A Comparison of Student Behavioral Engagement in Traditional Live Coding and Active Live Coding Lectures

Anshul Shah
ayshah@ucsd.edu
University of California, San Diego
USA

Fatimah Alhumrani
falhumrani@ucsd.edu
University of California, San Diego
USA

William G. Griswold
bgriswold@ucsd.edu
University of California, San Diego
USA

Leo Porter
leporter@ucsd.edu
University of California, San Diego
USA

Adalbert Gerald Soosai Raj
asoosairaj@ucsd.edu
University of California, San Diego
USA

ABSTRACT

Live coding is a recommended teaching practice in which an instructor dynamically programs in front of students. However, findings related to students' engagement during live coding are mixed. Some works have reported that live coding *seems to improve* student engagement while others regard live coding as an activity in which students passively observe the instructor without asking questions or following along. Active live coding, in which students extend a live coding example and discuss with peers, incorporates active learning with the traditional live coding approach. We conducted a quasi-experimental study in which one section of an advanced introductory programming course was taught using active live coding (ALC) and the other was taught using traditional live coding (TLC). The goal of this work is to compare students' behavioral engagement in the two lectures using a classroom observation protocol called the Behavioral Engagement Related to Instruction (BERI) protocol. Our results from the 2,790 observations we collected indicate that traditional live coding engages only 65% of students, on average. However, we found a "persisting engagement" effect of active live coding, where students were significantly *more* engaged in the traditional live coding components of a lecture up to 20 minutes *after* the active live coding component. Notably, the two lecture groups performed similarly on the Post-Lecture Questions, which were administered after each lecture as a review of the lecture material. Therefore, our results indicate an improved student engagement due to active live coding, but do not show a corresponding improvement in conceptual knowledge.

CCS CONCEPTS

• **Social and professional topics** → **CS1**; **Model curricula**.

KEYWORDS

live coding, active learning, student behavioral engagement

ITiCSE 2024, July 8–10, 2024, Milan, Italy

ACM ISBN 979-8-4007-0600-4/24/07.
https://doi.org/10.1145/3649217.3653537

ACM Reference Format:
Anshul Shah, Fatimah Alhumrani, William G. Griswold, Leo Porter, and Adalbert Gerald Soosai Raj. 2024. A Comparison of Student Behavioral Engagement in Traditional Live Coding and Active Live Coding Lectures. In *Proceedings of the 2024 Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2024), July 8–10, 2024, Milan, Italy.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3649217.3653537

1 INTRODUCTION

Live coding is touted as a recommended teaching practice in computing education [3]. A consistent line of work has focused on the impact of live coding on students' in-class engagement [28], from Paxton's early work in 2002 [20] to Shah et al.'s work in 2023 [30]. Analyses of student engagement during live coding has relied either on the perceptions of instructors [4, 20, 25], who typically perceive high engagement while live coding, or surveys and interviews from students, who typically find live coding too fast, difficult to follow along, and impractical for note-taking [30, 32]. Further, as Gaspar and Langevin point out, live coding often becomes an activity in which students can passively observe the instructor without active engagement [12]—a problem that may be compounded through videos of live coding demonstrations [32]. Due to conflicting perceptions of and limited empirical works on students' behavioral engagement during live coding, a lingering question from instructors may be: *how engaged are my students, really, when I live code?*

Motivated by concerns about students' engagement during live coding, we conducted a quasi-experimental study to compare traditional live coding (TLC)—the typical instructor-led form of live coding where students observe the instructor program [28]—to active live coding (ALC)—a variant of TLC in which all students actively write code between traditional live coding components. In these ALC components, students extend the code written by the instructor during live coding, discuss their code with a neighbor, and then watch the instructor live code a solution, similar to the process of Peer Instruction [22]. We used a classroom observation approach to empirically compare students' behavioral engagement in two lecture sections of the same advanced, introductory programming course: one lecture section taught using only TLC and another lecture section taught using TLC *and* ALC. For each lecture, we also asked students to answer pre- and post-lecture questions, allowing us to measure students' learning gain during the lecture. We use our classroom observations and students' lecture responses to answer the following two research questions:

- **RQ1:** How does behavioral engagement differ between students in traditional and active live coding lectures?
- **RQ2:** How does performance on pre- and post-lecture questions differ between students in traditional and active live coding lectures?

2 RELATED WORK

2.1 Behavioral Engagement

Fredricks et al. conducted a review of education research about "school engagement" and point out that there are three aspects of school engagement: behavioral engagement, emotional engagement, and cognitive engagement [9]. Behavioral engagement refers to students' physical presence and participation in academic activities, such as attending school and paying attention during lectures [9]. While behavioral engagement is strongly correlated with achievement [2, 9, 15], studies have used a variety of metrics to measure different dimensions of behavioral engagement. For example, some studies in computing education research have measured the impact of lecture or laboratory attendance [5, 19, 33]. Others have measured behavioral engagement through interaction with course materials in flipped [1] or online courses [16]. The variety of methods to measure behavioral engagement is largely a result of the different goals of these studies. For example, studies for predicting student success and identifying at-risk students have relied on easily-collected data such as lecture attendance so that other instructors can easily adopt the prediction models [18].

Because our quasi-experiment is narrowly focused on *lecture* techniques, we decided to measure *in-lecture* behavioral engagement using a classroom observation approach called the Behavioral Engagement Related to Instruction (BERI) protocol [17]. The BERI protocol, created by Lane and Harris, describes a methodology to directly observe students' behavior throughout a lecture to estimate classroom engagement [17]. The protocol was specifically designed for measuring engagement in large, in-person university lectures where observation of every single student is not feasible. In the BERI protocol, a classroom observer sits among students and identifies 10 students to observe for the entire lecture. Roughly every 10 to 15 minutes, the classroom observers evaluate each of the 10 students for 15-30 seconds based on a rubric to determine whether a student is "engaged" or "disengaged." If it is difficult to classify a student as engaged or disengaged, the observers should spend an extra 15-30 seconds to observe the student until a classification was clear according to a pre-defined rubric.

Lane and Harris measured the validity and reliability of the BERI protocol by testing it across multiple pairs of observers, in various class sizes, and at different locations in a classroom [17]. The average inter-rater reliability (IRR) of 2,154 observations from 6 pairs of observers was 96.5% and the results proved to be reliable at different course sizes [17]. The authors also tested whether 10 students is a sufficient sample size to assess engagement trends in a class by positioning three observers at various locations in a lecture hall. They found that although there were slight variations among different subgroups of students, the trends in the engagement were the same among the different groups of students, leading the authors to conclude that 10 students can serve as a proxy to understand engagement trends throughout a lecture [17]. We describe our implementation of the BERI protocol in Section 4.1.

2.2 Engagement During Live Coding

Research on live coding predominantly analyzes the traditional, instructor-led form of live coding. Studies have evaluated the effect of live coding on student learning [25, 26], cognitive load [24], engagement [20, 30], and programming processes [29, 30]. The findings have shown that students taught with live coding and those taught with static-code examples perform similarly on exams and demonstrate similar adherence to incremental development and debugging techniques [26, 29, 30]. In terms of student engagement, limited studies have used methods beyond surveying students or leveraging instructor perceptions. Our own prior work in which we conducted a quasi-experimental comparison of live coding to static-code examples [30] found that nearly 20% of students in the live coding group wrote that the lectures were too fast, whereas only 2% in the static-code group wrote that same feedback [30]. Students were also surveyed on the impact of their lecture style on facilitating note-taking and holding their attention. In *both* questions, there was a significant difference in favor of the static-code group (i.e., the live coding students felt their lecture style was *worse* at holding attention and facilitating note-taking). Similar student perspectives on the drawbacks of live coding for engagement have been reported by Stephenson [32] and Grønli and Fagernes [14].

Though these works offer insight on students' self-reported engagement, end-of-term survey questions or student feedback only offers a more high-level understanding of live coding. For example, questions related to students' engagement *throughout a lecture* cannot be answered by such methods. Therefore, our current work to use classroom observation throughout a course can offer instructors a deeper insight into students' engagement in a live coding lecture. Our use of pre- and post-lecture questions will also lend insight into the conceptual knowledge students gain throughout traditional live coding and active live coding lectures.

3 COURSE DESIGN

3.1 Lecture Design

In the Fall 2023 term at our R1 university, there were two lecture sections in our accelerated introductory programming course. Both lecture sections met on Tuesdays and Thursdays over the 10 week term, amounting to 20 total lecture sections. The two sections were taught by the same instructor, who has been using live coding in introductory programming courses for nearly 8 years. Table 1 summarizes the notable differences between the two lectures, such as the different timings and class sizes.

Table 1: The differences between the Active Live Coding (ALC) and Traditional Live Coding (TLC) lectures.

Active Live Coding (ALC)	9:30am	385 students	395-person room
Traditional Live Coding (TLC)	11:00am	196 students	200-person room

Both lecture sections always covered the same content; however, the 9:30am section—the Active Live Coding (ALC) lecture—included an active live coding component. During the active live coding period, students were given roughly 5 to 7 minutes to write part of the live coding example on their own and then discussed their solution with peers for 3 minutes. Nearly all lectures included one active live coding component, although three of the lectures included two active live coding components per lecture. This corresponding time in the 11am lecture—the Traditional Live Coding (TLC) lecture—was used for the instructor to live code the same code snippet that students wrote in the 9:30am section. During the traditional live coding components in both lectures, the instructor would ask the class to predict the output of the code example or to fill in a blank in a line of code that the instructor had written to promote engagement. The instructor would ask for a student volunteer for each question asked throughout the lecture.

The structure of the two lecture sections was consistent throughout the semester. Table 2 describes the different teaching activities conducted during lectures. Typically, each lecture began with course announcements for up to 10 minutes. Then, students were provided a link that was only active for 10 minutes to complete two Pre-Lecture Questions on Gradescope [13] for attendance. Following these preliminary activities, the instructor typically spent 10 to 15 minutes solving a problem from an in-class worksheet that covered material from the previous lecture. After this worksheet review, which typically concluded about 30 minutes into lecture, the instructor began teaching the new content using the traditional form of live coding in which the instructor screen-shares their code editor. While traditional live coding was used extensively in both the ALC and TLC lectures, the instructor used an active live coding component only in the ALC lecture. During this component, students created a fork of the instructor's Edstem [8] workspace, spent 5 to 7 minutes extending the code the instructor had written, then discussed their approach with students sitting near them for 3 minutes (similar to the process for Peer Instruction [21]). After the active live coding component, the instructor used traditional live coding to demonstrate a solution and continued introducing new content. In the TLC lecture, this same time was used for the instructor to live code the same code that students had been asked to write in the ALC lecture. Occasionally, the instructor hand-wrote notes on a sheet of paper that was projected to the class at least once a lecture for several (<10) minutes. The purpose of the handwritten notes was to emphasize certain pieces of conceptual knowledge, such as important vocabulary, or to draw out diagrams. By the end of the live coding and handwritten notes, there were typically 5 minutes left for students to complete two Post-Lecture Questions.

Table 2: The different instructional components used during the two lectures.

Activity	Description
Course Announcements	Instructor provides due date reminders and discusses course logistics
Pre-Lecture Questions	Students answer two multiple-choice questions on Gradescope for attendance credit
Worksheet Review	Instructor solves a problem from an in-class worksheet on a projector
Handwritten Notes	Instructor handwrites notes on a blank piece of paper in front of the class using a projector.
Traditional Live Coding	Instructor codes in front of students and prompts questions to the class
Active Live Coding	Students extend the code from the instructor's workspace for attendance credit
Post-Lecture Questions	Students answer two more questions on Gradescope for attendance credit

3.2 Participants

Our work is approved by our university's institutional review board with exempt status. The classroom observations are covered by this approval due to the observations being done in a public setting without sharing confidential student information. Prior to the term, we gave students a research consent form to opt-in to sharing their grades and survey data with the researchers. In total, 483 students of the 581 (83.1%) students consented to their data being used for research purposes. The results of the consent form were hidden from the instructor, who is a member of the research team, until after final course grades were submitted.

In order to understand our participants' backgrounds, we conducted a Pre-Course Survey in the first week of the term. We asked students about their age, race, preferred pronouns, major, and native language. Among our consenting students, 95.2% were between the ages of 18 to 22. In total, 50.7% of students were first-year undergraduates, 24.4% were second-years, 17.6% were third-years, and 6% were fourth-years. In terms of majors, 85.5% of students were taking the class to fulfill a major or minor requirement, as opposed to taking the class out of interest in CS (8.3%) or to switch into the CS major (4.3%). This high percent of students taking the course for their major or minor indicates that students had an incentive to perform well in the class for GPA records. Our gender distribution is 61.5% male-identifying and 35% female-identifying. The remaining students self-identified into categories of less than 10 students so we do not report these groups to preserve student privacy. The three most common racial groups were Asian/Asian American (61.6%), Latine (12.4%), and Caucasian (10.4%). Finally, 73.3% of our students were native English speakers whereas 25.9% were non-native English speakers, with the remaining students self-identifying as "Bilingual". Since the course is presented as an advanced introductory programming course, many students already have some prior programming experience. Our Pre-Course Survey indicated that 54.3% of students had some knowledge of Java and 71.9% of students had some knowledge of Python before the course.

4 METHODS

4.1 RQ1 Methods: Classroom Observation

To implement the BERI protocol as effectively as possible, the same observer (our *primary* observer) attend both lecture sections for each lecture throughout the term. Although Lane and Harris showed that a single observer analyzing a single group of 10 students is sufficient to capture class engagement trends [17], a secondary classroom observer accompanied the primary observer to roughly half of the lectures. As recommended by Lane and Harris,

Table 3: Condensed rubric used by classroom observers

	Engaged Criteria	Disengaged Criteria
Computer Use	Student is taking notes or has instructor's code open	Student is using the computer for non-class purposes or is doing homework for the class
Looking/ Listening	Student is looking at the instructor and nodding along to demonstrate attention	Student is not looking at the projector and instead is distracted by phone, laptop, etc.
Student Interaction	Student is not talking to neighbors during instruction	Student is talking or laughing with neighbors during instruction

our observers were familiar with the course material and teaching methods that the instructor used. Further, since the authors of the BERI protocol discussed the importance of using trained observers, our primary observer spent the first week doing a "pilot-test" of the protocol. As a result, our lecture observations began at the third lecture. The primary observer then trained the secondary observer for one lecture before the secondary observer began collecting data.

For each pair of lectures on the same day, our observers positioned themselves in the same location in the two lecture halls ensure consistency in the observations (i.e., if the observer sat in the back-middle of the ALC lecture on a certain day, they must sit in the back-middle of the TLC lecture for that same day). Both observers tracked their seating locations and the locations of the 10 students for each lecture to ensure an even distribution of the entire lecture hall throughout the term. Throughout the term, we were able to cover nearly the entire classroom with some seating locations have multiple observations throughout the term.

Since students were encouraged to bring laptops to lecture, we needed to be able to know whether students were using the computer in an engaged manner (i.e., for note-taking, active live coding, etc). Therefore, the observers carefully selected 10 students at the start of lecture such that the students' face and laptop were in view, as these were important factors according to the rubric (see Table 3). Every 10 to 15 minutes, our observers recorded the instructional activity (Pre-LQ, Worksheet Review, Live Coding, etc) and assigned a label ("E" for Engaged or "D" for Disengaged) for each of the ten students they selected. When students were classified as Disengaged, our observers wrote down the reason for disengagement (irrelevant computer use, socializing with neighbors, sleeping, etc). For the observations during the Active Live Coding components, our observers made sure to check the students' laptops to make sure each student was coding on the instructor's Edstem workspace (as opposed to working on a programming assignment, working on material from another class, etc). The two observers messaged each other throughout the lecture to ensure that both observers conducted their data collection at the same time. In general, the observers aimed to be as discreet as possible so students were not aware that they were being observed.

In order to not influence the instructor's teaching methods, the results of the classroom observation were never shown to the instructor of the course until after the course was over. During the weekly research meetings to discuss the experiment, the instructor of the course, who is an author on this paper, left the room so that the observers could reflect on the observations without impacting the instructor's teaching approach.

4.2 RQ2 Methods: Pre- and Post-Lecture Questions

To answer our second question, we analyzed student responses to the pre- and post-lecture questions. Each set of pre- and post-lecture questions included two multiple-choice questions hosted on Gradescope [13]. The link was only active when the instructor allowed students to work on the problems to ensure that only students who were present in lecture could answer. In fact, the post-lecture questions form required students to submit a picture of the in-class worksheet handed out during lecture. We initially aimed for the pre- and post-lecture questions to be isomorphic questions, but halfway into the term we decided to have the pre- and post-lecture questions be the exact same questions due to concerns about whether our questions were truly isomorphic. The questions covered the content for that specific lecture in order to a) measure students' prior knowledge on the topics and b) measure students' learning gain on those topics during the lecture [22].

Our learning gain calculation is loosely based on the Weighted Learning Gain metric, presented by Porter et al. in their work to measure the impact of Peer Instruction [22]. Due to different correctness levels between the two lectures in the pre-lecture questions, our calculation of learning gain focuses only on the Potential Learner Group (PLG)—the group who answered the pre-lecture quests *incorrectly*. Our learning gain metric for each question is the percentage of PLG students that correctly answered the post-lecture question.

5 RESULTS

5.1 RQ1 Results

Table 4 compares students' lecture attendance throughout the quarter. Although the average attendance rates are represented as percentages out of 100, we applied a two-sample t-test [23] (as opposed to a z-test of proportion [27]) since these values are *scores* out of 100 rather than purely binomial distributions. We found no significant difference between the groups and a very small effect size.

Table 4: Comparison of average lecture attendance.

Lecture Condition	N	Mean	Std dev	t stat	p val	eff. size
ALC Lecture	303	87.3%	18.1%	-0.64	0.52	0.064
TLC Lecture	150	88.5%	19.1%			

Our classroom observation data allowed for a multi-faceted analysis of student engagement. Figure 1 shows the average number of students that were engaged during each of the instructional activities in both lectures. The sample sizes indicate the number of students that were observed during each instructional activity throughout the term for a specific lecture group. For example, we observed 180 students in the ALC lecture during Pre-Lecture Questions. There is no data for Active Live Coding during the TLC lecture because no active live coding was used in the TLC lecture.

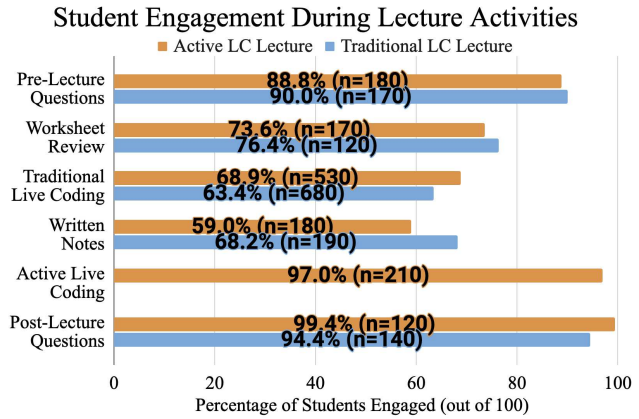

Figure 1: Comparison of behavioral engagement between the two lecture groups.

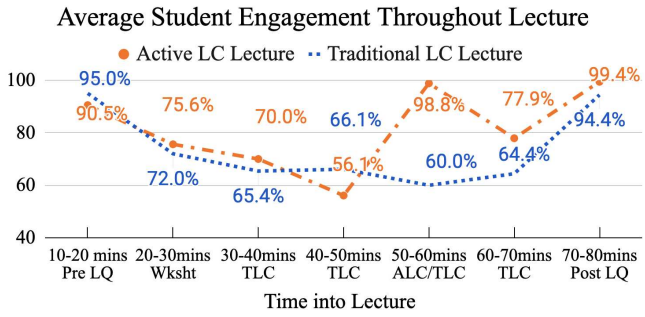

Figure 2: Average engagement throughout lectures.

Figure 2 shows students' average engagement throughout lecture based on the number of minutes into lecture. To create this figure, we identified the most common instructional activity and the average engagement for that activity for each ten-minute increment in the 80-minute lectures. For example, the most common activity from 10-20 minutes into lecture was the Pre-Lecture Questions, from 20-30 minutes it was Worksheet Review, etc. The engagement values represent the average engagement for that activity *within that specific 10-minute window* (i.e., the average engagement for Traditional Live Coding components between 30 to 40 minutes into lecture is 70% for the ALC lecture and 65.4% for the TLC lecture). Note that for the 50-60 minute period, the figure depicts the engagement for Active Live Coding for the ALC lecture but Traditional Live Coding in the TLC lecture.

Figure 3, which focuses only on the ALC lecture, compares students' engagement during traditional live coding components *before* and *after* active live coding. The engagement in the traditional live coding components *before* ALC was 61.5% but rose to 77.7% in the traditional live coding components *after* ALC. Using a two-sample t-test [23], the difference of these means is significant, with a p-value < 0.001 and Cohen's d of 1.16—a large effect size [10].

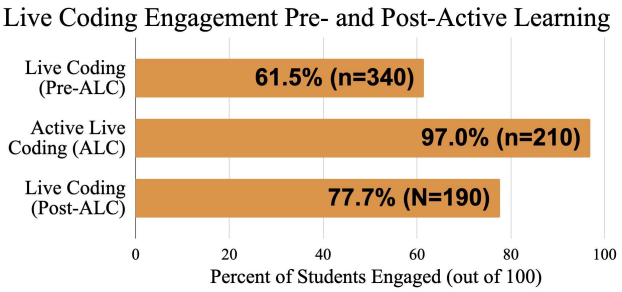

Figure 3: Student engagement during live coding before and after active live coding in the ALC lecture.

5.2 RQ2 Results

Our learning gain metric represents the proportion of students who got the Pre-Lecture Question incorrect but answered the corresponding Post-Lecture Question correctly. Since our learning gain metric is sensitive to the proportion of students in the Potential Learning Group (i.e., if there was only one student in the PLG, then that student answering the post-lecture question correctly would lead to a 100% learning gain), we did a preliminary analysis of the correctness of students on the Pre-Lecture Questions. The groups had similar rates of incorrectness throughout the term, with the ALC lecture having an average of 43.1% of students in the PLG and the TLC lecture having 40.8% of students in the PLG.

Table 5: Comparison of students' learning gain.

Lecture Condition	Num Questions	Learning Gain	z stat	p val	eff. size
ALC	2768	50.7%	-1.85	0.064	0.062
TLC	1323	53.7%			

Though we also conduct individual comparisons for the Pre- and Post-Lecture Questions on a question-by-question basis, we report the aggregate learning gain as a summary of the results. Table 5 shows that there is a 3 percentage point difference in the aggregate learning gain throughout the term. The "Num Questions" column represents the number of pairs of questions we analyzed during the quarter from the Potential Learning Group. This number represents only the questions According to our z-test of proportions [27], this difference is not statistically significant with an α threshold of 0.05 and our Cohen's h effect size for binary data [7] is low.

6 DISCUSSION

6.1 Interpretation of Results

Our data shows that active live coding not only re-engaged students, but also had a *persisting* impact on engagement (Figures 2 and 3). Students typically started the lectures with high engagement during the Pre-Lecture Questions but then slowly lost engagement over the next 40 minutes. Within the ALC Lecture, the engagement for traditional live coding was only 61.5% *before* the active learning component (Figure 3), but peaks at 97% *during* active live coding. The high engagement during active live coding is unsurprising: students were told that their attendance would be based on whether

they completed the coding activity and were instructed to discuss their code with a neighbor. Following this active live coding component, however, student engagement dropped from 97%, but still stayed at 77.7% during the traditional live coding components *after* active live coding. This different pre- and post-ALC engagement represents a 26.3% improvement to student engagement, a significant difference. In fact, the average overall engagement during traditional live doding in the ALC lecture is about 5 percentage points higher than the engagement during traditional live coding in the TLC lecture (68.9% vs 63.4%, Figure 1). This difference is roughly 1 out of every 20 students (5%) being engaged during live coding.

We reason that this persistent re-engagement occurs because the active coding component may provide students intrinsic motivation to watch the instructor live code. Since students complete the coding activity on their own before discussing with peers, we saw many students get stuck or realize that they are unsure how to approach a problem. We suspect that this moment of realization could motivate students to observe how the instructor approaches the problem. Drawing upon the theory of Cognitive Apprenticeship [6, 31], which is often cited in live coding research [28], students are able to *reflect* on their own approach by seeing the instructor's approach. In contrast, students in the TLC lecture only see the instructor's approach without having attempted the problem on their own, preventing the same reflection moment. Therefore, the students in the ALC lecture have a greater investment into the coding example that students in the TLC lecture do not have.

In general, our observations showed that traditional live coding engages between 60-70% of students at any given time, roughly similar engagement as handwritten notes on a projector. Based on our observations during traditional live coding, students tended to use their laptops for non-lecture purposes, such as doing homework for another course, browsing courses to take in the next quarter, or working on the programming assignment for this class (which still counts as behaviorally disengaged). Our observations build upon the results from Shah et al., who showed that students felt live coding was too fast and did not hold their attention [30]. Specifically, our results shed light on the risks of students being encouraged to use their laptop to follow along with the instructor's live coding lesson. In fact, we observed many students exhibiting blatant disengagement by playing games or watching videos on their laptop. Disengaged computer use was by far the most common cause of disengagement that we observed. In fact, our observers compiled a list of computer games that students were playing, including Chess, Wordle, Geoguessr, 2048, and more. Given the prevalence of disengaged computer use, an interesting follow-on study could evaluate the effect of a "no electronics during lecture" policy on students' engagement using classroom observations.

Despite the optimistic findings related to engagement during active live coding lectures, students' performance on Pre- and Post-Lecture Questions did not indicate a stronger learning impact of active live coding. The learning gain in the TLC lecture was three percentage points higher than in the ALC lecture (Table 5). These results do not seem to align with previous work that shows a link between engagement and achievement [2, 9, 15]. However these findings may actually highlight the impact of cognitive and emotional engagement. As Fredricks et al. point out in their work about school engagement, students can exhibit behavioral engagement, yet still be cognitively or emotionally disengaged (and vice versa) [9]. It is certainly possible that we observed students to be behaviorally engaged in the ALC based on their physical characteristics when they were, in fact, cognitively disengaged because they were thinking about something unrelated to the class, such as social plans or homework in a different course. These undetected forms of engagement could explain learning gain results we saw.

6.2 Threats to Validity

We encountered several threats to validity during our study due to the differences between the two lectures. For example, the ALC lecture was held at 9:30am while the TLC lecture was at 11am. This difference in lecture timing may have impacted some of our results, such as attendance and in-class engagement. Some students may have decided to simply not attend the 9:30am lecture due to its earlier start time. Further, the difference in the sizes of the lectures could have impacted students' in-lecture engagement. Based on our observers' and instructor's perceptions, the ALC lecture, which was double the size of the TLC lecture, seemed to be noisier than the TLC lecture. Physically, the lecture hall for the ALC lecture was much larger than the lecture hall for the TLC lecture. We suspect that proximity to the instructor can impact students' engagement (i.e., students further away could be more prone to disengagement). Lastly, our instructor reflected that the 11am lectures seemed to run more smoothly since the instructor had already taught the same content in the 9:30am lecture. The instructor reported feeling more prepared and encountered fewer technical issues in the 11am lecture. These important differences between the two lectures could certainly have impacted the engagement we observed.

Second, the Pre- and Post-Lecture Questions were all multiple choice questions that targeted students' conceptual understanding, code comprehension, or code completion (i.e., "fill-in-the-blank" questions) abilities. However, live coding primarily targets students' program generation and code writing abilities rather than conceptual knowledge and code comprehension (which is primarily targeted through static-code examples) [30]. Therefore, the content of the Pre- and Post Lecture Questions may not be the best representation of students' true learning gain via live coding.

7 CONCLUSION

Our study aimed to observe and measure students' behavioral engagement during Traditional Live Coding and an Active Live Coding lectures. Traditional live coding engages typically between 60-70% of students, with the lowest engagement occurring roughly halfway through an 80-minute lecture. Our results showed that active live coding *increases* students' engagement during the activity and had a persisting effect on engagement *after* the active component. However, this increased engagement did not translate to a higher learning gain during lectures, as the two groups demonstrated similar learning gain during lectures. Our work also relies on a promising classroom observation protocol that can be used for future studies to help instructors understand the effect of various lecture techniques and class policies on student engagement.

ACKNOWLEDGMENTS

This work was supported in part by NSF award 2044473.

REFERENCES

[1] Ashish Aggarwal and Akshay Ashok. 2022. Exploring the Differences in Students' Behavioral Engagement With Quizzes and Its Impact on their Performance in a Flipped CS1 Course. In *Proceedings of the 22nd Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '22)*. Association for Computing Machinery, New York, NY, USA, Article 24, 11 pages. https://doi.org/10.1145/3564721.3564740

[2] Ugur Akpur. 2021. Does Class Participation Predict Academic Achievement? A Mixed-Method Study. *English Language Teaching Educational Journal* 4, 2 (2021), 148–160.

[3] Neil C. C. Brown and Greg Wilson. 2018. Ten quick tips for teaching programming. *PLOS Computational Biology* 14, 4 (04 2018), 1–8. https://doi.org/10.1371/journal.pcbi.1006023

[4] Russel E. Bruhn and Philip J. Burton. 2003. An Approach to Teaching Java Using Computers. *SIGCSE Bull.* 35, 4 (Dec 2003), 94–99. https://doi.org/10.1145/960492.960537

[5] Donald Chinn, Judy Sheard, Angela Carbone, and Mikko-Jussi Laakso. 2010. Study habits of CS1 students: what do they do outside the classroom?. In *Proceedings of the Twelfth Australasian Conference on Computing Education - Volume 103* (Brisbane, Australia) *(ACE '10)*. Australian Computer Society, Inc., AUS, 53–62.

[6] Allan Collins, John Seely Brown, Ann Holum, et al. 1991. Cognitive apprenticeship: Making thinking visible. *American educator* 15, 3 (1991), 6–11.

[7] Rajarshi Dey and Madhuri S. Mulekar. 2018. *Effect Size as a Measure of Difference Between Two Populations.* Springer New York, New York, NY, 715–726. https://doi.org/10.1007/978-1-4939-7131-2_110195

[8] Edstem. 2023. *Edstem.* https://edstem.org/

[9] Jennifer A Fredricks, Phyllis C Blumenfeld, and Alison H Paris. 2004. School engagement: Potential of the concept, state of the evidence. *Review of educational research* 74, 1 (2004), 59–109.

[10] David C. Funder and Daniel J. Ozer. 2019. Evaluating Effect Size in Psychological Research: Sense and Nonsense. *Advances in Methods and Practices in Psychological Science* 2, 2 (2019), 156–168. https://doi.org/10.1177/2515245919847202

[11] Alessio Gaspar and Sarah Langevin. 2007. Active learning in introductory programming courses through student-led "live coding" and test-driven pair programming. In *International Conference on Education and Information Systems, Technologies and Applications, Orlando, FL.*

[12] Alessio Gaspar and Sarah Langevin. 2007. Restoring "Coding with Intention" in Introductory Programming Courses. In *Proceedings of the 8th ACM SIGITE Conference on Information Technology Education* (Destin, Florida, USA) *(SIGITE '07)*. Association for Computing Machinery, New York, NY, USA, 91–98. https://doi.org/10.1145/1324302.1324323

[13] Gradescope. 2024. *Gradescope.* https://www.gradescope.com/

[14] Tor-Morten Grønli and Siri Fagernes. 2020. The live programming lecturing technique: A study of the student experience in introductory and advanced programming courses. In *Norsk IKT-konferanse for forskning og utdanning.*

[15] Kristin E. Harbour, Lauren L. Evanovich, Chris A. Sweigart, and Lindsay E. Hughes. 2015. A Brief Review of Effective Teaching Practices That Maximize Student Engagement. *Preventing School Failure: Alternative Education for Children and Youth* 59 (2015), 13 – 5. https://api.semanticscholar.org/CorpusID:143101189

[16] Nien-Lin Hsueh, Bilegjargal Daramsenge, and Lien-Chi Lai. 2022. Exploring the Influence of Students' Modes of Behavioral Engagement in an Online Programming Course Using the Partial Least Squares Structural Equation Modeling Approach. *Journal of Information Technology Education: Research* 21 (2022), 403–423.

[17] Erin S. Lane and Sara E. Harris. 2015. A New Tool for Measuring Student Behavioral Engagement in Large University Classes. *Journal of College Science Teaching* 44, 6 (2015), 83–91. http://www.jstor.org/stable/43632000

[18] Soohyun Nam Liao, Daniel Zingaro, Michael A. Laurenzano, William G. Griswold, and Leo Porter. 2016. Lightweight, Early Identification of At-Risk CS1 Students. In *Proceedings of the 2016 ACM Conference on International Computing Education Research* (Melbourne, VIC, Australia) *(ICER '16)*. Association for Computing Machinery, New York, NY, USA, 123–131. https://doi.org/10.1145/2960310.2960315

[19] Claudia Ott, Anthony Robins, Patricia Haden, and Kerry Shephard. 2015. Illustrating performance indicators and course characteristics to support students' self-regulated learning in CS1. *Computer Science Education* 25 (04 2015), 174–198. https://doi.org/10.1080/08993408.2015.1033129

[20] John Paxton. 2002. Live Programming as a Lecture Technique. *J. Comput. Sci. Coll.* 18, 2 (dec 2002), 51–56.

[21] Leo Porter, Cynthia Bailey Lee, and Beth Simon. 2013. Halving Fail Rates Using Peer Instruction: A Study of Four Computer Science Courses. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 177–182. https://doi.org/10.1145/2445196.2445250

[22] Leo Porter, Cynthia Bailey Lee, Beth Simon, and Daniel Zingaro. 2011. Peer Instruction: Do Students Really Learn from Peer Discussion in Computing?. In *Proceedings of the Seventh International Workshop on Computing Education Research* (Providence, Rhode Island, USA) *(ICER '11)*. Association for Computing Machinery, New York, NY, USA, 45–52. https://doi.org/10.1145/2016911.2016923

[23] Harry O. Posten. 1984. *Robustness of the Two-Sample T-Test.* Springer Netherlands, Dordrecht, 92–99. https://doi.org/10.1007/978-94-009-6528-7_23

[24] Adalbert Gerald Soosai Raj, Pan Gu, Eda Zhang, Arokia Xavier Annie R, Jim Williams, Richard Halverson, and Jignesh M. Patel. 2020. Live-Coding vs Static Code Examples: Which is Better with Respect to Student Learning and Cognitive Load?. In *Proceedings of the Twenty-Second Australasian Computing Education Conference* (Melbourne, VIC, Australia) *(ACE'20)*. Association for Computing Machinery, New York, NY, USA, 152–159. https://doi.org/10.1145/3373165.3373182

[25] Adalbert Gerald Soosai Raj, Jignesh M. Patel, Richard Halverson, and Erica Rosenfeld Halverson. 2018. Role of Live-Coding in Learning Introductory Programming. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '18)*. Association for Computing Machinery, New York, NY, USA, Article 13, 8 pages. https://doi.org/10.1145/3279720.3279725

[26] Marc J. Rubin. 2013. The Effectiveness of Live-Coding to Teach Introductory Programming. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (Denver, Colorado, USA) *(SIGCSE '13)*. Association for Computing Machinery, New York, NY, USA, 651–656. https://doi.org/10.1145/2445196.2445388

[27] Randall E. Schumacker. 2023. *Learning Statistics Using R.* SAGE Publications, Inc., 55 City Road, London, Chapter 12. https://doi.org/10.4135/9781506300160

[28] Ana Selvaraj, Eda Zhang, Leo Porter, and Adalbert Gerald Soosai Raj. 2021. Live Coding: A Review of the Literature. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1* (Virtual Event, Germany) *(ITiCSE '21)*. Association for Computing Machinery, New York, NY, USA, 164–170. https://doi.org/10.1145/3430665.3456382

[29] Anshul Shah, Vardhan Agarwal, Michael Granado, John Driscoll, Emma Hogan, Leo Porter, William Griswold, and Adalbert Gerald Soosai Raj. 2023. The Impact of a Remote Live-Coding Pedagogy on Student Programming Processes, Grades, and Lecture Questions Asked. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1* (<conf-loc>, <city>Turku</city>, <country>Finland</country>, </conf-loc>) *(ITiCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 533–539. https://doi.org/10.1145/3587102.3588846

[30] Anshul Shah, Emma Hogan, Vardhan Agarwal, John Driscoll, Leo Porter, William G. Griswold, and Adalbert Gerald Soosai Raj. 2023. An Empirical Evaluation of Live Coding in CS1. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1* (Chicago, IL, USA) *(ICER '23)*. Association for Computing Machinery, New York, NY, USA, 476–494. https://doi.org/10.1145/3568813.3600122

[31] Anshul Shah and Adalbert Gerald Soosai Raj. 2024. A Review of Cognitive Apprenticeship Methods in Computing Education Research. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (<conf-loc>, <city>Portland</city>, <state>OR</state>, <country>USA</country>, </conf-loc>) *(SIGCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 1202–1208. https://doi.org/10.1145/3626252.3630769

[32] Ben Stephenson. 2019. Coding Demonstration Videos for CS1. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (Minneapolis, MN, USA) *(SIGCSE '19)*. Association for Computing Machinery, New York, NY, USA, 105–111. https://doi.org/10.1145/3287324.3287445

[33] Ashok Kumar Veerasamy, Daryl D'Souza, Rolf Lindén, Erkki Kaila, Mikko-Jussi Laakso, and Tapio Salakoski. 2016. The Impact of Lecture Attendance on Exams for Novice Programming Students. *International Journal of Modern Education and Computer Science* 8 (05 2016), 1–11. https://doi.org/10.5815/ijmecs.2016.05.01