An Electroencephalography Study on Cognitive Load in Visual and Textual Programming

Sverrir Thorgeirsson
sverrir.thorgeirsson@inf.ethz.ch
ETH Zürich
Zürich, Switzerland

Chengyu Zhang
chengyu.zhang@inf.ethz.ch
ETH Zürich
Zürich, Switzerland

Theo B. Weidmann
tweidmann@ethz.ch
ETH Zürich
Zürich, Switzerland

Karl-Heinz Weidmann
karl-heinz.weidmann@fhv.at
University of Applied Sciences Vorarlberg
Dornbirn, Austria

Zhendong Su
zhendong.su@inf.ethz.ch
ETH Zürich
Zürich, Switzerland

ABSTRACT

This paper presents a comparative study of Algot, a visual programming language designed to bridge the syntax-semantics gap via liveness and programming by demonstration, and the textual programming language Python. We conducted an experimental, within-subjects study with 24 undergraduate computer science students who performed recursion-based tasks in each language while their cognitive load was measured using an electroencephalogram and a validated survey instrument. The students received a brief introduction to Algot, but were all familiar with Python. The students performed significantly better when programming in Algot, but the cognitive load levels were similar according to both instruments. Our results provide evidence that within the domain that was tested, Algot can be quickly learned, and that students do not find it more cognitively demanding than working in a familiar language.

CCS CONCEPTS

• **Human-centered computing** → **Empirical studies in HCI**; • **Social and professional topics** → **Computer science education**; • **Software and its engineering** → **General programming languages**.

KEYWORDS

electroencephalography, undergraduate education, CS1, recursion, cognitive load, visual programming, programming by demonstration, direct manipulation, live programming

1 INTRODUCTION

In the essay "Learnable Programming" from 2012 [70], interaction designer Bret Victor made a compelling case that traditional programming environments are too disconnected from the way humans naturally understand and interact with the world, thereby impeding the learning process for novices by obscuring the mechanics of computation. By demonstrating how programming environments could provide immediate, tangible feedback and visualization of data and control flow, Victor argued that programming could become much more accessible and intuitive. Some have taken inspiration from Victor's essay to develop new tools and programming environments, for example the live IDE extension Theseus [35], the machine infrastructure for digital fabrication Imprimer [69], and our visual programming language Algot [67, 71], which is intended to bridge the syntax-semantics gap of programming via its liveness and its built-in implementation of programming by demonstration.

While the arguments that Victor puts forward are primarily supported by intuition, three recent experimental studies on Algot may provide a tentative empirical foundation behind the ideas expressed in the essay. One physiological study found that undergraduate students experienced lower cognitive load when solving simple programming tasks in Algot than in Python [68], another found that secondary school students demonstrated a significantly stronger grasp of recursion after programming in Algot than after programming with the Make-a-Block feature in Scratch [66], and the third found that undergraduate students understood existing CS1-level programs composed in Algot significantly better than equivalent Python programs [22]. A difference in cognitive mechanisms or mental models that students employ when solving the tasks in Algot could plausibly explain the performance improvements; by allowing students to compose or view programs in a way that resembles their execution, students may be able to focus better on the underlying algorithms they wish to implement or comprehend. With a deeper understanding of whether Algot improves performance or lowers cognitive load for more complex tasks, computer science educators might have a stronger reason to consider applying its underlying ideas when constructing other educational tools, materials or languages, such as liveness, programming by demonstration, direct manipulation, or the default visibility of the program state.

In this paper, we report on a comprehensive, within-subjects study conducted on 24 undergraduate computer science students who solved recursion-based programming exercises in Algot and Python while the electrical activity of the brain was measured with a 19-channel electroencephalogram (EEG). Electroencephalography, a well-established functional neuroimaging technique, records brain electrical activity and is widely used in cognitive neuroscience research, including the study of cognitive load [3]. Our objective with this study is twofold:

(1) We aim to expand on our comparative study published in 2024 [68], which measured undergraduates' performance in Algot and Python on two simple programming tasks and used eye-tracking and galvanic skin measurements to estimate their cognitive load. We wish to find whether

students also perform better and experience lower cognitive load in Algot when implementing more complex algorithms that require recursion, as opposed to the simple tasks used in the 2024 study. We also wish to address the discrepancy in some of the physiological measurements of the 2024 study by using an EEG, a tool that is considered to give an objective measure of task difficulty [31].

(2) By using two measures of cognitive load, (i) a survey sometimes used in computer science (CS) research (the Computer Science Cognitive Load Component Survey; CS CLCS) [40] and (ii) electroencephalogram measurements, we hope to investigate the convergent validity of the former. Convergent validity means assessing whether different measures that purport to test the same construct yield similar results in practice.

Our research questions are the following:

RQ1 Do students perform better when solving recursion-based programming exercises in Algot compared to Python, thus indicating that Algot might be more effective for learning complex programming concepts?

RQ2 Do the survey results and the EEG data support the hypothesis that students experience lower cognitive load when programming in Algot as opposed to Python, particularly for recursion-based tasks that are traditionally considered challenging for novices?

RQ3 Is there a significant correlation between the subjective measures of cognitive load (as assessed by the CS CLCS) and the measures obtained through EEG, thereby providing evidence of the convergent validity of the CS CLCS in the context of programming education research?

To answer the research questions, we used quantitative methods to analyze the data from the 24 participants. The participants were first-year CS students who were familiar with Python but not Algot. Despite the difference in their knowledge of the two programming languages, our hypothesis was that students would perform better and experience lower cognitive load when programming in Algot than in Python, which would mirror the results from our earlier cognitive load study on the same languages [68]. We also hypothesized that there would be a significant correlation between the two measures of cognitive load.

2 BACKGROUND

Cognitive load theory (CLT), which is rooted in research by John Sweller from the 1980s [42, 59], provides insights into how our brain processes and learns information. The theory suggests that our working memory has a limited capacity for handling information and this directly impacts our ability to learn new things. When the information load exceeds our cognitive capacity, learning efficiency drops, making it harder to absorb and retain new knowledge. CLT has been very influential in educational psychology and has shaped educational design [55], for example in rejecting traditional instructional techniques for "[not taking] into account the limitations of the human cognitive architecture" [54].

Sweller has outlined three distinct forms of cognitive load, that is intrinsic, extraneous, and germane load (see Fig. 1), which he compared in a 1998 paper [63]. Intrinsic load is tied to the fundamental complexity of the subject matter and varies according to the learner's existing understanding of the topic. Sweller identifies "element interactivity" as the underlying mechanism of intrinsic load [60, 61], in which an element is "anything that needs to be or has been learned" [61]. When elements are learned in isolation (for example when memorizing word translations that do not depend on each other), the interactivity is low and so is the intrinsic load, but the opposite is true when the elements are highly interactive, or in other words, depend on each other.

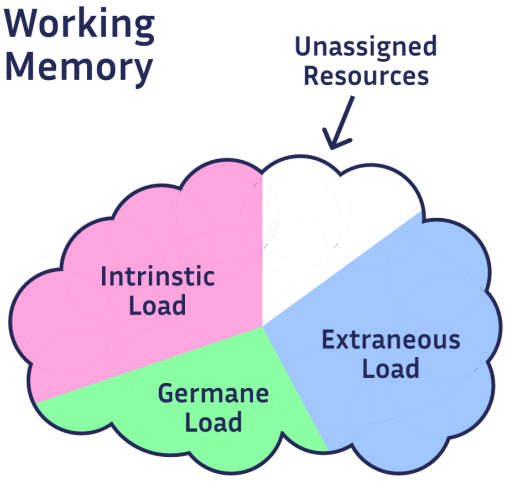

Figure 1: Our illustration of the three types of cognitive load according to Sweller's theory [63].

Sweller considers "schema acquisition" to be one of two critical learning mechanisms, with a schema defined as a "cognitive construct that organizes the elements of information according to the manner with which they will be dealt" [60]. Beckmann considers extraneous load as the result of "mental activities that do not directly contribute to schema acquisition and automation" [5]. This may be the result of presenting information that is not necessary for learning or by using poorly designed instructional designs. Sweller suggests that in case of learning designs with low element interactivity, high extraneous load may not significantly interfere with learning [60]. Additionally, previous research shows that the prior knowledge of students can decrease element interactivity [10], consequently decreasing the intrinsic cognitive load. In a 1998 paper [63], Sweller distinguishes between extraneous and germane cognitive load by considering the former as as reflection of "the effort required to process poorly designed instruction," while the latter "reflects the effort that contributes to the construction of schemas." Therefore, effective teaching strategies should focus on minimizing extraneous load while carefully balancing intrinsic load and enhancing germane load to promote effective learning.

Aspects of this model of cognitive load have been reconsidered in recent years. In response to, among else, the results of empirical studies, Sweller proposed a significant change in 2019, finding that germane cognitive load is not a contributor to total cognitive load, but rather contributes towards distributing extraneous working

memory resources elsewhere [62]. A review of cognitive load theory in CS education from 2022 [15] refers to the original as model "old CLT" and the revised model as "new CLT," noting that "germane load has been at the heart of many critiques" of the old version and that "new CLT" is closer to the original version of Sweller's theory from the 1980s than the 1998 version.

The three primary methods for measuring cognitive load are dual-task measures, questionnaires, and physiological measures [73]. Dual-task measures refer to the approach of asking participants to complete a secondary task along with the primary task. However, a disadvantage with this method is that performing the secondary task can significantly interfere with the performance on the primary task [44, 73]. Surveys are a more frequent measure; the Paas-scale [45] is a commonly used survey, and in CS education research, the Computer Science Cognitive Load Component Survey (CS CLCS) [40] by Morrison et al., an 11-item Likert-scale survey adapted from a validated assessment from 2013 [34], has also been used in some studies [18, 41]. Since both the CS CLCS and the assessment that it is based on predate Sweller's revised theory of CL, it measures three factors—intrinsic, extraneous and germane CL—and should be considered in the context of the "old CLT." A recent replication study by Zavgorodniaia et al. [75] found evidence for the internal reliability of the CS CLCS, meaning that the three factors it measures were internally consistent. A two-factor model measuring intrinsic and extraneous CL only, which had been tested successfully in the domain of language education [26], was found to be a worse fit for the data. However, as Zavgorodniaia et al. mention, their analysis does not address the construct validity of germane CL, which they consider an open question. Without construct validity, it is not clear whether the CS CLCS measures germane CL as defined by Sweller's original theory, some other dimension of cognitive load, or possibly something else.

Studies have identified a variety of physiological measurements which may be effective in measuring cognitive load due to their apparent convergent validity. For example, heart rate variability has been found to correlate with self-reported measures of cognitive load [57] and so has galvanic skin response [56]. Eye-tracking analysis is sometimes used [74], for example microsaccades and eye fixation [21] and pupil diameter analysis [30]; however, a disadvantage with this is that pupil diameter can be affected by many other factors than cognitive load, such as light conditions [23, 73].

Electroencephalography (EEG), which involves using an electrogram to measure the electrical activity of the brain, is also a well-established method in cognitive load research and is considered appropriate in educational psychology [3]. Unlike electrocorticography, which requires the surgical placement of electrodes on the brain [39], EEG is a non-invasive technique that only requires placing electrodes on the head using a cap designed for the purpose. EEG is considered an objective measure of the mental load of an activity [31], which is its primary advantage. Its limitations lie in its poor spatial resolution and susceptibility to motion artifacts such as blinking and moving and noise from electrical interference, breathing and heartbeat [3]. Some of those limitations can be addressed by the careful preprocessing of its output data.

Time-domain signals from the EEG are often transformed into the frequency domain to identify gamma, theta, alpha, beta and delta bands (see Fig. 2), which correspond to different brain states

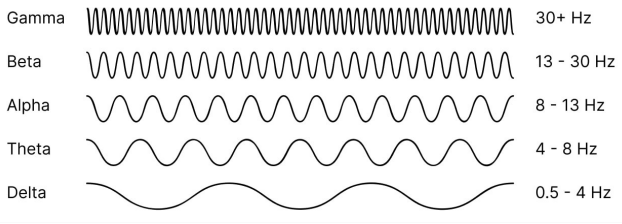

Figure 2: The five frequency bands which are typically considered in EEG research: gamma, beta, alpha, theta and delta.

and activities. Both the alpha and theta frequency power, meaning how much energy is transferred by the signal per unit of time, has for a long time been considered sensitive to cognitive load [27] and task difficulty is considered to increase with theta power and decrease with alpha power [19]; Holm et al. [25] proposed a index for "acute external and cumulative internal load" that they dubbed "brainbeat," which can be found by simply dividing the theta power with the alpha power. However, a 2022 review found that the theta channel alone is the best measure of cognitive load [11]. The association of the alpha channel with cognitive load is less straightforward according to the same review [11] and has been considered inconsistent by others [53], with some studies finding that power in the alpha band is lower in attention-demanding tasks [17] and others that increases in the alpha band corresponds to an increase in working memory load [43]. A 2014 study found indications that "increases in alpha activity are a behaviorally adaptive mechanism to meet the demands of increasing [working memory] load" [37]. A 2020 EEG study of 35 participants who were shown educational videos of varying linguistic complexity found that theta power is correlated with participants' self-reported intrinsic cognitive load [8]. The assessment instrument that was used for that study is the same one that the CS CLCS is based on.

EEG measurements have applications in education outside of cognitive load theory, for example to measure attention [28, 65] or emotional states [2]. A review from 2021 on EEG studies in education found that EEG signals are "mostly used to test students' concentration and meditation" and that most studies "lasted less than an hour, the sample sizes of EEG signals-based studies were limited, and the largest study group were university students" [4]. EEG has also been used for source localization, meaning the identification of regions within the brain where electrical activity originates; for example, a 1996 study assessed the role of the left and right brain hemisphere for solving arithmetic sign comprehension problems [16].

In CS education research, EEG equipment has not been used often, but some studies have been conducted; for example, a study from 2023 used light, portable EEG instruments on 36 students in a CS class to gauge their attention levels and found that they correlate with learning performance [64], and a 2019 study on nine participants found that the difficulty of programming problems could be accurately predicted using EEG data [46]. A 2016 EEG study on program comprehension with sixteen participants found a difference between novice and expert programmers in their brainwaves coming from some electrodes [33]. Some studies on

program comprehension have also looked at the difference frequency bands; a 2017 study found significant differences in alpha and theta bands when presenting participants with code samples that were meant to be confusing and non-confusing [72], and a 2018 study found that the alpha, beta and theta bands have high discriminatory power, allowing the identifications of code more difficult to understand [29]. Likewise, a 2021 study found that theta and beta bands might be useful for assessing the mental effort of different programming activities [38]. A 2022 study [47] on program comprehension combined EEG with eye-tracking to understand the relationship between efficacy and measures of programming experience. Among other results, the authors found that according to the brainbeat measure (theta power divided by alpha power), programmers with high efficacy experienced lower cognitive load, although the correlation was weak.

3 SYSTEM

Algot is a visual and graph-based educational programming language that is intended to bridge the gap between code syntax and its semantics [71]. Similar to block-based programming languages like Scratch, Algot limits the potential of syntax errors via its drag-and-drop interface, but unlike Scratch, Algot users do not have to manipulate or comprehend textual code in order to construct programs. Inspired by the essay "Learnable Programming" by interaction designer Bret Victor [70], Algot makes the program state visible at all times and allows programmers to interact with it using direct manipulation, meaning that in order to call a specific function, the programmer will select visually, in either order, the function itself and the specific parts of the state that they wish to use as function arguments.

In Algot, the program state is represented as a graph, a design decision made by its creators due to the modelling power of graphs [71]. The "state graph" is visible within the "state view" and consists of multiple connected components that can take the shape of trees, cyclic graphs, individual nodes, linked lists, or matrices. The nodes in these components can take numbers or short strings as values. By default, such as when creating a new node, their values are set to zero. By applying *base operations*, which are atomic operations that exist by default, the programmer can modify the values of the nodes or the structure of the graph. For example, in order to increment the value of a node containing a number, the user first clicks on the `Increment` base operation (which is shown in the right sidebar of the application) and then clicks the node in the state graph that should be incremented. Doing so will have an immediate, live visual effect, meaning that the programmer will see the value change in the visual representation of the state graph as soon as the operation has executed.

The Algot programming paradigm can be considered a form of "programming by demonstration," a common mode of programming in robotics [6]. To construct new programs, the programmer enters a three-layered demonstration view in which input nodes are first specified (including "pattern-matching" nodes that are a part of the same connected components as the input), followed by calling operations on the inputs in the same order that they should be executed. These operations can be called conditionally depending on the results of "queries," which are natural-language yes-or-no statements about the structure or values of specific nodes (e.g., whether two nodes share the same value). These queries are displayed as small panel windows in the interface and can be viewed at all times. This way, the programmer can ensure that an operation is only executed if a given statement is true or false, which are options that can be selected in the query panel. It is possible to use multiple, separate queries together, in which case an operation is only executed if every queries applies (e.g., if a node `a` has a smaller value than node `b`, *and* a node `c` does not have the value zero).

To execute the same operation repeatedly, the programmer is expected to use recursion, which, like in textual languages, means that they must call the same operation that they are currently defining. For instance, Fig. 3 shows that the programmer could apply the operation `Count Nodes`, which is currently being defined, on any two of the visible nodes, just like it would be possible to apply any other operation.

To illustrate how these ideas come together, consider an operation in Algot that populates a linked list with a specified number of Fibonacci numbers. A possible way to do this would be the following:

(a) Define the operation `Fib` with two input nodes that represent the linked list of Fibonacci numbers (`k`) and an index (`idx`) that determines how many Fibonacci numbers should be added. The node `k` is expected to have the value `1` and should have a parent with the value `0`, since those are the two first Fibonacci numbers. After this, three nodes are now visible in the demonstration view: `idx`, `k`, and `k`'s parent node.
(b) Apply the query `Is Zero?` on the node `idx` to determine branching paths within the operation's logic. In this case, only the negative case matters, so the *false* case in the query window should be selected.
(c) Append a new node to `k` using the base operation `Add Child`, and then call `Sum` on `k`'s parent, `k`, and the newly created node. This will find the sum of the new node's two preceding values and store in the new node.
(d) Still with the negative case of `Is Zero?` on `idx` in effect, call the base operation `Decrement` on `idx` and call `Fib` recursively on the newly created node and `idx`. If the value of `idx` is zero after its value was decremented, this recursive call will not execute.

One difference between computing the above in Algot and another language such as Python is that the effects of each step is immediately visible. For instance, the programmer will see in the visual representation of the program state whether a new child was created after the `Add Child` operation is selected, which can potentially help the programmer better understand the meaning and effect of what they are doing and whether they made a mistake. For a more comprehensive treatment of Algot's semantics, we refer to our 2022 paper [71].

Algot has been the subject of three earlier empirical studies. One is a controlled, experimental, between-subjects study in which secondary school students were asked to compose recursive programs using Algot and the Make-a-Block feature in Scratch, followed by taking a post-test assessment on their conceptual and mechanical understanding of recursion [66]. Although neither group excelled

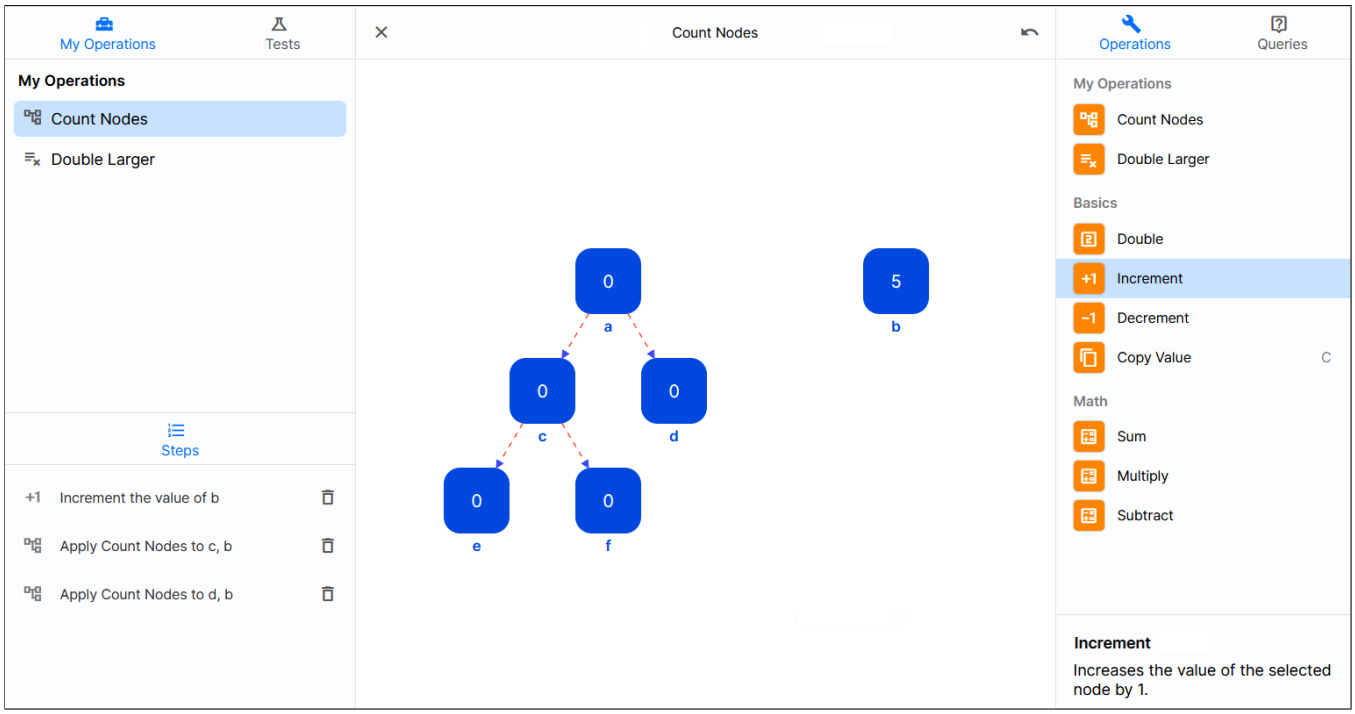

Figure 3: A screenshot showing the Algot environment used in the study, slightly edited to reduce whitespace.

at the programming tasks nor the assessment, the students using Algot performed significantly better than the Scratch group at both the tasks and the post-test. Another study, which is similar to ours, found that undergraduate students performed significantly better at solving programming tasks in Algot than in Python and exhibited lower cognitive load according to the post-test CS CLCS survey and according to some of the physiological measures used (skin conductance and eye-tracking) [68]. However, the tasks used were simple, featuring only simple constructs such as functions and variable assignment. Lastly, one study on program comprehension featuring typical CS1 programs from a CS1 problem repository found that students displayed a better understanding of most existing programs in Algot than Python, in particular those featuring trees and matrices [22].

4 METHODS

Our study was conducted with first-year computer science students at the University of Applied Sciences Vorarlberg, Austria, who were all recruited from the course User-Centered Technologies. The students were familiar with programming in Python from other courses in their degree program and had been taught introductory algorithms, data structures, and the problem-solving strategy of recursion.

After signing consent forms, participants were asked to answer two demographic questions about themselves (age and gender in a free-form question) and given a pretest (Section 4.2). They were also asked if they had worked with Algot before and were asked to self-assess their experience with Python on a 10-point scale. Then they were given an approximately 10-minute tutorial on Algot and the Python environment used for the study and a refresher on recursion and the notion of graphs and trees (Section 4.3). Following this, students solved recursion-related problems in Algot and Python under a within-subjects design with counterbalancing, in which two equally large and randomly chosen groups programmed in Algot first and the other in Python (Section 4.4), while their cognitive load was measured with an electroencephalogram (EEG; Section 4.5). Students were also recorded to helped identify possible artifacts during the EEG data analysis. Students were given 15 minutes for each set of problems, which amounted to 30 minutes in total. Finally, they were asked to fill out the Computer Science Cognitive Load Component Survey (CS CLCS) [40] about the Algot-based tasks and the Python-based tasks on their perceived cognitive load during the study (Section 4.5). While this survey is based on an older theory of cognitive load, it was chosen for two reasons: to make the study results more comparable with the results from a recent cognitive load study on Algot and Python in which the CS CLCS was chosen [68], and since instruments based on the more recent version of the theory have not yet been widely adapted and tested, especially in CS education.

We used quantitative methods to analyze the data. We applied classical paired-samples t-tests to help find whether we could reject the null hypothesis and accept the alternative hypothesis, i.e., that programming in Algot induces lower cognitive load. We did the same to determine if there was a statistical difference in performance. We also calculated the correlation with the self-reported and EEG-based measurements using Spearman's ρ. All calculations were conducted in the statistical software JASP [36] (v. 0.18.1).

Participants were compensated with the equivalent of approximately 30 USD in the local currency and with a small bonus on the final exam, which was considered acceptable by the local ethics committee due to the educational value of the study and the connection to the intended course outcomes.

4.1 Ethics

The ethics application was an important process due to the nature of the data that we intended to collect from the study participants. We received ethics approval for our research from the ethics committee at the institution in which the study took place. Standard ethical practices were followed, such as allowing students to withdraw from the study at any time, and to keep the data secure with no access to third parties. Furthermore, students were allowed to participate in the study without using the EEG equipment, without specifying a reason, and still receive full compensation for their participation. We also followed the stipulation made by the ethics committee that the teacher of the course from which the students were recruited was not allowed to participate in the data collection; instead, only researchers with no relationship to the students did so.

4.2 Pretest

We administered a pretest with two questions from the recent Nebraska Assessment of Computing Knowledge (NACK; see Fig. 4), chosen because the assessment is publicly accessible and has recently been validated on a large number ($N = 1318$) of undergraduate computer students using item response theory [48]. Since students were only expected to spend about an hour in total on the entire experiment, we could not administer the entire assessment.

The purpose of the pretest was twofold. First, we wished to gain a better understanding of the computing knowledge background of the participants, which is important since low scores would imply that the participants lack foundational knowledge, compromising the validity of the study. Second, we wished to improve the generalizability of our findings; in case of follow-up studies, researchers can compare their pretest results with ours to help find if differences in computing knowledge among samples may explain variations in outcomes.

4.3 Tutorial

We gave the participants a short general introduction to Algot, including the state view and how to apply base operations on nodes. We then showed them how to compute the following operations in Algot:

(1) Increase the value of an input node by two.
(2) Increment the value of the child of an input node.
(3) Conditionally increment the value of an input node if it has the same value as another input node.
(4) Increment all the values in a list using a recursive function.

We also showed participants how to use the online Python IDE, including how to test their solution on example inputs and against the hidden test cases that were used to evaluate their solution. The order they were given the Algot and Python parts of the tutorial was the same as the order they were assigned for solving the tasks. Students were also shown how to solve a specific recursive problem in both Algot and Python, namely how to count the number of nodes in a tree that are larger than a given value (see the Python example in Fig. 5). This problem resembled the ones they were asked to solve in the study.

4.4 Tasks

The participants were asked to solve two tasks in Algot and Python:

(1) Compute the number of nodes in a binary tree (and in Algot, to store it in one of the input nodes; see a screenshot in Fig. 3).
(2) Double the values of all nodes in a linked list that are larger than a given value.

The tasks were chosen since they tested two different types of data structures (trees and linked lists), required recursion, and we considered it realistic to solve them in 15 minutes in each language, given the prior knowledge of the students. The participants were provided with a helper sheet explaining the terminology (trees, binary trees, nodes, root nodes, and linked lists). Similar to the 2024 study [68], students were provided with a list of functions in Python they could use to solve the task such as `get_value`, `get_children`, `copy_value(a,b)` and `has_outgoing(a)`, and similar base operations and base queries in Algot.

4.5 Electroencephalogram

The EEG equipment used for the study was Neurofax EEG-1200, a 10-20 system manufactured by Nihon Kohden. We used a sampling frequency of 200 Hz and all nineteen provided channels (see Fig. 6). To analyze the data, we used EEGLAB [7], a MATLAB toolbox designed for electrophysiological research that is widely used by EEG researchers [12]. To process the data files from the NeuroFax device, we used EEGLAB's NihonKoden extension released by Makoto Miyakoshi [1]. We began by applying a high-pass filter set at 1 Hz, following the procedure recommended by the EEGLAB developers. High-pass filters on EEG data have been shown to improve performance [13] and although they have a small distorting effect, they are a common pre-processing step as they help reduce noise [58]. Following this, we used *Clean Rawdata*,[1] an EEGLAB extension used by many authors [14, 20, 52], to process the data in the following way, using default settings whenever possible:

(1) We removed channels with low activity (flat for more than 5 seconds), contained much noise (higher standard deviation than 4), or were poorly correlated with other channels (lower correlation than 0.8).
(2) We removed data using Artifact Subspace Reconstruction (ASR) [9], a data artifact correction method that has become increasingly popular in the EEG community [49].
(3) Lastly, we rejected bad portions of data where a set percentage of channels (25%) passed a standard deviation threshold. The threshold window was set to the predefined value of "-Inf, 7" standard deviations.

After the preprocessing of the data, we applied power spectral analysis to compare the cognitive load of the participants while

[1] See https://eeglab.org/others/EEGLAB_Extensions.html or the GitHub repository https://github.com/sccn/clean_rawdata.

1. Given the following code snippet, what are the values of `persons[0]` and `persons[1]`?

```
int persons[10];
persons[0]=1;
persons[1]=2;
int temp=3;
temp=persons[0];
persons[0]=persons[1];
persons[1]=temp;
```

2. Why are algorithms necessary in computational problem solving?

I. The concept of algorithm can be used to define the notion of decidability – whether an outcome can be achieved by following a set of steps.
II. An algorithm is a blue-print for the actual implementation of a solution, enabling the conversion of a conceptual solution to a program.
III. Expressing solutions in algorithms allow us to solve problems without having to deal with programming details that might be specific to a particular programming language.
IV. Algorithms are needed for programs to compile.

Figure 4: Questions 5 and 8 from the Nebraska Assessment of Computing Knowledge [48], which were used on the pretest ahead of the study. Each question came with five and four multiple choice options, respectively; see the original 2020 paper for more information.

```
# Root is a node in a binary tree. b is a number
def count_larger(root, b):
    count = 0
    if get_value(root) > b:
        count = 1

    for child in get_children(root):
        count += count_larger(child, b)

    return count
```

Figure 5: Example of recursive code in Python that was shown in the tutorial to students. The functions `get_value` and `get_children` were provided and explained.

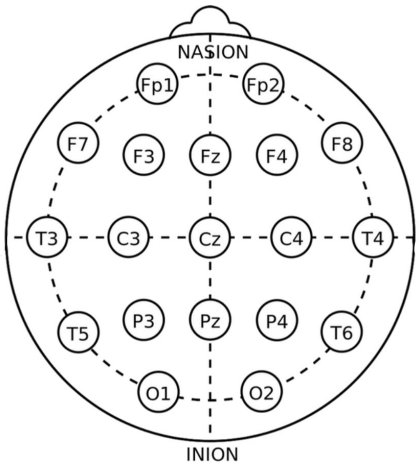

Figure 6: The standard 10-20 EEG model, showing all 19 channels.

programming in Algot and Python. We decomposed the time domain signals into the frequency domain over two bands: alpha (8-13 Hz) and theta (4-8 Hz), and then calculated the absolute power of the bands using the unit $\frac{\mu V^2}{Hz}$, which is suitable for describing the power distribution over the frequency of a signal that has small voltage levels such as those found in brainwave activity. The code we used to find the power values can be found in the Swartz Center for Computational Neuroscience's wiki page "Makoto's useful EEGLAB code" along with a description of the above unit.[2]

We also compared the EEG data with the results from the CS CLCS. We are not familiar with a widely-accepted procedure for using EEG to distinguish between the types of cognitive load in either Sweller's original theory or Sweller's revised 2019 theory of cognitive load. As a simplification, we took the average values of the extraneous and intrinsic components and compared it against the theta value, which we consider the most reliable form of cognitive load according to Chiki et al.'s recent meta-review [11].

A sketch showing a part of the study setup can be seen in Fig. 7.

5 RESULTS

32 students registered for the study, of which one did not make an appearance. 24 participated in the main EEG component of the study. Each participant was tested at a time for approximately one hour. The study spanned three days in total with about ten participants attending each day.

One student was randomly selected to test the study materials without EEG monitoring a day before the study began. This was to lower the risk of software failure and to help identify errors. Five additional other participants tested the equipment in a pilot setting on the first day with limited tutoring and with only five active EEG channels, which helped us refine the experimental setup, ensure the equipment was working properly, and help those working with the data collection familiarize themselves with the equipment, to ensure the comfort and safety of all study participants and staff.

[2]See https://sccn.ucsd.edu/wiki/Makoto%27s_useful_EEGLAB_code#How_to_extract_EEG_power_of_frequency_bands_.2806.2F06.2F2020_updated.29.

Figure 7: A sketch showing a part of the testing environment used in the study, showing the EEG cap in red and a video camera attached to the monitor. Not drawn are the electrode leads extending from the top of the cap onto the EEG recording system and the keyboard used in the study below the monitor.

Of the other remaining 25 students who participated, one chose not to use the EEG equipment. The other 24 participants were monitored with the EEG equipment using all nineteen channels. The median age of the participants was 22 ($\mu = 23.0, \sigma = 3.8$). 23 were male and one was female. This section contains a summary of their results. The raw data reported here is fully included in Appendix A.

70.8% and 75.0% of the participants answered questions 1 and 2 correctly on the pretest, respectively (see Fig. 4). This is substantially higher than the student score on the study in which the NACK was validated (58% and 62%, respectively) [48]. The mean value for the participants' Python self-assessed knowledge was 5.2 ($\sigma = 1.2$, min 3.0, max 7.0) on a 1-to-10 scale. No participant had worked with Algot before.

5.1 Task performance

The performance on the Algot and Python tasks of the study can be seen in Table 1. The table shows the number of tasks that students solved completely correctly, i.e., passed all hidden test cases, which we ensured sufficed for correctness. We note that although we did not do a complete taxonomy of the incorrect solutions from students, many of them made substantial progress even when they were not able to completely solve the tasks in the 15 minutes given.

We computed classical paired samples t-test to determine if there is a statistical significance in the mean performance. We used the Shapiro–Wilk normality test to determine if we should use the Student's t-test or non-parametric Wilcoxon signed-rank t-test (a widely accepted procedure for both paired and independent samples [51]), and finding significant results (suggesting deviations from normality), we applied the Wilcoxon's t-test. The result was $p = 0.033 < 0.05$, suggesting we could reject the null hypothesis. We also computed the effect size using Cohen's d for repeated measures (Cohen's d_{rm}), finding the value 0.408 (SE = 0.214), indicating that the mean student programming the tasks in Algot performs approximately 0.4 standard deviations above the mean student programming in Python.

	Algot Performance	Python Performance
Mean	0.708	0.375
Std. Deviation	0.908	0.647

Table 1: Descriptive statistics showing how many of the two tasks students solved completely correctly (i.e., passed all test cases) on the Algot and Python versions of the tasks. The minimum and maximum possible values are 0.0 and 2.0.

We observed that many students made two types of mistakes when programming in Algot:

- For the first task, on counting the number of nodes in a binary tree, the students attempted to use the query *Has Outgoing?* on a given node to determine whether or not to proceed with the operation logic, resulting in programs that only count the non-leaves of a tree.
- The students struggled with including all arguments in the recursive call, perhaps confusing the intended "counter" node with a global variable.

The errors students made in Python were more varied, with many students demonstrating the recursion-related errors identified by Hamouda et al. [24] such as neglecting to define a base case or not being able to form a recursive call that reaches the base case. Unresolved syntax errors also occurred, but we did not observe syntax errors causing otherwise correct programs to fail.

5.2 Electroencephalogram

Preprocessing the EEG data indicated that the electrode signal was mostly useful; out of the 24 participants, there were only four instances when a channel (or in one case, two channels) had to be removed from the data analysis due to a poor correlation with the other channels (see the procedure in the method section). We find it unlikely that this had a meaningful effect on the results.

The descriptive statistics of the mean power levels in the theta and alpha bands for the Algot and Python epochs are shown in Table 2. The results were inconclusive; on average, the theta values were very close and slightly, but not meaningfully, higher for Python than Algot (Cohen's $d_{rm} = 0.090$). We consider those the most reliable indication of cognitive load. The alpha values were also very similar but slightly higher for Algot (Cohen's $d_{rm} = 0.292$). For completeness, we also completed the mean "brainbeat" value–the theta power divided with the alpha power–which were also similar but slightly higher for Python (Cohen's $d_{rm} = 0.342$) which would indicate a higher cognitive load in Python. Wilcoxon's signed-rank t-test (chosen for the same reason as before) gave a p-value of $0.08 > 0.05$, meaning the null hypothesis could not be rejected.

5.3 Cognitive Load Survey

We calculated the mean scores on the Computer Science Cognitive Load Component Survey (CS CLCS) and found that the scores were similar across the intrinsic, extraneous and germane cognitive load (CL). The results are reported in Table 3. On the intrinsic and extraneous CL (undesirable CL), Algot scored lightly lower (Cohen's d_{rm} was 0.138 and 0.135, respectively), and on the germane CL (more

	Theta (A)	Theta (P)	Alpha (A)	Alpha (P)	Brainbeat (A)	Brainbeat (P)
Mean	0.0347	0.0327	0.0150	0.0131	2.37	2.61
Std. Deviation	0.0180	0.0232	0.0079	0.0103	0.51	0.87
Minimum	0.0136	0.0119	0.0065	0.0070	1.50	1.24
Maximum	0.0826	0.1166	0.0382	0.0585	3.17	5.50

Table 2: Descriptive statistics of the powers of the alpha and theta bands when programming in Algot (A) and Python (P).

desirable CL), Algot scored slightly higher (Cohen's $d_{rm} = 0.144$). However, the effect size is low for each component comparison.

We computed a paired samples t-test to find if the difference is statistically significant. We first computed a normality check and found that the intrinsic values passed ($p = 0.089$) but that the extraneous and germane ones did not ($p = 0.002$ and $p = 0.0024$), but for the sake of consistency, we applied the Wilcoxon signed-rank t-test on all. We found that all the p-values were clearly above 0.05 (0.2011, 0.5094 and 0.3134, respectively), so the null hypothesis could not be rejected.

	Intrinsic	Extraneous	Germane
Mean (A)	4.1528	1.7639	4.4896
Mean (P)	4.3750	2.0139	4.2813
Std. Dev. (A)	1.9609	1.7207	2.3619
Std. Dev. (P)	1.7481	2.0276	1.8450

Table 3: Descriptive statistics from the CS CLCS for Algot (A) and Python (P).

5.4 Correlations

For comparing the data, we chose Spearman's ρ as our correlation coefficient since it makes no assumption of normality or linearity and is not sensitive to outliers. For the Python component of the study, we found a negative correlation between the theta power values and the average of the intrinsic and extraneous component scores on the CS CLCS: Spearman's $\rho = -0.532, p = 0.0075$. For Algot, there was a much weaker and less statistically significant relationship: Spearman's $\rho = 0.1372, p = 0.5226$. We found no meaningful relationship between the mean alpha power values and the survey scores.

There was a weak positive but not statistically significant correlation between the student's self-assessment of their Python proficiency and their performance on the Python tasks (Spearman's $\rho = 0.243, p = 0.253$). The pretest performance was positively correlated to both the performance on the Algot tasks (Spearman's $\rho = 0.355, p = 0.089$) and the Python tasks (Spearman's $\rho = 0.377, p = 0.070$).

5.5 Free-form feedback

We invited students to leave comments or feedback about their study experience. We received twelve comments from the students in the main part of the study and two additional comments from the students in the pilot group. Since the comments were few and brief, we did not find it meaningful to conduct a qualitative analysis. Three students reported that they found the tasks difficult and two additional students noted that they would have liked more time to solve the tasks. Three students left positive comments about Algot, e.g., "It was a great opportunity to take part in such a study and see how all of that works. Also it was a great topic and I hope the idea of programming with Algot gets developed more because it is very intuitive to work like that and see what you are doing. Thank you!". Other comments suggested improvements in the study environments, e.g., to split up the provided Python code into two files, and to use an additional color for some of the Algot nodes.

6 DISCUSSION

In our study, we sought to answer three research questions. We found the answers reported below.

RQ1 We found that students did perform statistically significantly better on the study tasks when presented in Algot instead of Python, despite the participants' lack of familiarity with Algot but prior exposure to Python. This is aligned with the results from the 2024 study which compared students' performance in Algot and Python when solving simple tasks, which reported the same results direction [68]. However, the difference is less pronounced in this study, which may reflect the increased difficulty of the tasks; as the tasks become more challenging, it is likely that students require more exposure to Algot in order to solve them successfully. It is possible that the cognitive load demands of working in a new language increase as the tasks become more difficult and unfamiliar–possibly due to higher element interactivity and the lack of pre-existing schemas to efficiently process and integrate new information–and that at the same time, the cognitive load of working in a familiar language does not increase as much when the tasks become more complex.

RQ2 The students' cognitive load appeared similar across both tasks according to both the EEG analysis and the survey results (CS CLCS). This is in contrast to the results from the 2024 study. As research suggests that element interactivity decreases with higher prior knowledge [10], the students might have experienced lower cognitive load had they been more familiar with Algot, but this should be corroborated in future research. To help the reader interpret our EEG results here, we note that some other EEG studies have also found that both alpha and theta power can increase when comparing one task or condition to another one [32, 50], although in our case, the increase was not significant.

RQ3 With our data, we were not able to find convincing evidence for a correlation between the EEG data and components

of cognitive load according to the CS CLCS survey. While there was a statistically significant negative correlation between theta power values for the Python tasks and the self-reported mean cognitive load on the intrinsic and extraneous components, there was no such correlation for the Algot tasks. While it is possible that the theta band more accurately reflects cognitive load for programming in Python, we find it more likely that the results occurred by chance.

Overall, considering the statistically significant difference in performance, we think that the results provide some evidence that Algot can be comparatively helpful for beginners in composing non-trivial programs. However, despite the difference in performance, the cognitive load levels appeared to be similar according to both the survey and the EEG results. It is important to note that the participants had very different exposure to the two languages; on one hand, they were familiar with Python, but had only been given a brief tutorial on Algot. Although the cognitive load levels were similar despite this imbalance, we do not think it necessarily means that cognitive load levels would have been lower in Algot if the participants had been equally familiar with the two languages. However, we think this is a hypothesis worth exploring in future studies.

We believe it is plausible that the performance results can be at least partially explained by the principles put forward in Victor's *Learnable Programming* essay [70] on, for example, keeping the program state visible by default. However, we note that while Algot was inspired by the essay, Victor describes visual extensions to textual programming rather than purely visual programming. Future work on programming language design for CS education might explore on a more granular level how such proposed features impact learning.

We think that this study may encourage educators or researchers to consider alternative programming paradigms, such as those implemented in Algot, in CS educational settings, particularly for novices, and calls for longitudinal studies to explore whether languages like Algot have long-term benefits. Future work could also explore whether and to what extent skills or knowledge gained while programming in Algot transfer onto more mature, textual languages. We also believe that the functional neuroimaging technique used in the study, the EEG, has been underexplored in CS education research. While we compared very different approaches to programming in our study, future experiments could investigate more targeted approaches or interventions, or compare how different groups of learners solve programming assignments (such as experts and novices). While we found the data collection with our device to be time-consuming (as it spanned several days), the advent of portable and inexpensive EEG devices has made it economically feasible to test many students at once, making studies of this type more accessible and scalable.

6.1 Limitations

Our study has several important limitations:

- The study environment may have had a negative impact on the ecological validity of the study. Students do not typically write programs while wearing EEG headsets and using unfamiliar hardware (keyboard, mouse and computer) and software. If students had solved the assignments on their own computers in the comfort of their own home, it is likely that their performance would have been better. We note, though, that students did not appear to feel uncomfortable using the EEG equipment and did not raise issues about the hardware or the physical environment during the study or in the free-form feedback.
- We believe that the lopsided gender ratio has a negative impact on the external validity of the study. Ideally, more female students would have participated.
- As noted in the discussion section, there is different interpretations of how EEG data should be used to determine cognitive load, in particular when it comes to the alpha band. If the scientific consensus on this changes, it is possible that the data could be interpreted differently.
- As we also noted, the study was imbalanced since the participants were familiar with Python but had no prior exposure to Algot. The difference in performance may possibly have been even more pronounced had the participants been on equal footing in both languages, but we cannot attest that based on our existing data, so it should be corroborated in future studies. In a future work, researchers may also wish to investigate whether this would affect the cognitive load levels.
- The survey we used to measure cognitive load [40] is based on the "old cognitive load theory," which is founded on a view of germane cognitive load that has been reinterpreted under the "new cognitive load theory" from 2019 [15, 62]. If germane cognitive load, as understood under the older version of the theory, has a poor construct validity, this could have a negative impact on the validity of the study. Although within the domain of computer programming, confirmatory factor analysis on a similar assessment based on the new theory did not appear to be a good fit [75], it is possible that other such instruments may prove in the future to be more appropriate than the instrument we used.

We note that the results are necessarily limited by the choice of tasks. We believe that the task choice was reasonable in the context of our study, but other tasks involving different algorithms or computing concepts would likely have impacted the performance and cognitive load of the study participants. We also note Algot or Python are distinct in many ways, so it is difficult to attribute the difference in performance to any specific points of difference between the two of them, be it Algot's visible program state, the support for direct manipulation, or some other features in either language. While we did not observe syntax errors being the sole error in any Python solutions, it is possible that they contributed towards some students failing to arrive at a correct solution.

7 CONCLUSION

We found evidence that undergraduate students programming in the visual programming language Algot perform better at solving recursion-based tasks in Algot than Python, even though the students were familiar with Python but had only received a short tutorial about Algot as a part of the experimental protocol. We did,

however, not find evidence that students experience lower cognitive load while programming the tasks in Algot. We did not find strong evidence that the two measures of cognitive load that we used, a survey and an electroencephalogram, are correlated; however, both measures indicated that the cognitive load level among the students was similar. Our study contributes to the growing body of research on visual programming languages and their impact on learning and finds some evidence that Algot can be useful resource in undergraduate computer science education.

8 ACKNOWLEDGEMENTS

We thank Sandra Wiklander for many of the figures used in the paper, Walter Ritter for helping us with the study setup, and the Human-Centred Technologies Research Centre at the University of Applied Sciences Vorarlberg for allowing us to use their laboratory. We also acknowledge the support of Interreg project Algot ABH020 in carrying out this project.

REFERENCES

[1] 2024. List of plug-ins available for download in EEGLAB 2019.1 and later versions. https://sccn.ucsd.edu/eeglab/plugin_uploader/plugin_list_all.php. Accessed: 2024-03-05.

[2] Omar AlShorman, Mahmoud Masadeh, Abdelhadi Alzyoud, Md Belal Bin Heyat, Faijan Akhtar, et al. 2020. The effects of emotional stress on learning and memory cognitive functions: an EEG review study in education. In *2020 Sixth International Conference on e-Learning (econf)*. IEEE, 177–182.

[3] Pavlo Antonenko, Fred Paas, Roland Grabner, and Tamara Van Gog. 2010. Using electroencephalography to measure cognitive load. *Educational psychology review* 22 (2010), 425–438.

[4] Fatima Bashir, Ahmed Ali, Toufique Akbar Soomro, M Marouf, M Bilal, and Bhawani Shankar Chowdhry. 2021. Electroencephalogram (EEG) Signals for Modern Educational Research. In *Innovative Education Technologies for 21st Century Teaching and Learning*. CRC Press, 149–171.

[5] Jens F Beckmann. 2010. Taming a beast of burden–On some issues with the conceptualisation and operationalisation of cognitive load. *Learning and instruction* 20, 3 (2010), 250–264.

[6] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. 2008. Survey: Robot programming by demonstration. *Springer handbook of robotics* (2008), 1371–1394.

[7] Clemens Brunner, Arnaud Delorme, and Scott Makeig. 2013. Eeglab–an open source Matlab toolbox for electrophysiological research. *Biomedical Engineering/Biomedizinische Technik* 58, SI-1-Track-G (2013), 000010151520134182.

[8] Leidy J Castro-Meneses, Jan-Louis Kruger, and Stephen Doherty. 2020. Validating theta power as an objective measure of cognitive load in educational video. *Educational Technology Research and Development* 68, 1 (2020), 181–202.

[9] Chi-Yuan Chang, Sheng-Hsiou Hsu, Luca Pion-Tonachini, and Tzyy-Ping Jung. 2018. Evaluation of artifact subspace reconstruction for automatic EEG artifact removal. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 1242–1245.

[10] Ouhao Chen, Slava Kalyuga, and John Sweller. 2017. The expertise reversal effect is a variant of the more general element interactivity effect. *Educational Psychology Review* 29 (2017), 393–405.

[11] Samy Chikhi, Nadine Matton, and Sophie Blanchet. 2022. EEG power spectral measures of cognitive workload: A meta-analysis. *Psychophysiology* 59, 6 (2022), e14009.

[12] Rupak Kumar Das, Anna Martin, Tom Zurales, Dale Dowling, and Arshia Khan. 2023. A survey on EEG data analysis software. *Sci* 5, 2 (2023), 23.

[13] Arnaud Delorme. 2023. EEG is better left alone. *Scientific Reports* 13, 1 (2023), 2372. https://doi.org/10.1038/s41598-023-27528-0

[14] Arnaud Delorme and Jeffery A Martin. 2021. Automated data cleaning for the Muse EEG. In *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 1–5.

[15] Rodrigo Duran, Albina Zavgorodniaia, and Juha Sorva. 2022. Cognitive Load Theory in Computing Education Research: A Review. *ACM Transactions on Computing Education (TOCE)* 22, 4 (2022), 1–27.

[16] Jonathan BB Earle, Patricia Garcia-Dergay, Anthony Manniello, and Christopher Dowd. 1996. Mathematical cognitive style and arithmetic sign comprehension: a study of EEG alpha and theta activity. *International journal of psychophysiology* 21, 1 (1996), 1–13.

[17] Tolgay Ergenoglu, Tamer Demiralp, Zubeyir Bayraktaroglu, Mehmet Ergen, Huseyin Beydagi, and Yagiz Uresin. 2004. Alpha rhythm of the EEG modulates visual detection performance in humans. *Cognitive brain research* 20, 3 (2004), 376–383.

[18] Barbara J Ericson, Lauren E Margulieux, and Jochen Rick. 2017. Solving parsons problems versus fixing and writing code. In *Proceedings of the 17th Koli Calling international conference on computing education research*. 20–29.

[19] Alan Gevins, Michael E Smith, Linda McEvoy, and Daphne Yu. 1997. High-resolution EEG mapping of cortical activation related to working memory: effects of task difficulty, type of processing, and practice. *Cerebral cortex (New York, NY: 1991)* 7, 4 (1997), 374–385.

[20] Michael Glassen, Gregory Ames, Guang Yue, Karen J Nolan, and Soha Saleh. 2023. EEG Based Cortico-Muscular Connectivity During Standing Early Post Stroke. In *2023 45th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, 1–4.

[21] Lucian José Gonçales, Kleinner Farias, and Bruno C da Silva. 2021. Measuring the cognitive load of software developers: An extended Systematic Mapping Study. *Information and Software Technology* 136 (2021), 106563.

[22] Oliver Graf, Sverrir Thorgeirsson, and Zhendong Su. 2024. Assessing Live Programming for Program Comprehension. In *Proceedings of the 29th Innovation and Technology in Computer Science Education Conference (ITiCSE 2024)*. ACM, Milan, Italy.

[23] Halima Habieb-Mammar and Franck Tarpin-Bernard. 2004. CUMAPH: Cognitive User Modeling for Adaptive Presentation of Hyper-documents. An Experimental Study.. In *International conference on adaptive hypermedia and adaptive web-based systems*. Springer, 136–145.

[24] Sally Hamouda, Stephen H Edwards, Hicham G Elmongui, Jeremy V Ernst, and Clifford A Shaffer. 2017. A basic recursion concept inventory. *Computer Science Education* 27, 2 (2017), 121–148.

[25] Anu Holm, Kristian Lukander, Jussi Korpela, Mikael Sallinen, Kiti MI Müller, et al. 2009. Estimating brain load from the EEG. *The Scientific World Journal* 9 (2009), 639–651.

[26] Dayu Jiang and Slava Kalyuga. 2020. Confirmatory factor analysis of cognitive load ratings supports a two-factor model. *The Quantitative Methods for Psychology* 16, 3 (2020), 216–225.

[27] Wolfgang Klimesch. 1999. EEG alpha and theta oscillations reflect cognitive and memory performance: a review and analysis. *Brain research reviews* 29, 2-3 (1999), 169–195.

[28] Wolfgang Klimesch, Michael Doppelmayr, Harald Russegger, Thomas Pachinger, and Jens Schwaiger. 1998. Induced alpha band power changes in the human EEG and attention. *Neuroscience letters* 244, 2 (1998), 73–76.

[29] Makrina Viola Kosti, Kostas Georgiadis, Dimitrios A Adamos, Nikos Laskaris, Diomidis Spinellis, and Lefteris Angelis. 2018. Towards an affordable brain computer interface for the assessment of programmers' mental workload. *International Journal of Human-Computer Studies* 115 (2018), 52–66.

[30] Krzysztof Krejtz, Andrew T Duchowski, Anna Niedzielska, Cezary Biele, and Izabela Krejtz. 2018. Eye tracking cognitive load using pupil diameter and microsaccades with fixed gaze. *PloS one* 13, 9 (2018), e0203629.

[31] Naveen Kumar and Jyoti Kumar. 2016. Measurement of cognitive load in HCI systems using EEG power spectrum: an experimental study. *Procedia Computer Science* 84 (2016), 70–78.

[32] Jim Lagopoulos, Jian Xu, Inge Rasmussen, Alexandra Vik, Gin S Malhi, Carl F Eliassen, Ingrid E Arntsen, Jardar G Sæther, Stig Hollup, Are Holen, Svend Davanger, and Øyvind Ellingsen. 2009. Increased theta and alpha EEG activity during nondirective meditation. *The journal of alternative and complementary medicine* 15, 11 (2009), 1187–1192.

[33] SeolHwa Lee, Andrew Matteson, Danial Hooshyar, SongHyun Kim, JaeBum Jung, GiChun Nam, and HeuiSeok Lim. 2016. Comparing programming language comprehension between novice and expert programmers using EEG analysis. In *2016 IEEE 16th international conference on bioinformatics and bioengineering (BIBE)*. IEEE, 350–355.

[34] Jimmie Leppink, Fred Paas, Cees PM Van der Vleuten, Tamara Van Gog, and Jeroen JG Van Merriënboer. 2013. Development of an instrument for measuring different types of cognitive load. *Behavior research methods* 45 (2013), 1058–1072.

[35] Tom Lieber, Joel R Brandt, and Rob C Miller. 2014. Addressing misconceptions about code with always-on programming visualizations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2481–2490.

[36] Jonathon Love, Ravi Selker, Maarten Marsman, Tahira Jamil, Damian Dropmann, Josine Verhagen, Alexander Ly, Quentin F Gronau, Martin Šmíra, Sacha Epskamp, et al. 2019. JASP: Graphical statistical software for common statistical designs. *Journal of Statistical Software* 88 (2019), 1–17.

[37] Peter Manza, Chui Luen Vera Hau, and Hoi-Chung Leung. 2014. Alpha power gates relevant information during working memory updating. *Journal of Neuroscience* 34, 17 (2014), 5998–6002.

[38] Júlio Medeiros, Ricardo Couceiro, Gonçalo Duarte, João Durães, João Castelhano, Catarina Duarte, Miguel Castelo-Branco, Henrique Madeira, Paulo de Carvalho, and César Teixeira. 2021. Can EEG be adopted as a neuroscience reference for

assessing software programmers' cognitive load? *Sensors* 21, 7 (2021), 2338.
[39] Kai J Miller, Pradeep Shenoy, John W Miller, Rajesh PN Rao, Jeffrey G Ojemann, et al. 2007. Real-time functional brain mapping using electrocorticography. *Neuroimage* 37, 2 (2007), 504–507.
[40] Briana B Morrison, Brian Dorn, and Mark Guzdial. 2014. Measuring cognitive load in introductory CS: adaptation of an instrument. In *Proceedings of the tenth annual conference on International computing education research*. ACM, 131–138.
[41] Briana B Morrison, Lauren E Margulieux, and Mark Guzdial. 2015. Subgoals, context, and worked examples in learning computing problem solving. In *Proceedings of the eleventh annual international conference on international computing education research*. 21–29.
[42] Elizabeth Owen and John Sweller. 1985. What do students learn while solving mathematics problems? *Journal of educational psychology* 77, 3 (1985), 272.
[43] Recep A Ozdemir, Jose L Contreras-Vidal, Beom-Chan Lee, and William H Paloski. 2016. Cortical activity modulations underlying age-related performance differences during posture–cognition dual tasking. *Experimental brain research* 234 (2016), 3321–3334.
[44] Fred Paas, Juhani E Tuovinen, Huib Tabbers, and Pascal WM Van Gerven. 2016. Cognitive load measurement as a means to advance cognitive load theory. In *Cognitive Load Theory*. Routledge, 63–71.
[45] Fred GWC Paas. 1992. Training strategies for attaining transfer of problem-solving skill in statistics: a cognitive-load approach. *Journal of educational psychology* 84, 4 (1992), 429.
[46] Ramaswamy Palaniappan, Aruna Duraisingam, Nithyakalyani Chinnaiah, and Murugappan Murugappan. 2019. Predicting Java computer programming task difficulty levels using EEG for educational environments. In *International Conference on Human-Computer Interaction*. Springer, 446–460.
[47] Norman Peitek, Annabelle Bergum, Maurice Rekrut, Jonas Mucke, Matthias Nadig, Chris Parnin, Janet Siegmund, and Sven Apel. 2022. Correlates of programmer efficacy and their link to experience: A combined EEG and eye-tracking study. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 120–131.
[48] Markeya S Peteranetz and Anthony D Albano. 2020. Development and evaluation of the Nebraska Assessment of Computing Knowledge. *Frontiers in Computer Science* 2 (2020), 11.
[49] Malgorzata Plechawska-Wojcik, Monika Kaczorowska, and Dariusz Zapala. 2019. The artifact subspace reconstruction (ASR) for EEG signal correction. A comparative study. In *Information systems architecture and technology: proceedings of 39th international conference on information systems architecture and technology–ISAT 2018: part II*. Springer, 125–135.
[50] Sébastien Puma, Nadine Matton, Pierre-V Paubel, Éric Raufaste, and Radouane El-Yagoubi. 2018. Using theta and alpha band power to assess cognitive workload in multitasking environments. *International Journal of Psychophysiology* 123 (2018), 111–120.
[51] Justine Rochon, Matthias Gondan, and Meinhard Kieser. 2012. To test or not to test: Preliminary assessment of normality when comparing two independent samples. *BMC medical research methodology* 12 (2012), 1–11.
[52] Soha Saleh, Michael Glassen, Kamyar Momeni, Manikandan Ravi, Akhil Bheemreddy, Armand Hoxha, Erica Garbarini, Guang Yue, and Gail Forrest. 2022. Corticomuscular Connectivity during Walking in Able Bodied and Individuals with Incomplete Spinal Cord Injury. In *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, 2332–2335.
[53] Sergei Schapkin, Jonas Raggatz, Markus Hillmert, and Irina Böckelmann. 2020. EEG correlates of cognitive load in a multiple choice reaction task. *Acta neurobiologiae experimentalis* 80, 1 (2020), 76–89.
[54] Wolfgang Schnotz, Stefan Fries, and Holger Horz. 2009. Some motivational aspects of cognitive load theory. *Contemporary motivation research: From global to local perspectives* (2009), 86–113.
[55] Wolfgang Schnotz and Christian Kürschner. 2007. A reconsideration of cognitive load theory. *Educational psychology review* 19 (2007), 469–508.
[56] Yu Shi, Natalie Ruiz, Ronnie Taib, Eric Choi, and Fang Chen. 2007. Galvanic skin response (GSR) as an index of cognitive load. In *CHI'07 extended abstracts on Human factors in computing systems*. 2651–2656.
[57] Soroosh Solhjoo, Mark C Haigney, Elexis McBee, Jeroen JG van Merrienboer, Lambert Schuwirth, Anthony R Artino, Alexis Battista, Temple A Ratcliffe, Howard D Lee, and Steven J Durning. 2019. Heart rate and heart rate variability correlate with clinical reasoning performance and self-reported measures of cognitive load. *Scientific reports* 9, 1 (2019), 1–9.
[58] Narayan P Subramaniyam. 2018. *Pitfalls of Filtering the EEG Signal*. https://sapienlabs.org/lab-talk/pitfalls-of-filtering-the-eeg-signal/ Accessed: 2024-03-25.
[59] John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive science* 12, 2 (1988), 257–285.
[60] John Sweller. 1994. Cognitive load theory, learning difficulty, and instructional design. *Learning and instruction* 4, 4 (1994), 295–312.
[61] John Sweller. 2010. Element interactivity and intrinsic, extraneous, and germane cognitive load. *Educational psychology review* 22 (2010), 123–138.
[62] John Sweller, Jeroen JG van Merriënboer, and Fred Paas. 2019. Cognitive architecture and instructional design: 20 years later. *Educational Psychology Review* 31 (2019), 261–292.
[63] John Sweller, Jeroen JG Van Merrienboer, and Fred GWC Paas. 1998. Cognitive architecture and instructional design. *Educational psychology review* 10 (1998), 251–296.
[64] Hengtao Tang, Miao Dai, Xu Du, Jui-Long Hung, and Hao Li. 2023. An EEG study on college students' attention levels in a blended computer science class. *Innovations in Education and Teaching International* (2023), 1–13.
[65] Hengtao Tang, Miao Dai, Shuoqiu Yang, Xu Du, Jui-Long Hung, and Hao Li. 2022. Using multimodal analytics to systemically investigate online collaborative problem-solving. *Distance Education* 43, 2 (2022), 290–317.
[66] Sverrir Thorgeirsson, Lennart Lais, Theo Weidmann, and Zhendong Su. 2024. Recursion in Secondary Computer Science Education: A Comparative Study of Visual Programming Approaches. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE 2024)*. Portland, Oregon.
[67] Sverrir Thorgeirsson and Zhendong Su. 2021. Algot: an educational programming language with human-intuitive visual syntax. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 1–5.
[68] Sverrir Thorgeirsson, Theo Weidmann, Karl-Heinz Weidmann, and Zhendong Su. 2024. Comparing Cognitive Load Among Undergraduate Students Programming in Python and the Visual Language Algot. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education (SIGCSE 2024)*. Portland, Oregon.
[69] Jasper Tran O'Leary, Gabrielle Benabdallah, and Nadya Peek. 2023. Imprimer: Computational Notebooks for CNC Milling. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. 1–15.
[70] Bret Victor. 2012. Learnable programming: Designing a programming system for understanding programs. *URL: http://worrydream. com/LearnableProgramming* (2012).
[71] Theo B Weidmann, Sverrir Thorgeirsson, and Zhendong Su. 2022. Bridging the Syntax-Semantics Gap of Programming. In *Proceedings of the 2022 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. 80–94.
[72] Martin K-C Yeh, Dan Gopstein, Yu Yan, and Yanyan Zhuang. 2017. Detecting and comparing brain activity in short program comprehension using EEG. In *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1–5.
[73] Beste F Yuksel, Kurt B Oleson, Lane Harrison, Evan M Peck, Daniel Afergan, Remco Chang, and Robert JK Jacob. 2016. Learn piano with BACh: An adaptive learning interface that adjusts task difficulty based on brain state. In *Proceedings of the 2016 CHI conference on human factors in computing systems*. 5372–5384.
[74] Johannes Zagermann, Ulrike Pfeil, and Harald Reiterer. 2016. Measuring cognitive load using eye tracking technology in visual computing. In *Proceedings of the sixth workshop on beyond time and errors on novel evaluation methods for visualization*. 78–85.
[75] Albina Zavgorodniaia, Rodrigo Duran, Arto Hellas, Otto Seppala, and Juha Sorva. 2020. Measuring the cognitive load of learning to program: A replication study. In *United Kingdom & Ireland Computing Education Research conference*. 3–9.

APPENDIX

The pretest, task performance, survey, and EEG results of all participants who underwent the full EEG measurement are presented in Table 4. For privacy reasons, demographic information and the freeform feedback is not included here, nor are the results from potentially identifiable participants (e.g., those who participated in the pilot sessions or the participant who elected not to have their data analyzed with the EEG equipment).

ID	PT	SA	A-θ	A-α	P-θ	P-α	P-A	P-P	IN-A	EX-A	GE-A	IN-P	EX-P	GE-P
1	0	6	0.0826	0.0277	0.0418	0.0145	0	0	5.33	2.67	4.00	4.00	1.67	4.00
2	2	7	0.0632	0.028	0.0275	0.0148	1	1	6.00	0.00	5.75	6.33	2.33	5.25
3	0	3	0.0265	0.016	0.0189	0.0152	0	0	5.00	5.00	2.00	6.00	4.33	2.00
4	2	3	0.0203	0.0109	0.0119	0.0080	0	0	3.00	2.33	2.00	3.00	1.67	2.50
5	2	5	0.0632	0.025	0.0531	0.0175	2	2	0.33	0.00	4.00	0.67	0.00	4.00
6	1	5	0.0435	0.0144	0.0303	0.0096	0	1	7.00	5.33	6.00	2.67	2.33	6.50
7	2	6	0.0245	0.0110	0.0237	0.0078	2	2	3.67	1.67	1.75	3.00	0.67	1.50
8	2	5	0.0226	0.0151	0.0261	0.0121	2	1	4.00	0.00	6.75	3.00	0.00	6.75
9	2	6	0.0278	0.0109	0.0226	0.0085	0	0	2.00	1.67	1.25	5.33	1.00	3.25
10	2	7	0.0364	0.0148	0.0188	0.0090	1	0	7.00	3.33	5.00	7.33	3.67	5.00
11	2	6	0.0421	0.0208	0.0314	0.0140	0	0	3.33	0.00	3.00	3.33	0.00	3.00
12	2	3	0.0188	0.0074	0.0187	0.0072	1	0	3.00	0.67	6.50	6.00	6.67	2.00
13	1	3	0.0181	0.0076	0.0350	0.0106	2	0	0.00	0.00	9.25	2.67	0.00	8.75
14	1	5	0.0374	0.0136	0.0270	0.0117	0	0	2.00	0.00	3.75	2.33	3.33	4.00
15	2	6	0.0315	0.0188	0.0316	0.0196	0	1	5.67	4.33	2.00	4.00	3.67	2.00
16	1	6	0.0307	0.0135	0.0202	0.0089	0	0	6.00	2.67	5.25	5.67	2.67	5.25
17	0	6	0.0190	0.0071	0.0842	0.0153	0	0	3.33	1.67	2.00	4.33	0.00	3.75
18	1	5	0.0195	0.0123	0.0205	0.0109	0	0	4.33	0.00	6.50	5.33	0.00	4.00
19	2	6	0.0697	0.0382	0.1166	0.0585	2	1	4.33	0.00	6.00	4.33	0.33	4.75
20	2	4	0.0136	0.0065	0.0193	0.0075	0	0	4.67	4.33	5.75	3.00	5.67	5.25
21	1	5	0.0352	0.0111	0.0230	0.0082	2	0	7.00	2.00	10.0	7.33	2.00	7.25
22	2	6	0.0306	0.0113	0.0393	0.0110	2	0	3.67	1.33	4.00	4.67	1.00	4.50
23	2	5	0.0271	0.0088	0.0209	0.0070	0	0	2.67	1.00	2.25	3.67	0.00	2.25
24	1	5	0.0292	0.0097	0.0232	0.0071	0	0	6.33	2.33	3.00	7.00	5.33	5.25

Table 4: Some raw data from the 24 participants whose EEG data was collected using all nineteen channels. The participant IDs have been scrambled for privacy reasons and do not necessarily reflect the order of participation. PT: Pretest score (0-2). SA: Self-reported proficiency with Python (1-10). A/P-θ/α: Power over the theta and alpha bands for the Algot and Python tasks, respectively, over the signal frequency, expressed as $\frac{\mu V^2}{Hz}$. P-A/P-P: Total score on the Alpha and Python tasks (0-2). IN/EX/GE-A/P: Score on the intrinsic, extraneous and germane CL components on the CS CLCS for the Algot and Python tasks, respectively (0-10).