intervene.

3 Eye-gaze and eye tracking in programming and learning

3.1 Gaze metrics and what they indicate

Eye tracking records where a person looks and for how long. Skaramagkas et al. [8] reviewed the metrics used in studies of emotional and cognitive processes, including fixation-based measures, saccade measures, blink behavior, and pupil response. In broad terms, fixations and their durations are read as signs of attention and processing effort, saccades describe how attention moves between regions, and pupil and blink measures are related to workload and arousal. Because these relationships depend on the task and are not one-to-one, a gaze metric is better treated as evidence about a state than as proof of it.

3.2 Gaze in program comprehension

Kather et al. [7] used eye tracking to study how advanced students comprehend programs at the algorithmic level. They compared code written with merged and with sequenced (abutted) composition of code blocks of different familiarity. Their results suggest that readers retrieve and build schemata when code is composed of familiar templates, which may help performance, and they found indicators of higher element interactivity for some compositions, meaning that more mental effort is needed to integrate the parts. The key message for this survey is that gaze on source code reflects how hard the reading task is. Code is not read like prose, and reading patterns change with abstraction and complexity.

3.3 Emotion-related information in eye signals

Tarnowski et al. [9] asked whether emotional states can be recognized from eye movements and pupil data. They identified 18 gaze- and pupil-related features that were useful for classifying emotional states, using movie clips as stimuli and dedicated eye-tracking equipment. This supports the idea that the eyes carry affective information. The setting, however, is far from a coding session, and the need for specialized hardware limits direct use in ordinary classrooms.

3.4 Gaze-aware intelligent assistants

The closest work to our theme is that of Danry et al. [6]. Their assistant turns gaze data from wearable glasses into structured observations that are passed to a multimodal LLM, so that the assistant can reason about the user's cognitive needs. In a controlled study with 36 participants, the gaze-aware assistant was compared with a text-only LLM assistant. Users spoke significantly fewer words with the gaze-aware version, which the authors read as more efficient interaction, and they report improved cognitive outcomes. The tasks involved general reading and comprehension and not programming, and misinterpretation of gaze remains a risk.

3.5 How gaze could complement LLM assistance

Gaze can tell an LLM something a text box cannot: where the learner is looking and how effortful that looking is. In a programming setting, repeated returns to the same line, a long fixation on an error message, or little attention to a hint that has just appeared might indicate difficulty or disengagement, which would be consistent with the comprehension findings in [7] and the metric review in [8]. These are hypotheses for future work and not established results. Practical obstacles include calibration, hardware cost, and the ambiguity of a long fixation, which may mean concentration, confusion, or a wandering mind.

4 Emotion recognition and engagement

4.1 Facial expressions, engagement, and frustration

Grafsgaard et al. [11] analyzed a video corpus of computer-mediated tutoring with 65 students, using an automated facial-expression tool to track movements such as brow raising and lowering, eyelid tightening, and mouth dimpling. Upper-face movements were predictive of engagement, frustration, and learning, and mouth dimpling was a positive predictor of learning and self-reported performance. The study matters because it was carried out in a learning setting and links expression to outcomes that a tutor cares about. Its limits are also instructive. It needs a camera, it depends on reference labels supplied by humans or learners, and similar facial movements can accompany different inner states, for example frustration and concentration.

4.2 Other behavioral routes to emotion

Eye-based recognition [8], [9] is a second route, but its evidence comes from movie stimuli and special hardware, as noted above. A third route is behavioral. Sim et al. [15] studied emotions in text-based mental-health conversations by combining keystroke dynamics with sentiment analysis of the typed text, with the aim of passive emotion sensing. Their findings are preliminary and the domain is helpline support and not education, but they show that typing behavior and typed content can be fused and that useful signals need not come from a camera.

4.3 Emotion and learning behavior

The reviewed work connects emotion to learning in two ways. In [11], expression-based indicators of frustration and engagement were predictive of learning-related outcomes, which is the main reason for wanting to detect them. In [1] and [2], emotion appears only indirectly, through confusion over compiler errors and through preferences about tone. Together they suggest that affect is both a result of difficulty and a factor that shapes how a learner responds to help. An assistant that ignores it risks offering the right content at the wrong moment.

4.4 Comparing facial, gaze, and interaction signals

Facial expression is rich but requires a camera, raises privacy concerns, and is sensitive to lighting, head pose, and individual expressiveness. Gaze is directly tied to attention and comprehension [7], [8] and can show where a difficulty lies, but it usually needs eye-tracking hardware [9]. Keystroke and interaction signals are the least intrusive and the most available inside a code editor, but they are indirect: a pause may reflect thinking, distraction, or confusion. No single signal gives a complete or reliable answer.

4.5 Challenges in inferring internal states

Emotion is an internal state, and every method above observes only behavior. Several difficulties follow. Ground-truth labels come from self-report or from observers and are themselves noisy. Behavior differs between individuals and cultures. Results obtained with induced emotions or scripted tasks may not carry over to emotions that arise naturally while programming. Categories also overlap, so a system may mistake effortful concentration for frustration. For these reasons, outputs from such systems should be treated as probabilistic and used to make gentle adjustments, not to label learners.

5 Keystroke dynamics and emotion

5.1 Typing features

Keystroke dynamics describe how a person types, not what is typed. The raw events are key-down and key-up times. From them researchers derive hold (dwell) time, which is the interval between pressing and releasing a key; key-to-key (flight or latency) intervals between consecutive keys; typing speed; pause lengths; and error-related behavior such as backspace and correction frequency. Cockell and Halak [13] built a number-pad authentication prototype that extracts characteristics of each key press and compares them with a stored user profile, and Sae-Bae and Memon [20] use n-gram latency features computed from keystroke timing to build user templates.

5.2 Typing patterns and emotional states

Epp et al. [12] showed that typing rhythm captured while people typed fixed text could be used to classify emotional states. Lee et al. [14] found in a controlled experiment that emotions induced through auditory stimuli influenced keyboard typing, particularly keystroke timing. Sim et al. [15] added text sentiment to keystroke information for emotion sensing in chat. Taken together, these studies indicate that emotion leaves a measurable mark on typing and that the mark can be captured with an ordinary keyboard.

5.3 Authentication and verification research

A large part of keystroke research asks who is typing, not how the person feels. The IEEE BigData 2023 Keystroke Verification Challenge [17] provided large-scale datasets and evaluation protocols for keystroke-based verification. Sae-Bae and Memon [20] proposed a metric that quantifies how distinguishable a user's keystroke template is and relates it to the false acceptance rate, testing it on two public datasets that include constrained-text and free-text typing. This body of work is relevant here for two reasons. It shows that typing patterns are strongly individual, so any emotion model needs a per-user baseline, and it supplies mature methods for feature extraction and evaluation that can be adapted to learner-state sensing.

5.4 Reliability and limitations

The emotion studies [12], [14] used fixed text or induced emotions in laboratory settings. Programming is different: it mixes code symbols, auto-completion, copying and pasting, and long thinking pauses, and pasting was noted as an important behavior in [16]. Whether the findings transfer to free-form coding has therefore not been shown. Because typing is so individual [20], a model trained on one group of learners may not fit another, and keyboard type and device can also change timing.

5.5 Privacy considerations

Keystroke data can identify people [13], [17], [20], so it behaves like biometric data, and raw logs may contain the code or text that was typed. A responsible design would store timing features instead of content, process data locally where possible, obtain informed consent, and let learners opt out.

5.6 Why keystrokes suit programming environments

A programming environment already produces keystroke and edit events. Collecting them needs no extra hardware, no camera, and no wearable; it is passive and continuous; and it can be combined with compile and run events [16]. For learners who are uncomfortable with cameras, or in settings where cameras are impractical, keystroke analysis offers a lower-intrusion way to follow effort and hesitation, with the caveat that the signal is indirect.

6 Cognitive load and programming behavior

6.1 Observing programming behavior

Gubanyi and Soh [16] collected keystroke and compilation data from CS1 students during assignments and exams and computed compiler-error metrics and behavior-based measures. Students behaved differently in exams than on baseline assignments, and lower- and higher-performing students differed in behavior. An analysis of pasting suggested that over-reliance on external resources during assignments was linked to weaker preparation for compile-and-run exam prompts. The authors recommend tracking programming process data to monitor student experience and engagement.

6.2 Cognitive diagnosis and cognitive load

Chi et al. [18] propose a cognitive-diagnosis method for programming ability that uses keystroke features collected during programming and clustering to classify programming patterns, with the aim of supporting personalized teaching. Suryani et al. [19] analyzed interaction behavior in online learning, such as time spent, input, and click patterns, against cognitive load measured with NASA-TLX, and reported associations between them. The setting in [19] was general online learning and not coding.

6.3 Candidate indicators of learner state

Combining these studies with the gaze and typing work [7], [8], [12], [14], several candidate indicators can be listed: repeated compile-and-run cycles that end in the same error [1], [16]; long pauses; high backspace or correction rates; slowed or irregular typing; large pasted blocks [16]; and gaze that keeps returning to one region of code [7]. These are plausible signals drawn from the literature. How they map to confusion, frustration, or overload during programming has not been established by the reviewed papers and would need validation.

6.4 Support for an adaptive programming tutor

These findings support behavior-triggered help. Instead of waiting for a student to click a help button or for a compiler error to occur, a tutor could watch for patterns of struggle, decide how much help to give (a small nudge or a full explanation), and avoid interrupting productive work. Process data as in [16] and [18] could also feed longer-term learner profiles. The central open question is how to tell productive struggle from unproductive struggle, and none of the reviewed studies answers it.

7 Multimodal integration, comparison, and research gaps

7.1 Why one modality is not enough

The areas reviewed above can be linked in one chain: LLMs provide the help, programming behavior shows what the learner is doing, gaze and keystrokes show how the learner is coping, and emotion or cognitive-state estimates indicate how the help should be shaped. Each link has blind spots when used alone. LLM feedback without sensing is blind to the learner [1]–[5]. Gaze can show where attention is but not why [7], [8]. Keystroke timing is cheap but ambiguous [12], [14]. Facial cues need a camera and may confuse different states [11]. Combining signals can resolve some ambiguity. For example, a long pause while gaze rests on an error message suggests something different from the same pause while the learner looks away from the screen. Evidence that fusion helps is still limited but suggestive: keystrokes were fused with text sentiment in [15] and gaze was fused with an LLM in [6]. Table 1 summarizes the reviewed studies.

Table 1. Comparative summary of the reviewed studies.

Ref.	Main focus	Data / signal	Method / setting	Major contribution	Limitation / gap
[1]	GPT hints for compiler errors (Pankiewicz & Baker)	Compiler errors; student ratings	Controlled study, intro programming course, GPT-4	Auto-generated hints on error; students rated them highly useful	Triggered by the error; not by learner state; outcomes varied by error type
[2]	Student views of LLM feedback (Zhang et al.)	ChatGPT feedback on Java labs; survey	CS1 survey with thematic analysis	Feedback seen as formative; code-aware prompts preferred	Perception only; opinions on tone divided; no affect sensing
[3]	LLMs in education (Kasneci et al.)	None (conceptual)	Review and discussion	Maps opportunities and challenges of LLMs	No implemented system or learner-state sensing
[4]	Student views of ChatGPT (Yilmaz & Karaoglan Yilmaz)	Student self-reports	Study of learner perceptions	Perceived gains in confidence, debugging help, understanding	Self-reported; no objective behavioral or emotional data
[5]	Exercise and explanation generation (Sarsa et al.)	LLM-generated exercises and explanations	LLM generation and evaluation	Material can be generated at scale with teaching potential	Not adapted to learner state; needs instructor review
[6]	Gaze-aware LLM assistant (Danry et al.)	Eye gaze from wearable glasses; speech	Controlled study (n = 36) vs. text-only LLM assistant	Fewer spoken words; more efficient interaction	Not programming tasks; risk of misreading gaze

Ref.	Main focus	Data / signal	Method / setting	Major contribution	Limitation / gap
[7]	Gaze in program comprehension (Kather et al.)	Eye tracking; interviews	Advanced students; merged vs. sequenced code	Gaze reflects schema use and complexity handling	Comprehension, not emotion; small sample
[8]	Eye-tracking metrics (Skaramagkas et al.)	Fixation, saccade, blink, pupil metrics	Literature review	Consolidates metrics for emotional and cognitive processes	Survey-level; no coding-specific model
[9]	Emotion from eye tracking (Tarnowski et al.)	Gaze and pupil features	Classification with movie-clip stimuli	Identifies 18 useful gaze/pupil features	Not programming; dedicated eye-tracking hardware
[10]	Federated learning (Yang et al.)	Distributed datasets	Concept and applications	Training without centralizing sensitive data	Not evaluated for affective tutoring; non-IID data and convergence issues
[11]	Facial expression, engagement, frustration (Grafs-gaard et al.)	Video; facial movements (CERT)	Tutoring video corpus, 65 students	Upper-face movements predict engagement, frustration, learning	Needs camera and reference labels; automation at scale not shown
[12]	Emotion from typing (Epp et al.)	Keystroke timing	Fixed-text typing; classification	Typing rhythm helps identify emotional states	Controlled fixed text; limited transfer to free-form coding
[13]	Keystroke authentication (Cockell & Halak)	Key-press characteristics	Number-pad token prototype; 32-user study	Design and analysis of keystroke-based verification	Identity verification, not emotion or cognition
[14]	Emotion and typing (Lee et al.)	Keystroke timing; auditory stimuli	Controlled experiment	Induced emotion alters typing patterns	Emotion induced artificially, not natural coding
[15]	Keystroke plus text emotion sensing (Sim et al.)	Keystrokes; text sentiment	Emulated helpline chat; preliminary	Shows keystroke and sentiment can be fused	Mental-health chat, not programming; preliminary
[16]	Programming behavior in CS1 (Gubanyi & Soh)	Keystroke and compilation logs	Assignments vs. compile-and-run exams	Behavior differs by context and by performance level	Assessment focus; no emotion classification
[17]	Keystroke verification benchmark (Stragapede et al.)	Large keystroke datasets	Challenge with evaluation protocols	Large-scale benchmark for keystroke systems	Verification focus; not emotion or cognition
[18]	Cognitive diagnosis from keystrokes (Chi et al.)	Programming keystroke features	Clustering of programming patterns	Diagnoses programming ability to support teaching	Not designed for real-time state detection
[19]	Cognitive load from interaction (Suryani et al.)	Duration, input, clicks; NASA-TLX	Online learning behavior analysis	Links interaction behavior to cognitive load	General online learning, not code editors
[20]	Template distinguishability (Sae-Bae & Memon)	Keystroke templates; two public datasets	Quality metric related to false acceptance	Shows users differ in template distinctiveness	Authentication focus; not emotion or cognition

7.2 Research gaps identified from the survey

The following gaps are identified from the twenty studies reviewed here. They describe what we did not find in this set of papers and are not claims about the whole literature.

1. **Limited integration of LLMs with learner-state detection.** The LLM studies [1]–[5] do not use sensed learner state, and the sensing studies [7]–[9], [11]–[20] do not connect to LLM feedback. Only [6] combines sensing with an LLM, and it is outside programming.
2. **Limited use of several signals together.** Most studies analyze one signal. Exceptions such as [6] and [15] show fusion is possible but are not in programming education.
3. **Difficulty separating states.** The studies use different state sets and settings [9], [11], [12], and no shared set of states for programming (for example confusion, frustration, boredom, concentration) was found.
4. **Weak realism of evidence.** Emotions were induced [14], text was fixed [12], stimuli were movie clips [9], or the domain was not programming [6], [15], [19].
5. **Individual differences.** Typing is highly individual [20], and gaze and expression also vary, yet per-learner calibration is rarely examined for tutoring.
6. **Static rather than adaptive feedback.** Tone, timing, and depth of help were fixed in [1], [2], although learners differed in their preferences [2].
7. **Privacy and ethics.** Concerns are raised in [3], and [10] offers a privacy-preserving training idea, but no reviewed study tests sensing-based tutoring against these concerns.
8. **Missing evaluation of learning benefit.** Evidence about LLM help is mostly perceptions and short-term ratings [1], [2], [4], and we found no study testing whether emotion-aware assistance improves programming learning.

8 Proposed research direction, challenges, and future scope

8.1 Conceptual emotion-aware programming tutor

This section is a proposal motivated by the reviewed literature. It has not been implemented or experimentally validated in this paper. What already exists, according to the reviewed studies, is LLM hinting and feedback [1], [2], [5], gaze- and expression-based state indicators [6]–[9], [11], typing-based emotion signals [12], [14], [15], and behavior-based indicators of struggle [16], [18], [19]. What is proposed here is their combination in one tutor, shown in Figure 1.

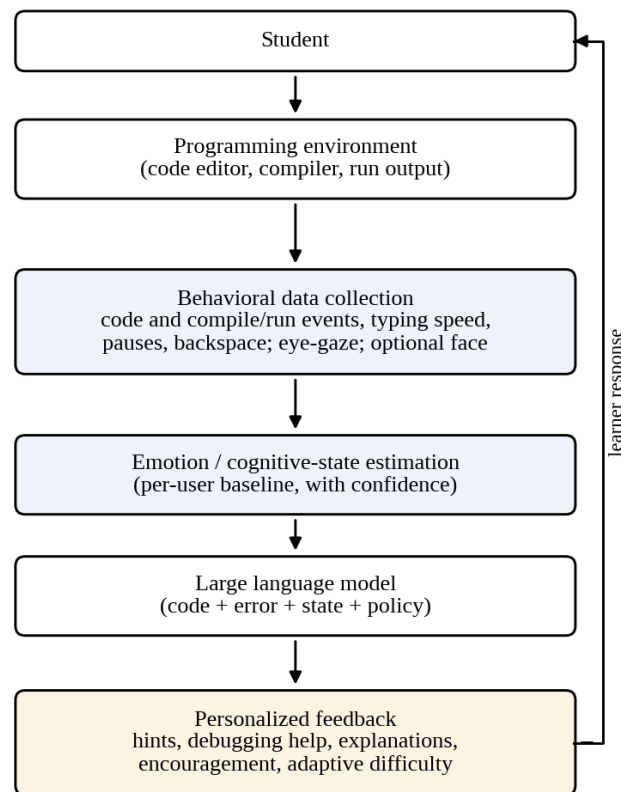


Figure 1. Conceptual emotion-aware programming tutor (proposed direction; not experimentally validated).

The student works in a programming environment, which supplies code submissions and compile and run results. A data-collection layer adds typing speed, pause duration, backspace and correction behavior, and, where a tracker is available and the learner consents, eye-gaze information. Optional facial-expression information could be added with explicit consent. The streams are time-aligned and compared with the learner's own baseline [20]. A state-estimation layer then produces a coarse estimate, for example engaged, confused, frustrated, or bored or overloaded, together with a confidence value. The LLM receives the code, the error, recent history, the state estimate, and a teaching policy, and produces contextual hints, debugging assistance, code explanations, encouragement, adaptive difficulty, or other personalized support.

Illustrative policies, which are hypotheses to be tested, include the following. Repeated identical errors with long pauses might trigger a short conceptual explanation with an example. Signs of frustration might call for a smaller step and a supportive tone. Signs of productive flow might call for withholding hints. Low confidence in the state estimate should lead the tutor to ask the learner a brief question instead of assuming. The tutor should avoid giving full solutions by default, so that the LLM supports learning and does not replace it.

8.2 Challenges and ethical considerations

Privacy, consent, and security. Gaze, face, and keystroke data are sensitive, and keystrokes can identify individuals [13], [17], [20]. Consent should be informed and optional, only features should be stored where possible, and data should be encrypted. Federated learning [10] is one way to train models without centralizing raw data, although it has its own difficulties, such as non-IID data and slow con-

vergence. *Bias and misclassification.* Models trained on one group may perform worse for others, and a wrong estimate could give unhelpful or patronizing feedback. *Individual differences.* Per-user baselines [20] and learner control over adaptation are needed. *Multimodal synchronization.* Signals with different sampling rates and latencies must be aligned in time. *LLM reliability.* Generated hints can be wrong or misleading [3], so outputs should be checked against test results where possible. *Ethical use.* Emotion estimates should not be used for grading or surveillance, learners should know what is inferred and why, and they should be able to turn adaptation off.

8.3 Future research possibilities

Several directions follow from the gaps in Section 7: collecting free-form programming datasets with synchronized keystroke, compile, and gaze data and with learner self-reports; testing which signals add value over interaction logs alone; checking whether webcam-based gaze is accurate enough given that dedicated hardware was used in [9]; building per-learner baselines; running randomized comparisons of emotion-aware tutoring against static LLM feedback with learning gain, not only satisfaction, as the outcome; exploring on-device or federated training; and studying how learners react to being sensed. Smaller locally hosted LLMs could also be examined as a way to reduce cost and keep data on the learner's machine.

9 Conclusion

This survey reviewed twenty studies on LLM-based programming assistance, gaze analysis, emotion recognition, keystroke dynamics, and cognitive-load indicators. LLM research shows that hints, feedback, and explanations are well received and are most useful when tied to the student's own code [1], [2], [4], [5], while also warning about reliability, bias, privacy, and over-reliance [3]. Eye-tracking research shows that gaze reflects attention, comprehension effort, and possibly emotion [7]–[9], and a gaze-aware LLM assistant has been tested outside programming [6]. Keystroke and interaction research shows that typing rhythm changes with emotion and that programming activity differs between learners and contexts [12], [14]–[16], [18], [19], with the authentication literature [13], [17], [20] providing mature methods and a reminder that typing is individual and privacy-sensitive.

The main gaps are the separation between LLM assistance and learner-state sensing, the limited use of several signals together, the difficulty of separating emotional and cognitive states in realistic programming tasks, individual differences, the reliance on static feedback, ethical concerns, and the absence of evidence that emotion-aware help improves learning. An emotion-aware programming tutor that combines code and compile behavior, typing, optional gaze, and an LLM is a promising direction, but at present it is a hypothesis. Careful, privacy-respecting experiments are needed before it can be recommended for classroom use.

References

- [1] Pankiewicz M, Baker RS. Navigating compiler errors with AI assistance—A study of GPT hints in an introductory programming course. In: *Proc. 29th Annu. ACM Conf. Innovation and Technology in Computer Science Education (ITiCSE)*; 2024.
- [2] Zhang Z, Dong Z, Shi Y, Price T, Matsuda N, Xu D. Students' perceptions and preferences of generative artificial intelligence feedback for programming. *Proc AAAI Conf Artif Intell.* 2024;38(21):23250–23258.
- [3] Kasneci E, et al. ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences.* 2023;103:102274.

-
- [4] Yilmaz R, Karaoglan Yilmaz FG. Augmented intelligence in programming learning: Examining student views on the use of ChatGPT. *Computers in Human Behavior: Artificial Humans*. 2023;1(2):100005.
 - [5] Sarsa S, Denny P, Hellas A, Leinonen J. Automatic generation of programming exercises and code explanations using large language models. In: *Proc. ACM Conf. Int. Computing Education Research (ICER)*; 2022. p. 27–43.
 - [6] Danry V, Hernandez J, Wilson A, Maes P, Amores J. From gaze to guidance: Interpreting and adapting to users' cognitive needs with multimodal gaze-aware AI assistants. *arXiv:2604.08062*. 2026.
 - [7] Kather P, Duran R, Vahrenhold J. Through (tracking) their eyes: Abstraction and complexity in program comprehension. *ACM Trans Comput Educ*. 2022;22(2):17.
 - [8] Skaramagkas V, et al. Review of eye tracking metrics involved in emotional and cognitive processes. *IEEE Rev Biomed Eng*. 2023;16:260–277.
 - [9] Tarnowski P, Kołodziej M, Majkowski A, Rak RJ. Eye-tracking analysis for emotion recognition. *Comput Intell Neurosci*. 2020;2020:2909267.
 - [10] Yang Q, Liu Y, Chen T, Tong Y. Federated machine learning: Concept and applications. *ACM Trans Intell Syst Technol*. 2019;10(2):12.
 - [11] Grafsgaard JF, Wiggins JB, Boyer KE, Wiebe EN, Lester JC. Automatically recognizing facial expression: Predicting engagement and frustration. In: *Proc. 6th Int. Conf. Educational Data Mining (EDM)*; 2013. p. 43–50.
 - [12] Epp C, Lippold M, Mandryk RL. Identifying emotional states using keystroke dynamics. In: *Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI)*; 2011. p. 715–724.
 - [13] Cockell R, Halak B. On the design and analysis of a biometric authentication system using keystroke dynamics. *arXiv:1909.10841*. 2019.
 - [14] Lee PM, Tsui WH, Hsiao TC. The influence of emotion on keyboard typing: An experimental study using auditory stimuli. *PLoS ONE*. 2015;10(6):e0129056.
 - [15] Sim KYH, Fortuno KT, Choo KTW. Towards understanding emotions for engaged mental health conversations. *arXiv:2406.11135*. 2024.
 - [16] Gubanyi M, Soh LK. Evaluating programming assignments and compile-and-run prompts in CS1 by observing student programming behaviors. In: *Proc. 21st ACM Conf. Int. Computing Education Research (ICER)*; 2025. p. 210–226.
 - [17] Stragapede G, et al. IEEE BigData 2023 keystroke verification challenge (KVC). In: *Proc. IEEE Int. Conf. Big Data*; 2023.
 - [18] Chi X, Guo X, Sheng Y. Methods for cognitive diagnosis of students' abilities based on keystroke features. *Applied Sciences*. 2025;15(9):4783.
 - [19] Suryani M, Santoso HB, Fathoni Aji R, Hadi S, Schrepp M. *User interaction behavior analysis for cognitive load detection in online learning processes*.
 - [20] Sae-Bae N, Memon N. Distinguishability of keystroke dynamic template. *PLoS ONE*. 2022;17(1):e0261291.