

The Journal of Computing Sciences in Colleges

**Papers of the 17th Annual CCSC
MidSouth Conference**

April 11th and April 12th, 2025
University of the Ozarks
Clarksville, AR.

Abbas Attarwala, Editor
California State University, Chico

Scott Sigman, Regional Editor
Drury University

Volume 40, Number 10

April 2025

The Journal of Computing Sciences in Colleges (ISSN 1937-4771 print, 1937-4763 digital) is published at least six times per year and constitutes the refereed papers of regional conferences sponsored by the Consortium for Computing Sciences in Colleges.

Copyright ©2025 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

Table of Contents

The Consortium for Computing Sciences in Colleges Board of Directors	7
CCSC National Partners	9
Welcome to the 2025 CCSC MidSouth Conference	10
Regional Committees — 2025 CCSC MidSouth Region	11
Ctrl+Alt+Engage: Teaching in a World Where Everything Competes for your Students’ Attention	12
<i>Michael B. Flinn, Frostburg State University</i>	
Autograding Techniques	13
<i>Dwayne Towell, Lipscomb University</i>	
A Gentle Introduction to LR Parsing	15
<i>David Middleton, Arkansas Tech University</i>	
CS2Mulch: Physical Manipulatives for Teaching Advanced Data Structures	17
<i>Mark Goadrich, Hendrix College; Gabriel Ferrer, Hendrix College</i>	
Explainable Artificial Intelligence (XAI) for Better Healthcare Decisions: A Review of XAI Methods in Cardiology and Neuroscience	18
<i>Robin Ghosh, Arkansas Tech University; Sailaja Pidugu, Arkansas Tech University; Angelina Das, Arkansas Tech University; Hayin Tamut, Arkansas Tech University</i>	
An Algorithm for the Heaviest Common Subsequence and Substring Problem	20
<i>Rao Li, University of South Carolina Aiken; Jyotishmoy Deka, Tezpur University; Kaushik Deka, National Institute of Technology Silchar; Vibhoo Srivastava, Indian Institute of Information Technology Kalyani; Madhuryya Bhattacharyya, Assam Engineering College</i>	
Addressing Participation and Feedback Challenges in Large Computer Science Classes	27
<i>Abbas Attarwala, California State University, Chico; Ed Lindoo, Regis</i>	

University; Paul Viotti, California State University, Chico

A Study Exploring the Use of LEGO® and Scratch to Engage Middle School Students in Programming 39

Abbas Attarwala, California State University, Chico; Ed Lindoo, Regis University

Incorporating Building Data Pipelines in Data-Engineering Curriculum for Supporting Classification Models in Production 49

Irene Garcia Montoya, University of San Francisco; Kabir Nawani, University of San Francisco; Fred Serfati, University of San Francisco; Gaurav Goyal, University of San Francisco; Sai Pranavi Avadhanam, University of San Francisco; Mahesh B. Chaudhari, University of San Francisco

Interactive Tool for Quantum Cryptography BB84 Algorithm Demonstration 58

Olivera Grujic, California State University, Stanislaus

Teaching Pointers through Visualizing 67

Gongbing Hong, Georgia College and State University; Yi Liu, Georgia College and State University; Frank Richardson, Georgia College and State University

Scaffolding Mobile Programming with a Student-Centered and Asset-Based Framework 79

Bilal Shebaro, St. Edward's University

Transformative Experiences in Early CS Research: How Undergraduate Research Participation Shapes Identity and Belonging 88

Renqingka (Rinchen Khar), University of Massachusetts Amherst; Neena Thota, University of Massachusetts Amherst

What Are They Actually Teaching?: A Literature Review Of Gamification in K-12 Computer Science Education 98

Paul Kim, Bridgewater State University; Junseok Hur, Bridgewater State University

Towards a Better Understanding of Proof By Induction 112

Charles Hoot, Northwest Missouri State University

Replication and Extension of Experiments with a Novel Weightless Neural Network	121
<i>Victor Nault, Haverford College; David Wonnacott, Haverford College</i>	
Esquemático: A Functional Programming Language for Native Spanish Speakers	135
<i>Kelly Farran, Ohio Wesleyan University; Sean McCulloch, Ohio Wesleyan University</i>	
Assessing Impact of Role-Playing on Introductory Programming Education: A Comparative Study	148
<i>Kriti Chauhan, Austin Peay State University; Leong Lee, Austin Peay State University</i>	
Design Guidelines for a Rigorous Deep Learning Course	161
<i>Mukulika Ghosh, Missouri State University; Jamil Saquer, Missouri State University</i>	
Gamification in Public Event Night: Evaluating the Impact of Unplugged Activity on K-12 Students' Computational Thinking in a Short Time Frame	171
<i>Seong Yeon Cho, Bridgewater State University; Jae Geun Lee, Bridgewater State University; Paul Kim, Bridgewater State University</i>	
Reviewers — 2025 CCSC MidSouth Conference	182

The Consortium for Computing Sciences in Colleges Board of Directors

Following is a listing of the contact information for the members of the Board of Directors and the Officers of the Consortium for Computing Sciences in Colleges (along with the years of expiration of their terms), as well as members serving CCSC:

Bryan Dixon, President (2026),
bcdixon@csuchico.edu, Computer
Science Department, California State
University Chico, Chico, CA 95929.

Shereen Khoja, Vice
President/President-Elect (2026),
shereen@pacificu.edu, Computer
Science, Pacific University, Forest Grove,
OR 97116.

Abbas Attarwala, Publications Chair
(2027), aattarwala@csuchico.edu,
Department of Computer Science,
California State University Chico,
Chico, CA 95929.

Ed Lindoo, Treasurer (2026),
elindoo@regis.edu, Anderson College of
Business and Computing, Regis
University, Denver, CO 80221.

Cathy Bareiss, Membership Secretary
(2025),
cathy.bareiss@betheluniversity.edu,
Department of Mathematical &
Engineering Sciences, Bethel University,
Mishawaka, IN 46545.

Judy Mullins, Central Plains
Representative (2026),
mullinsj@umkc.edu, University of
Missouri-Kansas City, Kansas City, MO
(retired).

Michael Flinn, Eastern Representative
(2026), mflinn@frostburg.edu,
Department of Computer Science &
Information Technologies, Frostburg
State University, Frostburg, MD 21532.

David Naugler, Midsouth
Representative (2025),
dnaugler@semo.edu, Brownsburg, IN
46112.

David Largent, Midwest
Representative (2026),
dllargent@bsu.edu, Department of
Computer Science, Ball State University,
Muncie, IN 47306.

Mark Bailey, Northeastern
Representative (2025),
mbailey@hamilton.edu, Computer
Science Department, Hamilton College,
Clinton, NY 13323.

Ben Tribelhorn, Northwestern
Representative (2027), tribelhb@up.edu,
School of Engineering, University of
Portland, Portland, OR 97203.

Mohamed Lotfy, Rocky Mountain
Representative (2025),
mohamedl@uvu.edu, Information
Systems & Technology Department,
College of Engineering & Technology,
Utah Valley University, Orem, UT
84058.

Mika Morgan, South Central
Representative (2027),
mikamorgan@wsu.edu, Department of
Computer Science, Washington State
University, Pullman, WA 99163.

Karen Works, Southeastern
Representative (2027), keworks@fsu.edu,
Department of Computer Science,
Florida State University - Panama City
Panama City, FL 32405

Michael Shindler, Southwestern
Representative (2026), mikes@uci.edu,
Computer Science Department, UC
Irvine, Irvine, CA 92697.

Serving the CCSC: These members are serving in positions as indicated:

Bin “Crystal” Peng, Associate Editor, bin.peng@park.edu, Department of Computer Science and Information Systems, Park University, Parkville, MO 64152.

Brian Hare, Associate Treasurer & UPE Liaison, hareb@umkc.edu, School of Computing & Engineering, University of Missouri-Kansas City, Kansas City, MO 64110.

George Dimitoglou, Comptroller, dimitoglou@hood.edu, Department of Computer Science, Hood College, Frederick, MD 21701.

Megan Thomas, Membership System Administrator, mthomas@cs.csustan.edu, Department of Computer Science, California State University Stanislaus, Turlock, CA 95382.

Karina Assiter, National Partners Chair, karinaassiter@landmark.edu, Landmark College, Putney, VT 05346.

Ed Lindoo, UPE Liaison, elindoo@regis.edu, Anderson College of Business and Computing, Regis University, Denver, CO 80221.

Deborah Hwang, Webmaster, hwangdjh@acm.org.

CCSC National Partners

The Consortium is very happy to have the following as National Partners. If you have the opportunity please thank them for their support of computing in teaching institutions. As National Partners they are invited to participate in our regional conferences. Visit with their representatives there.

Gold Level Partner

Rephactor

ACM2Y

Code Grade

GitHub

Welcome to the 2025 CCSC Midsouth Conference

After a brief hiatus, 2025 marks the return of the CCSC: Midsouth Conference. The 2025 Midsouth Conference Committee is pleased to welcome everyone to our 18th annual conference in Clarksville, Arkansas hosted by the University of the Ozarks. This year's conference includes research papers on a variety of CS education topics, tutorials and workshops on classroom techniques. The conference also features faculty posters, student posters, and student papers.

We are especially excited to have as our keynote speaker, Dr. Michael Flinn, Chair and Distrupter-in-residence of the Department of Computer Science and Information Technologies at Frostburg State University. Dr. Flinn will be speaking to us about "nudging faculty, students, and institutions to rethink what's possible when curiosity meets action."

This conference would not be possible without the effort of many people. We extend our gratitude to the authors who submitted their work, the reviewers who ensured a high-quality program with a 58% paper acceptance rate, the committee members who planned and organized the conference, and to the session moderators. The CCSC Board of Directors also deserves acknowledgment for their support of the conference. Without their financial support, the conference would not have been possible. Special thanks also goes to CCSC Board Members Michael Flinn, Eastern Region Representative, Ed Lindoo, CCSC Treasurer, and Cathy Bareiss, CCSC Membership Director, for their efforts organizing the conference within a short timeframe. Finally, we extend our gratitude to the the administration and staff of the University of the Ozarks and to the CCSC National Partners for their support.

We extend a very warm and delightful welcome to all presenters and attendees and encourage everyone to enjoy our program and the University of the Ozarks. Thank you again to all members of the 2025 Conference Committee who provide the necessary time and dedication to the conference with grace and commitment.

Scott Sigman
Drury University
Conference Chair

2025 CCSC Midsouth Conference Steering Committee

Scott Sigman, Co-Chair	Drury University, MO
Rickey Casey, Co-Chair, Student Papers	Campbellsville University, KY
Ed Lindo, Treasurer	Regis University, CO
Michael Flinn, Publicity Chair	Frostburg State University, MD
Dwayne Towell, Student Posters	Lipscomb University, TN
Cathy Barnes, CCSC Membership Chair	Bethel University, IN
Stacy Prowell, Committee Member	Tennessee Tech University, TN
Vincent Scovetta, Committee Member	Campbellsville University, KY
Lisa Singleton, Committee Member	Campbellsville University, KY
Shellon Blackman-Lees	Campbellsville University, KY

Ctrl+Alt+Engage: Teaching in a World Where Everything Competes for your Students' Attention*

Keynote

Dr. Michael B. Flinn,
Frostburg State University

Dr. Michael B. Flinn is Professor, Chair, and Disrupter-in-Residence (unofficial title, very official energy) in the Department of Computer Science and Information Technologies at Frostburg State University, where he also coordinates the Master of Applied Computer Science program. With nearly 25 years in computing education, Dr. Flinn challenges the status quo by asking one big question: *How do we truly engage students in a world flooded with AI agents, infinite scroll, and attention-hacking algorithms?*



His answer? Disrupt the system — with purpose. Whether it's creating the Bobcat Innovation Launch Pad, standing up a low-cost eSports arena, launching mandatory programming competitions, or empowering undergrads to host high school students for student-led CTF cybersecurity events, Dr. Flinn finds ways to make learning meaningful, social, and energizing.

In this interactive keynote, Dr. Flinn brings a growth mindset lens to the conversation—nudging faculty, students, and institutions to rethink what's possible when curiosity meets action. Expect challenged assumptions, fresh ideas, and a few ready-to-deploy hacks to make your teaching hit different. Be ready to discuss, interact, and pose questions—not just to Dr. Flinn, but to each other. This session is about building shared momentum, not just consuming content.

*Copyright is held by the author/owner.

Autograding Techniques*

Conference Tutorial

Dwayne Towell
Computer Science
Lipscomb University
Nashville, TN 37204
`dwayne.towell@lipscomb.edu`

Abstract

Autograders excel at providing immediate feedback, but are traditionally limited to input-processing-output type programs, limiting their use to introductory classes or unrealistic contexts. In addition, they lack the ability to differentiate between good solutions and mediocre ones, thus potentially reenforcing student misperceptions about code quality. In this workshop, you will learn tricks, existing tools, and scoring techniques, developed over twenty years, to allow you to expand the types and goals of automated feedback you provide your students.

Description

Autograders excel at providing immediate feedback to students, yet they lack the ability to differentiate between good solutions and mediocre ones. For example, it is not uncommon for a data-structures student's solution to be overly complex or involve convoluted logic. However, if the correct output is generated the typical autograder will indicate the student's work is just as good as a student whose work is simple and elegant. This missing, yet crucial, feedback potentially reenforces the student's misperceptions about their code quality. It would be better if we could indicate to the student that while their solution technically works, it could be much better.

Similarly, autograders are traditionally limited to input-processing-output (IPO) type programs, like traditional programming contests. Unlike contests,

*Copyright is held by the author/owner.

autograders usually provide students with a complete failing test case, which can be used to reproduce the deficiency. However, because of their IPO architecture, their use is usually limited to introductory classes. This is particularly troubling since introductory students may not have mastered I/O operations yet. When an IPO architecture is used in more advanced classes, for example an OS scheduler, the autograded solution looks nothing like the "in situ" version. It would be nice to provide feedback on "real" solutions, but this requires a change to our overall architecture.

In addition to the two scenarios noted above, there are many other desirable types of testing and feedback. For example, it may be desirable to limit available "solutions" when the goal of the assignment is to learn a particular coding technique (like recursion) or tool (like C++ STL). Sometimes it is useful to verify a fundamental component, such as a base class or utility function, is working correctly. Occasionally it is useful to test for the presence of error recovery paths. It would be beneficial if our autograding toolkit had more than one tool, 'cause not everything is a "nail".

Autograders have become much more capable but many instructors still rely on IPO-style feedback. Many of the autograders available today allow problem authors to utilize almost any existing tool as part of the scoring process. In this workshop, we will explore the tricks and techniques, discovered over twenty years of developing an autograder that has provided millions of results to thousands of students in dozens of different classes. By using these techniques, you not only expand the types of technical feedback provided, you also expand the kinds of assignments that can be automatically scored.

Biography

As an undergraduate, Dwayne placed sixth in the International Computer Programming Contest team. After graduating, he led teams developing CD-ROM titles, for companies such as Hasbro, Matel, Intel, and Disney. At Texas Tech University, he obtained a MS in Software Engineering and a PhD in Computer Science, before creating Athene, an autograder used by several private universities. Later, he helped found Docket Navigator, the premiere research tool that monitors, reports, and analyzes every daily action in every US patent case. He currently teaches Computer Science at Lipscomb University and continues to develop new autograder techniques.

A Gentle Introduction to LR Parsing*

Conference Tutorial

David Middleton
Department of Computer Sciences
Arkansas Tech University
Russellville, AR 72801
`dmiddle@cs.atu.edu`

Yacc and Lex provide a powerful tool to automate finding implicit structure hidden within a flat sequence of symbols. However, Yacc is fragile in that it can easily fail with obscure messages. We learn LR Parsing to fix these.

In a module taking six to eight hours, students will understand how these tools work, which serves two purposes. First, it expands the toolbox students take with them. Second, introduced in courses such as Graph Algorithms, Theory of Programming Languages, and Theory of Computation among others, students come to appreciate the power that theory brings to practice.

The ultimate purpose is to enable a computer to find implicit hierarchical structure hidden within an explicit, flat, sequence of symbols. For this, an algorithm must be provided, meaning we must know how to find implicit structure in a flat sequence. The structure being sought is described with grammars, introducing terminals and non-terminals, rules, derivations and parse trees.

A slight but useful detour at this point explores finite automata (FA) and their equivalence to non-deterministic FA, and regular expressions.

Parsing is viewed as using a particular sequence of symbols to direct a careful attempt at constructing a matching parse tree, the hidden structure of that sequence.

Top-down parsing keeps a stack of structures still to be found, even while terminal symbols are consumed from the input. The evolving concatenation of the found terminals and the remaining structures form a leftmost derivation.

Bottom-up parsing keeps a stack of structures already found. The evolving concatenation of structures found and terminals still to be used form a rightmost derivation (in reverse). Grammar rules, viewed as finite automata, are

*Copyright is held by the author/owner.

overlapped to create the Characteristic Finite State Machine underlying LR parsing.

CS2Mulch: Physical Manipulatives for Teaching Advanced Data Structures*

Conference Workshop

Mark Goadrich and Gabriel Ferrer
Mathematics and Computer Science
Hendrix College, Conway, AR 72032
`{goadrich,ferrer}@hendrix.edu`

This workshop will guide participants on how to use the CS2Mulch project, which provides engaging manipulatives specifically developed to physically demonstrate concepts in advanced data structures. Instructors can use these tools to support lessons on sorting algorithms, binary search trees, red-black trees, heaps, sets, and hash tables, including cuckoo hashing and bloom filters.

Two original decks of cards have been designed and tested that can form the basis for interactive classroom games and collaborative peer exercises in CS2. The Acorns deck is primarily useful for sorted data structures. This deck consists of 70 cards, numbered with integers from 00-69. The Menagerie deck facilitates hashed data structures. This deck consists of 110 cards. 64 of these cards display an animal, along with two integers that are the result of passing the name through two simple hash functions, Murmur3 and FNV, modulo 8. Twenty-six circular chits are also available, red on one side and black on the other, labeled with each letter of the alphabet. The CS2Mulch decks, along with lesson plans and supporting materials, are publicly available¹, either for PDF download, purchase through on-demand printing at The Game Crafter, or virtually through the online board game sandbox software Tabletopia.

In this workshop, the presenters will demonstrate the use of each deck by guiding attendees through specific exercises in sorting algorithms, insertion and deletion in binary trees, and multiple implementations of hash tables. Following these demonstrations, there will be time to ask for feedback and brainstorm new uses of these tools.

*Copyright is held by the author/owner.

¹<https://mgoadric.github.io/cs2mulch/>

Explainable Artificial Intelligence (XAI) for Better Healthcare Decisions: A Review of XAI Methods in Cardiology and Neuroscience*

Robin Ghosh¹, Sailaja Pidugu², Angelina Das³, Hayin Tamut⁴

^{1,3,4}Department of Computer and Information Science

²Department of Health Information Management

Arkansas Tech University

Russellville, AR 72801

{rghosh, spidugu, adas, htamut}@atu.edu

Abstract

Artificial intelligence (AI) is a technology intended to imitate human intelligence, enabling effective problem-solving abilities, and it has been recognized as a key area of computer science. AI-based applications have grown exponentially across various everyday domains over the last ten years, encompassing medicine and healthcare. The initial release of ChatGPT in the latter half of 2022 propelled AI to the center of public attention, leading to widespread recognition of its capabilities. The rapid integration of AI in healthcare applications has sparked major concerns regarding their ability to generate an interpretable and accurate model for clinical decisions. This requirement for transparency and trust in healthcare AI-based applications has led to the development of Explainable AI (XAI). XAI refers to a collection of techniques and approaches employed by AI applications to transparently explain how they arrived at their decisions and understand why they made that specific prognosis. Though there are numerous surveys conducted for the XAI applications in the healthcare field, the main aim of this study is to present a survey that summarizes various XAI methodologies and illustrates the techniques employed to enhance interpretability in machine learning (ML) models that apply to the disciplines of neuroscience and cardiology, with the intent of enhancing decision-making in clinical AI-based models. Following a comprehensive research study, we carefully filtered and reviewed

*Copyright is held by the author/owner.

25 research articles that cover several XAI techniques that are applied in cardiology and neuroscience applications to enhance their interpretability. According to our analysis, the SHAP method appears to be the most popular method of explainability among the reviewed studies. In addition, we provide future research initiatives to tackle the major challenges and limitations of XAI applications.

An Algorithm for the Heaviest Common Subsequence and Substring Problem*

Rao Li¹, Jyotishmoy Deka², Kaushik Deka³,
Vibhoo Srivastava⁴, and Madhuryya Bhattacharyya⁵

¹Dept. of Computer Science, Engineering, and Mathematics
University of South Carolina Aiken, Aiken, SC 29801

raol@usca.edu

²Dept. of Electrical Engineering
Tezpur University, Tezpur, Assam 784028
India

jyotishmoydeka62@gmail.com

³Dept. of Computer Science and Engineering
National Institute of Technology Silchar, Cachar, Assam 788010
India

jagatdeka20@gmail.com

⁴Dept. of Computer Science and Engineering
Indian Institute of Information Technology Kalyani
Kalyani, Nadia, West Bengal 741235
India

vibhoosrivastava24@gmail.com

⁵Dept. of Electronics and Telecommunication Engineering
Assam Engineering College
Jalukbari, Guwahati, Assam 781013
India

madhuryyabhattacharyya@gmail.com

*Copyright ©2025 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

Abstract

Let Σ be an alphabet. For each letter in Σ a positive weight is assigned to it. The weight of a string S over Σ is defined as the sum of the weights of the letters in S . Let X and Y be two strings over an alphabet Σ . The heaviest common subsequence and substring problem for two strings X and Y is to find a string Z such that Z is a subsequence of X , a substring of Y , and the weight of it as large as possible. In this note, we propose an algorithm to solve the heaviest common subsequence and substring problem for two strings.

1 Introduction

Let Σ be an alphabet and S a string over Σ . For each letter α in Σ , a positive weight, denoted $W(\alpha)$, is assigned to it. The weight of a string S , denoted $W(S)$, is defined as the sum of the weights of the letters in S . A subsequence of a string S is obtained by deleting zero or more letters of S . A substring of a string S is a subsequence of S consists of consecutive letters in S . Let X and Y be two strings over an alphabet Σ . The longest common subsequence problem for X and Y is to find the longest string which is a subsequence of both X and Y . The longest common substring problem for X and Y is to find the longest string which is a substring of both X and Y . Both the longest common subsequence problem and the longest common substring problem have been well-studied in last several decades. They have applications in different fields, for example, in molecular biology, the lengths of the longest common subsequence and the longest common substring are the suitable measurements for the similarity between two biological sequences. More details on the algorithms for the first problem can be found in [2], [3], [4], [5], [6], [11], [13], [14], and [15] and the second problem can be found in [1], [7], [9] and [16]. Motivated by the two problems above, Li et al. [12] introduced the longest common subsequence and substring problem for two strings. The longest common subsequence and substring problem for two strings X and Y is to find the longest string which is a subsequence of X and a substring of Y . Li et al. [12] proposed an algorithm to solve the longest common subsequence and substring problem for two strings.

To further generalize all the problems above, we in this note introduce the heaviest common subsequence and substring problem. Let X and Y be two strings over an alphabet Σ such that each letter in Σ is assigned a positive weight. The heaviest common subsequence and substring problem for X and Y is to find a string Z such that Z is a subsequence of X , a substring of Y , and the weight of it as large as possible. If the weight of each letter in Σ is equal to 1, then the heaviest common subsequence and substring problem for X and Y becomes the longest common subsequence and substring problem for

X and Y . Thus the heaviest common subsequence and substring problem for X and Y is a generalization of the longest common subsequence and substring problem for X and Y . We propose an algorithm to solve the heaviest common subsequence and substring problem for two strings.

2 The Recursive Structures of the Algorithm

In order to present our algorithm, we need to prove some facts which are the recursive structures for our algorithm. Before proving the facts, we need some notations as follows. Let $S = s_1s_2...s_l$ be a string over an alphabet Σ . The i th prefix of S is defined as $S_i = s_1s_2...s_i$, where $1 \leq i \leq l$. Conventionally, S_0 is defined as the empty string. The l suffixes of S are the strings of $s_1s_2...s_l$, $s_2s_3...s_l$, ..., $s_{l-1}s_l$, and s_l . Let $X = x_1x_2...x_m$ and $Y = y_1y_2...y_n$ be two strings. We define $Z[i, j]$ as a string satisfying the following conditions, where $1 \leq i \leq m$ and $1 \leq j \leq n$.

- (1) It is a subsequence of X_i .
- (2) It is a suffix of Y_j .
- (3) Under (1) and (2), its weight is as large as possible.

Fact 1. Let $U = u_1u_2...u_r$ be a heaviest string which is a subsequence of X and substring of Y . Then $W(U) = \max\{W(Z[i, j]) : 1 \leq i \leq m, 1 \leq j \leq n\}$.

Proof of Fact 1. For each i with $1 \leq i \leq m$ and each j with $1 \leq j \leq n$, we, from the definition of $Z[i, j]$, have that $Z[i, j]$ is a subsequence of X and substring of Y . By the definition of U , we have that $W(Z[i, j]) \leq W(U)$. Thus $\max\{W(Z[i, j]) : 1 \leq i \leq m, 1 \leq j \leq n\} \leq W(U)$.

Since $U = u_1u_2...u_r$ is a heaviest string which is a subsequence of X and substring of Y . There is an index s and an index t such that $u_r = x_s$ and $u_r = y_t$ such that $U = u_1u_2...u_r$ is a subsequence of X_s and a suffix of Y_t . From the definition of $Z[i, j]$, we have that $W(U) \leq W(Z[s, t]) \leq \max\{W(Z[i, j]) : 1 \leq i \leq m, 1 \leq j \leq n\}$.

Hence $W(U) = \max\{W(Z[i, j]) : 1 \leq i \leq m, 1 \leq j \leq n\}$ and the proof of Fact 1 is complete.

Fact 2. Suppose that $X_i = x_1x_2...x_i$ and $Y_j = y_1y_2...y_j$, where $1 \leq i \leq m$ and $1 \leq j \leq n$. If $Z[i, j] = z_1z_2...z_a$ is a string satisfying conditions (1), (2), and (3) above. Then we have

[1]. If $x_i = y_j$, then $W(Z[i, j]) = W(P) + W(y_j)$, where P is a heaviest string which is a subsequence of X_{i-1} and a suffix of Y_{j-1} .

[2]. If $x_i \neq y_j$, then $W(Z[i, j]) = W(Q)$, where Q is the heaviest string which is a subsequence of X_{i-1} and a suffix of Y_j .

Proof of [1] in Fact 2. Suppose $P = p_1p_2\dots p_b$ is a string satisfying the following conditions.

- (4) It is a subsequence of X_{i-1} .
- (5) It is a suffix of Y_{j-1} .
- (6) Under (4) and (5), its weight is as large as possible.

Since $P = p_1p_2\dots p_b$ is a subsequence of X_{i-1} , a suffix of Y_{j-1} , and $x_i = y_j$, $P_1 = p_1p_2\dots p_by_j$ is a subsequence of X_i and a suffix of Y_j . From the definition of $Z[i, j]$, we have $W(P) + W(y_j) \leq W(Z[i, j])$.

Notice that $z_a = y_j = x_i$ and $Z[i, j] = z_1z_2\dots z_a$ is a string satisfying conditions (1), (2), and (3) above. We have that $z_1z_2\dots z_{a-1}$ is a string which is a subsequence of X_{i-1} and a suffix of Y_{j-1} . From the definition of $P = p_1p_2\dots p_b$, we have that $W(Z[i, j]) - W(z_a) = W(Z[i, j]) - W(y_j) \leq W(P)$. Thus $W(Z[i, j]) = W(P) + W(y_j)$, where P is a heaviest string which is a subsequence of X_{i-1} and a suffix of Y_{j-1} .

Proof of [2] in Fact 2. Suppose $Q = q_1q_2\dots q_c$ is a string satisfying the following conditions.

- (7) It is a subsequence of X_{i-1} .
- (8) It is a suffix of Y_j .
- (9) Under (7) and (8), its weight is as large as possible.

Since $Q = q_1q_2\dots q_c$ is a subsequence of X_{i-1} and a suffix of Y_j , $Q = q_1q_2\dots q_c$ is a subsequence of X_i and a suffix of Y_j . From the definition of $Z[i, j]$, we have $W(Q) \leq W(Z[i, j])$.

Since $Z[i, j] = z_1z_2\dots z_a$ is a string satisfying conditions (1), (2), and (3) above, $z_a = y_j \neq x_i$. Thus $z_1z_2\dots z_a$ is a string which is a subsequence of X_{i-1} and a suffix of Y_j . From the definition of $Q = q_1q_2\dots q_c$, we have that $W(Z[i, j]) \leq W(Q)$. Thus $W(Z[i, j]) = W(Q)$, where Q is a heaviest string which is a subsequence of X_{i-1} and a suffix of Y_j .

Hence the proof of Fact 2 is complete.

3 An Algorithm for the Heaviest Common Subsequence and Substring Problem

Based on the Fact 1 and Fact 2 in Section 2, we can design an algorithm for the heaviest common subsequence and substring problem. Once again, we assume that $X = x_1x_2\dots x_m$ and $Y = y_1y_2\dots y_n$ are two strings over an alphabet Σ . In the following Algorithm A, W is an $(m+1) \times (n+1)$ array and the cells $W(i, j)$, where $1 \leq i \leq m$ and $1 \leq j \leq n$, store the weights of the strings such

that each of them satisfies the following conditions.

- (1) It is a subsequence of X_i .
- (2) It is a suffix of Y_j .
- (3) Under (1) and (2), its weight is as large as possible.

ALG A(X, Y, m, n, W)

1. Initialization: $W(i, 0) \leftarrow 0$, where $i = 0, 1, \dots, m$
 $W(0, j) \leftarrow 0$, where $j = 0, 1, \dots, n$
 $maxWeight = -1$
 $xIndexAtMaxWeight = -10$
 $yIndexAtMaxWeight = -10$
2. **for** $i \leftarrow 1$ **to** m
3. **for** $j \leftarrow 1$ **to** n
 if $x_i = y_j$ $W(i, j) \leftarrow W(i - 1, j - 1) + W(y_j)$
 else $W(i, j) \leftarrow W(i - 1, j)$
 $maxWeight = \max\{maxWeight, W(i, j)\}$
 $xIndexAtMaxWeight = i$
 $yIndexAtMaxWeight = j$
4. $u = xIndexAtMaxWeight$
 $v = yIndexAtMaxWeight$
 String S = an empty string
 while ($u > 0$ and $v > 0$)
 if ($x_{u-1} = y_{v-1}$)
 $S = S + y_{v-1}$
 $u = u - 1$
 $v = v - 1$
 else
 $u = u - 1$
5. **return** S and its weight which is $maxWeight$

Because of the Fact 1 and Fact 2 in Section 2, it is clear that Algorithm *A* is correct. Since filling each entry in the array W of size $(m+1)(n+1)$ needs a constant time, the time complexity of Algorithm *A* is $O(mn)$. Obviously, the space complexity of Algorithm *A* is also $O(mn)$. We implemented Algorithm *A* in Java and the program can be found at <https://sciences.usca.edu/math/~mathdept/rli/HCSS/HCSubseqSubstr.pdf>.

4 Conclusion

In this paper, we introduce the heaviest common subsequence and substring problem for two strings which generalizes the the longest common subsequence and substring problem for two strings in [12]. We also propose an algorithm for solving the heaviest common subsequence and substring problem for two strings. In the future, we will design new algorithms for the heaviest common subsequence and substring problem for two strings with improved time complexity and/or space complexity.

5 Acknowledgment

The authors would like to thank the referees for their suggestions which improve the initial version of the paper.

References

- [1] A. Amir, P. Charalampopoulos, S. P Pissis, and J. Radoszewski, Dynamic and Internal Longest Common Substring, *Algorithmica* 82 (2020) 3707-3743.
- [2] A. Apostolico, String editing and longest common subsequences, in: G. Rozenberg and A. Salomaa (Eds.), *Linear Modeling: Background and Application*, in: *Handbook of Formal Languages*, Vol. 2, Springer-Verlag, Berlin, 1997.
- [3] A. Apostolico, Chapter 13: General pattern matching, in: M. J. Atallah (Ed.), *Handbook of Algorithms and Theory of Computation*, CRC, Boca Raton, FL, 1998.
- [4] L. Bergroth, H. Hakonen, and T. Raita, A survey of longest common subsequence algorithms, in: *SPIRE*, A Coruña, Spain, 2000.
- [5] C. Blum et al., Solving longest common subsequence problems via a transformation to the maximum clique problem, *Computers and Operations Research* 125 (2021), 105089.
- [6] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, Section 15.4: Longest common subsequence, *Introduction to Algorithms* (second edition), MIT Press, Cambridge, MA, 2001.
- [7] M. Crochemore, C. Iliopoulos, A. Langiu, and F. Mignosi, The longest common substring problem, *Mathematical Structures in Computer Science / FirstView Article / April 2016*, pp 1-19.

- [8] M. Djukanovic, G. Raidl, and C. Blum, Finding Longest Common Subsequences: New anytime A^* search results, *Applied Soft Computing* 95 (2020), 106499.
- [9] D. Gusfield, II: Suffix Trees and Their Uses, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press, 1997.
- [10] D. Hirschberg, A linear space algorithm for computing maximal common subsequences, *Comm. ACM* 18 (June 1975) 341-343.
- [11] D. Hirschberg, Serial computations of Levenshtein distances, in: A. Apostolico and Z. Galil (Eds.), *Pattern Matching Algorithms*, Oxford University Press, Oxford, 1997.
- [12] R. Li, J. Deka, and K. Deka, An algorithm for the longest common subsequence and substring problem, *Journal of Mathematics and Informatics* 25 (2023) 77-81.
- [13] Y. Li, An efficient algorithm for LCS problem between two arbitrary sequences, *Mathematical Problems in Engineering* (2018), Article ID 4158071, <https://doi.org/10.1155/2018/4158071>.
- [14] S. Mousavi and F. Tabataba, An improved algorithm for the longest common subsequence problem, *Computers and Operations Research* 39 (2012) 512-520.
- [15] C. Rick, New algorithms for the longest common subsequence problem, Research Report No. 85123-CS, University of Bonn, 1994.
- [16] P. Weiner, Linear pattern matching algorithms. In: 14th Annual Symposium on Switching and Automata Theory, Iowa City, Iowa, USA, October 15-17, 1973, pp. 1-11.

Addressing Participation and Feedback Challenges in Large Computer Science Classes*

Abbas Attarwala Ph.D.¹, Ed Lindoo Ph.D.² and Paul Viotti Ph.D.³

¹Computer Science Department
California State University
Chico, CA, 95973

aattarwala@csuchico.edu

²Computer Information Systems
Anderson School of Business
Regis University
Denver, CO, 80221

elindoo@regis.edu

³Political Science Department
California State University
Chico, CA, 95973

pviotti@csuchico.edu

Abstract

Teaching large classes, particularly in computer science (CS), presents notable challenges that can hinder student learning. Common issues include low student participation, often due to students feeling hesitant or uncomfortable speaking up in large groups, and difficulties in providing timely feedback. To address these challenges, this study introduces a straightforward and practical solution: a dual-component quiz system integrated into laboratory sessions. In this approach, students first complete a quiz individually, followed by a collaborative phase where they

*Copyright ©2025 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

retake the same quiz in small groups. This structure promotes peer feedback and reinforces understanding. The grading scheme allocates 60% of the total score to individual performance and 40% to group performance, encouraging both personal accountability and collaborative engagement. To assess the effectiveness of this method, a comparative analysis was performed using course evaluations from five sections of CS 132 at Boston University. Two sections, serving as the baseline, did not include the quiz component, while three sections implemented the new system. The results of the Welch Statistical Test (WST) revealed that the collaborative quiz system significantly improved student participation, promptness of feedback, and quality of feedback to students.

1 Introduction

Teaching large classes, such as those with more than 100 students, presents notable challenges that can hinder student learning. A major issue is the reluctance of many students to participate in such environments, often due to feelings of shyness or intimidation [7]. As a result, only a small fraction of students actively engage, while the majority remain passive. In addition, timely and constructive feedback is crucial for effective learning [14], but delivering it in large classes is difficult, even with the support of teaching assistants. The sheer volume of grading leads to delays, and by the time students receive feedback, they may have already moved on from the material, diminishing its impact on their learning.

Williams [17] emphasized that active participation and timely feedback are critical to effective learning. He found that students who actively participate in class are more likely to retain information, develop critical thinking skills, and achieve stronger academic performance. Timely feedback enables students to identify and correct mistakes, reinforce their understanding of key concepts, and apply improvements while the material remains fresh. In contrast, delayed feedback reduces its effectiveness, potentially creating comprehension gaps that can hinder future learning. Inspired by the findings of Williams [17], we revisited student course evaluation data collected at Boston University. The evaluation data specifically measured instructor performance on eight attributes: (1) Effectiveness in explaining concepts, (2) Ability to stimulate interest in the subject, (3) Encouragement of class participation, (4) Fairness in grading, (5) Promptness in returning assignments, (6) Quality of feedback provided to students, (7) Availability outside of class, and (8) Overall instructor rating. Our analysis specifically focused on the attributes of Encouragement of class participation, Promptness in returning assignments, and Quality of feedback provided to students. Our goal was to determine whether the incorporation of a collaborative quiz component into lab sessions significantly impacted these

targeted attributes based on student teaching course evaluation data.

Based on prior research [7, 13], we hypothesized that a collaborative quiz approach would encourage students to reflect on their individual efforts, promote peer learning, and facilitate immediate feedback from classmates, teaching assistants, and instructors. Our quiz system was thus designed with two parts: an individual component, where students first attempted the quiz on their own, and a collaborative component, where they retaken the same quiz in small groups. Subsequent studies have since reinforced the value of quizzes, frequent formative assessments, and immediate feedback in enhancing learning outcomes and student engagement [8, 9, 11, 17], indirectly supporting the effectiveness of our collaborative design choices.

The grading scheme was also adjusted to incentivize participation and collaboration, with 60% of the quiz grade based on individual performance and 40% on group performance. The goal is to ensure that by the end of the lab session, student concerns are addressed through group discussions, enhancing their understanding and retention of the material. More formally, the research question is stated as, *How does the implementation of a collaborative quiz component in large class-size courses impact student engagement, participation, and the effectiveness of feedback compared to without such a component?*

In this study, we compare student course evaluation data from five different sections of CS 132. Two sections of CS 132, taught in Fall 2017 and Fall 2018, did not include any quiz component. Three other sections i.e., Spring 2019 Section A, Spring 2019 Section B, and Fall 2019 incorporated the collaborative quiz component. A WST test was used to determine whether differences in course evaluations between sections without quizzes and sections with the collaborative quiz component were statistically significant.

By assessing the effectiveness of this approach, the goal is to provide insight into how structured peer collaboration and timely feedback can enhance learning experiences in large classroom settings. The findings may have broader implications for educational practices and policies in higher education, particularly within STEM disciplines.

2 Literature Review

Chen and Liu [3] identified flipped learning, and classroom approaches as particularly well-suited for STEM courses, noting their frequent application in these disciplines. They explained that flipped learning shifts direct instruction from the group setting to the individual learning space (or vice versa), transforming the group space into a dynamic, interactive environment.

Samaila and Al-Samarraie [11] pointed out that a major issue in teaching with a flipped classroom model in CS is that many students avoid watching

pre-class videos or watch them right before the lectures, which can negatively impact learning. They suggest adopting a quiz-based flipped classroom, where quizzes within the videos increase student interaction with the pre-class content, thereby promoting greater engagement in the classroom as well. Unlike the approach of embedding quizzes within pre-class videos, our method integrates a collaborative quiz component directly into the classroom setting. This allows students to engage with the material both individually and in groups, facilitating immediate feedback and peer learning, which can enhance understanding and retention of the subject matter.

Group quizzes, where students collaborate, are supported by social interdependence theory [7] and social constructivism [15, 6], both of which highlight the role of student dialogue in improving the learning experience. Research by [10] indicates that students perform significantly better on final exams and projects when quizzes are open-book and allow for collaboration, such as discussing questions in pairs. Similarly, [9] found that students who participated in frequent quiz-based assessments achieved higher final exam grades compared to those who only took the final exam. Furthermore, [13] reported that collaborative learning groups not only improve student learning outcomes but also create positive attitudes toward both subject matter and peers. Their study on collaborative testing in an introductory Sociology course revealed that, compared to a control group, students involved in collaborative testing completed more assigned readings and demonstrated improved attitudes toward learning and the testing process.

Bandura contends that self-efficacy plays a crucial role in learning environments, directly supporting our collaborative quiz approach [1]. As students collaborate on quizzes, they observe peers successfully applying concepts, which enhances their own self-efficacy and willingness to participate in subsequent learning activities. This theoretical perspective is consistent with our finding that students who initially provided incorrect responses demonstrated better comprehension after engaging in peer discussions.

Wenger’s communities of practice framework [16] provides additional theoretical grounding for our findings. Collaborative quizzes create temporary micro-communities in which students develop shared understanding through negotiation of meaning. This social learning process facilitates deeper conceptual engagement than individual study alone. Similarly, Boud et al. describe a “multiplier effect” in which students simultaneously give and receive feedback, addressing the delayed feedback limitations that Williams [17] identified as problematic in large classes. The peer learning framework of Boud et al. [2] sheds light on how peer feedback often proves more accessible and actionable than instructor feedback alone.

Simsek [12] found that a five-point Likert scale is the most preferred re-

sponse format for measuring individual characteristics in self-report instruments such as questionnaires and surveys. They further noted that both parametric and non-parametric analyses using this item type have been widely reported in previous research. In their study, [12] analyzed the suitability of the Wilcoxon-Mann-Whitney test, Welch's t-test, and Student's t-test for Likert-type data. The results indicated that the Wilcoxon-Mann-Whitney test is more effective for analyzing smaller datasets, whereas WST performs better with larger datasets, particularly when the variances between the two groups are unequal.

3 Methodology

Williams [17] made a key observation that inspired this study: students struggle to learn effectively when feedback is delayed or when they lack engagement during class. Based on this insight, the hypothesis is that incorporating a collaborative quiz component, which requires small group interactions, will enhance students' understanding of the material through peer discussions. This approach allows students who initially answer quiz questions incorrectly to learn from their peers and address gaps in their understanding during the lab session. In support of this, [12] reported that students who initially provided incorrect responses demonstrated better comprehension after participating in peer discussions. As detailed in Section 6, the results show a statistically significant improvement in both class participation and the quality of feedback provided to students.

This study analyzed end-of-semester course evaluations to compare semesters in which the collaborative quiz component was implemented with those in which it was not. The objective was to determine whether the inclusion of the collaborative quiz led to statistically significant improvements in student engagement, the quality of feedback received, and the promptness of feedback delivery. Although [9] did not specify whether a collaborative quiz component was introduced, the focus of this study is to evaluate whether such a component enhances student participation, improves the quality of feedback, and ensures timely feedback delivery.

For this study a statistical analysis was performed to evaluate the effectiveness of the collaborative group quiz component introduced in CS 132. Two sections of CS 132 from Fall 2017 and Fall 2018, were compared, with no collaborative quiz component, against three sections of CS 132, specifically two sections of Spring 2019 and one section from Fall 2019, where the collaborative quiz component was implemented. The null hypothesis (H_0) and alternative hypothesis (H_1) were established as follows:

- H_0 : There are no differences in the mean ratings of each of the eight

evaluation questions between sections with and without the collaborative quiz component.

- H_1 : There is a significant difference in the mean ratings for each of the eight evaluation questions between sections with and without the collaborative quiz component, which could be positive or negative.

Although we conducted our statistical tests for completeness across all eight evaluation attributes, the primary focus of this study is specifically on the three attributes identified earlier: Encouragement of class participation, Promptness in returning assignments, and Quality of feedback provided to students. The inclusion of all eight attributes in the hypothesis testing serves to provide a comprehensive perspective, but we emphasize these three attributes due to their particular relevance to assessing the impact of the collaborative quiz component. To evaluate these hypotheses, a two-tailed WST was used for each of the eight evaluation questions. Before performing statistical tests, the alpha level was set at 10%. The analysis was performed in six pairs of sections, comparing each of the two sections without the collaborative quiz component to each of the three sections that incorporated the collaborative quiz component. A two-tailed test is selected because it enables the detection of increases and decreases in teaching effectiveness, irrespective of the direction. The Student t-test presumes equal variances between the compared groups. According to the data presented in Section 5, this assumption does not hold. Therefore, Student's t-test might result in misleading conclusions. In contrast, WST does not require the assumption of equal variances, which makes it more suitable for the course evaluation data [4, 5] in this paper. This test provides a more reliable assessment by adjusting the degrees of freedom according to the sample sizes and variances of each group. The $t_statistic$ and *Degree of Freedom* are calculated for the WST:

$$t_statistic = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}} \quad Deg\ of\ Freedom = \frac{\left(\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}\right)^2}{\frac{(\sigma_1^2/n_1)^2}{n_1-1} + \frac{(\sigma_2^2/n_2)^2}{n_2-1}}$$

μ_1, μ_2 are the means of the two groups, σ_1^2, σ_2^2 are their variances, and n_1, n_2 are the sample sizes. In our analysis, subscript 1 refers to the “No Quiz” group and subscript 2 refers to the “With Quiz” group. Using WST allows us to more accurately determine whether the observed differences in teaching effectiveness are statistically significant, thus providing a better foundation for potential pedagogical adjustments based on empirical evidence.

4 Quiz Structure

Quizzes were administered every two to three weeks during the 15-week semester during scheduled lab sessions. In the first phase, students completed the quiz individually within 25 minutes, under closed-book conditions, without any aids, and using pen and paper. After collecting individual quizzes, students were organized into groups of three or four, with the teaching assistant or instructor facilitating group formation to ensure balanced teams. The students were then allowed to rearrange the furniture so that they could sit together and engage in collaborative discussions.

After forming the groups, the same quiz printed on a new sheet is distributed to each team. Students are given 20 minutes to complete the quiz collaboratively. The initial individual attempt allows students to identify areas of difficulty which they can address during the group discussion. Throughout this phase, students are encouraged to communicate within their groups and may also seek guidance from the instructor or teaching assistant. In addition, students are encouraged to reference the textbook as needed. This setup transforms the classroom into a dynamic and interactive environment in which students actively engage with one another. A notable dynamic emerges as students who performed well on the individual quiz often take on a teaching role explaining concepts and answers to their peers. This peer teaching is further incentivized by the grading structure which allocates a portion of the overall grade to the collaborative component.

5 Data

The course evaluations from CS 132 were collected at the end of each semester, specifically during the last week, at Boston University. These evaluations were conducted online, allowing students to complete them on their phones or computers. Table 1 presents data from course evaluations for the Fall 2017 and Fall 2018 semesters during which no collaborative quiz component was involved. In contrast, Table 2 includes data from course evaluations for Spring 2019 Section A, Spring 2019 Section B, and Fall 2019 where a collaborative quiz component was integrated into the course.

During the evaluations, students rated the instructor performance on a five-point Likert scale from 1 (poor) to 5 (excellent) in eight questions. For Question #3, “Encouragement of Class Participation,” the highest mean rating of 4.25 was achieved in Spring 2019 Section B with the collaborative group component, and the lowest rating of 3.93 in Fall 2018 without the collaborative quiz. In Question #5, “Promptness in returning assignments,” the highest mean rating was 4.03 in Spring 2019 Section A, and the lowest was 3.2 in Fall

Table 1: Teaching course evaluations of CS 132 from Fall 2017 and Fall 2018 for classes taught without the collaborative quiz component.

	Fall 2017			Fall 2018		
	N	SD	M	N	SD	M
Effectiveness in explaining concepts	62	1.08	4.03	95	1.18	3.72
Ability to stimulate interest in subject	62	1.02	4.08	95	1.05	4.09
Encouragement of class participation	62	1.03	4.1	95	1.14	3.93
Fairness in grading	62	1	4.35	95	0.95	4.18
Promptness in returning assignments	62	1.06	4	95	1.16	3.2
Quality of feedback to students	62	1.08	4.11	95	1.07	3.84
Availability outside of class	62	1	4.32	95	0.89	4.14
Overall rating of instructor	62	0.98	4.31	95	1.02	4.02

Table 2: Teaching course evaluations of CS 132 from Spring 2019 Section A, Spring 2019 Section B and Fall 2019 for classes taught with the collaborative quiz component.

	Spring 2019 A			Spring 2019 B			Fall 2019		
	N	SD	M	N	SD	M	N	SD	M
Effectiveness in explaining concepts	53	1.12	4.02	60	1.18	3.82	99	1.01	4.14
Ability to stimulate interest in subject	53	1.19	3.85	60	1.09	4.15	99	1.06	4.06
Encouragement of class participation	53	0.91	4.13	60	1.04	4.25	99	1.11	4.1
Fairness in grading	52	0.93	4.15	60	1.12	4.25	98	0.89	4.32
Promptness in returning assignments	53	1.02	3.72	60	1.12	4.03	98	1.13	3.71
Quality of feedback to students	53	1.05	4.04	60	1.13	4.17	99	0.94	4.2
Availability outside of class	53	1	4.17	60	0.92	4.42	99	0.83	4.49
Overall rating of instructor	53	0.92	4.23	60	1.1	4.22	99	0.85	4.42

2018. For Question #6, “Quality of feedback to students,” the highest mean rating was 4.2 in Fall 2019 and the lowest was 3.84 in Fall 2018.

An important question arises from this observation: Are the differences in average ratings across these three questions statistically significant when comparing semesters with the collaborative group quiz component to those without it? Understanding whether these differences are statistically significant is crucial. It allows us to determine whether the observed differences are the result of the collaborative quiz component or are simply random occurrences. However, it is important to acknowledge that other factors could have influenced the ratings between semesters, including variations in class size and differences in student cohorts. Aside from these factors, the course material, assignments, and delivery methods remained consistent across all sections.

Student course evaluation data across all eight questions (Tables 1 and 2) were analyzed using a two-tailed WST with hypotheses described in Section 3. While statistical analysis results for all attributes are presented in Section 6 for completeness, this study specifically emphasizes three evaluation questions aligned with the research question in Section 1: (1) Encouragement of class participation, (2) Promptness in returning assignments, and (3) Quality of feedback provided to students.

6 Results and Discussion

The results of this study as seen in Table 3 demonstrate the impact of integrating a collaborative quiz component in CS 132 on 1) Encouragement of Class participation, 2) Promptness in returning assignments, and 3) Quality of feedback to students. In Table 3, row 3, there is one statistically significant result indicating that the collaborative quiz component improved class participation. Even in instances where the results were not statistically significant, the trend in all six pairs consistently showed an improvement in participation. This suggests that the collaborative component positively influences students' willingness to engage during class.

Timely feedback is essential for students to understand their mistakes and improve their learning strategies. The results for promptness in returning assignments, shown in row 5, reveal three statistically significant improvements in sections with the collaborative quiz component. This finding indicates that the immediate feedback provided during collaborative sessions is highly valued by the students. Although the overall assignment feedback, which includes grading outside the lab time, might still experience delays, the instantaneous peer and instructor feedback during the quizzes significantly enhances the students' perception of prompt feedback.

As discussed by [14] and [17] high-quality feedback is vital for students to grasp complex concepts and correct their misunderstandings. In row 6, two statistically significant improvements were observed in the quality of feedback for sections that included the collaborative quiz component. Additionally, although the remaining pairs did not show statistically significant differences, all (except for one) indicated an improvement in feedback quality for the collaborative sections. This consistent trend underscores the importance of immediate and constructive feedback provided during collaborative activities which plays a crucial role in reinforcing students' learning.

These results are significant for several reasons. Firstly, they demonstrate that incorporating collaborative components in quizzes can enhance student participation, ensure timely feedback, and improve the overall quality of feedback. This approach effectively addresses common challenges in large classroom settings such as student shyness and delays in receiving meaningful feedback. Additionally, the findings indicate that students value and benefit from immediate feedback which collaborative activities can provide efficiently. These findings carry significant implications for educational strategies, indicating that collaborative quizzes can be utilized by educators to create a more dynamic and interactive learning atmosphere.

Table 3: Statistical Analysis of all eight questions. DF is Degree of Freedom, tStat is t statistic, and CI is the 90% confidence interval. Statistical significant results are **bolded**.

Evaluation Question	Spring 2019 Section A with Quiz		Spring 2019 Section B with Quiz		Fall 2019 with Quiz	
	Fall 2017 with No Quiz	Fall 2018 with No Quiz	Fall 2017 with No Quiz	Fall 2018 with No Quiz	Fall 2017 with No Quiz	Fall 2018 with No Quiz
Effectiveness in explaining concepts	DF = 108.88 tStat = 0.049 pValue = 0.961 CI = [-0.332, 0.352]	DF = 112.48 tStat = -1.532 pValue = 0.128 CI = [-0.625, 0.025]	DF = 118.26 tStat = 1.024 pValue = 0.308 CI = [-0.130, 0.550]	DF = 125.61 tStat = -0.514 pValue = 0.608 CI = [-0.422, 0.222]	DF = 123.13 tStat = 0.645 pValue = 0.520 CI = [-0.393, 0.173]	DF = 184.95 tStat = -2.658 pValue = 0.009 CI = [-0.681, -0.159]
Ability to stimulate interest in subject	DF = 103.15 tStat = 1.103 pValue = 0.271 CI = [-0.116, 0.576]	DF = 96.87 tStat = 1.226 pValue = 0.223 CI = [-0.085, 0.565]	DF = 118.83 tStat = -0.366 pValue = 0.715 CI = [-0.887, 0.247]	DF = 122.10 tStat = -0.339 pValue = 0.736 CI = [-0.354, 0.234]	DF = 133.43 tStat = 0.119 pValue = 0.905 CI = [-0.238, 0.288]	DF = 191.80 tStat = 0.198 pValue = 0.843 CI = [-0.220, 0.288]
Encouragement of class participation	DF = 112.87 tStat = -0.166 pValue = 0.869 CI = [-0.230, 0.270]	DF = 128.45 tStat = -1.168 pValue = 0.245 CI = [-0.484, 0.084]	DF = 119.78 tStat = -0.800 pValue = 0.425 CI = [-0.461, 0.161]	DF = 134.07 tStat = -1.797 pValue = 0.075 CI = [-0.615, -0.023]	DF = 136.82 tStat = 0.000 pValue = 1.000 CI = [-0.285, 0.285]	DF = 191.11 tStat = -1.052 pValue = 0.294 CI = [-0.437, 0.097]
Fairness in grading	DF = 112.17 tStat = 1.110 pValue = 0.269 CI = [-0.099, 0.499]	DF = 109.62 tStat = 0.187 pValue = 0.852 CI = [-0.237, 0.297]	DF = 117.51 tStat = 0.520 pValue = 0.604 CI = [-0.219, 0.419]	DF = 110.48 tStat = -0.401 pValue = 0.689 CI = [-0.359, 0.219]	DF = 118.71 tStat = 0.193 pValue = 0.847 CI = [-0.228, 0.288]	DF = 189.24 tStat = -1.056 pValue = 0.292 CI = [-0.359, 0.079]
Promptness in returning assignments	DF = 111.40 tStat = 1.441 pValue = 0.152 CI = [-0.042, 0.602]	DF = 119.65 tStat = -2.829 pValue = 0.005 CI = [-0.825, -0.215]	DF = 119.08 tStat = -0.152 pValue = 0.880 CI = [-0.358, 0.298]	DF = 128.89 tStat = -4.432 pValue = 0.000 CI = [-1.140, -0.520]	DF = 136.03 tStat = 1.643 pValue = 0.103 CI = [-0.002, 0.582]	DF = 190.37 tStat = -3.093 pValue = 0.002 CI = [-0.783, -0.237]
Quality of feedback to students	DF = 111.11 tStat = 0.352 pValue = 0.726 CI = [-0.260, 0.400]	DF = 109.39 tStat = -1.103 pValue = 0.272 CI = [-0.501, 0.101]	DF = 119.27 tStat = -0.300 pValue = 0.765 CI = [-0.392, 0.272]	DF = 120.49 tStat = -1.807 pValue = 0.073 CI = [-0.633, -0.027]	DF = 116.31 tStat = -0.510 pValue = 0.590 CI = [-0.366, 0.186]	DF = 186.61 tStat = -2.486 pValue = 0.014 CI = [-0.599, -0.121]
Availability outside of class	DF = 110.23 tStat = 0.892 pValue = 0.424 CI = [-0.160, 0.460]	DF = 97.57 tStat = -0.182 pValue = 0.856 CI = [-0.304, 0.244]	DF = 119.70 tStat = -0.575 pValue = 0.566 CI = [-0.388, 0.188]	DF = 122.50 tStat = -1.869 pValue = 0.064 CI = [-0.528, -0.032]	DF = 112.01 tStat = -1.119 pValue = 0.266 CI = [-0.422, 0.082]	DF = 189.66 tStat = -2.830 pValue = 0.005 CI = [-0.554, -0.146]
Overall rating of instructor	DF = 111.99 tStat = 0.451 pValue = 0.653 CI = [-0.214, 0.374]	DF = 117.27 tStat = -1.280 pValue = 0.203 CI = [-0.482, 0.062]	DF = 117.43 tStat = 0.477 pValue = 0.635 CI = [-0.223, 0.403]	DF = 118.54 tStat = -1.134 pValue = 0.259 CI = [-0.492, 0.092]	DF = 115.99 tStat = 0.729 pValue = 0.468 CI = [-0.360, 0.140]	DF = 183.05 tStat = -2.961 pValue = 0.003 CI = [-0.623, -0.177]

7 Conclusion

This study highlights the potential of structured peer collaboration to mitigate common instructional challenges in large classroom settings, offering a scalable solution to improve educational experiences in higher education. The study demonstrated that integrating a collaborative quiz component into a large CS course can significantly enhance student engagement as well as the quality and timeliness of feedback. The analysis focused on three key areas: encouraging class participation, ensuring prompt return of assignments, and improving the quality of feedback provided to students. The results revealed that the collaborative quiz component positively influenced class participation with one statistically significant result and consistent improvements observed across all section comparisons. Regarding the promptness of returning assignments, three statistically significant results indicated that students benefited from the immediate feedback facilitated by the collaborative quizzes. Similarly, the quality of feedback showed two statistically significant improvements. These findings emphasize the value of interactive and real-time feedback systems in improving the learning experience, especially in large classroom environments.

The collaborative quiz component attempts to address common challenges such as student shyness and delayed feedback by having students work together in small groups on a collaborative quiz. This study revealed the potential of collaborative learning techniques to create a more engaging educational environment. Although the results of this study are encouraging, further research

is suggested to determine the general applicability of these conclusions. Future investigations should examine the implementation of collaborative quiz elements in diverse courses and educational environments. Through expanding this research, we can further understand the extensive effects of collaborative learning strategies on student performance. Although this study provides strong evidence that the incorporation of collaborative quizzes can significantly enhance various aspects of the learning experience in large CS classrooms, these findings also have the potential to improve teaching methods and educational practices in a variety of disciplines.

Acknowledgement

We gratefully acknowledge the use of OpenAI's ChatGPT for proofreading, grammatical checks, and other text editing tasks.

References

- [1] Albert Bandura. *Self-efficacy: The exercise of control*. New York: W.H. Freeman and Company, 1997.
- [2] David Boud, Ruth Cohen, and Jane Sampson. *Peer Learning in Higher Education: Learning From and With Each Other*. London: Routledge, 2001.
- [3] Jyun-Chen Chen and Chia-Yu Liu. "Using knowledge building and flipped learning to enhance students' learning performance in a hands-on STEM activity". In: *Journal of Computer Assisted Learning* 40.6 (2024), pp. 3474–3485.
- [4] Marie Delacre, Daniël Lakens, and Christophe Leys. "Why psychologists should by default use Welch's t-test instead of Student's t-test". In: *International Review of Social Psychology* 30.1 (2017), pp. 92–101.
- [5] Ben Derrick, Deirdre Toher, and Paul White. "Why Welch's test is Type I error robust". In: *The quantitative methods for Psychology* 12.1 (2016), pp. 30–38.
- [6] Marjorie Gabain. *"The" Language and Thought of the Child*. Routledge & Kegan Paul, 1971.
- [7] David W Johnson, Roger T Johnson, and Karl Smith. "The state of cooperative learning in postsecondary and professional settings". In: *Educational psychology review* 19 (2007), pp. 15–29.

- [8] Lauren E Margulieux, Briana B Morrison, and Adrienne Decker. “Reducing withdrawal and failure rates in introductory programming with subgoal labeled worked examples”. In: *International Journal of STEM Education* 7 (2020), pp. 1–16.
- [9] Ivica Pesovski and Vladimir Trajkovik. “The influence of frequent quiz-based exams on learning performance among computer science students”. In: *ICERI2022 Proceedings*. IATED. 2022, pp. 5693–5699.
- [10] Ali R Rezaei. “Frequent collaborative quiz taking and conceptual learning”. In: *Active Learning in Higher Education* 16.3 (2015), pp. 187–196.
- [11] Kamaludeen Samaila and Hosam Al-Samarraie. “Reinventing teaching pedagogy: the benefits of quiz-enhanced flipped classroom model on students’ learning outcomes and engagement”. In: *Journal of Applied Research in Higher Education* (2023).
- [12] Ahmet Salih Şimşek. “The power and type I error of Wilcoxon-Mann-Whitney, Welch’s t, and student’s t tests for likert-type data”. In: *International Journal of Assessment Tools in Education* 10.1 (2023), pp. 114–128.
- [13] Suzanne R Slusser and Rebecca J Erickson. “Group quizzes: an extension of the collaborative learning process”. In: *Teaching Sociology* 34.3 (2006), pp. 249–262.
- [14] Marjolein Versteeg et al. “Peer instruction improves comprehension and transfer of physiological concepts: a randomized comparison with self-explanation”. In: *Advances in Health Sciences Education* 24 (2019), pp. 151–165.
- [15] Lev Semenovich Vygotsky and Michael Cole. *Mind in society: Development of higher psychological processes*. Harvard university press, 1978.
- [16] Etienne Wenger. *Communities of Practice: Learning, Meaning, and Identity*. Cambridge: Cambridge University Press, 1998.
- [17] Andrew Williams. “Delivering Effective Student Feedback in Higher Education: An Evaluation of the Challenges and Best Practice.” In: *International Journal of Research in Education and Science* 10.2 (2024), pp. 473–501.

A study exploring the use of LEGO[®] and Scratch to engage middle school students in programming*

Abbas Attarwala Ph.D.¹ and Ed Lindoo Ph.D.²

¹Computer Science Department
California State University
Chico, CA, 95973

aattarwala@csuchico.edu

²Computer Information Systems
Anderson School of Business

Regis University
Denver, CO, 80221

elindoo@regis.edu

Abstract

This study presents a one-day outreach workshop designed for middle school students, where the focus was on teaching block-based programming using Scratch alongside LEGO[®] Education SPIKE[™] kits. Using the inherent appeal of LEGO[®] construction, the workshop aimed to make programming fun and exciting through a hands-on project: building and programming a motorized LEGO[®] car. A key component of the session involved having students articulate, in their own words, how the car should move in a square—for example, describing that “the car moves 15 cm in the north direction, then makes a 90-degree turn to the east, followed by another 15 cm movement, a 90-degree turn to the south, and finally moves 15 cm before a 90-degree turn to the west.” This verbal algorithm was then collaboratively translated into Scratch code

*Copyright ©2025 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

that controlled sensor inputs, motor outputs, and sound effects. The experience provided valuable observations on the strengths and challenges of short-duration programming workshops and laid the foundation for future, more extensive outreach initiatives.

1 Introduction

Introducing programming to young learners is most effective when the learning process is engaging and relatable [6]. Middle school students in particular, benefit from methods that combine hands-on activities with abstract thinking. Understanding that LEGO® captivates many students, this study’s workshop was designed around this familiar and enjoyable medium. The session started with students describing, in plain language, how a LEGO® car should move in a square. In the workshop, a group of six students initially reached a consensus on a rough description of the car’s movement. One student demonstrated the motion visually by holding the car and using hand gestures to show how it would move straight, then turn, then move straight again, and continue in this pattern until it returned to its starting point. To further clarify the motion, the car’s movement was traced on a piece of paper marked with the cardinal directions, offering students a concrete visual reference. Building on this demonstration, the group refined their description to a more precise set of instructions: “move 15 cm in the north direction, then make a 90-degree turn to the east, followed by another 15 cm movement, a 90-degree turn to the south, and finally move 15 cm before making a 90-degree turn to the west.”

This natural language description served as a foundation for developing algorithmic thinking. Students collaborated to translate their verbal instructions into Scratch code, enabling them to control the car’s motors and sensors. They also incorporated sound effects to enhance the overall experience and engagement. The workshop, held over a three-hour session at California State University, Chico, successfully combined LEGO® construction with computational thinking, demonstrating that programming concepts can be introduced in a fun, interactive, and collaborative environment.

2 Literature Review

Research on programming education for middle school students has explored various approaches, including LEGO®-based robotics, Scratch programming, and tangible computing tools. De Lira [2] found that project-based programming camps improve computational thinking, while [11] demonstrated how integrating Scratch with physical computing improved engagement. Paparo [12] analyzed how middle school students engaged in advanced Scratch concepts,

such as variables and procedures, through structured challenges.

Jormanainen [5] highlighted the motivational impact of educational robotics. Their large-scale empirical study (n=1440) showed that students at the age of 9-10 years (Grades 3-4) are significantly more motivated towards such a learning tool than the students of age 11-12 years (Grades 5-6). Furthermore, they proved that young girls in particular find robotic programming motivating and they are eager to learn more. This indicates that appealing tools play a key role when teaching programming concepts to young school children. Sung [14] introduced Concrete Computational Concepts Programming Environments (3CPEs), using metaphors to simplify computational concepts. Their findings indicate that 3CPEs have the potential to offer elementary students a suitable environment to learn computational concepts. This is accomplished by offering a low entry point, making it easy for students to begin, while also providing a high ceiling that encourages them to take on progressively more challenging projects.

Waldhor [15] developed BrickMusicTable, a LEGO® based music sequencer for creative coding. Their resulting musical instrument is a sequencer that produces sound based on visually tracked bricks placed on a LEGO® ground plate. The prototype recognizes different sizes and colors of bricks and their position on the sequencer as notes. This proved to be an exceptional motivator in student attention to detail and learning. Lott [10] explored LEGO® as a hands-on learning tool, leading to a LEGO® based creative learning space at Rutgers University. Lott's goal was to create an affordable, compact, and easy-to-install space within the Rutgers library that would engage students. The initiative was influenced by the research by Lotts on LEGO®, which showed that hands-on, mind-on learning leads to deeper and more meaningful engagement with concepts and their possibilities. Lotts also believed that active learning workshops could help individuals discover how hands-on experiences, community building, and play can positively impact their organizations.

Korei [8] demonstrated how LEGO® robots promote STEM engagement and interdisciplinary learning. Their findings emphasized how robotic activities in education spark interest in STEM and motivate students to engage with technology. They concluded that Robotics in Education (RiE) serves as a valuable tool to involve students in scientific and technological creativity while creating the development of technological knowledge. Beyond technical skills, RiE also promotes soft skills such as communication and teamwork. Within RiE, educational robotics focuses on improving learning experiences by creating, implementing, refining, and validating pedagogical activities, tools, and technologies in which robots play an integral role. Kolne [7] studied LEGO® robotics for children with disabilities, showing its role in STEM learning, therapy goals, and the development of social skills. The study focused on the

activity settings of the HB FIRST® Robotics program; a group-based initiative tailored to meet the needs of children with disabilities. The program aimed to provide opportunities for participants to develop STEM skills while simultaneously working on therapy goals, creating self-confidence, independence, communication, and teamwork, all within a play-centered environment. In general, [7] found that the robotics program facilitated extensive interaction between children and adults, as well as meaningful social engagement with peers in all activity settings. These observations aligned with the program’s objective of integrating the teaching of STEM and robotics principles into a social environment that nurtured critical thinking, problem-solving, and teamwork skills.

Likewise, [3] introduced LEGO® Kintsugi, a teaching method that combines LEGO® robotics with emotional intelligence, promoting resilience and creativity. In their study, they explored the impact of the LEGO® Education SPIKE™ environment on children with autism and disabilities. Their results showed that using LEGO® Education SPIKE™ Prime and its structured lesson plans significantly improved students’ digital skills, particularly benefiting those with special educational needs. The LEGO® Kintsugi methodology integrates LEGO® robotics with the Japanese art of Kintsugi, emphasizing resilience and the beauty of imperfection in the learning process. By encouraging students to reflect on and rework their robotic creations, the approach positively impacted the development of life skills, especially in managing and coping with emotions. [3] concluded that LEGO® Kintsugi could serve as an innovative teaching method for STEAM disciplines—Science, Technology, Engineering, Art, and Mathematics—which emphasize interdisciplinary learning through creativity, problem-solving, and critical thinking. This approach blends emotional intelligence with technical skills, supporting students in developing competencies relevant to fields such as artificial intelligence.

Korkmaz [9] assessed educational robots’ technological integration, identifying LEGO® kits as highly adaptable and user-friendly. Danahy [1] reviewed how universities have integrated LEGO® robotics into engineering curricula, demonstrating its effectiveness in teaching sensor accuracy, motor control, and rapid prototyping. Specifically, [1] highlighted Tufts University, The University of Nevada, Reno, the Arizona State University Polytechnic Campus and The University of Notre Dame which have incorporated LEGO® products into their curricula for over 15 years. The researchers noted the enthusiasm with which students engaged in complex robotics challenges early in their engineering education. They emphasized that LEGO® Mindstorms products enable engineers and nonengineers to explore critical concepts such as sensor accuracy, motor latency, response times, and task prioritization without requiring extensive knowledge of circuit design, assembly-level programming, or artificial

intelligence. In addition, these tools provide students with accessible opportunities to investigate product design and prototyping.

Shang [13] found that STEM robotics camps significantly enhanced computational thinking and self-efficacy among rural students. The study also revealed gender differences in participation: girls tended to favor creative activities, while boys preferred structured and competitive tasks. Although boys and girls exhibited similar self-ratings for their ability to learn and their expectations of success in mathematics, girls who participated in the camp expressed a strong interest in pursuing future careers in STEM fields. Francis [4] showed that programming robots to navigate polygons reinforced geometric concepts, incorporating measurement, proportional reasoning, and hands-on role play to strengthen understanding. Notably, [4] creatively engaged students by having them act out and identify the steps needed before beginning programming on the computer.

In contrast to previous work, our study differs in several significant ways. First, it focuses on a short, one-day workshop rather than multi-week programs. Second, it uniquely integrates LEGO® construction and Scratch coding in a single session, bridging physical computing with block-based programming. Unlike [5], who studied long-term robotics education, this approach prioritizes immediate engagement through an interactive LEGO® car project. Although [14] used predefined metaphors, this study relies on student-generated verbal instructions to develop algorithmic thinking before coding. Waldhor [15] explored LEGO® for music-based programming, whereas this research emphasizes robotic movement, sensor integration, and engineering concepts. Inspired by [10], this study highlights the role of LEGO® in promoting creativity and active learning. Unlike [8], who used robotics for mathematical visualization, this approach focuses on real-world movement and problem solving.

Kolne [7] emphasized social engagement and therapy goals, while this study focuses on computational thinking and algorithmic problem solving. Filipone [3] linked LEGO® robotics to emotional intelligence and resilience building, whereas this work emphasizes STEM learning through interactive programming activities. While [9] assessed the suitability of robots for preschool learners, this study explores how LEGO® and Scratch can introduce computational thinking into middle school education. Unlike [1], which investigated the role of LEGO® in university engineering programs, this research focuses on initial STEM education. Shang [13] examined gender differences in robotic STEM camps, while this study emphasizes cooperative, practical problem-solving independent of gender. Finally, while [4] focused on polygon navigation and geometric learning, in contrast, this study highlights algorithmic problem solving through real-world robotics applications.

Future work will expand the workshop to five days, incorporate pre- and

post-assessments with statistical analysis, and refine the curriculum to enhance student learning. By integrating hands-on building, computational thinking, and collaborative problem solving, our study provides a novel perspective on how short-term workshops can effectively introduce programming to middle school students.

3 Day of Workshop

During the workshop, a single LEGO® Education SPIKE™ kit was shared among six students. The instructor closely monitored their progress and facilitated the turn-taking as they collaboratively built the LEGO® car equipped with sensors. The initial assembly phase was straightforward as the students were already familiar with reading the instruction manual and following step-by-step instructions. However, occasional guidance was provided to explain unfamiliar components. For instance, when the students connected the wire to the LEGO® motor, the instructor paused to demonstrate its functionality by assembling a simple piece of code. This demonstration helped the students recognize that certain LEGO® components were not merely static but could perform various actions based on programmed instructions. Through this hands-on learning experience, they began to grasp the fundamental concept that connecting wheels to the motor enabled movement. When the motor was activated, it set the wheels in motion, ultimately propelling the car forward.

Following construction, the instructor led an interactive activity to further explore the concept of movement. By manually pushing the car across a sheet of paper, a pen traced its path. The paper had been premarked with the cardinal directions: north, south, east, and west to facilitate the exercise. As the car moved, students were encouraged to describe its motion in their own words, providing directions such as moving a certain distance north, turning 90 degrees east, and continuing with additional movements and turns. This exercise served as the foundation for the next phase of the workshop. Although most of the students had limited prior exposure to programming, they engaged in trial and error, experimenting with various programming constructs in Scratch. Through collaborative problem solving, they successfully matched the appropriate constructs to their verbal instructions. Ultimately, the group developed a program that directed the car's movement as described, effectively transforming everyday language into a functional piece of code. Figure 1 presents the SCRATCH code that was developed collaboratively with the students during the workshop, while Figure 2 shows the LEGO® car they built.



Figure 1: SCRATCH code developed in the workshop to make the LEGO® car move in a square shape.



Figure 2: Actual LEGO® car built in the workshop using LEGO® Education SPIKE™ Kit.

4 Lessons Learned

One key observation from the workshop was that using LEGO® as the primary medium provided significant motivation. The students were excited about the opportunity to build a tangible object that they could relate to, which made the subsequent introduction of programming concepts feel natural and engaging. Another highlight was the development of algorithmic thinking. By asking students to describe the movement of a car in a square using everyday language, for example “moving 15 cm north, then making a 90-degree turn to the east, etc.”, the workshop provided a concrete basis for translating these ideas into Scratch code. This exercise underscored the value of starting with natural language descriptions to bridge the gap between real-world instructions and computational logic.

Through a collaborative process to encode the motion of the car, the students shared and experimented with different features of Scratch programming to make the car move. By choosing suitable Scratch programming constructs that reflected their verbal directives, they not only improved their coding abilities, but also promoted teamwork and critical thinking. In addition, incorporating the audio feature of the LEGO® Education SPIKE™ kit, having students record and play back sound effects during specific movements, added an element of fun and creativity to the workshop, further deepening their participation.

Despite these successes, the three-hour session presented some challenges.

Sharing one computer among several students occasionally limited individual participation, particularly for those who were less assertive. This experience has important implications for the design of future workshops, suggesting that adjustments in equipment allocation and session structure might be necessary to ensure that every student has ample hands-on time.

5 Future Work

Building on this experience, several enhancements are envisioned for future iterations of the workshop. Instead of a single-day session, a five-day workshop and each day allocating three hours to the workshop are planned to allow for a more gradual introduction of concepts, with each day dedicated to constructing and programming a new LEGO[®] artifact that increases in complexity. Furthermore, investing in more LEGO[®] Education SPIKE[™] kits will reduce the student-to-kit ratio, ensuring that each participant enjoys ample hands-on time, a change that will particularly benefit shy or less vocal students by allowing them to engage more fully with the project. Future sessions will also incorporate pre- and post-workshop surveys to collect quantitative data on students' programming knowledge and interest in computer science (CS). The data collected will be analyzed using statistical tests, such as a Welch test, to objectively assess the impact of the workshop. Finally, the curriculum will be refined iteratively based on ongoing feedback and observations, with potential integration of additional multimedia elements and more challenging programming tasks to further enhance student engagement and learning outcomes.

6 Conclusion

This one-day outreach workshop demonstrated that integrating LEGO[®] construction with block-based programming can serve as an effective and engaging introduction to CS for middle school students. By first inviting students to describe, in their own words, how a LEGO[®] car should move in a square, the session successfully bridged the gap between natural language and algorithmic thinking. The subsequent translation of these descriptions into Scratch code, complete with sensor inputs, motor outputs, and sound effects, provided a memorable and tangible demonstration of how programming works. Although time constraints and equipment sharing presented challenges, the experience offers valuable insights that will inform future more extensive iterations of the workshop. With extended duration, improved access to equipment, and a focus on empirical assessment, future sessions have the potential to further enhance student engagement and create a sustained interest in CS.

Acknowledgement

We gratefully acknowledge the use of OpenAI's ChatGPT for proofreading, grammatical checks, and other text editing tasks.

References

- [1] Ethan Danahy et al. "Lego-based robotics in higher education: 15 years of student creativity". In: *International Journal of Advanced Robotic Systems* 11.2 (2014), p. 27.
- [2] Carla De Lira, Rachel Wong, and Olusola Adesope. "A systematic review on the effectiveness of programming camps on middle school students' programming knowledge and attitudes of computing". In: *Journal of Computing Sciences in Colleges* 38.1 (2022), pp. 89–98.
- [3] Alfonso Filippone and Antonio Bevilacqua. "Lego-Kintsugi: a new teaching and educational methodology to enhance life skills and digital life skills, create well-being and educate happiness". In: *Italian journal of health education, sport and inclusive didactics* 8.3 (2024).
- [4] Krista Francis and Michael Poscente. "Building number sense with Lego® robots". In: *Teaching children mathematics* 23.5 (2016), pp. 310–312.
- [5] Ilkka Jormanainen and Markku Tukiainen. "Attractive educational robotics motivates younger students to learn programming and computational thinking". In: *Eighth International Conference on Technological Ecosystems for Enhancing Multiculturality*. 2020, pp. 54–60.
- [6] Tanja Karp et al. "Generation NXT: Building young engineers with LEGO®s". In: *IEEE Transactions on Education* 53.1 (2009), pp. 80–87.
- [7] Kendall Kolne, Sunny Bui, and Sally Lindsay. "Assessing the environmental quality of an adapted, play-based LEGO® robotics program to achieve optimal outcomes for children with disabilities". In: *Disability and rehabilitation* 43.25 (2021), pp. 3613–3622.
- [8] Attila Körei, Szilvia Szilágyi, and Ingrida Vaičiulytė. "An Educational Robotics Approach to Drawing and Studying Central Trochoids at the University Level". In: *Sustainability* 16.22 (2024), p. 9684.
- [9] Özgen Korkmaz et al. "PEDAGOGICAL CLASSIFICATION OF EDUCATIONAL ROBOTS IN PRE SCHOOL TEACHING". In: *International Online Journal of Primary Education* (2023).

- [10] Megan Lott. “Playing with LEGO®, Learning about the Library, and “Making” Campus Connections: The Rutgers University Art Library Lego Playing Station, Part One”. In: *Journal of Library Administration* 56.4 (2016), pp. 359–380.
- [11] Callan J Noak et al. “Introducing Programming to Middle School Students to Increase Knowledge and Interest in Computer Science”. In: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2*. 2022, pp. 1098–1098.
- [12] Giulia Paparo, Marco Hartmann, and Mareen Grillenberger. “A Scratch Challenge: Middle School Students Working with Variables, Lists and Procedures”. In: *Proceedings of the 21st Koli Calling International Conference on Computing Education Research*. 2021, pp. 1–10.
- [13] Xiaojing Shang et al. “Effects of robotics STEM camps on rural elementary students’ self-efficacy and computational thinking”. In: *Educational technology research and development* 71.3 (2023), pp. 1135–1160.
- [14] Ching-Ying Sung et al. “3CPEs: Concrete Computational Concepts Programming Environments for Elementary Computer Science Education”. In: *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*. 2024, pp. 1–7.
- [15] Merlin Waldhör, Oliver Hödl, and Peter Reichl. “BrickMusicTable: A LEGO brick-based tabletop sequencer”. In: *Proceedings of the International Conference on Mobile and Ubiquitous Multimedia*. 2024, pp. 412–418.

Incorporating Building Data Pipelines in Data-Engineering Curriculum for Supporting Classification Models in Production*

Irene Garcia Montoya, Kabir Nawani, Fred Serfati,
Gaurav Goyal, Sai Pranavi Avadhanam,
Mahesh B. Chaudhari

Data Institute, University of San Francisco
San Francisco, CA 94515

{igarciamontoya,knawani,fserfati,ggoyal,savadhanam,mchaudhari}@usfca.edu

Abstract

Teaching students to build automated data pipelines at the graduate level is becoming increasingly important for educators in preparing them for data engineering positions. Almost every data-driven organization needs software engineers with experience building data platforms in the cloud, such as Google Cloud Platform (GCP). As part of a graduate-level course, this paper illustrates how students apply the concepts from the course to build an automated data pipeline using a workflow manager like Apache Airflow and MongoDB within a cloud computing environment like GCP. The students demonstrate the use of minimal cloud resources to build a small end-to-end data pipeline that automatically pulls fresh data from different sources and feeds it to machine learning models. Such a data ingestion and integration pipeline enables students to continuously train their models with newer data and improve model efficiency. The sample data pipeline in this paper integrates movie-themed datasets to train and deploy a multiclass classification model based on RoBERTa. The project illustrates how continuously bringing in fresh data results in better recommendations for the latest movies, enhancing user selection

*Copyright ©2025 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

and overall recommendation quality. This type of course-wide project allows students to apply and consolidate the skills they have learned across multiple data science and data engineering courses to solve real-world problems.

1 Introduction

As organizations increasingly rely on data-driven decision-making, the ability to design and manage scalable data pipelines has become a critical skill for data professionals. These data pipelines bring data from various sources together to provide unified access to the data within the organization. Deploying such data pipelines in Cloud environments such as Google Cloud Platform (GCP) or Amazon Web Services (AWS) helps in scaling the pipeline infrastructure whenever the data volume increases. This data is then leveraged by various teams such as data scientists, business analysts, and others. Thus, preparing graduate students in the field of cloud computing and data engineering is becoming important so that these students are prepared for such jobs. The Master of Science in Data Science (MSDS) is a graduate program at the University of San Francisco that focuses on a curriculum around data science and data engineering principles [11].

It is important to design projects that keep course materials as the main focus but also encourage students to apply techniques from previous courses. By building end-to-end projects, students can gain a more holistic perspective of various data engineering concepts and become more well-rounded engineers. Through these group projects, the students learn to collaborate and communicate effectively with their team members and also collectively solve an interesting problem using various concepts and systems they have studied as part of the course.

In this paper, we present a graduate-level project where students apply cloud computing skills and their knowledge of various distributed data systems from a graduate-level course at the university that teaches data science and data engineering courses as part of the curriculum. The students have already gained prior knowledge in model training and validation, data warehousing, etc., from previous courses offered as part of the graduate program. This course has a component of a group-project to build an end-to-end autonomous pipeline using different systems the students have studied as part of the course. In this paper, we explain the architecture and execution of the ETL (Extract, Transform, Load) pipeline, tailored for processing a comprehensive preexisting dataset of movie summaries combined with a second ETL pipeline that is triggered weekly to fetch fresh data from the Web. In the end, the datasets are fed into a RoBERTa-based classification model [8]) that labels each new data

point (newly released movies) with its predicted classification label (movie genre). To process the large volume of data in the starter dataset, we use Apache Spark [5] which provides a distributed framework to process data faster using a multi-node cluster. For the orchestration and automation of the routine data ingestion and subsequent jobs, we use Apache Airflow [4]. Data ingested through the pipeline is indexed and stored in MongoDB, a document storage database [9], so that it can be repurposed to retrain the model and improve the efficiency and the accuracy of the model.

2 Data Ingestion

2.1 Training Dataset

As a jumping point, we make use of CMU Movie Summary Corpus dataset. The dataset includes movie plot summaries and other movie-associated meta-data collected by David Bamman, Brendan O'Connor, and Noah Smith at Carnegie Mellon University [1]. The dataset contains 42,306 movie plot summaries extracted from Wikipedia as well as the aligned metadata extracted from Freebase. The data model of the dataset can be observed in Figure 1.

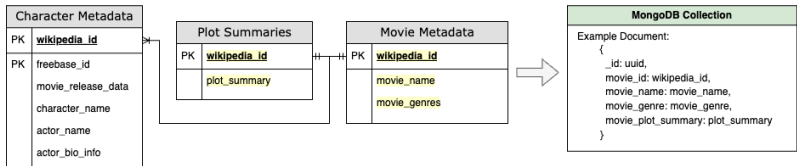


Figure 1: Source data model and transformed JSON document.

To prepare the dataset for model training, we perform the following transformations through PySpark User Defined Functions (UDFs):

1. Each movie can have more than one genre assigned. Thus, every movie record is expanded so that each combination of the movie and the genre appears separately in the MongoDB collection. Each combination is uniquely identified in the collection for faster lookup.
2. The dataset contains a wide array of genres, many of which are subtypes of another genre (e.g. “Mockumentary” is a type of “Comedy”) or are combinations of two or more genres (e.g. “Action Comedy” is expanded into “Action” and “Comedy”). Genres with commonalities are grouped together to simplify the classification problem, and genres without a significant enough presence in the dataset (e.g. “Beach Film” or “Airplanes and Airports”) are deleted.

3. The plot summaries are also stripped of HTML content, links, and special characters. We also corrected common misspellings, ensuring the summaries are clean and conducive to text analysis.

The end result of this PySpark ETL is remotely uploaded to a new MongoDB collection through a MongoClient connection.

2.2 Testing Dataset

The testing data is produced by the main ETL pipeline of the project. This pipeline is responsible for scraping the test data for our model and will be eventually used in production to get the latest data on a weekly basis. This pipeline does not handle a high data volume, however, it is much more complex than our training ETL. The main ETL is explained in detail in Section 3, whereas, more information about the training ETL is provided in Section 4.

The purpose of the model is to classify newly released movies, so the new data is used to answer two questions: *“What are the newly released movies?”* and *“What are their plots?”*. Following are the two sources from where the data is obtained:

1. Box Office website by IMDB [7]: To capture the latest movie releases, we look for the names of the movies that appear highlighted as “New This Week” (HTML class “mojo-annotation-isNewThisWeek”). The extracted data from the targeted object is then formatted using XPath and scanned to get the URL of the individual movie web page. The data from that web page is also scraped to retrieve the IMDb ID of the movie. The IMDb ID of the movie can be found as part of the hyperlinks in the movie profile web page, as shown in Figure 2.
2. OMDb API [6]: Through GET requests to the OMDb API using the IMDb ID we have scraped, we retrieve the official genres and plot summary of each movie. The extracted data is cleaned and transformed to match the schema of the training dataset to ensure consistency with the schema. These official genres will be used in the tests to assess the accuracy of the model.

Similar to the training dataset, the newly acquired data is loaded into another MongoDB collection.

3 Airflow Orchestration

The Airflow workload consists of 3 directed acyclic graphs (DAG) that interconnect with each other, as seen in Figure 3:

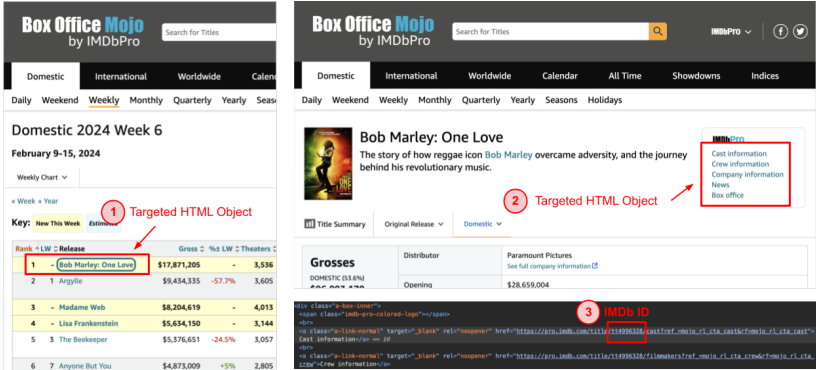


Figure 2: Targeted HTML objects as seen in web browser

1. API Call DAG (ACD): This is the main DAG. It is triggered weekly, and it drives the whole data pipeline. After being activated, it triggers the Web Scraping DAG (WSD) and passes the necessary information to find the correct web page to scrape. After the WSD returns a list of IMDb IDs, the ACD dynamically creates tasks to make API calls to the OMDb API. This is done through Airflow’s SimpleHttpOperators. Once the data is retrieved, formatted, and successfully stored, the ACD triggers the Genre Classification DAG (GCD). Once the GCD is completed successfully, the Airflow scheduler marks the job as finished with the status of “success”.
2. Web Scraping DAG (WSD): This DAG is triggered by the ACD. The goal of this DAG is to fetch the HTML text from the web through a SimpleHttpOperator and dynamically create the necessary tasks to extract the IMDb ID of each movie (Figure 2). On completion of all the tasks inside this DAG, it returns the final list of IMDb IDs to the ACD.
3. Genre Classification DAG (GCD): The GCD DAG is the last step in the pipeline. The purpose of this DAG is to load the pretrained model artifact from Google Cloud Storage [10], and generate classifications for the new movies. The results are stored in a collection in the MongoDB database.

4 Model Training

As mentioned in the introduction, we are tackling a Multiclass Logistic Classification (MLC) problem. In order to simplify the complexity of assigning the

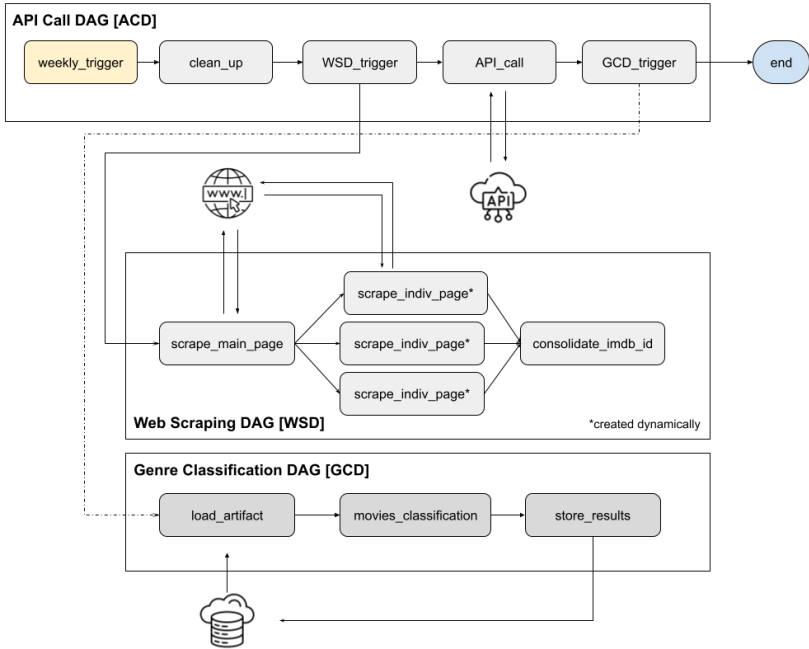


Figure 3: Airflow orchestration workflow

correct label from a potentially large dataset, we decompose our MLC task into a series of binary classification problems. In this approach, a separate logistic regression model is trained for each label, independently making the decision of whether that genre label applies to a given movie.

Using this methodology for MLC problems allows us to simplify the decision boundaries of the model, making training and optimization easier. It scales better when the number of possible labels increases. The smaller binary classification models can be trained or run in parallel allowing for modularity, parallelization, and easier integration of new labels, while maintaining both flexibility and computational efficiency. Finally, this technique helps us with interpretability, as each classifier focuses on a single label, making debugging and feature influence analysis more straightforward.

4.1 Model Choice

For the model, we employ a transformer-based neural network architecture [2], specifically using the RoBERTa model from Hugging Face transformers library

[3]. We chose the RoBERTa model because it has a proven track record of success in natural language processing tasks. Although it is not the focus of this paper, it is worth mentioning that its effectiveness comes from its advanced attention mechanism and its pretraining done on extensive and diverse text corpora.

4.2 Training Pipeline

After creating the training dataset with our ingestion pipeline (Section 2.1), we use it to fine-tune the RoBERTa model, using binary cross entropy (BCE) as the loss function. This is standard practice for MLC tasks that have been broken down into binary classification problems, as we explained. BCE optimizes each label independently, producing individual probabilities for each class, which is useful when labels are not mutually exclusive (e.g., *drama* and *romance*).

Multiple iterations of the model were cross-validated against a validation set to assess performance. The final model was selected based on their F1 score, precision, and recall. These metrics are crucial because MLC tasks often involve imbalanced data, where accuracy alone can be misleading. Precision ensures that assigned labels are correct, recall captures how many relevant labels are identified, and the F1 score balances both.

The selected model from our training workflow is saved as a model artifact in Google Cloud Storage. This artifact will be loaded when it is time to make predictions on new data.

4.3 Testing Pipeline

At the end of Airflow Orchestrating (Figure 3), the Genre Classification DAG will load the model artifact and use it to make predictions on the new movie data acquired (Section 2.2). Because the official genre has also been extracted from the OMDb API, we can check the performance of our model in real time.

5 Conclusion

In this paper, we have demonstrated how students can build an automated data pipeline using minimum resources in GCP to ingest and integrate various datasets that can be used to train and test different data science models. This type of group project work enhances communication and collaboration between students and also helps them reinforce their learning by applying their skills to solving interesting and practical problems. As part of the graduate-level course focusing on distributed data systems, the students were able to build a complete Machine Learning pipeline that requires no human intervention

by integrating MongoDB and Google Cloud services for storage and Apache Airflow for orchestration of various tasks. The paper illustrated how all the data used in the project is indexed and stored in MongoDB and is used to train and test a fine-tuned RoBERTa multiclass classification model.

Because of limited time during the course (7-week course) and limited GCP cloud credits (\$50 per person), there is room for future improvements in the automated data pipeline. For example, we could improve the architecture of the pipeline by closing the machine learning loop to allow for retraining the model using the new data, thus continuously improving the model. Furthermore, we could containerize our application using Docker and implement a fully cloud-based deployed web-application that can be accessed by the public. This was not done because of time restrictions and not enough Google Cloud credits. Leveraging unused metadata in the original dataset is another idea that could potentially improve the efficiency of the model. Finally, some work can be done to analyze the cost of maintaining such pipelines in production. Incorporating building even a small automated end-to-end pipeline into the course curriculum helps students to be prepared for related jobs in the industry by showcasing the work through their resumes and interviews.

References

- [1] David Bamman, Brendan T. O'Connor, and Noah A. Smith. "Learning Latent Personas of Film Characters". In: *Annual Meeting of the Association for Computational Linguistics*. 2013. URL: <https://api.semanticscholar.org/CorpusID:4986998>.
- [2] Judith E. Dayhoff. *Neural network architectures: an introduction*. USA: Van Nostrand Reinhold Co., 1990. ISBN: 0442207441.
- [3] Hugging Face. *Hugging Face's Transformer Library*. 2024. URL: <https://huggingface.co/docs/transformers/en/index>.
- [4] The Apache Software Foundation. *Airflow: Workflows as code*. 2024. URL: <https://airflow.apache.org/docs/apache-airflow/stable/index.html>.
- [5] The Apache Software Foundation. *PySpark: Python API for Apache Spark*. 2024. URL: <https://spark.apache.org/docs/latest/api/python/index.html>.
- [6] Brian Fritz. *The Open Movie Database*. 2024. URL: <https://www.omdbapi.com/> (visited on 11/15/2024).
- [7] IMDbPro. *Box Office Mojo by IMDbPro*. 2024. URL: https://www.boxofficemojo.com/weekly/2024W08/?ref=bo_wly_table_2 (visited on 11/15/2024).
- [8] Yinhan Liu et al. "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *CoRR* abs/1907.11692 (2019). arXiv: 1907.11692. URL: <http://arxiv.org/abs/1907.11692>.
- [9] MongoDB. *MongoDB Inc.* 2024. URL: <https://www.mongodb.com>.
- [10] Google Cloud Platform. *Google Inc.* 2024. URL: <https://cloud.google.com>.
- [11] University of San Francisco. *Data Science, MS | University of San Francisco*. 2025. URL: <https://www.usfca.edu/arts-sciences/programs/graduate/data-science>.

Interactive Tool for Quantum Cryptography BB84 Algorithm Demonstration*

Olivera Grujic
Computer Science
California State University, Stanislaus
Turlock, CA 95382
`ogrujic@csustan.edu`

Abstract

Quantum cryptography could be a solution to a challenge of a quantum computer being able to break RSA encryption potentially very soon. The BB84 protocol is elegant and simple, and yet not easy to grasp by students who lack background in probability, statistics or physics. In an undergraduate Fundamentals of Cybersecurity class, we provided students with hands-on practice of BB84 using the 3D printed laboratory kit that illustrates the algorithm in action. The exercise enabled students to roll binary dice and run their own experiments using lasers, light filters, and detectors, so they get empirical understanding how and why the stats work that make the encryption reliable. It also enabled students to work in pairs and exchange secret messages which contributed to active learning teaching practice. We collected students' self-reported learning benefits via survey. Most responded positively to the tool being useful, engaging, and increasing their understanding of the algorithm.

1 Introduction

A student asked what would happen if a quantum computer broke RSA encryption[7], which lead me, the author of this paper, to investigate. The BB84 quantum key distribution (QKD) scheme[1] was introduced a long time ago

*Copyright ©2025 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

and yet it isn't explained in several cybersecurity textbooks. Recent advances in artificial intelligence (AI) and quantum computing make RSA encryption more likely to be broken. Many universities are starting to offer courses on quantum computing[6] and post quantum cryptography[3]. I decided to fill in the gap in cybersecurity curriculum and teach quantum cryptography in my undergraduate level introductory cybersecurity class offered to CS majors who completed data structures, but not probability and physics.

Several freely available videos exist that explain the BB84 algorithm, even using animation or props such as candy to illustrate it. However, they do not provide the resulting outcome of an actual run. I introduced a hands-on lab assignment using the physical lab apparatus. I believe that when students complete their runs, they will get better understanding why and how the stats used by the algorithm work (with some risk involved). They also learn some concepts from wave physics by combining several different light filters. I surveyed students for their views. Most of them have responded very positively overall, considering the tool useful, engaging, and increasing their understanding of the algorithm. The student feedback is discussed in detail in section 3.

2 Materials and Methods

I briefly describe the algorithm, the toolkit and the assignment here.

The BB84[1], protocol is based on quantum mechanics, specifically on the difficulty in measuring photons, which is based on Heisenberg's uncertainty principle[5]. The only way to find out about the polarization of a photon is by using a polarizing light filter. For example, if the photon is vertically polarized, it'll pass through a vertical filter, but it will not pass through the horizontal filter. However, if it is diagonally polarized, it might or might not pass through the filter.

The initial version of the protocol is related to the counterfeiter's problem[4]: the counterfeiter must use the correct orientation of the Polaroid filter to identify a photon's polarization, but she does not know which orientation to use because she doesn't know the polarization of the photon. Bennett and Brassard (BB) applied this idea to cryptography in 1984, hence the name BB84[2]. Quantum key distribution allows Alice (the sender) and Bob (the receiver) to agree on a key and Eve (the eavesdropper) cannot intercept it without making errors.

Alice uses rectilinear and diagonal filters (encoded as 0 or 1) to send photons (binary message) to Bob. She switches between these two filters in an unpredictable way. Since Bob cannot know which filter Alice chooses, he chooses his own (rectilinear or diagonal), and his choice of a filter will be wrong approximately half the time. He only gets to measure the photon once because

a photon is indivisible and receives either 0 or 1. Therefore, Alice then talks to Bob (for example, over the telephone) and tells him which polarization she used for each photon, but not how she polarized each photon. They ignore the photons for which Bob used a different filter than Alice.

The sequence of bits for which Alice and Bob used the same filter is random, because it was derived from Alice's original sequence which was also random. Therefore, it is the secret symmetric key exchanged between Alice and Bob, which they can use to encrypt and decrypt their messages. If Eve were to intercept and measure the photons, she would've measured some using the incorrect detector and would've misinterpreted some of the bits that make up the sifted key.

The quantum cryptography toolkit was 3D printed and assembled locally by students, according to the guidelines provided by the Scientific Outreach team at the Institute for Quantum Computing at the University of Waterloo. There is a similar professional kit that can be purchased, but it costs almost ten times more. The idea behind this assignment wasn't so much to build it (though we did!), but to borrow it from physics and implement it in computer science and cybersecurity. The kit uses a laser to send photons (light), and a microcontroller (Arduino) based detector to receive them. The photons are sent and received through rectilinear and diagonal polarizing light filters.

The lab assignment is done in pairs. One student is a sender (Alice) and another one is a receiver (Bob). The coin toss decides which string of bits Alice is going to send, which filter she is going to use for each bit, rectilinear (+) or diagonal (x), and which filter Bob is going to use to receive it. Alice flips a coin 50 times to send each binary digit (0 or 1) and 50 times to use + or x filter. Bob flips a coin 50 times to decide whether he is receiving with + or x filter. Alice sends what she planned, Bob records what he received (0 or 1). The quantum key gets derived after Alice and Bob compare the filters used for each bit. They only keep the bits where they both used the same basis to form the cryptographic key.

All 21 students enrolled in class (undergraduates in their third and fourth year of study), submitted their lab reports with worksheets and their answers to seven lab questions and six questions from the questionnaire.

3 Results

Here, I show information on the participant's profile, their prior knowledge of the topic, and the effect of the assignment on their learning.

3.1 Participants Profile

The first two questions were intended to verify the expectation that students who completed the assignment are majoring in computer science and are not already familiar with quantum cryptography. All 21 students indicated they were computer science majors. 18 (86%) indicated they were unfamiliar with the concept (BB84 protocol or the lab kit). Only one student (<5%) indicated they were moderately familiar, and couple of students (<10%) indicated they were somewhat familiar. No student (0%) indicated they were very familiar with this topic.

3.2 Learning Preferences

The third question asked for learning preferences and allowed for multiple answers. The methodologies available for selection were: watching videos, interactive examples, building projects, reading text, quizzes, and group discussion.

All 21 students submitted their answers to this question. Each student selected three out of six methodologies (on average). There were 66 total responses. The top two categories combined include more than half of total responses 36 out of 63 (56%). Those are: interactive examples (selected by 20 out of 21 students or >95%) and watching videos (selected by 17 out of 21 students or >80%). The remaining categories received small percentage of all responses compared to the top two: reading text (14%), building projects (14%), group discussions (11%), and quizzes (6%). It is not surprising to see the quizzes ranked on the bottom. Figure 1 shows the distribution of students' responses to this question.

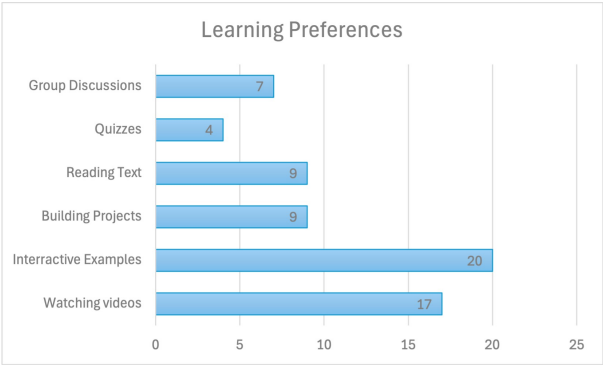


Figure 1: Preferred learning strategies when learning new concepts.

3.3 Effectiveness of the Tool

The fourth question asked to what extent a student felt their understanding of the subject improved after using the tool. The responses vary across the scale (extremely, very, moderately, slightly, and not at all), which is to be expected, as this is subjective. Most students, 18 out of 21 (>85%), thought the tool was helpful to some extent, while only three students (14%) didn't find it helpful at all. Among students who found the tool helpful, 10 (>47%) thought it was only moderately helpful, while two found it extremely helpful (10%) and three found it very helpful (15%). Figure 2 shows this distribution.

The fifth question inquired about the overall sentiment. Out of 21 students who responded, one of them indicated they didn't complete the assignment (4.76%). The responses of the remaining 20 students vary among all four options. Most, eight out of 20 (40%) found the lab fun, but not easy, six out of 20 (30%) found the lab fun and easy, four (20%) found it neither easy nor fun, and two (10%) found it easy, but not fun. Figure 3 shows the distribution.

3.4 Qualitative Results

The last question asked students to comment on the lab. All (21) students provided detailed responses. I highlight some to illustrate the themes that emerged.

Feedback on the assignment: “I enjoy how it allowed me to better understand the process of creating the key”; “Seeing how quantum encryption works firsthand was neat”; “Since I am a tactile as well as visual learner, seeing things and being able to work with my hands did wonders for my understanding”; “I do enjoy a lab with physical aspects to it, I learn better when working with

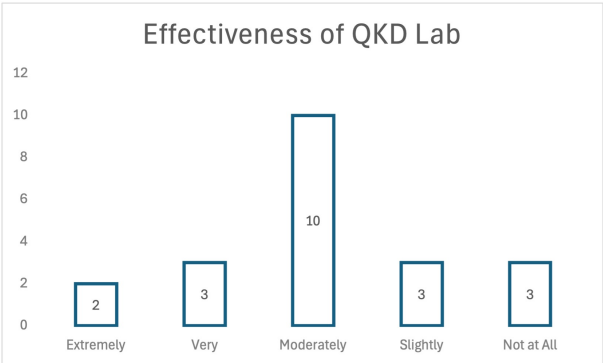


Figure 2: Improvement of BB84 understanding after completing the lab.



Figure 3: Student evaluation of the lab assignment.

concepts than simply reading about them”; “Did not do the lab. The in-person aspect of the lab was a huge turn off”.

Feedback on groupwork: “Questions were a lot easier to work through as a group, especially when having the physical equipment there as a tool”; “I enjoyed working in a group since I can enjoy the experience of learning with someone else and I love doing lab as that is when I learn best”; “I also thought having a partner for a lab made things far more enjoyable than working solo”; “I like working with other students, it’s easier to solve problems and share ideas”; “I really liked working with some of my classmates for this lab rather than doing it by myself. Overall, the lab was a great experience”.

Feedback on hardware: “It was fun messing with the wires, trying to fix the BB84 machine. The light sensors were very sensitive, so the room had to be dark”; “If it weren’t for the fact that we went through a debugging process of the lasers, light filters, and Arduino microcontroller, it would’ve been moderately fun”; “Overall, I didn’t care too much for this lab, maybe it was due to the issues with the testing kit, or simple lack of interest”; “I honestly couldn’t get the laser to behave the way we needed it to behave. So, after myself and a few other students attempted it, I just cut my losses”; “We ended up just completing what we could and leaving the rest because the results were not consistent”.

3.5 Discussion

The data confirms the expectation that the assignment had the potential for teaching this increasingly important and relevant concept. The preferred learning method of the students were the interactive examples, which is what the lab kit and the assignment were intended to provide. The second preferred learn-

ing method was watching videos, which is what the video tutorial I provided was intended to address.

This data suggests that almost a quarter of the students (23.8%) rated the toolkit above average (very and extremely useful), and almost three quarters of them (71.42%) rated it average or above (very, extremely and moderately useful). One of the three students who indicated they didn't find it helpful at all didn't complete the lab but filled out the survey. This suggests that exactly three quarters of the students who completed the lab rated it useful or above (15 out of 20, 75%). More than half of students, 13 out of 20 (65%), thought the assignment was fun. Perhaps engaging would've been a better choice of words to make it more in line with the principles of active learning.

The highest rated response is the one that matches my intentions and expectations, to make it fun, but not too easy so that learning takes place. I think the lab was doable, but not easy. The underlying physics and math concepts are difficult. My intention in that regard can be compared to teaching a person how to drive without teaching them how the engine works. And just like learning how to drive takes practice, getting the experiment to run in practice requires patience and persistence.

On the flip side, there were several students who encountered issues with the lab kit. While physics professors are likely used to dealing with malfunctioning lab kits, some CS faculty are not. Interestingly, there were some students who simply resisted having to use the apparatus located in the physical world, expecting all their assignments could be completed online.

4 Summary and Conclusions

I presented a hands-on lab assignment using the 3D printed laboratory kit that demonstrates the quantum cryptography protocol (BB84) in action. This algorithm is based on the principles of quantum mechanics and wave physics, and therefore difficult to understand by computer science majors.

The lab assignment enabled computer science students to run physics experiments using lasers, light filters, and detectors, so they gain empirical understanding of the reliability of the encryption produced by this algorithm. The toolkit also illustrated the quantum cryptography's main limitation: the protocol doesn't work for long distances (yet).

The assigned exercises implemented active learning teaching practice. Students worked in pairs. They were able to correctly answer several questions after observing the outcome of the experiments. They exchanged encrypted messages using the resulting key. Additionally, I demonstrated that a single toolkit (3D printed and assembled by the students for approximately 10% of the price of the original kit) could serve 21 college students within a week.

The survey results showed most students thought the tool was useful, engaging, and increased their understanding of the algorithm. They also indicated students preferred to learn using interactive tools. Next, I plan to test its limits on high-school and potentially middle schools' students.

5 Acknowledgements

I thank Michael Shindler for reading the manuscript, Jake Weigel and his student assistants for 3D printing and assembling the kit, and my students for providing valuable and detailed feedback.

As the surveys are anonymized, and the activity was part of classroom lab evaluation, this study falls outside of the purview of the Institutional Review Board protocol.

References

- [1] C. H. Bennett and G. Brassard. “Quantum cryptography: Public key distribution and coin tossing”. In: *Theoretical Computer Science* 560 (2014). Theoretical Aspects of Quantum Cryptography – celebrating 30 years of BB84, pp. 7–11. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2014.05.025.
- [2] C. Bernhardt. “Alice, Bob, Eve, and the BB84 Protocol”. In: *Quantum Computing for Everyone*. The MIT Press, 2019. Chap. 3, pp. 53–55.
- [3] T. J. Borrelli, M. Polak, and Sław Radziszowski. “Designing and Delivering a Post-Quantum Cryptography Course”. In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2024. Portland, OR, USA: Association for Computing Machinery, 2024, pp. 137–143. DOI: 10.1145/3626252.3630823.
- [4] E. A. Cohen and S. Singh. “The Code Book: The Evolution of Secrecy from Mary, Queen of Scots to Quantum Cryptography”. In: *Foreign Affairs* 78.6 (1999). ISSN: 0015-7120. DOI: 2307/20049567.
- [5] B. Ishak. “Quantum physics: what everybody needs to know, by M. G. Raymer”. In: *Contemporary Physics* 59.1 (2018), pp. 87–88. DOI: 10.1080/00107514.2017.1403467.
- [6] J. Liu and D. Franklin. “Introduction to Quantum Computing for Everyone: Experience Report”. In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2023. Toronto ON, Canada: Association for Computing Machinery, 2023, pp. 1157–1163. DOI: 10.1145/3545945.3569836.

- [7] R. L. Rivest, A. Shamir, and L. Adleman. “A method for obtaining digital signatures and public-key cryptosystems”. In: *Commun. ACM* 26.1 (Jan. 1983), pp. 96–99. issn: 0001-0782. doi: 10.1145/357980.358017.

Teaching Pointers through Visualizing*

Gongbing Hong, Yi Liu, and Frank Richardson
Information Systems and Computer Science
Georgia College and State University
Milledgeville, GA 31061
{gongbing.hong,yi.liu,frank.richardson}@gcsu.edu

Abstract

Pointers, a way for indirect data addressing, can be a very challenging topic for programming students, especially for those who are in the beginning stage of their studies. Very often, students maintain misconceptions about pointer/reference variables regarding what information they hold and how that information is used. As a result, they often struggle to produce working code for various programming projects that require the use of pointers. It is the authors' belief that *ad hoc* drawings, as often used in our classrooms, are not adequate. Professors should adopt a *standardized* set of drawing symbols in the teaching of this topic and use more drawings in their instruction to provide a visual representation of the relationship between pointer/reference variables and the referenced data. These visual representations will help negate common misconceptions students develop regarding the data access through pointers. We propose a standard set of drawing symbols to be used in textbooks, videos, and classroom instruction. Through case studies, we will illustrate the use of these drawing symbols.

1 Introduction

As a teaching topic, pointers (C/C++) and references (Java/C#) for indirect data access are known to be some of the hardest topics for beginning programming students. A survey with 37 respondents listed the topic of pointers at

*Copyright ©2025 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

a difficulty level of 6.054 out of 7 among the 28 programming topics surveyed [10]. Even students beyond entry-level can still struggle with pointer handling. Therefore, helping students better understand the concepts of pointers and the effects of pointer operations has always been of great importance and interest among instructors.

From our experience, the main reason why students often have difficulty in handling pointers is due to their misconceptions about pointer/reference variables, specifically, what information is stored in the variable and how that information is used to access the desired data. Teaching students at their early stage of learning to correctly visualize various attributes possessed by a variable is critical and will assist in rectifying theforementioned misconceptions. Additionally, the proper visual representation of these attributes will help in clarifying the effects of pointer operations. To achieve this, it is critical to develop a proper and standard set of drawing symbols for use in educational materials for the students.

Below is a piece of Java code showing a simple pointer assignment:

```
String first = new String("Hello"); // line 1
String second = first;                // line 2
```

After learning that Java object reference variables behave like pointers, students usually have no problem visualizing the effect of line 1 in the above code. In one class setting, 14 out of 19 students were able to visualize it correctly. However, when it comes to line 2, many students are often confused about the effect of the pointer assignment. When students were presented with two choices shown in Figure 1 and asked which one correctly illustrates the effect of the code on line 2, 12 out of 18 students responded with incorrect answer (Choice 2). If such misconceptions are allowed to continue, it is not hard to imagine that these students will have trouble in writing working code for projects that involve the use of pointers/references.

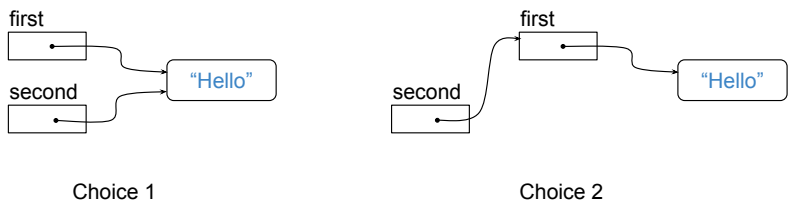


Figure 1: Which is the correct drawing for the pointer assignment?

We hypothesize that, due to the lack of a set of “standardized” drawing symbols for the teaching of indirect data addressing, students have not been *adequately* taught to handle pointers/references correctly and effectively. It is

our belief that a standardized set of drawing symbols for the learning of indirect data addressing is required to offer a clearer and more effective tool for teaching the usage and workings of pointer/reference variables. This set of symbols must provide a simple and easily understandable visual representation of pointer attributes with sufficient details to eliminate any ambiguities. Also, the set will need to be easily applicable to all aspects of pointer-related programming activities ranging from simple assignment to dynamic object allocation to the building of large data structures.

In this paper, we take an initial step in proposing a set of drawing symbols to be standardized and adopted by the computer science education community. We further present case studies to illustrate the use of these drawing symbols and their advantages in the teaching of pointers. We hope that this set of drawing symbols will be adopted and extended by the community for wider use.

2 Pointer Drawing Symbols

In this section, we introduce various drawing symbols for the illustration of pointers and their operations. Other drawing symbols that may not be directly related to pointers but are considered essential are also introduced.

2.1 Variables and Values Stored in Variables

A variable represents a *named* piece of memory (at a particular address). Depending on specific programming languages, the types of values that can be stored in a variable may be restricted. In Java, a variable can only store either a primitive value or a reference value. A Java variable cannot store an object directly. Instead, a Java variable can store a reference value to indirectly address an object. On the contrary, C++ does not have this restriction. C++ variables can store an object either directly or indirectly through a pointer. While the internal mechanics of pointers and references are different, conceptually they are the same. The information stored within a pointer/reference variable provides an indirect path to the desired data.

In our drawing, a variable is represented by a rectangular box (Figure 2) with the variable name above it. Optionally, an address or reference location can be supplied above the box following an @ symbol.

Representing variables with a rectangular box has its issues. It is important to stress to the students that such a box:

- can *never* be empty or emptied (the box always contains a value; it can be a random garbage value initially)

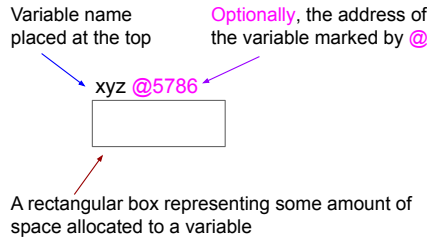


Figure 2: Variable drawing symbol.

- cannot hold a new value while preserving the old one (the old value gets erased when a new value is assigned)

Care needs to be taken when using a box analogy. To beginning programming students, the box analogy is not always clear [3, 11]. This is because, in real life, a box can be empty and can hold multiple items.

To illustrate that a variable contains a garbage value when it is first created, we represent the initial garbage value as “??” in the rectangular box for the variable in the drawing (Figure 3).

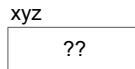


Figure 3: Symbol of a variable containing an initial garbage value.

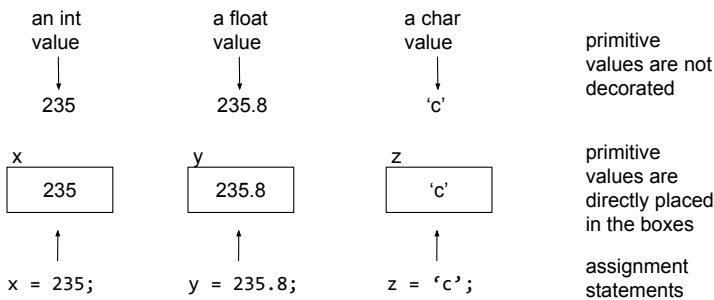


Figure 4: Symbols for primitive values and their placement in a variable

Primitive values are not “decorated” in the drawing and are directly placed into the rectangular box representing a variable (Figure 4). On the other hand, objects (and **struct** values) are “decorated” with a round-cornered rectangle

because they are composite (Figure 5) before they are placed in the rectangular box of an object/struct variable (Figure 6). Java does not support object variables while C++ does. In Java, all objects are stored in anonymous memory.

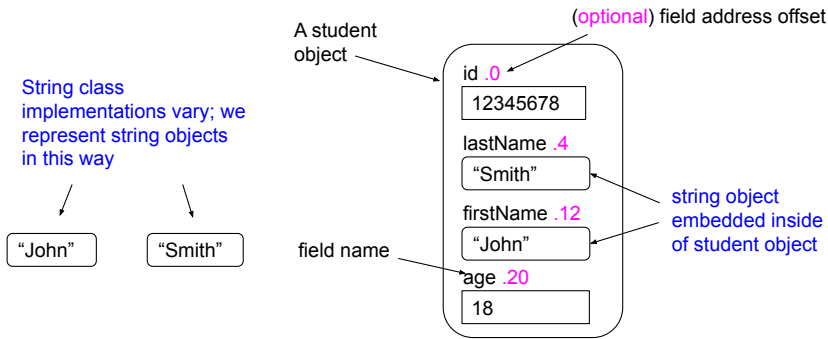


Figure 5: Symbols for objects/struct values

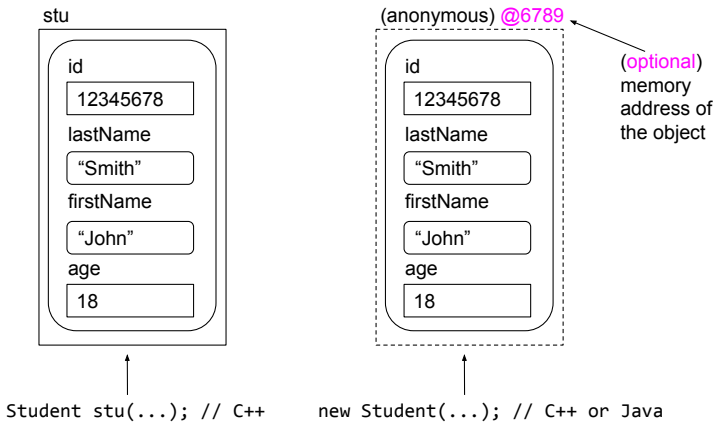


Figure 6: Symbols for objects/struct values placed in a variable

Depending on specific programming languages such as Java, embedding an object within another object may not be possible. However, this system allows users to informally draw objects as embedded within another object. If this informal drawing causes confusion, a formal drawing must be used (we will cover this later).

2.2 Pointer Variables and Pointer Values

There have been various efforts to hide the details of pointers from beginning programming students or even mature programmers *for* safety. However, to demystify pointers, students should be taught that pointer/reference variables are no different from primitive value variables except that they only store memory addresses (numbers) of other objects. In other words, pointers use addresses to indirectly reference objects stored elsewhere, locatable through those addresses. Students should be taught that handling addresses directly is *rarely* a good idea. In addition, addresses are not always known *a priori*, which is why arrows are used in the drawing to visualize pointers (values) (Figure 7).

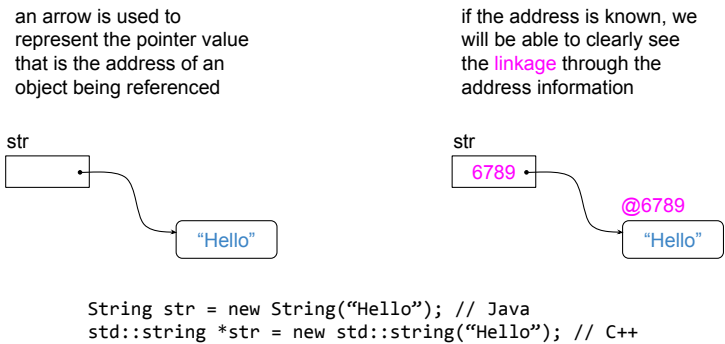


Figure 7: Symbol of a pointer referencing an object

When a programming language such as Java does not support object embedding, every field of the object will be either a primitive variable or a reference variable represented by a rectangular box in the drawing. One example of the drawing is given in Figure 8 where an object references other objects, creating the illusion of object embedding. Students should be taught to visualize this scenario, through which the difference between a shallow copy and a deep copy can be readily explained.

When a pointer variable contains a *null* value or contains a garbage value initially, it is drawn as shown in Figure 9. In the drawing, we depict an uninitialized pointer pointing to an “explosion” object to show the danger of using such a pointer. An informal drawing of a pointer, as indicated and widely used, should never be used in front of a student.

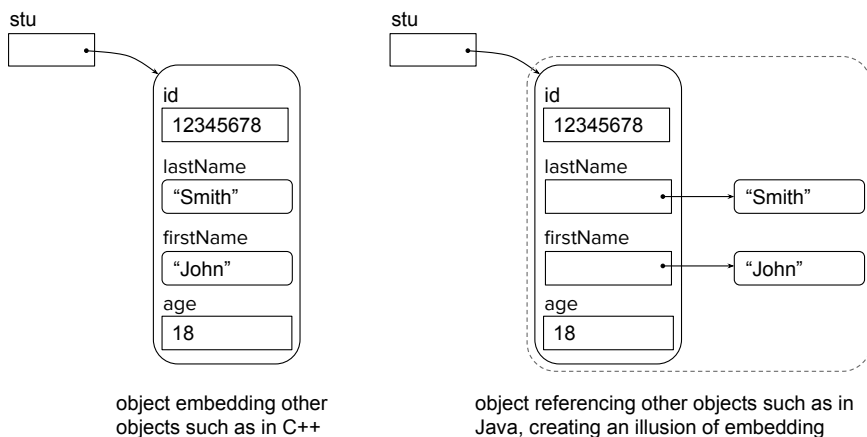


Figure 8: Object embedding vs object referencing other objects.

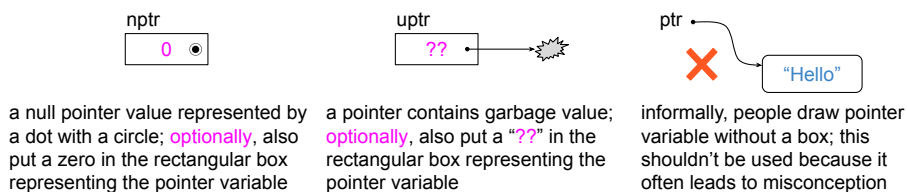


Figure 9: Symbols for null pointer and uninitialized pointer.

3 Case Studies

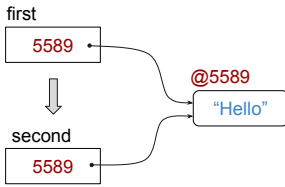
In this section, we present several case studies to illustrate the use of the drawing symbols we proposed and the advantages of these symbols.

3.1 Case Study 1: The Effect of Pointer Assignment

Pointer assignment has always been a mystery to beginning programming students. On one hand, students have been taught that pointer assignment makes a pointer point to an object. On the other hand, some students are surprised to learn that the correct choice is Choice 1 instead of Choice 2 in Figure 1, which shows that the net effect of assigning one pointer to another is *really* just to make two pointer variables point to the same object.

Since students already understand that an assignment statement is essentially a copy operation that transfers the value from the right-hand side to the left-hand side, the correctness of Choice 1 in Figure 1 can be clearly explained

```
String first = new String("Hello"); // line 1
String second = first; // line 2
```



Line 1: Students understand that the assignment makes the pointer point to the object. Using a made-up address, students learn it is the address info that establishes the linkage, which is visualized by an arrow from the pointer variable to the object.

Line 2: Pointer assignment simply copies the address info (not any different from other types of assignments). Therefore, line 2 makes the two pointers point to the same object because they have identical pointer value (address).

Figure 10: Effect of pointer assignment.

with a drawing like Figure 10, which uses a made-up address.

3.2 Case Study 2: What does this code do?

```
// Java code
class Node {
    int id;
    Node next;
}

Node head = new Node(); // (1)
head.id = 66;

Node ptr = new Node(); // (2)
ptr.id = 23;

head.next = ptr; // (3)

ptr = head; // (4)

head = ptr.next; // (5)
```

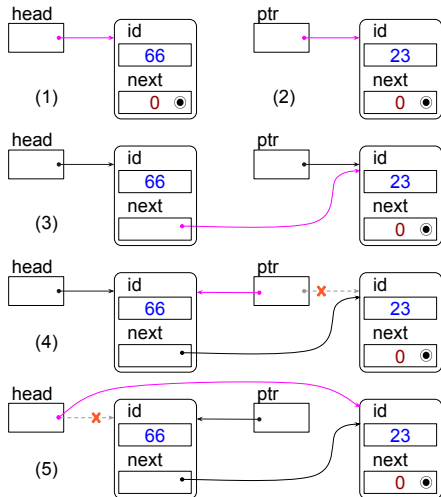
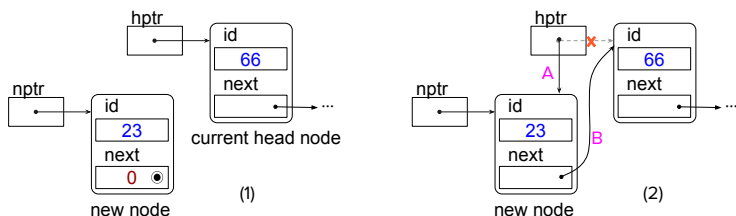


Figure 11: What does the code do?

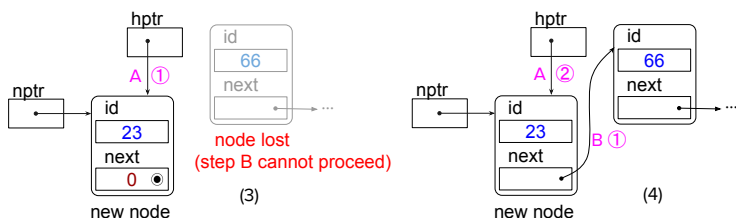
Once students have been taught the meanings of various symbols, as an exercise, students can be given a piece of code to figure out what the code exactly does *through drawing*. We found such exercises to be extremely helpful.

One example is the code given in Figure 11 along with the expected student response. As a result, students learn that the code links two nodes and then



Given the new node, make it the new head node for the linked list

Through drawing, it is not difficult for students to figure out two steps A and B are needed



Wrong order: A first, B second. Through drawing, students learn that if A goes first, current head node is lost forever because of no more reference to it. B cannot be done.

Right order: B first, A second. Through drawing, students learn that, when dealing with pointers that requires multiple steps, the order of the steps is critical. Whenever in doubt, always visualize it through drawing.

Figure 12: The order of pointer operations.

makes the two pointers exchange their nodes.

3.3 Case Study 3: This Order or That Order?

Take the example of adding a node as the new head of a linked list. Through drawing, students can often quickly figure out which individual operations are needed but are not always clear on which order of operations is the correct one. Students find a drawing, as shown in Figure 12, to be extremely helpful. The next natural step is to encourage students to come up with the code following the correct order of operations.

4 Related Work

Teaching pointers through drawings is not a new idea. Programming textbooks have used drawings to explain pointers, but no standardized set of drawing symbols has been adopted. The lack of a *standardized* set of drawing symbols for teaching pointers is the issue we are concerned with. We believe that the

computer science education community needs to come together and adopt a *standardized* set of drawing symbols for this challenging teaching topic.

In the classic K&R C programming book [6], the authors used simple drawings to explain how C pointers work in a way that is *fully consistent* with the actual memory model. A pointer variable is represented by a rectangular box labeled with the variable name. The box contains a solid circle-headed arrow pointing to another box representing another variable, an array, or a structure. To handle modern objects, *reference types*, which are reference-counted smart pointers, were invented. A more sophisticated set of drawing symbols is needed to deal with the complexity of reference types and objects. To some degree, the drawing symbols we propose can be seen as an extension of the drawing symbols used in the K&R book.

When it comes to explaining reference types, introductory programming textbooks (e.g., [4, 8]) use drawings sparingly, perhaps due to the fact that these modern languages try to hide the details of pointers. When variables and the objects they reference are drawn, they are often drawn indistinguishably: both variables and objects are drawn as rectangles, for example. In [4], the drawings of objects are not even consistent. Sometimes objects are drawn as rectangles and sometimes as round-cornered rectangles. The drawing symbols we propose clearly distinguish between a variable and an object.

To explain pointers in the context of certain data structures such as linked lists and binary trees, drawings are *indispensable*. Textbooks (e.g., [7, 1]) use drawings extensively. However, to communicate the same concept/idea, each book uses its own different set of drawing symbols. For example, [7] uses a UML-like rectangle to represent objects, while [1] uses a round-cornered rectangle. The UML standard was developed for software design. Its use of a rectangle to represent an object is not necessarily adequate for teaching purposes, with one criticism given in [5]. In addition, [7] labels a reference variable with $x =$, where x is the variable name. This is quite unconventional.

Teaching pointers is hard. It appears that no single strategy is effective for all students, and visualization through drawings remains a common strategy. Researchers have tried other methods to help teach pointers with varying degrees of success. For example, Yamashita et al. [12] developed a visualization tool that can visualize C pointers in simple running programs. jGRASP, a lightweight IDE for novice Java programmers, supports visualization [2] of Java programs including Java reference variables. Others have resorted to using computer games to teach C/C++ pointers [9].

5 Conclusion and Future Work

Beginning programming students (and even some students beyond entry-level) often have various misconceptions about pointers. As a result, they often struggle to write working code involving pointers.

In this paper, we argue that such a problem can be remedied from the start if professors use a right set of drawings to clearly visualize the meaning of pointers and the effect of pointer operations without ambiguity. One issue here is that we have not adopted a *standardized* set of drawing symbols with clear definitions. To this end, we took an initial step to propose a set of drawing symbols in the hope of being adopted and extended for wider use. We further presented case studies to demonstrate the use and advantages of this set of drawing symbols for the teaching of pointers.

Our initial experimental work in the classroom indicates that this set of drawing symbols is quite effective in illuminating the pointer concepts to the students. From a *Data Structures* course taught by the same instructor, we collected some data on the students' performance before and after the introduction of the drawing symbols. These students had an equivalent of one year of Java programming lessons before taking the course. For a pointer assignment problem slightly more complicated than the one shown in Figure 1, only 25% of the 16 students in Fall 2023 were able to answer it correctly before the introduction of the drawing symbols. After the introduction, the number of students who were able to answer it correctly increased dramatically (77% of the 18 students in Fall 2024 and 70% of the 24 students in Spring 2025). Similarly, before the introduction of the drawing symbols, less than 25% of the students in Fall 2023 were able to tell the difference between the `==` operator and the `equals()` method in Java. After the introduction of the drawing symbols, in a recent exam, for a question on this topic, the average grade for a class of 24 students was 84% correct. There do not appear to be any other factors that could have caused the improvement in the students' performance.

Future work will include the design and implementation of more in-class experiments to test the effectiveness of the drawing symbols. We would also like to involve other community members in the design and implementation of such experiments. We would also welcome feedback from the community on the proposed drawing symbols and their usage.

Acknowledgments

The authors would like to acknowledge the anonymous reviewers of the CCSC Midsouth Conference for their invaluable feedback and constructive suggestions, which helped enhance the quality of this paper.

References

- [1] Frank M. Carrano and Timonthy M. Henry. *Data Structures and Abstractions with Java*. Fifth Edition. Pearson, 2019.
- [2] James H Cross. “Using the new jGRASP canvas of dynamic viewers for program understanding and debugging in Java courses”. In: *Journal of Computing Sciences in Colleges* 29.1 (2013), pp. 37–39.
- [3] Benedict Du Boulay. “Some difficulties of learning to program”. In: *Journal of Educational Computing Research* 2.1 (1986), pp. 57–73.
- [4] Tony Gaddis and Godfrey Muganda. *Starting Out with Java: From Control Structures through Objects*. Fourth Edition. Pearson, 2019.
- [5] David Gries. “A principled approach to teaching OO first”. In: *Proceedings of the 39th SIGCSE technical symposium on Computer science education*. 2008, pp. 31–35.
- [6] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language*. Second Edition. Upper Saddle River, New Jersey: Prentice Hall P T R, 1988.
- [7] Elliot B. Koffman and Paul A. T. Wolfgang. *Data Structures: Abstract and Design Using Java*. Fourth Edition. Hoboken, New Jersey: John Wiley and Sons, 2021.
- [8] Y. Daniel Liang. *Introduction to Java Programming*. Tenth Edition. Pearson, 2015.
- [9] Monica M McGill et al. “If memory serves: Towards designing and evaluating a game for teaching pointers to undergraduate students”. In: *Proceedings of the 2017 ITiCSE Conference on Working Group Reports*. 2018, pp. 25–46.
- [10] Iain Milne and Glenn Rowe. “Difficulties in learning and teaching programming – views of students and tutors”. In: *Education and Information technologies* 7 (2002), pp. 55–66.
- [11] Alaaeddin Swidan, Felienne Hermans, and Marileen Smit. “Programming misconceptions for school students”. In: *Proceedings of the 2018 ACM Conference on International Computing Education Research*. 2018, pp. 151–159.
- [12] Koichi YAMASHITA et al. “Learning support system for understanding pointers based on pair of program visualizations and classroom practices”. In: *International Conference on Computers in Education*. 2020, pp. 658–663.

Scaffolding Mobile Programming with a Student-Centered and Asset-Based Framework*

Bilal Shebaro
Department of Computer Sciences
St. Edward's University
Austin, TX 78704
`bshebaro@stedwards.edu`

Abstract

This paper discusses the redesign of the Mobile Programming course with the goal of fostering a student-centered, inclusive learning environment. This redesign emphasizes asset-based teaching and integrates generative AI tools to align with modern technological advancements. By encouraging students to propose and develop mobile applications inspired by their personal interests, cultural backgrounds, and societal concerns, the course seeks to enhance engagement, creativity, and innovation while fostering a sense of ownership and meaningful connection to the learning process. This approach not only aligns with our university's mission to promote inclusive education but also leverages the diversity of the student body to create a dynamic and transformative learning experience.

1 Introduction

Over the past decade, universities worldwide have incorporated mobile programming courses into their curricula, enabling students to develop applications for the two dominant platforms: Apple and Android devices. These courses typically rely on traditional teaching methods, such as project-based or

*Copyright ©2025 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

application-based approaches, that focus heavily on platform features and capabilities. Students are often tasked with creating either one large application over the semester or multiple mini-apps, each showcasing a specific concept, skill, or feature [8]. These projects are commonly sourced from textbooks, online resources, or instructor-designed templates, leaving limited room for students to explore their unique interests or address societal concerns through their work.

Recognizing the limitations of traditional approaches, this paper introduces a redesign of our Mobile Programming course that centers on inclusivity and student learning. This reimagined course leverages asset-based teaching and integrates generative AI tools to empower students to propose and develop mobile applications that reflect their personal interests, diverse backgrounds, and societal issues they are passionate about addressing [3]. By allowing students to drive the content of the course applications and projects, the course not only enhances engagement and motivation, but also fosters a sense of ownership and personal connection to the learning process [5].

Another key limitation of traditional teaching approaches in mobile programming is the uniformity of the implemented applications: all students typically work on the same applications, whether through a single semester-long application or smaller, predefined assignments. While this method can effectively teach technical concepts, it often fails to engage students on a deeper level, as it does not account for their individual interests or experiences. In contrast, our redesigned course allows each student to work on their own unique mobile application projects, tailored to their personal assets, while applying the same core concepts taught in class. This approach not only fosters creativity and innovation, but also enhances motivation and engagement, as students see their ideas come to life in meaningful ways [1].

The remainder of this paper is organized as follows: Section 2 outlines the course redesign and structure of the Mobile Programming course, providing examples of its building blocks. Section 3 details the design of homework assignments and the strategies employed to ensure the appropriate use of generative AI. Section 4 focuses on the components of the course project, and the paper concludes with final remarks in Section 5.

2 Course Design and Structure

The redesigned Mobile Programming course focuses on developing applications for Apple's iPhone and iPad devices, guiding students through the entire process of designing and building a mobile app from start to finish. The course is structured as a hands-on, lab-style experience, where students learn to code in Swift, Apple's versatile and high-level programming language, and adopt the

Model-View-Controller (MVC) architecture to write clean, maintainable code using the Xcode Integrated Development Environment (IDE). The course is implemented in a 16-week, three-credit undergraduate college course. Classes are held twice a week, with each session lasting 75 minutes.

2.1 Student Asset Maps

To support this vision of a student-centered learning environment, our course redesign begins by tasking students with developing their own asset maps, which highlight the unique strengths, interests, and experiences they bring to the class as shown in Figure 1. The process of constructing asset maps begins with the instructor modeling the practice, explaining their purpose, demonstrating how they are built, and sharing their own asset map. This approach not only helps students understand how to create their own maps but also offers a glimpse into the instructor’s personal assets, fostering connection and mutual understanding within the classroom. After that student are asked to construct their own asset maps that will later serve as a foundation to identify topics that resonate with the students’ passions and backgrounds, ultimately driving their choice of mobile applications to develop throughout the course [9, 7].

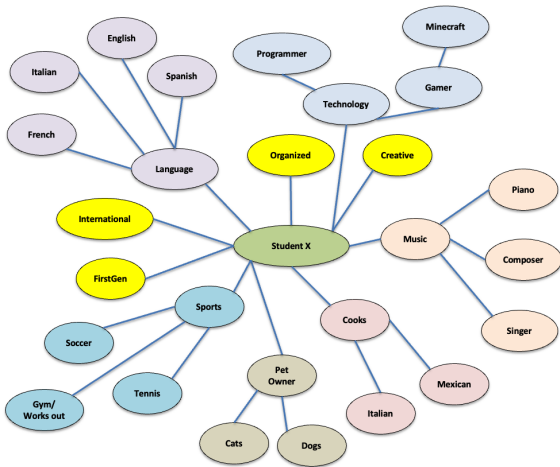


Figure 1: Asset Map example.

2.2 Structure: Foundations and Building Blocks

The course structure is divided into two key components: foundational concepts and building blocks. Drawing inspiration from architectural principles, where

a solid foundation must be laid before constructing the rest of the structure, this course applies a similar approach to mobile app development. In this context, the foundational concepts refer to the core methods for building a mobile application's graphical user interface. In iOS programming, this is primarily achieved using either the UIKit or SwiftUI frameworks. These foundational elements equip students with a strong technical base by introducing them to both tools, which are essential for effective app design and development.

The building blocks represent the additional structures built upon that foundation. In our course, these building blocks represent individual course concepts that students progressively learn and apply. Each building block is demonstrated through mini-apps that are directly inspired by the classroom asset maps. Each building block that can be built using either one of the two building foundations (UIKit or SwiftUI) will result in a mini-app. With students bringing diverse assets from each other into the classroom, the completion of each building block will lead to a variety of different mini-apps, showcasing students' assets while all utilizing the same concepts dedicated to the building block.

While teaching the foundational concepts in our current course design takes approximately two weeks, each mini-app resulting from each building block is usually completed over at most a two-week period, comprising four 75-minute sessions.

Once the foundations are set, the rest of the course is designed as building blocks. Figures 2 and 3 display two examples of building blocks that show how the course is designed to scaffold learning through a structured sequence of steps, ensuring students grasp foundational concepts before applying them to their individual mini-apps and projects [4, 10]. Below is a step-by-step breakdown of how each this building block is taught:

Learning Objectives: The block begins with clearly defined learning objectives that focus on critical technical skills.

Check-in: For approximately three minutes, students have the opportunity to connect with each other, fostering a collaborative and friendly atmosphere. This time is often used to share updates on their progress in their courses or casual conversations about activities from the past weekend. Following this, the instructor opens the floor for any questions about prior lectures or assigned homework, ensuring that any concerns are addressed before moving forward with new material.

Next, the instructor employs the “Five E’s” framework: Explore (I do), Engage (we do), Experience and Expand (they do), and finally Evaluate (in an assessment form).

Explore: The instructor introduces new concepts and demonstrates an example application.

Scaffolding Steps	Building Block # 4: Camera & Photo Album Frameworks
Learning Objectives	<ul style="list-style-type: none"> • Access and use the device camera to capture photos and videos. • Handle permissions and privacy settings. • Access and manage the photo library using the Photos framework. • Retrieve, display, and delete photos and videos from the photo album. • Handle camera and photo album permissions and privacy settings. • Save & load photos and videos to the device's photo library. • Implement photo and video editing functionalities.
Check-in	<ul style="list-style-type: none"> • Students check-in together • Check for questions about previous lecture • Check for questions about assigned homework
Explore	<ul style="list-style-type: none"> • Introduce students to the Camera and Photos frameworks to access the Camera app and manage the photo library. • Demonstrate retrieving, displaying, and managing photos and videos. • Show how to request and manage user permissions for accessing the camera and photo library. • Demonstrate techniques for saving, processing, and manipulating images and videos.
Engage	<ul style="list-style-type: none"> • Mobile application topic selection based on student assets • Discussion: Design app story & User Interface
Experience	<ul style="list-style-type: none"> • Coding / testing / debugging • Think-Pair-Share
Expand (class music playlist playing)	<ul style="list-style-type: none"> • In teams or individually • Extend app capabilities / self learn / stretch goals / exceed expectation • Assist others
Evaluate	<ul style="list-style-type: none"> • Assign homework/mini-project: <ul style="list-style-type: none"> ◦ Define milestones ◦ Share rubric ◦ Topic selection based on student assets
Check-out	<ul style="list-style-type: none"> • Run app on device or simulator • Exit ticket survey

Figure 2: Building block for Camera and Photo Album.

Scaffolding Steps	Building Block # 5: Maps & Location Services Frameworks
Learning Objectives	<ul style="list-style-type: none"> • Embed maps into iOS applications using MapKit. • Customize map annotations. • Handle user interactions with maps (e.g., tapping, dragging, zooming). • Access and manage user location data. • Implement location tracking and updates. • Apply custom map styling and appearance, and integrate custom markers and callouts.
Check-in	<ul style="list-style-type: none"> • Students check-in together • Check for questions about previous lecture • Check for questions about assigned homework
Explore	<ul style="list-style-type: none"> • Introduce the MapKit framework and its components. • Demonstrate how to create and manage map annotation. • Explain the Core Location framework and its functionalities. • Show how to access and update user location data.
Engage	<ul style="list-style-type: none"> • Mobile application topic selection based on student assets • Discussion: Design app story & User Interface
Experience	<ul style="list-style-type: none"> • Coding / testing / debugging • Think-Pair-Share
Expand (class music playlist playing)	<ul style="list-style-type: none"> • In teams or individually • Extend app capabilities / self learn / stretch goals / exceed expectation • Assist others
Evaluate	<ul style="list-style-type: none"> • Assign homework/mini-project: <ul style="list-style-type: none"> ◦ Define milestones ◦ Share rubric ◦ Topic selection based on student assets
Check-out	<ul style="list-style-type: none"> • Run app on device or simulator • Exit ticket survey

Figure 3: Building block for Maps and Location Services.

Engage: The instructor encourages the students to apply the new concepts in a meaningful way that aligns with their personal interests and their asset maps. This ensures their mini-apps are personally relevant. During discussions, both the instructor and students work together on the mini-apps story design and user interface before students could apply the concepts taught in the corresponding building block.

Experience: This provides students with a hands-on practice with coding and debugging their mini-apps, and collaborating together through the Think-Pair-Share method, encouraging peer interaction and idea exchange.

Expand: This step encourages students to extend their learning beyond the core concepts of the building block by exploring advanced or extended functionalities. Students may work individually or in teams to challenge themselves by implementing additional app capabilities or more sophisticated features. This phase is a crucial component of the course, as the field of mobile programming is vast and continually evolving, making self-learning an essential skill for mastering the domain.

During the **Experience** and **Expand** phases, where students primarily work independently, the instructor occasionally plays a classroom music playlist curated from students' favorite songs, collected through a survey conducted at the beginning of the semester. This adds a personal touch to the course, fostering a sense of belonging and showing students that they are valued and cared for, enhancing their excitement and connection to the class.

Evaluate: The instructor assigns homework or mini-projects with clearly defined milestones and an accompanying rubric. Students select topics aligned with their personal assets, ensuring the work is both meaningful and relevant. This step emphasizes meeting the instructor-defined milestones, providing a structured framework for students to demonstrate their understanding and progress.

Check-out: Upon the completion of this building block mini-app, students show off their running version of the completed mini-app on a device or the Xcode simulator to evaluate its functionality.

3 Homework Assignments and Generative AI

The homework assignments in this course are carefully designed to reinforce the concepts taught during each building block, ensuring students gain a solid understanding of the material. Following the completion of one or two building blocks, students are given a homework assignment that evaluates their comprehension and ability to apply the building block(s) concepts independently. Each assignment includes clearly defined milestones to guide progress, but students have complete freedom to choose the topic and content of their applications.

This flexibility allows students to tailor their work to align with their interests, backgrounds, and asset maps, fostering a deeper personal connection to the learning process.

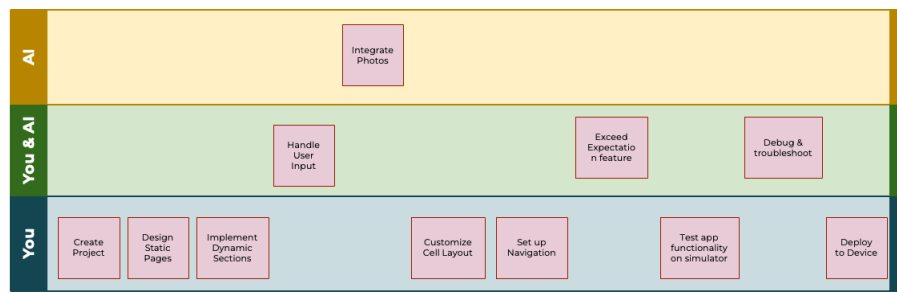


Figure 4: Homework tasks breakdown between student and AI (framework inspired from Dave Birss).

Generative AI plays a crucial role in supporting students throughout their homework and projects. As shown in Figure 4, each homework assignment is broken down into tasks that specify the extent to which AI can be used: AI-Generated Tasks, Student Collaboration with AI, and Student-Only Work. This structured approach aims for students to use AI responsibly while maintaining a strong foundation in programming and problem-solving skills.

4 Student Course Projects

A major focus of the course is the end-of-semester project, where students implement their best work in a comprehensive application. The course project requires each student to design a mobile application that incorporates one of the foundational concepts (UIKit or SwiftUI), utilizes at least four building blocks covered in class, and includes data persistence. The app should address a specific need or solve a meaningful problem for the student or a targeted user group, demonstrating greater complexity and originality than the in-class examples or homework assignments.

Students are asked to:

- 1. Propose an App Idea:** Students select a topic that resonates with their personal experiences, cultural background, or asset map. The app idea must address a real-world problem or fulfill a specific need and is subject to

instructor approval.

2. Prototype the App: Using industry-standard prototyping tools, such as Figma [6] or Marvel [2], students create a detailed blueprint of their app, focusing on user experience and interface design.

3. Implement the App: Students bring their app to life by coding it using one of the foundational frameworks (UIKit or SwiftUI) and incorporating their chosen set of building blocks and data persistence method.

The flexibility of the app topic encourages students to create applications that are personally meaningful. For many, this project represents an opportunity to develop an application they have always wanted to create but lacked the necessary skills prior to the course. At the end of the semester, students present their projects, demonstrating their technical achievements, creativity, and problem-solving abilities.

5 Conclusion

The redesigned Mobile Programming course transforms traditional teaching by fostering a student-centered, inclusive learning environment where students create apps inspired by their unique assets, such as about their pets, hobbies, favorite music and celebrities, hometowns, or places they were born or previously visited. By integrating asset-based teaching and generative AI tools, the course empowers students to develop applications that reflect their personal experiences and interests while mastering essential technical skills, and to thrive in the rapidly evolving mobile technology industry. Aligned with the university's mission to promote inclusive education and its strategic goals of courageous teaching and transformative learning, this course cultivates socially conscious, technically proficient graduates ready to make meaningful contributions to society.

References

- [1] Tracie Marcella Addy, Derek Dube, Khadijah A. Mitchell, and Mallory SoRelle. *What Inclusive Instructors Do: Principles and Practices for Excellence in College Teaching*. Routledge, 2021.
- [2] Marvel App. Marvel: Design and prototype for apps and websites. <https://marvelapp.com>.

- [3] Lori Assaf and Sean Justice. Asset-based computational thinking pedagogy early childhood teachers' use of asset-based computational thinking pedagogy: Centering students' expertise and life experiences. 04 2024.
- [4] University at Buffalo. Scaffolding content - office of curriculum, assessment and teaching transformation - university at buffalo. <https://www.buffalo.edu/catt/teach/develop/build/scaffolding.html>.
- [5] Nicholas Bowman. A brief intervention to improve college success and equity. *Science (New York, N.Y.)*, 380:457–458, 05 2023.
- [6] Figma Inc. Figma: Collaborative interface design tool. <https://www.figma.com>.
- [7] Erika L. Mein. Asset-based teaching and learning with diverse learners in postsecondary settings. 2018.
- [8] Mohammed Seyam, D. Scott McCrickard, Shuo Niu, Andrey Esakia, and Woongsup Kim. Teaching mobile application development through lectures, interactive tutorials, and pair programming. In *2016 IEEE Frontiers in Education Conference (FIE)*, pages 1–9, 2016.
- [9] Elisabeth A. Stoddard and Geoff Pfeifer. Diversity, equity, and inclusion tools for teamwork: Asset mapping and team processing handbook. 2020.
- [10] Northern Illinois University. Instructional scaffolding to improve learning - northern illinois university. <https://www.niu.edu/citl/resources/guides/instructional-guide/instructional-scaffolding-to-improve-learning.shtml>.

Transformative Experiences in Early CS Research: How Undergraduate Research Participation Shapes Identity and Belonging*

Renqingka (Rinchen Khar)¹ and Neena Thota²

¹College of Education

²Manning College of Information & Computer Sciences

University of Massachusetts Amherst

Amherst, Massachusetts 01002

`rrenqingka@umass.edu`

`nthota@umass.edu`

Abstract

Computer science students face numerous challenges during the initial stages of their academic journey, particularly those from underrepresented groups. Establishing a sense of belonging and developing a computing identity are crucial factors in student retention and success. This study investigates how participation in the Early Research Scholar Program (ERSP), a year-long research apprenticeship for early undergraduate students, influences these critical aspects of student development. Through a qualitative thematic analysis of bi-weekly reflective surveys collected over one semester from 24 first-year computer science students, we examined the evolution of students' identity as computing researchers and their emotional responses to the research process. Our findings reveal that creating a supportive and inclusive environment led to transformative experiences that significantly impacted students' sense of belonging and identity formation as researchers. Specifically, early exposure to computer science research, dual-mentoring from faculty and graduate students, and peer support emerged as key factors positively

*Copyright ©2025 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

influencing these transformations. By the program’s conclusion, 85% of participants reported positive changes in their identity and confidence as computer scientists, while 87% indicated an increased sense of belonging in the field. This study contributes valuable insights into effective practices for fostering diversity, inclusion, and student success in computer science education through early undergraduate research experiences.

1 Introduction

The increasing enrollment in Computer Science (CS) programs ([8],[9]) has led to a growing interest in providing early research experiences for undergraduate students. These experiences have been shown to offer numerous benefits, such as improved retention rates and course performance ([3],[2],[1]), and promoting greater diversity in computing in the long term ([4],[10],[12],[1]). Our institution participates in the Early Research Scholar Program (ERSP), a structured program designed to provide a diverse group of undergraduate CS students with a year-long, group-based research apprenticeship under the guidance of both faculty and graduate student mentors. While prior work by ERSP partners has examined the role of mentoring, the scalability of group-based undergraduate research models for underrepresented groups, and the long-term impact on diversity in computing ([3],[2],[12]), there is a lack of research on the factors contributing to the transformative experiences of undergraduate students in the program. This study aims to bridge this gap by investigating the impact of early undergraduate research participation on CS students’ identity as computer scientists and their sense of belonging in the field. We conducted a qualitative thematic analysis of students’ open-ended reflections on their evolving identity as computing researchers and their emotional responses to the research process. Our findings provide reliable insights into the experiences of ERSP students, highlighting the program’s benefits in fostering high confidence levels and a sense of belonging within the CS discipline. The results also corroborate earlier findings on the effectiveness of a dual-mentoring approach, with faculty and graduate student mentors playing complementary roles in supporting undergraduate researchers. By examining the factors that contribute to transformative experiences in early undergraduate research, this study contributes to the growing body of knowledge on best practices for promoting diversity, inclusion, and success in CS education.

2 Related Work

In 2014, the University of California San Diego’s Department of Computer Science and Engineering created the ERSP to offer early research opportunities

to computer science students ([4]). The program mainly admits students from underrepresented groups in CS, including women. The Center for Evaluating the Research Pipeline (CERP) conducted a yearly survey from 2014 to 2016 to assess the lasting effects of early formal research experiences on undergraduate computing students ([1]). The studies found that the experiences positively impacted students’ understanding of mentor support, retention in the major, sense of belonging, and identity change as researchers.

There is evidence that undergraduate research experiences contribute to developing students’ science identity in STEM disciplines, with students reporting being more confident, less intimidated, and leaving the program with a sense of accomplishment ([11]). Such experiences can be considered transformational, as they often lead to persistence in STEM disciplines and interest and enrollment in doctoral programs. In ERSP, the group-based model with dual mentoring has proved successful in showing that students in the program have more confidence and interest in a computing research career ([1]). In particular, women and minorities who are given opportunities for early research show an increased interest in continuing in the discipline and engaging in research ([3],[2],[4]).

A sense of belonging motivates undergraduates to thrive in their learning environments ([7],[14]). A long-term study with a larger sample size investigating the research experiences of first-year CS undergraduates ([12]) found that perceived mentor support predicted a sense of belonging for underrepresented students. Similar to other REU programs, ERSP has helped boost the participants’ self-efficacy and sense of belonging in CS, motivating them to pursue advanced degrees in the field and correlating with better grades for women and racially minority students ([1]). Undergraduate research experiences that include faculty-student interactions and mentoring have also proved beneficial for women and other underrepresented students in many disciplines, as well as in STEM and CS ([4]), ([5], [11]). A qualitative study of the mentoring practices in ERSP ([3]) showed that strong mentorship that provided project guidance and technical support was beneficial for a successful undergraduate research experience. The high retention rates in the program (98% for the cohort reported in their study) showed that early undergraduates benefit from project-related and emotional support.

3 Research Methodology

3.1 Study Context and Participant Selection

The Early Research Scholars Program (ERSP), launched at our institution in Fall 2021, provided first-year Computer Science (CS) students with valuable early research experiences. Following an open call for applications sent

to all incoming first-year students, we selected 28 participants using the established selection procedure ([4]) that looked at their motivation to do CS research, academic performance, and thoughts on how their participation contributed to diversity in CS. These students engaged in seven diverse research projects spanning key CS domains, including mobile app development, machine learning, human-computer interaction, blockchain, natural language processing, robotics, and cybersecurity. Each project offered participants hands-on experience developing essential technical, problem-solving, and collaboration skills in authentic research environments. The intentional diversity of projects enabled students to explore various CS specializations while gaining practical experience aligned with their specific interests. This early exposure to research methodologies has helped participants develop a deeper understanding of computer science beyond traditional coursework.

3.2 Research Design and Data Collection

To gain an understanding of the transformative experiences facilitated by ERSF, we conducted a qualitative study guided by the following research questions: RQ1: How does students' identity and confidence as computer scientists and researchers evolve throughout the program? RQ2: How does students' sense of belonging in CS evolve throughout the program? After obtaining ethics approval, we invited all 28 students enrolled in the Fall 2021 ERSF cohort to participate in the study. Twenty-four students provided informed consent and were subsequently included in the research. To minimize potential biases and power dynamics between researchers and student participants, we assigned specific roles to the research team members. The first author, a doctoral student in education, had no direct contact with the participants and was responsible for administering bi-weekly online surveys, analyzing the collected data, and collaborating on the writing of this report. The second author, a professor of computer science, introduced the students to the research process, served as the central faculty mentor, and assisted with weekly student meetings. To maintain confidentiality, the second author did not have access to the survey data until an anonymized dataset was provided a year later for analysis. Furthermore, the study design deliberately excluded demographic and identifying information, such as race and gender, from data collection to ensure participant confidentiality. Throughout the Fall 2021 semester, seven bi-weekly surveys were administered, each consisting of five open-ended questions framed as "reflections." The specific questions, which form the corpus of our data, were drawn from the following reflections:

- Reflection 1: Your identity as a researcher (Week 1)
- Reflection 2: Emotional responses to research (Week 3)

- Reflection 6: Emotional responses to research (Week 11)
- Reflection 7: End of Program Reflection (Week 14)

Each reflection prompt was designed to capture how participation in ERSF influenced students' identity as computer scientists and their sense of belonging in CS. The "Emotional response to research" reflection was administered at the beginning (Week 3) and towards the end of the semester (Week 11) to track the evolution of students' perceptions of the research process. These four selected bi-weekly surveys were chosen because they specifically focused on students' emotional experiences and identity-related perceptions in computer science. While all seven surveys provided useful data, these particular four surveys contained prompts that directly explored students' sense of belonging, emotional responses to programming challenges, and their evolving identities as computer scientists. This aligns with our paper's primary research questions on understanding the affective and identity-related dimensions of students' experiences in computer science. The other three surveys, while significant for the broader research project, focused on different aspects that fell outside the scope of this paper's specific investigation of emotional experiences and identity development.

3.3 Data Analysis

We employed an inductive thematic analysis approach, using participants' survey responses as our primary data source. Rather than relying on agreement metrics such as inter-rater reliability, we resolved uncertainties through discussion and consensus-building among the authors.

The analysis process followed a systematic approach to thematic analysis. Initially, the authors independently reviewed the survey responses using open coding ([6]), assigning descriptive codes to specific segments of text (e.g., "programming frustration," "peer support," "classroom participation"). Through weekly meetings, we discussed and refined these initial codes, ensuring consistency in our coding approach. As patterns emerged, we grouped related codes into broader categories, which then evolved into preliminary themes. For example, multiple codes related to students' changing perceptions of their programming abilities were grouped under the broader theme "Evolution of Identity and Confidence." Similarly, codes capturing students' interactions and feelings of inclusion formed the theme "Development of Sense of Belonging." After reaching consensus on these themes, we individually reanalyzed the data to ensure that the themes accurately represented the coded content. Finally, we examined how these themes aligned with our research questions, focusing on those most relevant to understanding students' identity development and sense of belonging in computer science.

4 Findings of the Study

This section presents findings of our study, addressing how students' identity and confidence as computer scientists and researchers (RQ1) and their sense of belonging in CS (RQ2) evolved throughout the ERSP program. To maintain confidentiality, all quotations are cited anonymously, with participants identified as S#.

4.1 Evolution of Identity and Confidence

Our analysis of the coded responses revealed that 85% of participants (19 out of 23) reported positive changes in their identity and confidence as computer scientists by the end of the program, while the remaining 15% (4 students) indicated neutral experiences. These changes manifested in three primary areas: identity as researcher, skill development, and self-efficacy. In terms of research identity development, 78% of participants demonstrated significant growth. Initial survey responses showed that only 13% of students had prior research experience or understanding. By the program's end, 78% of participants reported feeling more confident about pursuing research careers. This transformation is exemplified by S09's journey, whose perspective shifted from "ERSP... has not yet impacted this aspect of my identity" to recognizing research possibilities: "ERSP has made me feel more like I can pursue a career in Computer Science".

Regarding technical skill development, 91% of students reported acquiring new technical skills such as Android development, research communication, and problem-solving abilities. Analysis of pre- and post-program reflections revealed that initial apprehension about technical challenges, reported by 87% of students, transformed into confidence in specific areas. Students particularly noted growth in Android development (65%), research communication (82%), and problem-solving abilities (73%). Additionally, analysis of longitudinal responses showed that 83% of students developed stronger self-efficacy beliefs, as exemplified by S06's reflection: "I am capable of working in the real world." This transformation was similarly expressed by S09, whose perspective shifted from "ERSP... has not yet impacted this aspect of my identity" to recognizing research possibilities: "ERSP has made me feel more like I can pursue a career in Computer Science". The development of these self-efficacy beliefs appeared closely connected to students' increasing technical competence and growing sense of belonging within the computer science community.

4.2 Development of Sense of Belonging

Our analysis of survey responses revealed significant changes in students' sense of belonging throughout the program. Initially, 74% of participants reported experiencing imposter syndrome, 65% expressed anxiety about their technical abilities, and 48% worried about course load management. However, by the program's end, 87% of participants reported an increased sense of belonging, with 91% citing peer support as crucial to their experience. Our pre- and post-program reflections on emotional surveys 2 and 6 revealed that 83% (23 out of 28 students) positively viewed the program and felt it helped them belong to the CS community. The impact of faculty and mentor interactions was noted by 83% of participants, while 78% specifically highlighted the value of team-based research work. The program's emphasis on diversity and inclusion showed particular impact, with all women participants (14 students) highlighting the importance of gender representation. Overall, 91% of all participants positively commented on the program's inclusive environment. These findings align with reflections, such as S04's evolution from experiencing "sense of imposter syndrome" to finding "great friends" and building "rapport with professors."

5 Discussion

Our study yielded valuable insights into the experiences of undergraduate students participating in the Early Research Scholar Program (ERSP), an early research program designed to foster a sense of belonging and identity among underrepresented groups in computer science (CS). The findings reveal that students' identity and confidence as computer scientists and researchers (RQ1) and their sense of belonging in CS (RQ2) underwent significant transformations during the first semester of the program. Consistent with prior research on undergraduate research experiences ([1],[13]), we observed a growing sense of accomplishment among students and an increased interest in pursuing research careers. As students gained confidence, they became more willing to engage with challenges and developed stronger communication skills, motivation, and persistence—all crucial factors for success in research and academic programs. However, it is important to recognize that confidence alone does not guarantee success; the transformation of students' identity into that of computer scientists and their integration into the CS community played a vital role in their development. For many participants, pursuing a degree in computer science influenced their career opportunities, expanded their knowledge and skills, and deepened their understanding of the world around them. The cultivation of a sense of belonging emerged as a critical aspect of creating a safe and inclusive learning environment, particularly for marginalized individ-

uals. Our results highlighted the value students placed on peer support and the positive impact of an increasing sense of belonging on collaboration and inclusivity, especially for women-identified participants. These findings align with previous research emphasizing the importance of peer support in enhancing motivation, persistence, and emotional well-being among students ([7],[14]) Moreover, our study underscores the crucial role of mentorship in undergraduate research experiences. Participants reported that mentors provided invaluable technical and social support, guiding them through the research process and contributing to their overall success. The survey results demonstrate the unique and complementary roles played by faculty and graduate student mentors in supporting student researchers. The effectiveness of the dual-mentoring approach in our program highlights the importance of mentorship by both professors and graduate students in creating a supportive academic environment, corroborating earlier findings on the benefits of strong mentorship in undergraduate research ([3],[4],[5],[11]) This study contributes to the growing body of literature aimed at promoting undergraduate research experiences and fostering supportive, inclusive educational environments for all students. Our findings have significant implications for universities and colleges seeking to provide early research opportunities and promote diversity, equity, and inclusion in CS education. To facilitate transformative experiences for students on their path to becoming CS researchers, institutions should actively encourage peer support and graduate student mentorship. Peer support has been shown to enhance learning through collaboration and knowledge-sharing while building student confidence and self-esteem ([7], [14]) Furthermore, peer support can provide emotional support, which is particularly crucial for students grappling with stress or anxiety related to their academic workload. Mentors, in turn, offer knowledge, expertise, and guidance to help students navigate the challenges of their studies, develop essential skills, and build their confidence ([3],[4],[5],[11]) While this qualitative study offers valuable insights, it is essential to acknowledge its limitations. As with any qualitative research, the study is susceptible to validity threats related to subjective experiences. To mitigate these threats, we employed member checking and joint review to enhance the validity of our findings. Although the study's sample size is limited and the research was conducted in specific contexts, which may limit the generalizability of the findings to other populations or settings, it does not diminish the value of our results. Qualitative research often seeks to gain a deep understanding of complex phenomena that quantitative approaches may not fully capture. In conclusion, our study provides crucial insights into the experiences of undergraduate students participating in an early research experience program. The findings suggest that early research experiences contributed to students' career planning and confidence in their abilities while emphasizing

the importance of peer and mentor support in promoting a sense of belonging in their chosen field. Future research should consider conducting longitudinal studies with larger sample sizes to examine how students' identity and confidence as computer scientists and researchers evolve over successive semesters and among different cohorts participating in early research programs. By further investigating the factors that contribute to transformative experiences in early undergraduate research, we can continue to refine and improve programs designed to support and empower the next generation of diverse, confident, and successful computer science researchers.

6 Acknowledgments

We extend our heartfelt appreciation to the student participants in the program and the support from the lab professors and graduate mentors.

References

- [1] Christine Alvarado, Sergio Villazon, and Burcin Tamer. "Evaluating a scalable program for undergraduate CS research". In: *Proceedings of the 2019 ACM Conference on International Computing Education Research*. 2019, pp. 269–277.
- [2] Christine Alvarado et al. "Scaling and adapting a program for early undergraduate research in computing". In: *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education-Volume 1*. 2022, pp. 50–56.
- [3] Christine Alvarado et al. "The role of mentoring in a dual-mentored scalable CS research program". In: *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 2021, pp. 945–951.
- [4] Michael Barrow, Shelby Thomas, and Christine Alvarado. "Ersp: A structured cs research program for early-college students". In: *Proceedings of the 2016 ACM conference on innovation and technology in computer science education*. 2016, pp. 148–153.
- [5] Heidi B Carlone and Angela Johnson. "Understanding the science experiences of successful women of color: Science identity as an analytic lens". In: *Journal of Research in Science Teaching: The Official Journal of the National Association for Research in Science Teaching* 44.8 (2007), pp. 1187–1218.

- [6] Juliet Corbin and Anselm Strauss. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications, 2014.
- [7] Tierra M Freeman, Lynley H Anderman, and Jane M Jensen. “Sense of belonging in college freshmen at the classroom and campus levels”. In: *The Journal of Experimental Education* 75.3 (2007), pp. 203–220.
- [8] *Generation CS: Report on CS Enrollment* — [cra.org. https://cra.org/data/generation-cs/](https://cra.org/data/generation-cs/). [Accessed 26-03-2025].
- [9] Eric Roberts et al. “Rising cs enrollments: Meeting the challenges”. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. 2018, pp. 539–540.
- [10] Audrey Smith Rorrer, Joseph Allen, and Huifang Zuo. “A national study of undergraduate research experiences in computing: Implications for culturally relevant pedagogy”. In: *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. 2018, pp. 604–609.
- [11] Mark Santolucito and Ruzica Piskac. “Formal methods and computing identity-based mentorship for early stage researchers”. In: *Proceedings of the 51st ACM technical symposium on computer science education*. 2020, pp. 135–141.
- [12] Jane G Stout, N Burçin Tamer, and Christine J Alvarado. “Formal research experiences for first year students: A key to greater diversity in computing?” In: *Proceedings of the 49th ACM technical symposium on computer science education*. 2018, pp. 693–698.
- [13] Burçin Tamer and Jane G Stout. “Understanding how research experiences for undergraduate students may foster diversity in the professorate”. In: *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. 2016, pp. 114–119.
- [14] Vincent Tinto. *Leaving college: Rethinking the causes and cures of student attrition*. University of Chicago press, 2012.

What Are They Actually Teaching?: A Literature Review Of Gamification in K-12 Computer Science Education*

Paul Kim and Junseok Hur
Computer Science

Bridgewater State University
Bridgewater, MA 02324

p2kim@bridgew.edu, j1hur@student.bridgew.edu

Abstract

Gamification is a process of applying game-design elements (e.g., puzzle, storytelling, point scoring) into non-game areas such as education and health care, and several research projects utilized it to engage K-12 students in computing education where they can have playful experience while learning concepts regarding computer science. However, there was no report yet that summarized previous research projects to see what areas of computer science have been taught for K-12 students specifically via gamification, which can provide clear insight into what topics were less focused so far and need to be considered in the future. Therefore, in this paper, we review 23 published research articles and show which concepts and topics have been covered and how successfully they were taught through gamification. This report will help to see the trends of research in gamification for K-12 computing education and the possible future directions that can be pursued in this research field.

1 Introduction

Today, we can meet computer science techniques everywhere in the real world, such as artificial intelligence. And, as the computer science job market is grow-

*Copyright ©2025 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

ing, the demand for a workforce in computer science careers is higher than ever before. So, the importance of computer science education (computing education) is more demanding than before to prepare students for computer science careers. But since computer science can be perceived as a boring and less interesting subject among K-12 students [20], an effective setup of computing education is necessary to lower barriers to the field and to experience computer science topics.

For K-12 students (especially for students who do not have coding experience), the important thing in computing education is giving them an interest in computer science and helping them continue studying the area. One effective pedagogical tool is called gamification. Gamification is a process of gamifying non-game contexts such as education and health care with game-design elements like puzzles, storytelling, and point scoring [1]. Via gamification, K-12 students can have a playful experience while learning corresponding topics in computing education. Many research projects have helped K-12 students experience computer science topics using gamification. For example, Parham-Mocello et al. [22] used tic-tac-toe board games to teach abstraction/representation, and Wang et al. [28] developed a digital game in which students can learn about the inner workings of artificial intelligence.

Although various computer science topics have been covered for K-12 students via gamification, there is still no report that clearly shows what topics have been handled so far and how effectively they were taught via gamification. Such a report can help researchers, especially researchers who start research in this gamification area, to see which areas were less focused in K-12 computing education, and if areas were covered, how they were gamified and whether they were successful to provide a playful experience while students learn the corresponding concepts.

Therefore, in this paper, we selected and analyzed 23 research papers to see which computer science topics they covered and how they were taught via gamification. Furthermore, the effectiveness of each approach was investigated. This paper will show an analysis of these papers based on topics, gamification approaches, and effectiveness and give a sense of trends in this research area. It will also provide several suggestions regarding future directions in this K-12 computing education field.

2 Paper Selection

We mainly used four research databases, ACM Digital Library, IEEE Xplore, ScienceDirect, and ResearchGate. When we were searching for research papers, we narrowed the search topic to ‘K-12 computing education via gamification’. Although it is worth looking at papers that used gamification for college CS0,

we only included papers targeting K-12 students. We excluded papers that study game-design elements themselves in the gamification process even though they targeted K-12 students. For example, the paper [14] was not included because the purpose was to figure out how to make students focus more on gameplay not providing teaching via gamification. From the search, we selected 23 research articles.

3 Paper Organization

After selecting papers from research databases, to see clearly what and how they taught using gamification, we organized the papers based on which topic they taught. Then, we labeled each paper according to its gamification approach. In this section, we provide details of these categories: topic and gamification approach, used to organize the selected papers for our analysis.

3.1 Topic

The topic category is used to categorize papers based on which topic they covered via gamification. After reviewing the papers, we could find three topics ‘Computational Thinking’, ‘Basic Programming’, and ‘Computer Science Topic’, which were commonly covered by gamification for K-12 students. Thus, for the topic category, we sorted the papers based on these topics. The following sections explain about these topics to give a better understanding of them.

3.1.1 Computational Thinking

Computational thinking is a thinking skill that can break down a big problem into smaller parts and design algorithmic solutions for them. This is for training students to have an effective thinking process when pursuing a computer science field in the future. Computational thinking was highlighted by Wing [29] and is now known as an important skill not only in the computer science field but also in non-computer science fields such as English literature [24]. Research papers that developed the computational thinking of K-12 students in computing education were included in this category.

3.1.2 Basic Programming

Basic programming refers to programming in general, not a specific programming topic (ex. conditionals) or specific computer science topic (ex. machine learning). For example, Kelleher et al. [13] utilized the storytelling element

to engage more female students in studying programming. However, the paper did not mention specific programming concept they tried to teach so we included this paper in this basic programming category. But we excluded the paper like [2] as it covers a specific computer science concept, machine learning.

3.1.3 Computer Science Topics

This category is for papers that teach specific computer science topics to K-12 students using gamification. For example, the paper that taught about how logic gates work based on given values [7] and the paper that taught about computer memory model [21] belong to this category.

3.2 Gamification Approach

After sorting the papers based on three topic categories: computational thinking, basic programming, and computer science topics, we labeled them based on gamification approaches (plugged, unplugged, and hybrid) which are described in [4]. This section will explain each of the gamification approaches so that we can understand the label of a paper in our analysis later.

Plugged: This approach fully utilizes a technology like a computer. Papers in this category usually use digital games as a tool. Research projects in which students play digital games developed by research groups or from online resources like Code.org were included. But we also included the research works in which students develop their digital games to learn concepts.

Unplugged: This approach is opposite to the plugged one. It does not use technology at all and includes physical activities like playing board games with friends and building a structure using household materials like cardboard.

Hybrid: This is a mixed version of the previous two approaches. In this approach, unplugged and plugged can be combined sequentially. For example, Tsarava et al. [27] used unplugged treasure hunt activity and plugged programmable game (e.g., filling a glass with raindrops) in series to train computational thinking. But they can also be combined simultaneously. Kahila et al. [10] gave a classroom environment where students play plugged digital data profiling games while having unplugged collaboration with classmates.

4 Result

4.1 Topic and Approach

For this literature review, we selected 23 research articles published since 2007, and we organized them according to the topics they taught and the gamification approaches they used. Table 1 shows the papers organized according to these

categories and which gamification approach each paper used. Please note that for the computer science topic category (CS Topic in Table 1), we additionally provided which topics of the computer science field the papers taught.

Table 1: Paper organization according to categories of topic and gamification approach. The ‘Gamification Approach’ category is written as the ‘Approach’ category in the table.

Topic		Paper	Year	Approach
Computational Thinking		Kalelioglu [11]	2015	Plugged
		Moreno-León [19]	2016	Plugged
		Denner et al. [6]	2012	Plugged
		Lee et al. [15]	2014	Plugged
		Kazimoglu et al. [12]	2012	Plugged
		Tsarava et al. [26]	2018	Unplugged
		Del Olmo-Muñoz et al. [5]	2020	Hybrid
Basic Programming		Kelleher et al. [13]	2007	Plugged
		Tsarava et al. [27]	2017	Hybrid
		Parham-Mocello et al. [22]	2023	Unplugged
CS Topic	Computer Memory	Papastergiou [21]	2009	Plugged
	Game Development	Holly et al. [8]	2024	Plugged
	Introduction to Artificial Intelligence	Wang et al. [28]	2024	Plugged
	Cryptography and Secure Hashing Algorithm	Rayavaram et al. [23]	2024	Plugged
	Message Routing over Internet	Mano et al. [16]	2010	Unplugged
	Binary Number, Binary search, Sorting algorithm	Thies and Vahrenhold [25]	2013	Unplugged
	Loop and Conditional Logics	Merino-Armero et al. [17]	2022	Unplugged
	Logic Gates	Fees et al. [7]	2018	Unplugged
	Mechanism of Social Media	Kahila et al. [10]	2024	Hybrid
	Human Computer Interaction	Bollin et al. [3]	2021	Hybrid
	Introduction to Machine Learning	Adisa et al. [2]	2023	Plugged
	Bubble Sorting Algorithm	Hunter et al. [9]	2023	Hybrid
		Mladenović et al. [18]	2025	Unplugged

In Table 1, we can see that 7 papers out of 23 papers (32%) taught computational thinking. This skill has a larger number of papers compared to other topics which have at most 3 papers published (we are not considering CS Topic as one topic but as 12 individual topics). This is possible because computational thinking is an important skill for K-12 students which can help them be ready for studying computer science field. What we can also see in Table 1 is that less number of papers used a hybrid gamification approach. 11 papers used a plugged approach, 7 papers used an unplugged approach, and 5 papers used a hybrid approach. As described in [5] and [25], since the approach has the potential to provide better learning performance than the case of using only plugged or unplugged approaches, we hope to see more research projects exploring the potential in the future.

We also analyzed how many papers were published in which year (between 2007 and 2025) as shown in Figure 1 to see whether this research area is being conducted actively.

As seen in Figure 1, for most of the years, an average of 1 to 2 papers were published which can conclude that they are consistently researching gamifica-

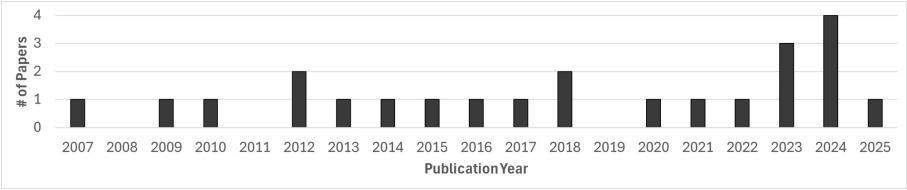


Figure 1: Chart showing how many papers were published in which year.

tion in K-12 computing education.

Figure 2 divided data of Figure 1 based on the topic category. In the table, CT, BP, and CS refer to Computational Thinking, Basic Programming, and Computer Science Topic categories, respectively.

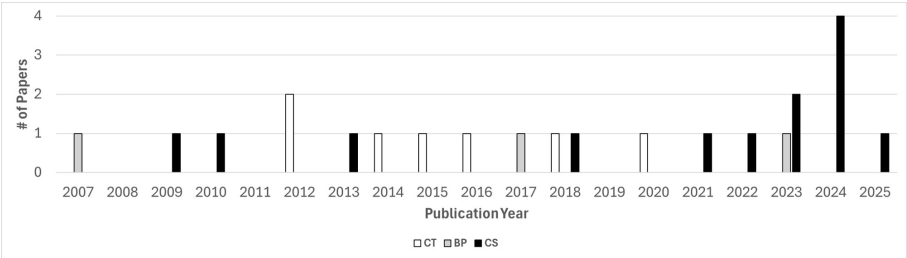


Figure 2: Chart showing how many papers were published for each topic category in which year. CT, BP, and CS refer to Computational Thinking, Basic Programming, and Computer Science Topic categories, respectively.

When we divided the data of Figure 1 in Figure 2, we could see one interesting point regarding papers of computer science topic category. 7 papers out of 13 papers (54%) of this category were published very recently (2023, 2024, and 2025). And the list of topics that they try to teach using gamification is:

- Social Media
- Game Development
- Artificial Intelligence
- Machine Learning
- Cryptography (Cybersecurity)
- Bubble Sorting Algorithm

It reveals that except for the sorting algorithm topic, researchers chose topics that became popular recently, which are also closely related to our real life (e.g., artificial intelligence [28]). Then, when the topic is introduced to students via gamification because they are already familiar with the topic, they

can understand its importance easily. Also, it will help them to see how computer science knowledge is related to their life and have more interest in the computer science field. Although it is still important to teach foundations of computer science such as computer memory model [21] and sorting algorithm [18], we need to note that teaching popular computer science topics via gamification is also important as it can help students to see that computer science field and their lives are closely related.

4.2 Effectiveness

In addition to analyzing papers based on topic and approach, we investigated how effective each approach was in teaching the corresponding topic. Table 2 gives a brief description of the gamification approach and its effectiveness with the target group level information. Please note that when we provided the targeting group level in Table 2, we followed the description given in the corresponding paper; some entries have specific age information, and other entries have school-level information like middle and high school. Also note that the mark section in Table 2 shows a level of effectiveness of each approach (S: Significant, M: Moderate, N: Not applicable).

9 papers that did not evaluate the learning outcomes of their approaches are marked as N in Table 2. They usually proved that their approaches were appropriate for their educational objectives. For example, Tsarava et al. [26] conducted a pilot evaluation of adult participants, not their target age group to check whether their approach provides any tension and negative emotions which can give high barriers to students to experience corresponding topic. Also, Mano et al. [16] surveyed middle school teachers and undergraduate students to see whether their approach is enjoyable.

5 papers (marked as M in Table 2) had moderate effectiveness. For example, in [22], students succeeded in understanding concepts regarding algorithms, but when they were given advanced practices like organizing instructions to complete an algorithm, they struggled to find the correct order. In [11], participants had no significant improvement in developing reflective thinking skills. But, they reported that students still enjoyed the gamification approach and were motivated to keep studying.

The papers marked as S in Table 2 demonstrated significant improvement in learning outcomes. For example, in [8], even students without prior knowledge could learn about the game development process by their gamified application. In [5], using the series of unplugged and plugged gamification approaches, participants could have the skills achievement and high motivation to continue learning. In [13], in addition to having successful learning outcomes, students were also motivated to spend extra spare time on programming. In [18], students had a significant retention rate in which they improved their knowledge

more in a test taken two weeks after the gamified activity.

In addition to evaluating learning outcomes, some papers provided extra interesting findings that we did not put on the table. Papers such as [11], [21], and [5] analyzed the difference in learning performance between male and female students with gamification and reported that there was no noticeable difference even though boys usually have more contact and interest in computing techniques. Moreover, Kalelioğlu [11] showed that female students completed programming activities more quickly and accurately than male students. Other papers such as [19] and [23] investigated whether there is a difference in learning performance between grade levels when they use a gamification tool. For example, Rayavaram et al. [23] saw that high school students were much better than middle school students in learning asymmetric cryptography and concluded that the topic is better suited to high school students. Moreno-León et al. [19] compared 6th grade and 2nd grade students and saw that 6th grade had more improvement in a survey. But since 2nd grade did not show a decrease in learning outcomes, the research group mentioned that it is still OK to introduce a gamified environment to lower grade level students.

We investigated the effectiveness of each approach in Table 2, but it is hard to say which approach is better as they have different topics to teach with different target groups. We hope that Table 2 can be used as a useful reference when researchers need to know what kind of approach was effective to teach which topic.

5 Limitation and Future Work

The literature review was based on restricted search criteria in which we selected gamification papers aimed only at K-12 computing education. Thus, it analyzed a relatively small number of papers in which our analysis result is hard to generalize. Our future research work will try to expand the search criteria, such as including other educational environments like higher education, and investigate whether gamification approaches and their effectiveness become different depending on educational levels. With the expanded search of papers, we look forward to seeing a more generalized picture of gamification tools in computer science education.

6 Conclusion

In this report, we selected 23 research articles searched from several research databases such as ACM Digital Library, IEEE Xplore, and ScienceDirect and analyzed which topics were covered and how they were taught by gamification to K-12 students. By analyzing the papers, we discovered several findings: 1)

Many papers in our selection used gamification to train students' computational thinking, as it is important not only for computer science but also for other non-computer science areas. 2) Most of the papers utilized plugged and unplugged gamification approaches, but fewer papers applied a hybrid approach (plugged + unplugged). 3) Recently, research projects have focused on topics that are closely related to our daily lives, such as cybersecurity and artificial intelligence. From the analysis result, we expect to see more research works that explore applying a hybrid gamification approach for K-12 computing education as the approach can utilize the benefits of both plugged and unplugged approaches. Also, we look forward to seeing more various projects that cover fast-growing computer science areas such as cloud computing so that K-12 students do not think of it as a mysterious magic box and have a better understanding to equip them with valuable knowledge and skills for their future careers.

Furthermore, we investigated the effectiveness of each gamification approach to better understand which approach was successful in teaching which topic. Some papers did not evaluate their learning outcomes and some papers made moderate effectiveness with their gamification approaches. Also, some papers made significant effectiveness in which students not only enjoyed the gamified activities but also had improvements in learning outcomes. Although not all gamification approaches provided remarkable effectiveness, it is not appropriate to say which approach is better than which approach, as they have different learning environments and different experiment settings. Hopefully, this report can be used as a useful reference for researchers who want to conduct research projects regarding gamification in K-12 computer science education.

Table 2: Table showing the description of the gamification approach and the effectiveness of each paper. Note that the mark column shows a level of effectiveness of each approach (S: Significant, M: Moderate, N: Not applicable)

Paper	Target	Approach Description	Effectiveness	Mark
Kaleoğlu [11]	Primary School	Playing online game from code.org	The approach did not have an effect on reflective thinking skills but gave students positive effect such as motivating to try more in programming.	M
Moreno-León [19]	Primary School	Developing computer game with Scratch	6th grade students had improved performance via the approach.	S
Demner et al. [6]	Middle School	Developing computer game	Participants were successfully engaged in computational thinking process that can prepare them for further study in computing.	S
Lee et al. [15]	10-15 yrs	Playing game designed in CTArcade system	The approach helped participants improve algorithmic thinking skill.	S
Kazinoğlu et al. [12]	High School	Playing digital game	It did not evaluate learning outcomes.	N
Tsarava et al. [26]	8-9 yrs	Playing board game	It did not evaluate learning outcomes.	N
Del Ohno-Muñoz et al. [5]	2nd Grade	Playing unplugged and plugged activities from code.org	The approach was not only beneficial in skills acquisition but also in motivation of participants.	S
Kelleher et al. [13]	Middle School	Alice Programming with storytelling element	The approach was successful in learning programming concepts; it motivated participants to spend extra time to continue programming.	S
Tsarava et al. [27]	3rd, 4th Grades	Playing series of unplugged and plugged activities	It did not evaluate learning outcomes.	N
Parham-Morello et al. [22]	Middle School	Playing physical game activity like tic-tac-toe	Participants succeeded in understanding topics but struggled with correctly organizing instructions for an algorithm.	M
Papastergiou [21]	High School	Playing digital game	The approach made much better learning performance than traditional lecture one.	S
Holly et al. [8]	Middle, High	Playing digital game	The approach made good learning outcomes even with students without prior theoretical knowledge.	S
Wang et al. [28]	3rd-12th Grades	Playing digital strategic game with LLM agent	It did not evaluate learning outcomes.	N
Rayavaram et al. [23]	Middle, High	Playing digital game with real-world scenario	Each topic of cryptography had different successful rate; (marked 60% in secure hashing algorithm test and marked 80% in symmetric cryptography test).	M
Mano et al. [16]	Middle School	Playing card game	It did not evaluate learning outcomes.	N
Thies and Vahrenhold [25]	Middle School	Playing activities from CSUnplugged.org	The approach helped participants reach to higher level learning stage regarding topics.	S
Merino-Armero et al. [17]	6th Grade	Imaginary robot activity with paper	The approach did not show significant improvement of learning outcomes in social science, as they are not related to computational thinking skills.	M
Fees et al. [7]	K-12	Building logic gate model with cardboard	It did not evaluate learning outcomes.	N
Kahila et al. [10]	5th, 8th Grades	Playing digital game with collaboration	The approach made good performance on profiling task but not good performance on reasoning about profiling.	M
Bollin et al. [3]	15-18 yrs	Developing digital games using HCI tool	It did not evaluate learning outcomes.	N
Adisa et al. [2]	Elementary	Playing digital game called S.P.O.T.	It did not evaluate learning outcomes.	N
Hunter et al. [9]	4th, 5th Grades	Physical dance activity with BBC Micro:Bit system	It did not evaluate learning outcomes.	N
Mladenović et al. [18]	Middle, High	Playing unplugged activities with cards	The approach did not affect students' immediate knowledge of topics but showed noticeable retention rate in which participants had improvements in their post-tests after two weeks.	S

References

- [1] A.C.T. Klock et al. A.M Toda. “Analysing gamification elements in educational environments using an existing Gamification taxonomy”. In: *Smart Learning Environments* 6 (2019). DOI: <https://doi.org/10.1186/s40561-019-0106-1>.
- [2] Ibrahim Oluwajoba Adisa et al. “S.P.O.T: A Game-Based Application for Fostering Critical Machine Learning Literacy Among Children”. In: *Proceedings of the 22nd Annual ACM Interaction Design and Children Conference*. IDC ’23. Chicago, IL, USA: Association for Computing Machinery, 2023, pp. 507–511. ISBN: 9798400701313. DOI: 10.1145/3585088.3593884.
- [3] Andreas Bollin et al. “HCI in K12 Computer Science Education – Using HCI as a Topic and a Didactic Tool”. In: *Proceedings of the 14th Biannual Conference of the Italian SIGCHI Chapter*. CHIItaly ’21. Bolzano, Italy: Association for Computing Machinery, 2021. ISBN: 9781450389778. DOI: 10.1145/3464385.3464717.
- [4] Gabriela de Carvalho Barros Bezerra et al. “Exploring the Use of Unplugged Gamification on Programming Learners’ Experience”. In: *ACM Trans. Comput. Educ.* 24.3 (Sept. 2024). DOI: 10.1145/3686165.
- [5] Javier del Olmo-Muñoz, Ramón Cózar-Gutiérrez, and José Antonio González-Calero. “Computational thinking through unplugged activities in early years of Primary Education”. In: *Computers & Education* 150 (2020), p. 103832. ISSN: 0360-1315. DOI: <https://doi.org/10.1016/j.compedu.2020.103832>.
- [6] Jill Denner, Linda Werner, and Eloy Ortiz. “Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts?” In: *Computers & Education* 58.1 (2012), pp. 240–249. ISSN: 0360-1315. DOI: <https://doi.org/10.1016/j.compedu.2011.08.006>.
- [7] Rachel E Fees et al. “Unplugged cybersecurity: An approach for bringing computer science into the classroom”. In: *International Journal of Computer Science Education in Schools* 2.1 (2018). DOI: 10.21585/ijcses.v2i1.21. URL: <https://www.ijcses.org/index.php/ijcses/article/view/21>.
- [8] Michael Holly, Lisa Habich, and Johanna Pirker. “GameDevDojo - An Educational Game for Teaching Game Development Concepts”. In: *Proceedings of the 19th International Conference on the Foundations of Digital Games*. FDG ’24. Worcester, MA, USA: Association for Computing Machinery, 2024. ISBN: 9798400709555. DOI: 10.1145/3649921.3649992.

- [9] Holly Hunter, Jamie Payton, and Christine Julien. “Expanding Elementary School Computer Science Education with an Introduction to Machine Learning Through Rhythmic Studies”. In: *Proceedings of the Twenty-Fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*. MobiHoc ’23. Washington, DC, USA: Association for Computing Machinery, 2023, pp. 463–467. ISBN: 9781450399265. DOI: 10.1145/3565287.3617623.
- [10] Juho Kahila et al. “Enhancing Understanding of Data Traces and Profiling Among K–9 Students Through an Interactive Classroom Game”. In: *Proceedings of the 19th WiPSCE Conference on Primary and Secondary Computing Education Research*. WiPSCE ’24. Munich, Germany: Association for Computing Machinery, 2024. ISBN: 9798400710056. DOI: 10.1145/3677619.3677635. URL: <https://doi.org/10.1145/3677619.3677635>.
- [11] Filiz Kalelioğlu. “A new way of teaching programming skills to K-12 students: Code.org”. In: *Computers in Human Behavior* 52 (2015), pp. 200–210. DOI: 10.1016/j.chb.2015.05.047.
- [12] Cagin Kazimoglu et al. “Learning Programming at the Computational Thinking Level via Digital Game-Play”. In: *Procedia Computer Science* 9 (2012). Proceedings of the International Conference on Computational Science, ICCS 2012, pp. 522–531. ISSN: 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2012.04.056>.
- [13] Caitlin Kelleher, Randy Pausch, and Sara Kiesler. “Storytelling alice motivates middle school girls to learn computer programming”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI ’07. San Jose, California, USA: Association for Computing Machinery, 2007, pp. 1455–1464. ISBN: 9781595935939. DOI: 10.1145/1240624.1240844.
- [14] Michael J. Lee, Amy J. Ko, and Irwin Kwan. “In-game assessments increase novice programmers’ engagement and level completion speed”. In: *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*. ICER ’13. San Diego, San California, USA: Association for Computing Machinery, 2013, pp. 153–160. ISBN: 9781450322430. DOI: 10.1145/2493394.2493410.
- [15] Tak Yeon Lee et al. “CTArcade: Computational thinking with games in school age children”. In: *International Journal of Child-Computer Interaction* 2 (July 2014). DOI: 10.1016/j.ijcci.2014.06.003.

- [16] Chad Mano, Vicki Allan, and Donald Cooley. “Effective in-class activities for middle school outreach programs”. In: *2010 IEEE Frontiers in Education Conference (FIE)*. 2010, F2E-1-F2E-6. DOI: 10.1109/FIE.2010.5673587.
- [17] José Miguel Merino-Armero et al. “Unplugged Activities in Cross-Curricular Teaching: Effect on Sixth Graders’ Computational Thinking and Learning Outcomes”. In: *Multimodal Technologies and Interaction* 6.2 (2022). ISSN: 2414-4088. DOI: 10.3390/mti6020013.
- [18] Monika Mladenović, Lucija Medak, and Divna Krpan. “Teaching the Bubble Sort Algorithm Using CS Unplugged Activities at the K-12 Level”. In: *ACM Trans. Comput. Educ.* 25.1 (Jan. 2025). DOI: 10.1145/3706120. URL: <https://doi.org/10.1145/3706120>.
- [19] Jesús Moreno-León. “Code to Learn: Where Does It Belong in the K-12 Curriculum?” In: *Journal of Information Technology Education: Research* 15 (2016), pp. 283–303.
- [20] Marina Papastergiou. “Are Computer Science and Information Technology still masculine fields? High school students’ perceptions and career choices”. In: *Comput. Educ.* 51.2 (Sept. 2008), pp. 594–608. ISSN: 0360-1315. DOI: 10.1016/j.compedu.2007.06.009.
- [21] Marina Papastergiou. “Digital Game-Based Learning in high school Computer Science education: Impact on educational effectiveness and student motivation”. In: *Computers & Education* 52.1 (2009), pp. 1–12. ISSN: 0360-1315. DOI: <https://doi.org/10.1016/j.compedu.2008.06.004>.
- [22] Jennifer Parham-Mocello et al. “Putting Computing on the Table: Using Physical Games to Teach Computer Science”. In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2023. Toronto ON, Canada: Association for Computing Machinery, 2023, pp. 444–450. ISBN: 9781450394314. DOI: 10.1145/3545945.3569883.
- [23] Pranathi Rayavaram et al. “Visual CryptoED: A Role-Playing and Visualization Tool for K-12 Cryptography Education”. In: *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE 2024. Portland, OR, USA: Association for Computing Machinery, 2024, pp. 1105–1111. ISBN: 9798400704239. DOI: 10.1145/3626252.3630963.
- [24] Cansu Tatar and Deniz Eseryel. “A literature review: Fostering computational thinking through game-based learning in K-12”. In: *The 42nd Annual Convention of The Association for the Educational Communications and Technology*. 2019, pp. 288–297.

- [25] Renate Thies and Jan Vahrenhold. “On plugging "unplugged" into CS classes”. In: *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*. SIGCSE ’13. Denver, Colorado, USA: Association for Computing Machinery, 2013, pp. 365–370. ISBN: 9781450318686. DOI: 10.1145/2445196.2445303.
- [26] Katerina Tsarava, Korbinian Moeller, and Manuel Ninaus. “Training Computational Thinking through board games: The case of Crabs & Turtles”. In: *International Journal of Serious Games* 5 (June 2018), pp. 25–44. DOI: 10.17083/ijsg.v5i2.248.
- [27] Katerina Tsarava et al. “Training Computational Thinking: Game-Based Unplugged and Plugged-in Activities in Primary School”. In: Oct. 2017.
- [28] Yuchen Wang et al. “Nemobot: Crafting Strategic Gaming LLM Agents for K-12 AI Education”. In: *Proceedings of the Eleventh ACM Conference on Learning @ Scale*. L@S ’24. Atlanta, GA, USA: Association for Computing Machinery, 2024, pp. 393–397. ISBN: 9798400706332. DOI: 10.1145/3657604.3664671.
- [29] Jeannette M. Wing. “Computational thinking”. In: *Commun. ACM* 49.3 (Mar. 2006), pp. 33–35. ISSN: 0001-0782. DOI: 10.1145/1118178.1118215.

Towards a Better Understanding of Proof By Induction*

Charles Hoot

School of Computer Science and Information Systems
Northwest Missouri State University
Maryville, MO 64468
`hoot@nwmissouri.edu`

Abstract

In this paper, some common difficulties that students experience with proofs by induction are discussed. These are used to form design goals for a WebApp to help students understand induction better. A WebApp implementing those goals is presented. Finally, results from a self reflection survey of students rating their understanding of induction are presented.

1 Introduction

Early on, proof by mathematical induction was identified as one of the harder concepts in mathematics. Some of the issues were not intrinsic to proof by induction but involved poor understanding of basic mathematic concepts such as exponentiation as mentioned by Baker in 1996.[1] Ernest in 1984 laid out three general categories of difficulties experienced by learners more directly related to the structure of a proof by induction.[2] To address these issues, researchers have proposed different theoretical approaches to understand induction. Ron and Dreyfuss, in 2004 proposed using visual/physical models to explain the recursion.[4] More recently based on literature reviews, Stylianides et al in 2017, concluded that there was a significant lack of intervention oriented studies on

*Copyright ©2025 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

proof by induction.[6] Most recently, in 2023, we see an intervention based on Parsons Problems (selecting and swapping the order of code fragments to accomplish a task) applied to induction with mathematical proof fragments.[3] Even with these new approaches, there are very few mentions at all of strong induction. My aim is to introduce a tool that will have objectives that will help students understand both weak and strong induction.

You can find the WebApp at the following URL: https://charles-hoot.github.io/Induction/induction_app.html

2 Specific Issues and Goals

As have been identified [4], there are three basic parts of an inductive proof that students struggle to understand:

1. The structure of the inductive proof
2. The induction basis
3. The inductive step

Within these general areas we will tease out some more specific concerns and transform those into goals that the WebApp will address.

2.1 Proof Structure

While proofs by induction from experienced mathematicians can be free form, students can struggle to complete all of the proof. For example, students may view the base case as an unimportant part of the proof and leave it out.[2] Another example is confusion between $P(n)$ and $P(n+1)$ and understanding that one does not need to prove $P(n)$ and that this is not a circular proof. It is expected that student proofs will relax over time as students get more familiar with inductive proofs, but initially more structure helps. *Goal: Provide strong sequencing through the proof. Clearly identify the induction hypothesis and what needs to be shown.*

2.2 Logical Dependency

A standard analogy for weak induction as presented in texts such as Rosen's book on discrete mathematics[5] is that induction is similar to a line of dominoes on edge. The first domino corresponds to the base case $P(0)$, the next is $P(1)$, and so forth. The proof of the base case is equivalent to knocking over the first domino. The inductive part of the proof shows that $P(k)$ implies $P(k+1)$ or equivalently knocking over a domino will knock down the next one. While this analogy will work for weak induction, it is not adequate for strong induction. We can still, however, express a dependence between various versions

of the predicate. Even for weak induction, giving a dependence provides an alternative view that can help students understanding. *Goal: Give students a tool that for a given integer value a , will show what it depends on by displaying one of three things: 1) $P(a)$ is outside the domain of values, 2) $P(a)$ must be shown directly, or 3) $P(a)$ depends on some set of values less than a .*

2.3 Application of the predicate

When constructing their proof, students will confuse the predicate with parts of the predicate. For example if the predicate is $P(n)$ is $1+2+3+\dots+n = \frac{n(n+1)}{2}$, students may misstate what $P(n_0)$, $P(k)$, and $P(k+1)$ are. They may loose terms from the sum, or drop the entire right or left hand side not realizing that what is left no longer has a truth value. *Goal: Automatically apply the predicate for the base case, induction hypothesis and inductive goal.*

2.4 Base case Identification

With weak induction, we need to identify a single base case. This is usually straightforward. For strong induction, we still need to identify a smallest base case, but there may be additional base cases that require a direct proof. This is further complicated because the number of extra base cases can change based on the inductive argument used. *Goal: Identify and report to the user any extra base cases automatically.*

3 Constructing the WebApp

The WebApp guides users to complete an inductive proof (weak or strong) in a series of well defined steps. There are some natural groupings of input and text widgets that users can interact with as seen in Figure 1 and Figure 2.. This addresses the goal of *Proof Structure*.

3.1 Step controls, Highlighting

As we go through the inductive proof the next step is highlighted and there is a text box on the upper right describing what the current step is. For steps with an input, if you change the value, updates will occur automatically. If you don't want to change the current value, there is a button at the bottom of the step description that will trigger any automatic updates and move you to the next step.

Induction Explorer

P(n) is

Postage of \$n\$ cents can be composed of 3 and 5 cent stamps

n_0 is

8

Base case

P(8) is

Postage of 8 cents can be composed of 3 and 5 cent stamps

Proof:

8 cents is one 5 cent and one 3 cent
Check

Proof finished

Next (No update)

Clear All

Sample Weak Induction

Sample Multi chain

Sample Tree

Figure 1: Upper half of the WebApp showing the predicate and base case for the multi-chain sample along with basic controls.

Multiple logic chains

skip back by 3

Induction hypothesis

Assume

P(8) ... P(10)

... P(k) is true when

$k \geq 10$

Show P(k+1) is true

Postage of (k+1) cents can be composed of 3 and 5 cent stamps

Proof:

To get postage of k+1 cents we use a 3 cent stamp
This leaves postage of $k+1-3 = k-2$
We know that k is at least 10, so k-2 is at least 8 and P(k-2) is in my list
By the IH postage of k-2 is composable and we are done
Check

Extra Base Cases

P(9) ... P(10)

9 cents is 3 by 3 cents
10 cents is 2 by 5 cents
Check

See dependencies for P(n)

14

P(14) depends on P(11)

Figure 2: Lower half of the WebApp showing the Induction and extra base cases for multi-chain sample with dependencies display.

3.2 Predicate, Base Case Value, Base Case Statement, Proof

This is the initial grouping for the WebApp shown in Figure 1. In this section, students will enter the predicate and the integer value n_0 of the base case. The WebApp then automatically generates the statement for the base case $P(n_0)$ addressing the goal of *Application of the Predicate*. The user is then directed to complete the proof which is inset to make it stand out and emphasize its importance. It is also reactive, in that if the user changes the predicate or base case value, the base case will be rewritten and the proof step will be highlighted.

3.3 Strategy, Inductive Hypothesis, Goal, Proof

In this section of the WebApp shown in Figure 2, we continue on to the Inductive portion of the proof. To start, the user needs to select the proof strategy they are going to use. The selections here are Single Chain (weak induction), Multi Chain (strong induction with a fixed step size back to the induction hypothesis) or General Tree (strong induction depending on two induction hypotheses). Once that decision has been made, for weak induction the WebApp will automatically generate the statement for the induction hypothesis $P(k)$ and the goal $P(k + 1)$.

For strong induction, we don't generate each of the statements that can be used as an induction hypothesis, but instead list the predicates with ellipses. (For example: $P(1)...P(k)$) In both cases, the user is directed to finish the inductive proof. This addresses the goal of *Application of the Predicate*.

3.4 Extra base cases

Depending on the strategy and values chosen, we may have extra bases cases that need to be completed. While all the base cases were listed before, we explicitly list out the extras in this section and have a space to list out the proofs. If there are no extra bases cases, this section will not appear. This addresses the goal of *Base Case Identification*.

3.5 Dependancies

In this section, the user has an input element where they can enter an integer value. The WebApp will then display the previous instances it depends on addressing the goal of *Logical Dependency*.

3.6 Sample Proofs

The piece of the WebApp are sample proofs that can be loaded into the app. There is one sample for each of the three proof strategies. Students can use the samples as a handy reference or as a basis for modification.

4 Survey

The following survey was given to 18 students in our Discrete Math course. The students had completed the unit on induction including a homework set and an assessment. For this survey, students were first asked to rate themselves on a scale from 0 to 10 with respect to the following seven statements.

Q1 I understand weak induction.

Q2 I understand strong induction.

Q3 I understand the basic proof pattern for weak induction.

Q4 I understand the proof patterns for strong induction.

Q5 I understand the dependence between different versions of $P(n)$ in an inductive proof.

Q6 I understand how to find base cases.

Q7 I understand how to use $P(n)$ to find $P(k)$ and $P(k+1)$

The students were then given the following directions for activities to use with the WebApp

1. Load the single chain (weak induction) sample and try different values for the dependency. You can enter a value directly. After that you can use the arrows to quickly move up and down.
2. Load the multi chain sample and try different values for the dependency. Make the skip value 5 and look at the chain of dependencies. How would you need to change the proof? What are the base cases needed in the proof?
3. Load the tree sample and again try different values for the dependency. Try changing the size of the left split. How small can it be? How large can it be? Did we need an extra base case?
4. Load the single chain and change the statement to $0 + 1 + 2 + \cdots + n = n(n+1)/2$
5. Enter a value for n .
6. Fill in base case proof.
7. Select single chain
8. Fill in inductive case proof.

After completing the activities, students were asked to do a post evaluation using the same seven questions as before. They were also given an open ended question soliciting advice and criticism on the functioning of the WebApp.

4.1 Numerical results

Of the 18 students, 14 responded to the survey. While this is a good response rate, the n value is too small to make strong conclusions. The following Table 1 gives the average and standard deviation for each of the questions.

Looking at the general understanding questions (Q1 and Q2), it is not surprising that students were more comfortable with weak induction (average 6.3) than strong induction (average 5.5). The average improvement for both was about the same with weak induction improvement (average 1.1) being slightly better than strong induction (average 0.9). On an individual basis, the amount of improvement reported was one of the values 0, 1, or 2. With weak induction only two students reported no improvement of their understanding. In contrast, five students reported no improvement of their understanding of strong induction.

The two pattern questions (Q3 and Q4) had basically the same values as for the first two questions. (The values are slightly higher as we might expect there to be more potential areas of misunderstanding besides the proof pattern. The final three questions (Q5, Q6, and Q7), are respectively looking at dependencies, finding bases cases, and use of $P(n)$ in a proof. The values are close to but a little smaller than those for the weak induction questions (Q1 and Q3). We do, however, see our largest pre to post average increases, with dependencies (average 1.1), base cases (average 1.2), and use of $P(n)$ (average 1.4)

Overall, the results from Q1 and Q2 show a modest improvement on the understanding of both weak and strong induction addressing the goal *Proof Structure*. The results from Q5 show a similarly modest improvement addressing the goal *Logical Dependency*. The results from Q6 show a modest improvement addressing the goal *Base Case Identification*. And finally, the results from Q7 show the best improvement addressing the goal *Application of the Predicate*. It is not surprising that this would be the easiest of the goals to improve on since it is mainly mechanistic.

4.2 Open ended responses

Almost all of the students responded with good, thoughtful feedback on what could make the WebApp better. Most of the students thought the WebApp was helpful and some reported that they would have liked for the WebApp to have been available during the unit on induction. The main suggestions for

Table 1: Pre and Post averages and StDev

	Statement	Pre Average/StDev	Post Average/StDev
Q1	Weak induction	6.3 / 1.86	7.4 / 1.91
Q2	Strong induction	5.5 / 1.95	6.4 / 2.17
Q3	Weak pattern	6.6 / 1.65	7.7 / 1.94
Q4	Strong pattern	5.7 / 1.86	6.8 / 1.37
Q5	Dependencies	6.1 / 2.25	7.3 / 1.14
Q6	Base cases	6.4 / 2.37	7.6 / 2.10
Q7	Using P(n)	6.4 / 2.31	7.7 / 1.98

improvement where more about the user interface as opposed to the content of the WebApp.

5 Future improvements

Aside from addressing the UI suggestions from the students, there are a number of things that could improve the WebApp. It would be useful to combine the pieces of the proof together into a single text output showing the complete proof. Incorporating LaTeX into the final proof construction would be a long range goal, especially as getting the predicate and proof pieces to comply with LaTeX could be tricky. Adding a link to the instructions from the WebApp to a page with suggested things to try would be helpful for anyone outside the course who does not have the survey available. More sample proofs would be useful as well. Expanding the WebApps capabilities to include structural inductions and recursive definitions would be similar topics that might not fit in the current framework, but might be added as a side page. One other line of inquiry, would be to see if it is better to have students use the WebApp from the start of the unit on induction or wait until later as presented in this paper.

6 Conclusions

A WebApp was constructed to meet goals addressing students understanding of proof by induction, both weak and strong. After doing a self assessment students were asked to perform some activities with the WebApp and then asked to reassess their understanding. The students also provided valuable feedback for incorporation in future versions of the WebApp. For a relatively modest investment in time, students reported modest improvements in the goals set for the WebApp.

References

- [1] JD Baker. “Students’ Difficulties with Proof by Mathematical Induction”. In: *Annual Meeting of the American Educational Research Association*. New York, NY, USA, 1996.
- [2] P. Ernest. “Mathematical Induction: A pedagogical discussion”. In: *Educational Studies in Mathematics* 2.15 (1984), pp. 173–189.
- [3] Seth Poulsen et al. “Efficiency of Learning from Proof Blocks Versus Writing Proofs”. In: Mar. 2023, pp. 472–478. DOI: 10.1145/3545945.3569797.
- [4] G . Ron and T. Dreyfuss. “The use of models in teaching proof by mathematical induction”. In: *Proceeding of the 18th conference of the International group for the Psychology of Mathematics Education* 4 (2004), pp. 113–120.
- [5] Kenneth Rosen. *Discrete Mathematics and its Applications*. 8th. McGraw Hill, 2018.
- [6] GJ Stylianides, AJ Stylianides, and K Weber. “Efficiency of Learning from Proof Blocks Versus Writing Proofs”. In: Mar. 2017, pp. 472–478. DOI: 10.1145/3545945.3569797.

Replication and Extension of Experiments with a Novel Weightless Neural Network*

Victor Nault¹ and David Wonnacott²

¹(*author, Haverford '24*)

`victornault@gmail.com`

²(*author's faculty advisor*)

Department of Computer Science

Haverford College

Haverford, Pa 19041

`davew@cs.haverford.edu`

Abstract

The field of Edge Intelligence has arisen to address latency and data security issues inherent to the older paradigm of deploying machine learning algorithms via cloud computing. However, the most-used machine learning algorithms are ill-suited to computationally-constrained Edge devices. Weightless Neural Networks are a lesser-known type of Neural Network that shows great promise for Edge Intelligence applications.

One particular recent Weightless Neural Network architecture, BTHOWeN, has open-source code, allowing for a deep analysis of their results. We report on our experiments conducted to replicate the original accuracy metrics for BTHOWeN's software implementation, as well as an extension where we performed an ablation study to determine the impact of several of the technical innovations used in the BTHOWeN architecture. Both were generally successful in confirming the original results presented, while revealing some deeper insights.

*Copyright ©2025 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

1 Introduction

Under a Cloud Computing paradigm, networked “terminal” devices send data to centralized data centers, where computation occurs, after which the results are sent back to the device. Edge Computing, as a concept, has emerged to deal with weaknesses inherent to Cloud Computing, such as unavoidable latency and potential for data privacy violations. Under this model, data from terminal devices is processed in auxiliary devices closer to the terminal devices (i.e. the “edge” of a network), such as a smartphone within a smart home. However, edge devices face limitations on computational power and energy usage, and any applications designed for the edge must take these into account [3]. Edge Intelligence is the application of artificial intelligence or machine learning within the edge, and has seen growth in recent years. However general machine learning algorithms are not designed for edge devices and their limitations [4].

An FPGA, or Field-Programmable Gate Array, is a device that consists of a mixture of hardware logic and configuration memory elements, that, together can be programmed or re-programmed into an arbitrary digital circuit [10]. For Edge Intelligence purposes, FPGAs provide significant advantages over the GPUs traditionally used as hardware accelerators: they’re more energy efficient, have lower latency, and better support concurrent processing for servicing multiple terminal devices [11].

In this paper, we review published work on a specific machine learning algorithm, the Weightless Neural Network (WNN), in relation to its relevance to Edge Intelligence, and discuss BTHOWeN [9], one particular WNN architecture intended for implementation on FPGAs. The paper that presents BTHOWeN, published in the proceedings of PACT ’22, has been given the “Artifacts Available” stamp by the ACM, as the authors (Susskind, Arora, Miranda, Villon, Katopodis, Araújo, Dutra, Lima, França, Breternitz, and John) have made their code for BTHOWeN publicly available, greatly increasing the reproducibility of their results, and enabling us to conduct our study. Specifically, we conducted first a replication experiment, in which we trained our own BTHOWeN models with the aim of matching the metrics Susskind et al. originally reported [9], and then extended our research with an ablation experiment, in which we disabled some of the technical innovations in the BTHOWeN model architecture in order to observe the impacts on accuracy and model size.

2 Background

Although few particulars of FPGAs or Edge Intelligence as a whole are necessary to sufficiently understand BTHOWeN, a solid grasp on Weightless Neural

Networks is ideal.

2.1 Weightless Neural Networks

A neural network consists of a sequence of layers; one way to think of each layer of a simple, feed-forward neural network is as a set of neurons, each taking a vector as input and computing a single scalar output value. The scalars produced by the set of neurons form the output vector [2]. In a Weightless Neural Network, each neuron is called a RAM Node, and each RAM Node operates like a truth table: n binary inputs are mapped to 2^n binary outputs, one for each combination of the inputs [9]. In other words, the inputs are mapped to outputs via an arbitrary boolean function. This mirrors the functionality of a core component of FPGAs: the Lookup Table (LUT), which can be used to perform an arbitrary boolean function on a number of inputs [6], which creates much simpler (and thus faster and less energy intensive) circuitry than what's needed for a standard neural network.

WNNs are also — in addition to being generally faster and more energy efficient — easier to train than other deep neural networks, as each entry in the training data only needs to be presented once to be fully learned, and as such training of WNNs can be up to 4 orders of magnitude faster than that of a standard neural network. However, they have two significant disadvantages: RAM nodes have no ability to generalize, and their size grows exponentially with the number of inputs. Nevertheless, with optimizations, WNNs can be well suited to Edge Intelligence applications [9].

2.2 Bleaching

RAM nodes learn entries in the training data after being presented with each only once. This allows for faster training than with regular neurons, but results in a greater tendency to overfit the training data, especially when the dataset is large. Bleaching is a technique that mitigates this. With bleaching, each RAM node has a counter associated with each bit input. During training, when an input is given to a RAM node, instead of setting it to evaluate to 1 for the pattern immediately, the counter is incremented. Then, after training, all bit patterns given to a RAM node a number of times greater than or equal to an arbitrary threshold are written to it, with the others discarded. The threshold can be decided after training, so that the best option may be selected empirically. Bleaching requires more memory during training, but inference works the same way as for a non-bleached model, so there's no additional memory use during inference, i.e. when implemented on an edge device [9].

2.3 Bloom Filters

Bloom filters provide a method of combining the accuracy of larger-input RAM nodes while maintaining the (much) lower memory requirements of smaller RAM nodes. A bloom filter is based on a hash table, and cannot confirm the *absence* of an element: the result 0 indicates the key is not a member, and 1 indicates it *possibly* is a member. The chance that a non-member element is reported as possibly a member increases as the bloom filter approaches its maximum capacity.

Bloom filters can be used in the place of RAM nodes to reduce memory usage. This is viable because larger RAM nodes are normally very sparse, so the probability of a bloom filter used as a substitute incorrectly returning a 1 for a non-member is low enough that accuracy is not significantly impacted [9].

3 BTHOWeN

BTHOWeN is a WNN architecture designed for Edge Intelligence purposes by Susskind et al [9]. It's similar to a previous WNN architecture, Bloom WiS-ARD, but iterates on previous improvements to WNNs via bleaching, bloom filters, hash function selection, and thermometer encoding. Their method of bleaching is standard, but there is no standard method to choose the counter thresholds. Susskind et al. chose one presented by Carvalho et al., which optimizes for minimizing runtime and memory usage [1].

Another small change involves the thresholds used in the architecture's thermometer encodings. In a standard (i.e. linear) thermometer encoding, every extra bit represents a fixed and equal integer increment over the previous number, e.g. 1111 would be twice as large as 0011. This is flexible, but the presence of outliers in the data will cause large numbers of bits to be allocated to represent very few values. BTHOWeN instead uses Gaussian thermometer encoding, in which each extra bit represents a variable increase in the underlying and approximated value dependent on the likelihood of the new value occurring within an assumed Gaussian, i.e. normal, distribution calculated from the inputs within the training data. The inputs are encoded so that, if a random input were chosen from the training data, it is equally likely to be 1111 or 0011 (or 0001, etc...), although 1111 would be larger. If the data presented during inference actually follows a normal distribution, small differences around the mean of the data are better accounted for than under linear encoding, at the cost of lower precision for values farther from the mean. Susskind et al. state that, during later testing, the normal distribution thresholds decreased mean error by about 13% [9].

More substantial is the choice to change the bloom filters used in prior work to instead use *counting* bloom filters. Just as bleaching adds a counter

Model Name	Bits /Input	Bits /Filter	Entries /Filter	Hashes /Filter	Size (KiB)	Test Acc.
MNIST-Small	2	28	1024	2	70.0	0.934
MNIST-Medium	3	28	2048	2	210	0.943
MNIST-Large	6	49	8192	4	960	0.952
Ecoli	10	10	128	2	0.875	0.875
Iris	3	2	128	1	0.281	0.980
Letter	15	20	2048	4	78.0	0.900
Satimage	8	12	512	4	9.00	0.880
Shuttle	9	27	1024	2	2.63	0.999
Vehicle	16	16	256	3	2.25	0.762
Vowel	15	15	256	4	3.44	0.900
Wine	9	13	128	3	0.422	0.983

Figure 1: Table of the 12 selected BTHOWeN models and related information. From Susskind et al., Table 3 [9]

for each RAM node input, a counting bloom filter has a counter for each of its inputs. During training, when a new bit pattern is presented to a counting bloom filter, it increments only the input(s) with the lowest counter value. For inference, the counting bloom filters are replaced with regular bloom filters, which are configured to return 0 instead of 1 for bit patterns previously seen too few times. This allows the accuracy increases of bloom filters and bleaching to be combined, whereas previous applications of bleaching could only apply to traditional RAM nodes.

Other innovations are also presented, but they are not directly relevant to our experimental results and are not described here; see the original BTHOWeN paper [9] or the author’s thesis [5].

3.1 Prior Experimental Results

Susskind et al. [9] created BTHOWeN models for nine datasets; they are listed in Figure 1, alongside the precision and composition of their architectural elements, memory usage, and accuracy. For most datasets, there was one model that clearly optimized memory usage while retaining high accuracy. The exception was MNIST, for which Susskind et al. selected three models with varying size to accuracy ratios, ideal for different use cases.

When compared to more typical quantized neural network models, specifically Convolutional Neural Networks (CNNs) and Multi-Layer Perceptrons (MLPs), BTHOWeN models maintain similar accuracy while consistently being much faster (66.7% to 90.0% latency reduction) and requiring less power

(56.2% to 90.7% total energy consumption reduction).

Compared to other WNN implementations, BTHOWeN also clearly shows improvements. On average, BTHOWeN models have 41% less error and are 51% smaller than Bloom WiSARD equivalents. For a previous WNN accelerator for MNIST models, the speed and energy efficiency is comparable, but is about 3% less accurate than the small BTHOWeN MNIST model and 5% less than the large [9].

4 Replication of Prior Experimental Results

Susskind et al. [9] provide both a software and hardware implementation of the BTHOWeN architecture – here we concern ourselves with the software implementation. Pre-trained “selected” models are provided in the BTHOWeN repository [8], and reproducing accuracy for these was trivial, as the related instructions in the repository README were simple and clear.

Reproducing the training procedure was slightly more complex, since, as noted in the BTHOWeN repository documentation, significant run-to-run variation in model accuracy is expected for the smaller datasets [8]. We briefly explored the possibility of eliminating this variation by ensuring that all our test runs were performed with the same versions of relevant libraries, and giving fixed seeds to all random number generators.

However, we still experienced run-to-run variation, and abandoned this attempt without an intensive search for the source of the variance, instead running the experiment repeatedly in hopes of replicating the published results. We trained models for each dataset a minimum of 5 times and added additional groups of 5 trainings until we reached either 30 trainings or got within a percentage point of the equivalent pre-selected model’s accuracy. (For simplicity, we simply ran 30 trainings for the especially small datasets.)

We additionally collected latency and power metrics for both our replication and extension experiments, and attempted replication of the BTHOWeN hardware implementation. However, due to space constraints, they have been omitted from this paper; they can be found in the (first) author’s undergraduate thesis for those interested [5].

5 Experiments to Add New Detail

As an extension to our replication experiments, we conducted an ablation study, where we “rolled back” some of the innovations Susskind et al. presented, via selectively disabling the code that provided functionality for them, in order to measure their impact on model accuracy and model size.

As mentioned previously, the advantage of Gaussian Encoding is that outliers in the training data do not skew the results as much. Testing to see how much this improvement actually affects the accuracy of the models was relatively easy: functionality for linear encoding is already present within the original BTHOWeN code, it just needs to be re-enabled. Results for Linear Encoding with MNIST are briefly mentioned in the original paper, but we extend to the other datasets and combine with disabled bleaching.

As also mentioned previously, bleaching prevents oversaturation by requiring an example to be seen more than once before it is learned. We measured the impact of their innovation of combining bleaching and bloom filters by forcing all of the thresholds for the RAM nodes to 1, which is equivalent to their original behavior of learning after one example.

The procedure for training models without Gaussian encoding, without bleaching, and with neither, closely resembles that of training the unmodified models. However, since there is no longer an original metric to replicate, we mirrored the number of training iterations needed to achieve near-maximum accuracy with a regular model.

As a final experiment, because the hyperparameters chosen for all the models were those selected to optimize their original (non-ablated) version, a hyperparameter sweep was done for a no-Bleaching model with the MNIST dataset, aiming to meet or exceed the accuracy of the original MNIST-Small model. We chose MNIST because it's the dataset Susskind et al. generally use for non-comprehensive experimentation (e.g. for Gaussian encoding), and because the variation in run-to-run accuracy common in other datasets would greatly hinder accurately choosing hyperparameters that maximized accuracy relative to model size, especially because variation increases substantially without Bleaching.

6 Results

The accuracy results of the replicated software inference runs, with the pre-selected models, are 1:1 with the those of the original paper's, with the odd exception of MNIST-Medium, which sees a slight decrease in accuracy. These results are constant across several inference runs.

The accuracy results of our software inference runs with custom-trained models (with identical hyperparameters to the pre-selected models) are detailed in Figure 2. Of note is that even with many runs, the models trained from scratch couldn't achieve accuracy within a percentage point of the pre-selected models for the Ecoli, Vehicle, and Wine datasets. More surprisingly, for Letter and Vowel, models were created that performed significantly better, with one of the Vowel models being almost 2% more accurate than the original!

Model	Original	Replication	Runs	Discrepancy
MNIST-Small	0.934	0.933	5	-0.1%
MNIST-Medium	0.943	0.941	5	-0.2%
MNIST-Large	0.952	0.953	5	0.1%
Ecoli	0.875	0.857	30	-1.8%
Iris	0.980	0.980	30	0%
Letter	0.900	0.905	10	0.5%
Satimage	0.880	0.873	20	-0.7%
Shuttle	0.999	0.997	10	-0.2%
Vehicle	0.762	0.752	30	-1%
Vowel	0.900	0.918	10	1.8%
Wine	0.983	0.966	30	-1.7%

Figure 2: Accuracy results of BTHOWeN software inference with custom-trained models.

Model	Best Replication	Best No Gauss.	Discrepancy
MNIST-Small	0.933	0.926	-0.7%
MNIST-Medium	0.941	0.936	-0.5%
MNIST-Large	0.953	0.944	-0.9%
Ecoli	0.857	0.866	0.9%
Iris	0.980	0.920	-6%
Letter	0.905	0.800	-10.5%
Satimage	0.873	0.858	-1.5%
Shuttle	0.997	0.944	-5.3%
Vehicle	0.752	0.741	-1.1%
Vowel	0.918	0.873	-4.5%
Wine	0.966	0.966	0%

Figure 3: Most Accurate Replicated Regular model versus Most Accurate Replicated model without Gaussian Encoding.

6.1 Novel Software Experiments

For the software extension with original hyperparameters, results are shown in Figures 3 through 6. Results are highly variable across datasets, and may not be easily generalizable. For instance, Ecoli without bleaching reaches a higher accuracy than the pre-trained “best” model provided by Susskind et al., while MNIST-Small without bleaching has its accuracy severely dented, and likewise for Letter without Gaussian encoding. Interestingly, Susskind et al. state in their paper that with Gaussian encoding, the mean error of the MNIST models is reduced by 12.9%, but the MNIST models are among the least affected by its removal, and the reduction in accuracy does not seem that steep. Training with neither Gaussian encoding nor Bleaching unsurprisingly results in the greatest decreases in accuracy, and is the only variation to not show increases

Model	Best Replication	Best No Bleach.	Discrepancy
MNIST-Small	0.933	0.808	-12.5%
MNIST-Medium	0.941	0.911	-3%
MNIST-Large	0.953	0.941	-1.2%
Ecoli	0.857	0.884	2.7%
Iris	0.980	0.960	-2%
Letter	0.905	0.906	0.1%
Satimage	0.873	0.849	-2.4%
Shuttle	0.997	0.999	0.2%
Vehicle	0.752	0.748	-0.4%
Vowel	0.918	0.903	-1.5%
Wine	0.966	0.932	-3.2%

Figure 4: Most Accurate Replicated Regular model versus Most Accurate Replicated model without Bleaching.

Model	Best Replication	Best No Gauss. or Bleach.	Discrepancy
MNIST-Small	0.933	0.762	-17.1%
MNIST-Medium	0.941	0.900	-4.1%
MNIST-Large	0.953	0.888	-6.5%
Ecoli	0.857	0.857	0%
Iris	0.980	0.9	-8.0%
Letter	0.905	0.788	-11.7%
Satimage	0.873	0.781	-9.2%
Shuttle	0.997	0.842	-15.5%
Vehicle	0.752	0.745	-0.7%
Vowel	0.918	0.888	-3.0%
Wine	0.966	0.966	0%

Figure 5: Most Accurate Replicated Regular model versus Most Accurate Replicated model without Gaussian Encoding or Bleaching.

in accuracy for any datasets over regular replication.

The results of the software extension to do a hyperparameter sweep on MNIST with a no-Bleaching model are shown in Figure 7. Our main point of reference was MNIST-Small, but the hyperparameters for MNIST-Medium and MNIST-Large are also shown for convenient comparison.

7 Reflections on Pedagogy and Education

This work informs our understanding of and computer science pedagogy as well as the field of weightless neural networks. Below, we offer reflections about the educational value of this work, providing thoughts of both the primary author, who was an undergraduate when the work was done, and the faculty advisor.

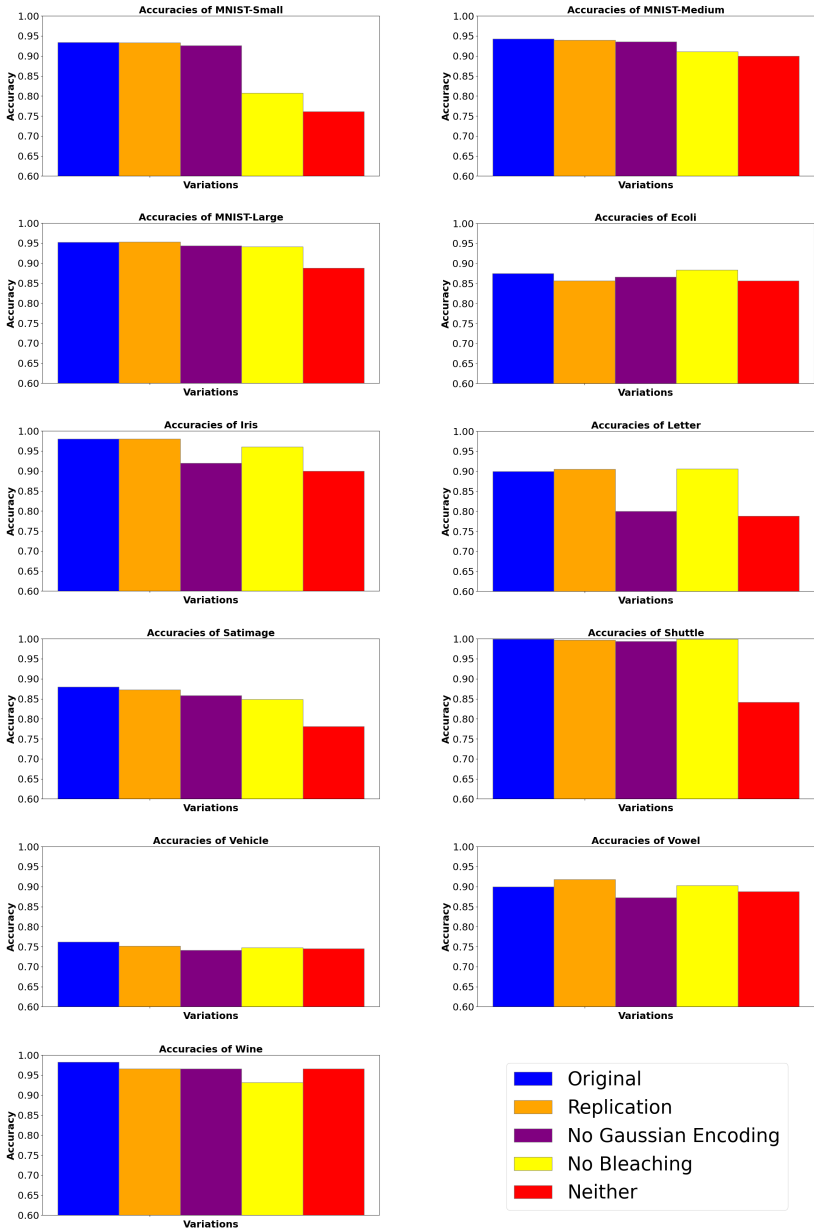


Figure 6: Software accuracy for all datasets, with the published results, replicated regular results, and results with the stripped-down models.

	“Small” Hyperparams	New Hyperparams	Change	(“Medium” HPs)	(“Large” HPs)
Bits per Input	2	5	3	(3)	(6)
Bits per Filter	28	28	0	(28)	(49)
Entries per Filter	1024	8192	7168	(2048)	(8192)
Hashes per Filter	2	1	-1	(2)	(4)
Accuracy	93.4%	93.9%	0.5%	(94.3%)	(95.2%)
Size	70.0 kB	936.7 kB	866.7 kB	(210 kB)	(960 kB)

Figure 7: Results of MNIST hyperparameter sweep.

7.1 Student Reflections

From my perspective, the most pedagogically interesting aspect of this work is that the faculty advisor did not have advanced knowledge of either of the fields this work is generally concerned with on a technical level, which are neural networks and FPGAs. This limited the amount of help available to me while performing the experiments — this is not to say there was none, however, for instance the idea to fix the RNG seeds to reduce run-to-run variation came from my advisor.

Said limited support probably made the experiments and the subsequent results less robust than they could have been. However, it may have made completing this work a more valuable learning experience for me. The large degree of independence meant that I had to come up with the design of the experiments myself, as well as gain the requisite depth of understanding of the subject material necessary to design meaningful experiments, which might have not been the case with an advisor who was involved with the work at a greater level.

The help the advisor gave in regards to the writing of the paper, the preliminary research done for it, getting it published, and wrangling the LaTeX for it were absolutely invaluable, however. Given the limited availability of research opportunities during a usual computer science undergraduate experience [7], research-focused skills such as these are not always developed during such an experience. I believe I have had greater opportunity than many to develop some of these skills, but I highly doubt this paper would be publishable without my advisor’s assistance.

7.2 Faculty Reflections on Teaching Experimental CS

This project arose from a combination of enthusiasm from a talented student and absence of faculty expertise. In retrospect, its structure could potentially address a number of issues in the undergraduate curriculum’s coverage of experimental techniques for computer science.

While undergraduate CS work often centers programming and some amount

of mathematical deduction (e.g., complexity proofs), simple experiments are within reach for even a first-year class. For example, students can time the execution of codes for algorithms of complexity $O(n^2)$, $O(n^3)$, and $O(2^n)$ and measure the degree to which the classic complexity analysis actually describes execution time. When small compute-bound codes are run on an idle machine, we’ve found the results to be quite accurate.

However, simple experiments can provide a false sense that experimental work is straightforward and works as expected. This is far from the truth, at the research level, where numerous challenges beset the would-be creator of a reproducible experiment on a cutting-edge system. Resource-intensive computations can expose performance dependencies on computer architecture, programming-language infrastructure, operating-system implementation, etc., especially for multi-core codes.

Complex codes that combine multiple libraries can also lead to challenges of reproducibility, e.g., due to library implementations that vary from machine to machine. Large code bases can even make it hard to be sure one has achieved a conceptually-simple goal like “make sure all random sequence generators can be started from standard seeds, to provide repeatable runs”.

Thus, when a result cannot be replicated for identical code and input, one is left to wonder which of many factors may be at play — an experience that’s quite distinct from an experiment on a small-scale system.

Teachers who focus on a deep introduction to experimental work could do so in a full-semester course that leads students through all phases of experimental work. But, if one wishes to shed light on this interesting topic with a smaller commitment of time and energy, we suggest replication of a result from the research literature. Even a paper bearing the “artifacts available” seal of approval can provide significant challenges. Design can be engaged on a smaller scope via the design of an extension, such as that discussed in our Section 5. If the lab comes pre-configured with all relevant tools, and students need only download the “artifact” and build and run it, interesting challenges can be explored within weeks rather than semesters.

8 Conclusion

Generally, replication for the selected models and for the training of new models were successful. In some cases, we did notice a few discrepancies that we are unable to explain, leading us to wonder about mistakes in the original paper. Specifically, the failure to replicate the accuracy result for the pre-selected MNIST-Medium was quite surprising. Additionally, it was surprising to see the variance between the original paper’s results and ours when running unaltered software models — getting within a percentage of the originally stated accura-

cies could not be achieved for some datasets, even with many training runs; for a couple of others, the stated accuracies were easily surpassed by wide margins. However, none of the possible mistakes we found undercut the core conclusions of the work to any significant degree.

The extension of the software experiments to selectively remove some of the innovations has revealed that both Gaussian encoding and Bleaching generally result in improvements, sometimes steep improvements, to WNN models. However, for some datasets, their improvements can be only mild, or they can even hurt accuracy to a degree that can be significant. The fact that the Ecoli model improved by close to 3% without bleaching from our replication of the original model, or 1% from the authors' original stated accuracy, raised concerns about the binary search algorithm to find the ideal bleaching threshold, as since removing bleaching amounts to forcing the bleaching threshold to 1, the algorithm should have chosen bleaching 1 even when bleaching was on. In the instances where removing bleaching appears to not significantly impact the accuracy of a model, this indicates that the binary search selected bleaching 1 as the ideal threshold already.

Our experiment with a hyperparameter sweep with a stripped-down architecture on MNIST shows that it is, in fact, possible to meet the accuracy of the original MNIST-Small model from Susskind et al. [9] even without one of their improvements. However, this comes at the cost of a much larger model size.

Overall, we feel our attempt at replication supports the core conclusions of Susskind et al.. BTHOWeN provides valuable innovations in the field of Weightless Neural Networks, making strides towards their practical implementation on FPGA. Our replication of their software implementation results confirms the overall accuracy of the design of BTHOWeN. Our experiments in isolation of individual improvements show the importance of Gaussian encoding (e.g., for Letter) and bleaching (e.g., for MNIST-Small and MNIST-Medium), and our attempt to increase model accuracy without bleaching shows its importance in creating accurate and practically-sized models.

References

- [1] Danilo S Carvalho et al. "B-bleaching: Agile overtraining avoidance in the wisard weightless neural classifier." In: *ESANN*. 2013.
- [2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.

- [3] Haochen Hua et al. “Edge computing with artificial intelligence: A machine learning perspective”. In: *ACM Computing Surveys* 55.9 (2023), pp. 1–35.
- [4] MG Sarwar Murshed et al. “Machine learning at the network edge: A survey”. In: *ACM Computing Surveys (CSUR)* 54.8 (2021), pp. 1–37.
- [5] Victor Nault. “Co-Design for Edge Intelligence: Binary Neural Networks”. Bachelor’s Thesis. Haverford College, 2024.
- [6] Anouar Nechi et al. “Fpga-based deep learning inference accelerators: Where are we standing?” In: *ACM Transactions on Reconfigurable Technology and Systems* 16.4 (2023), pp. 1–32.
- [7] Rhea Sharma et al. ““It’s usually not worth the effort unless you get really lucky”: Barriers to Undergraduate Research Experiences from the Perspective of Computing Faculty”. In: *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 1*. 2022, pp. 149–163.
- [8] Zachary Susskind et al. *BTHOWEN: Code to accompany "Weightless Neural Networks for Efficient Edge Inference"*. 2022. URL: <https://github.com/ZSusskind/BTHOWeN> (visited on 01/29/2024).
- [9] Zachary Susskind et al. “Weightless neural networks for efficient edge inference”. In: *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*. Association for Computing Machinery, 2022, pp. 279–290.
- [10] Kizheppatt Vipin and Suhaib A Fahmy. “FPGA dynamic and partial reconfiguration: A survey of architectures, methods, and applications”. In: *ACM Computing Surveys (CSUR)* 51.4 (2018), pp. 1–39.
- [11] Xiaofei Wang et al. “Convergence of edge computing and deep learning: A comprehensive survey”. In: *IEEE communications surveys & tutorials* 22.2 (2020), pp. 869–904.

Esquemático: A Functional Programming Language for Native Spanish Speakers*

Kelly Farran and Sean McCulloch

Department of Mathematics and Computer Science

Ohio Wesleyan University

Delaware, OH 43015

{klfarran,stmccull}@owu.edu

Abstract

The dominance of English-based syntax and documentation of programming languages presents an additional challenge for non-native English speakers learning programming. Due to the lack of multilingual resources available, non-native English speakers must often learn English in tandem with the programming language to be able to effectively utilize instructional materials, documentation, and support forums [5]. To address this issue, we have created *Esquemático*, a collection in the functional programming language Racket, a dialect of Scheme, which enables the use of Spanish keywords and syntax within the DrRacket IDE. The collection utilizes Racket's advanced macro system to modify its underlying syntactic structure, directly enabling functional programming with Racket in Spanish. Through the use of this macro system, our collection generates detailed error messages in Spanish which directly point to errors in syntax and logic. To aid students who are interested in learning Scheme through our collection, we have deployed a documentation site which provides a tutorial to begin your first program as well as detailed examples, definitions, and syntax patterns of all Spanish procedures provided in the collection.

*Copyright ©2025 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

1 Introduction

Computer programming is a skill in high demand across nearly every industry around the world. Despite this, the overwhelming majority of documentation, programming language syntax, instructional materials, and Q&A support forums are in English [17]. It has become commonly accepted that proficiency in English is necessary in order to learn computer programming; however, according to [5], nearly 95% of the world does not natively speak English. Thus, an unlevel playing field clearly exists in computer science education where the majority of students looking to learn programming face obstacles that can hinder learning and success — challenges not experienced by native English speakers. Recognizing these struggles is often difficult for native English-speaking programmers as it is uncommon for native English-speaking programmers to be put in a situation where they have to interpret code with non-English keywords and comments.

To understand the perspective of a non-native English speaker learning programming, consider the standard Hello World program in Java as an English speaker:

```
public class HelloWorld {  
    public static void main (String[] args) {  
        // Print Hello, World! to the terminal window  
        System.out.println("Hello, World!");  
    }  
}
```

Figure 1: The Hello World Program in English

This is the most basic introductory program used in almost every introduction to programming. Even without understanding the technicalities of programming, keywords like “class”, “main”, and “print” give insight and meaning into what the programming is doing: printing “Hello, World!” to the console window. Using an idea from Becker [1] who did a Hello World program in Irish (Figure 2), Figure 3 shows how that same program would look if all keywords were in Spanish. To a non-native Spanish speaker, it is much harder to understand the meaning of the program.

This is the core advantage English speakers have: the familiarity of the keywords (since they are English words) makes learning programming easier and more intuitive [10]. A non-Spanish speaker has to spend more time interpreting the keywords of the program in Figure 2 before they can begin to understand the functionality of the program. This time would be drastically increased in order to learn an entire framework of a programming language like

Java in Spanish. It is clear that learners who do not have an understanding of the natural language behind the keywords in a programming language are at a disadvantage when it comes to learning programming.

```
poiblí aicme DiaDuitDomhan {  
    poiblí statach folús príomh(Sreang[] args) {  
        // Priontáil "Dia duit, Domhan" go dtí an fuinneog teirminéal  
        Córas.amach.priontáilln("Dia duit, Domhan");  
    }  
}
```

Figure 2: The Hello World Program in Irish

```
clase pública HolaMundo {  
    pública estática vacío principal(Cadena[] args) {  
        // Escribe "Hola, Mundo" en la ventana del terminal  
        Sistema.fuera.imprimirln("Hola, Mundo");  
    }  
}
```

Figure 3: The Hello World Program in Spanish

This need to be conversant in a non-native language is also seen in the lack of comprehensive documentation and relevant forum discussions in languages other than English. For example, the community-driven Q&A forum Stack Overflow is one of the largest and most reputable platforms for troubleshooting and resolving challenges encountered when programming [12]. Stack Overflow's official policy on non-English questions instructs users not to submit non-English questions and directs non-English users to refer instead to alternative resources in their native language [15]. However, Stack Overflow in English has over 24 million questions answered, whereas Stack Overflow en Español has answered less than 1% of that amount [13, 14]. Similarly, textbooks and instructional materials have limited availability in languages other than English [5]. When instructional materials are available in another natural language, they often still include the use of English comments, identifiers, and function names. For example, the chapter "Bugs Y Errores" in the Eloquent JavaScript Español Textbook contains examples like the one in Figure 4 where despite the instructional descriptions being in Spanish, meaningful variable names and keywords are all written in English [6].

MODO ESTRICTO

JavaScript puede ser un *poco* más estricto al habilitar el *modo estricto*. Esto se hace colocando la cadena "use strict" en la parte superior de un archivo o en el cuerpo de una función. Aquí tienes un ejemplo:

```
function canYouSpotTheProblem() {  
  "use strict";  
  for (counter = 0; counter < 10; counter++) {  
    console.log("Happy happy");  
  }  
}  
  
canYouSpotTheProblem();  
// → ReferenceError: counter is not defined
```

Figure 4: A function with an error in Eloquent Java Español

In this example, the name of the function is “canYouSpotTheProblem”, and the name of the counter variable is “counter”. The purpose of the function is to create an intentional error in the code (using an identifier “counter” without first defining the identifier), and the name of the function reflects that purpose; however, without a knowledge of the meanings of the English words in the names of the identifiers, the purpose of the example becomes more difficult to deduce. Thus, we believe that non-native English speakers would benefit from a programming environment where the keywords, documentation, and error messages all were in the native language of the speaker.

2 Related Work

There are not many existing projects in the area of translating programming languages into a natural language other than English. *Latino* is a programming language influenced by Python and Lua which includes only Spanish keywords and built-in functions [16]. While there is some Spanish documentation for *Latino*, and while some error messages are in Spanish, other errors are reported in English. This may be a result of the difficulty of translating the entirety of a complex programming language like Python. *Tango* is a prototype language with syntactic constraints similar to Java which includes its own compiler to interpret Spanish keywords, however *Tango* is not a “fully functional and bug-free programming language” [18]. EsJS is a language closely modeled after JavaScript which provides an interface for JavaScript programming using Spanish keywords and syntax, yet error messages are still prompted to the user in English [9].

There is not a Racket collection or language which currently exists besides Esquemático which provides support for Racket programming in Spanish, however there is a project which provides support for Racket programming using Chinese characters called *ming-lang* [11]. *Ming-lang* is similar to Esquemático as its purpose is to provide an interface which allows users to program in Racket through procedures and keywords not in English. *ming-lang* provides extensive documentation in Chinese to aid users in learning how to use *ming-lang* and how to program using functional programming concepts, however, several examples in the *ming-lang* documentation show instances in which *ming-lang* procedures still display error messages in English. Similarly, *Deinprogramm* is a teaching language of Racket whose goal is to educate users on how to use Racket in German [2]. *Deinprogramm* provides detailed documentation in German on functional programming and using Racket, however it does not appear to provide support for Racket programming in German; its main focus is on documentation and education in using the built-in English procedures and keywords. *International Scheme* is a project which seeks to allow the translation of general Scheme into any natural language by providing macros like *translate-procedure-name* and *translate-syntax-name*, which users can use to define translation bindings for built-in Scheme procedures and keywords, however there does not currently exist an implementation of *International Scheme* which provides support for Spanish [7].

3 Design of Esquemático

We have created Esquemático to address both the disadvantage of programming keywords, identifiers, and errors being in English as well as the issue of the lack of available documentation. Esquemático not only provides support for Scheme syntax and error handling in Spanish, but comprehensive documentation and instructional materials in Spanish including a tutorial to starting your first program. We hope that after learning functional programming concepts using Esquemático, native Spanish speaking programmers may go on to apply their knowledge to more intuitively learn English-based programming languages.

3.1 Choice of Language

Racket is an ideal language for the implementation of Esquemático for two main reasons. Firstly, Racket is a functional programming language, which enables a modular approach to programming by focusing on functions and their composition. This was particularly appealing to us during the planning of Esquemático as we wanted our project to directly call the underlying English procedures rather than relying on simple text replacement. As programming

languages rely on strict syntax, an imprecise approach like text-replacement would likely lead to inconsistencies and ungrammatical phrases in the programming language which would lead to unpredictable errors at compile time. Additionally, text-replacement does not effectively address the issue of English error messages, as it does not interpret the given parameters or syntax. Moreover, functional programming is equipped with less complicated syntax, making it ideal for learning programming concepts without having to understand a more complicated syntax.

The second reason pertains to Racket's advanced macro system. Macros allow the syntax of a programming language to be altered by adding custom rules to the underlying implementation of the language. Racket's macros are more powerful than macros in other programming languages like C++ as Racket's macros can perform pattern matching, where syntactic expressions can be expanded uniquely based on different conditions [8]. Because of its macro capabilities, Racket is sometimes referred to as a 'language-oriented' programming language, as it is an ideal language for creating new programming languages [4]. For this reason, Racket is an ideal language for implementing a translation-focused project like Esquemático.

3.2 Interpreting a Spanish function call

Esquemático is a collection of Racket macros which collaborate in order to provide the overall functionality of a Spanish-language version of Scheme. Figures 5 and 6 give an example of this functionality for the procedure *longitud*, which is the *length* procedure for lists in standard English Racket. For each English Racket procedure for which an equivalent Spanish procedure is provided in Esquemático, there exists a macro wrapper (Figure 5) and an inner procedure (Figure 6) which work together to call the underlying English Racket procedure as well as catch any possible errors in the user's input.

When a user invokes a Spanish procedure provided in Esquemático such as *longitud*, it is passed to the macro wrapper, which performs a few preliminary checks. Firstly, the wrapper asks if the user passed a keyword followed by a series of arguments, in which case that procedure *appears* to be a legal call, and so it is passed to the inner procedure. If the user just passed a keyword with no arguments and no parenthesis, the macro affirms to the user that keyword is indeed bound to a valid procedure in Esquemático by printing out `#<procedimiento:procedure-name>`- the equivalent of English Racket's `#<procedure:procedure-name>`. Finally, if the user passed the keyword with some unrecognized syntax pattern, the user is informed in Spanish that they have made a syntax error.


```

(define-syntax longitud
  (lambda (stx)
    (syntax-case stx ()
      [(longitud . args)
       #'(inner-longitud . args)]
      [longitud
       #'(displayln "#<procedimiento:longitud>")]
      [else
       (error "\nlongitud: sintaxis no válida")]))

```

Figure 5: Macro wrapper for *longitud* procedure

If the keyword was passed with a series of arguments, the macro passes those arguments along to the inner procedure (called *inner-longitud* in Figure 6). The inner procedure then checks those arguments for errors in type and quantity according to the legal contract for that procedure. To facilitate this error-checking, we have defined a set of error-checking functions which is accompanied by a series of helper functions. One of these error-checking functions called in almost every procedure in Esquemático is *arity-check*, which takes a number of given arguments and a number of desired arguments (or a range for the number of desired arguments) for a procedure and returns true if the given arguments satisfy the conditions of the desired arguments. If the given arguments do not satisfy the conditions of the desired arguments, an error is generated which describes the location and cause of the error in Spanish and stops the program. Similarly, the procedure *contract-viol-check* calls upon helper functions *match-contract* and *get-contract* in order to determine whether or not the given arguments for a procedure are of the correct type or not. *Get-contract* maps every procedure in Esquemático to a legal contract (for example, “string, [integer]”) while *match-contract* determines whether a set of given arguments satisfies all of the requirements of the legal contract for the intended procedure. Depending on the procedure, the arguments will be assessed to look for relevant types of error in input (for example, the procedure for division *div* has a *divide-by-zero-check* which other procedures do not). In the case of the *longitud* procedure, the arguments are being checked for arity (calling the *arity-check* procedure and telling it that *longitud* can only accept one argument) as well as for contract violation (calling *contract-viol-check* and telling it that *longitud* must accept a list with any contents inside). If the arguments passed by the user pass all error checks, the underlying English *length* procedure is called with the arguments passed by the user to *longitud*.

```
(define inner-longitud
  (lambda args
    (if (and (arity-check (count-args args) 1 "longitud")
             (contract-viol-check args "(cualquiera)" "longitud"))
        (length (car args))
        (error "error al comprobar si hay errores en la entrada
                \npor favor consulte la documentación" ))))
```

Figure 6: Inner procedure for the *longitud* procedure

Because the inner procedures directly call upon the underlying English Racket procedures, if Esquemático didn't check for errors in user code, errors would be raised in English. Because Esquemático contains its own system for checking errors in user input, not only can errors be raised in Spanish, but the specific location of the error can be displayed to the user, aiding in usability and making Esquemático learner-friendly. For example, Figure 7 shows an error which has been thrown because the argument passed to *longitud* violated the legal contract for *longitud*; i.e. a non-list was passed to the procedure. The error message tells the user that the legal contract has been violated, that a list was expected but instead 7 was given, and that the error happened at 7, where a list was expected to be rather than an integer.


```
Welcome to DrRacket, version 8.7 [cs].
Language: scheme, with debugging [custom]; memory limit: 128 MB.
> (longitud 7)

  ..\..\Program Files\Racket\collects\esquematico\err-chks.rkt:45:10:
    longitud: violación de contrato
    esperado: (cualquiera)
    dado: 7
    error en: 7, se espera una lista
>
```

Figure 7: An error generated by Esquemático

3.3 Non-function keywords

Esquemático also provides functionality for keywords provided by Racket which are not procedures: for example, the keyword *define*, which is the keyword used to bind values to identifiers, or the keyword *cond* which signals to the compiler that a series of conditions are to be interpreted. For these keywords, Esquemático provides functionality through macros, but not through the inner-outer method described above. Instead, the calls to the underlying Racket keywords (typically done by the inner procedure) are just made inside of the

macro itself. This is because there are multiple distinct patterns for which legal calls to these types of keywords can be made. For example, the keyword *define* can be invoked legally in several different forms. One form is to use *define* to bind a single value to an identifier, for example:

```
(define counter 1)
```

This line binds the value of 1 to the identifier “counter”. Additionally, *define* can be used to define a procedure, in which an expression is bound to an identifier, where the value of that identifier is the value determined by evaluating the bound expression given some parameter values. For this operation, there are several different legal calls to *define*, for example using the *lambda* keyword:

```
(define contains-5?  
  (lambda (list)  
    (cond  
      ((null? list) #f)  
      ((eq? (car list) 5) #t)  
      (else (contains-5? (cdr list)))) ))
```

Figure 8: An example of a Scheme procedure bound using *define*

The *define* keyword can also be used to bind identifiers to expressions without the *lambda* expression. Because of all of these different legal forms, the macro for *define* in Esquemático, “*definir*”, must first decipher which form the user is trying to use, and then determine if the input provided by the user (syntax, parameter quantity, parameter type, ...) constitutes a legal call to *define* or not. For each different legal pattern for *definir*, there are different ways to raise errors because of incorrect input, and so the macro must check specifically for these errors and raise them appropriately in Spanish. If all checks for incorrect input are passed, a call to *define* is made, and the appropriate values are bound to the requested identifiers using Racket’s value binding system induced by *define*. In order to use our Spanish-language error checking procedures, *definir* generates a modified expression that wraps an arity-check around the user-provided expression. This ensures that our Spanish-language errors are also given on user-defined functions.

4 Results

This project has produced a fully functional interface for functional programming with Scheme in Spanish. To demonstrate the functionality of this interface, we provide Figure 9 and Figure 10. Figure 9 is an example of a function in standard English Scheme and Figure 10 is the Spanish equivalent of the procedure produced using Esquemático:

```
(define eq-list? ; returns true if two lists, lisa and lisb, are equal
  (lambda (lisa lisb)
    (cond
      ((and (null? lisa) (null? lisb)) #t)
      ((or (null? lisa) (null? lisb)) #f)
      ((and (and (not(list? (car lisa))) (not(list? (car lisb))))
            (and (eq? (car lisa) (car lisb))))
       (eq-list? (cdr lisa) (cdr lisb)))
      ((or (not(list? (car lisa))) (not(list? (car lisb)))) #f)
      (else (and (eq-list? (car lisa) (car lisb))
                  (eq-list? (cdr lisa) (cdr lisb)))))))
```


Figure 9: A procedure in standard Scheme which compares two lists for equality

```
(definir ¿lista-igual? ; verifica la igualdad entre dos listas: 'lisa', 'lisb'
  (lambda (lisa lisb)
    (condición
      ((y (¿nulo? lisa) (¿nulo? lisb)) 'verdadero)
      ((o (¿nulo? lisa) (¿nulo? lisb)) 'falso)
      ((y (y (no (¿lista? (pri lisa))) (no (¿lista? (pri lisb))))
         (y (¿ig? (pri lisa) (pri lisb))))
       (¿lista-igual? (res lisa) (res lisb)))
      ((y (no (¿lista? (pri lisa))) (no (¿lista? (pri lisb)))) 'falso)
      (sino (y (¿lista-igual? (pri lisa) (pri lisb))
                (¿lista-igual? (res lisa) (res lisb)))))))
```

Figure 10: The Spanish equivalent of the procedure in Figure 9 using Esquemático

In Figure 11, we see a few calls to the Spanish procedure *¿lista-igual?* which demonstrates its capabilities in both functionality as well as error handling ('verdadero' in Spanish meaning *true* and 'falso' in Spanish meaning *false*). When passed a single argument instead of two arguments, we get an arity mismatch, where Esquemático tells the user that two arguments were expected, but only one was provided.

```

> (¿lista-igual? '(a b c) '(d e f))
falso
> (¿lista-igual? '(a b c) '(a b c))
verdadero
> (¿lista-igual? '(1 2))
 ..\..\..\Program Files\Racket\collects\esquemático\err-chks.rkt:28:8:
¿lista-igual?: desajuste de aridad;
el número esperado de argumentos no coincide con el número dado
esperado: 2
dado: 1

```

Figure 11: A series of calls to *¿lista-igual?*

As the purpose of Esquemático is to teach and to make learning about functional programming more widely accessible, Esquemático is accompanied by an extensive documentation site which includes descriptions, examples, and all legal syntax patterns of all of the Spanish procedures and keywords included in Esquemático [3]. Also included in the documentation is a tutorial which serves to instruct first-time users on how to begin their first program using Esquemático as well as instructions for installation and configuration of DrRacket to utilize Esquemático. Combined with the functionality of the interface provided by Esquemático for DrRacket, we have created a learning environment which allows Spanish-speaking programming students to learn how to program in Racket/Scheme as well as continue on in their work and develop comprehensive projects of their own.

5 Future Work

An important direction for future work involves distributing Esquemático to Spanish-speaking communities to evaluate its effectiveness in enhancing programming education. This would allow us to better understand the impacts of engaging with programming concepts in one's native language as well as potentially identify areas for improvement with Esquemático. Additionally, while this paper focuses on providing support for Racket/Scheme programming in Spanish, future work in the area of programming language translation could include the modifying of Esquemático's macro model to provide support for Scheme programming in other natural languages. Esquemático's underlying structure is modular, making it easily ported into other natural languages only with the translation of the keywords and a handful of error messages. As each keyword in Esquemático is provided through an inner procedure and a macro wrapper, a search and replacement of a Spanish keyword with a translation of that keyword into another natural language would produce a callable procedure using that new keyword. Upon translating the error messages and

replacing all of the Spanish procedures with translated keywords, Esquemático would provide full support for that other language of translation. Included on the documentation site is a guide which provides instructions on how to port Esquemático to another natural language in this manner [3].

6 Conclusion

We have presented Esquemático: a Spanish-language programming environment designed to make Scheme programming more accessible to Spanish-speaking users. Our design includes a comprehensive implementation of 71 standard Scheme functions, accessible entirely in Spanish. To provide this functionality, user code is interpreted by our wrappers and then passed on to the native English Racket functionality. The macros we have written to interpret the Spanish keywords and procedures perform error checking, giving context-relevant error messages in Spanish to the user for both native Racket procedures as well as user-defined procedures. As such, the programming environment can be entirely conducted in Spanish, without the need to understand English keywords or identifiers.

Additionally, we have created a documentation website to support Spanish speakers in learning to use Esquemático and write programs using it. We hope that these contributions will reduce the barriers to learning programming that they currently face, making computer science education more inclusive and accessible.

References

- [1] Brett A. Becker. “Parlez-vous Java? Bonjour La Monde != Hello World: Barriers to Programming Language Acquisition for Non-Native English Speakers”. In: *Annual Workshop of the Psychology of Programming Interest Group*. 2019. URL: <https://api.semanticscholar.org/CorpusID:232221466>.
- [2] *Deinprogramm*. <https://pkgs.racket-lang.org/package/deinprogramm>.
- [3] *Equemático!* <https://esquematico.azurewebsites.net/Inicio>.
- [4] Matthias Felleisen et al. *The Racket Manifesto*. <https://felleisen.org/matthias/manifesto/index.html>.

- [5] Philip J. Guo. “Non-Native English Speakers Learning Computer Programming: Barriers, Desires, and Design Opportunities”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: Association for Computing Machinery, 2018, pp. 1–14. ISBN: 9781450356206. DOI: 10.1145/3173574.3173970. URL: <https://doi.org/10.1145/3173574.3173970>.
- [6] Marijin Haverbeke. *Javascript Eloquente (Spanish Translation of Eloquent Javascript, 4e)*. No Starch Press, 2024.
- [7] *International Scheme*. <https://github.com/metaphorm/international-scheme>.
- [8] *Introduction to Macros in Racket*. https://www2.cs.sfu.ca/CourseCentral/383/tjd/racket_macros.html.
- [9] JS Foundation. *JavaScript Español*. <https://es.js.org/>.
- [10] Caitlin Kelleher and Randy Pausch. “Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers”. In: *ACM Comput. Surv.* 37.2 (June 2005), pp. 83–137. ISSN: 0360-0300. DOI: 10.1145/1089733.1089734. URL: <https://doi.org/10.1145/1089733.1089734>.
- [11] *Ming Language*. <https://www.yanying.wang/ming/index.html>.
- [12] Iraklis Moutidis and Hywel T. P. Williams. “Community evolution on Stack Overflow”. In: *PLOS ONE* 16.6 (2021). <https://journals.plos.org/plosone/e0253010>. DOI: 10.1371/journal.pone.0253010. URL: <https://app.dimensions.ai/details/publication/pub.1138958611>.
- [13] *Preguntas más nuevas*. [urlhttps://es.stackoverflow.com/questions](https://es.stackoverflow.com/questions).
- [14] *Stack Overflow Newest Questions*. <https://stackoverflow.com/questions>.
- [15] *StackOverflow Non-English Question Policy*. <https://stackoverflow.blog/2009/07/23/non-english-question-policy/>. 2009.
- [16] Latino Language Team. *About Latino*. <https://manual.lenguajelatino.org/en/latest/About-Latino.htm>.
- [17] Ashok Kumar Veerasamy. “Teaching English based programming courses to English learners/non-native speakers of English”. In: *International Conference on Education and Management Innovation*. Feb. 2014.
- [18] Ashley M. Zegiestowsky. “Tango: A Spanish-Based Programming Language”. In: *Butler Journal of Undergraduate Research* 3.11 (2017).

Assessing Impact of Role-Playing on Introductory Programming Education: A Comparative Study*

Kriti Chauhan and Leong Lee

Department of Computer Science and Information Technology
Austin Peay State University
Clarksville, TN 37044

chauhank@apsu.edu, leel@apsu.edu

Abstract

Role-playing has been utilized as a teaching method for introductory object-oriented programming for decades. In one instance, an instructor removed role-playing activities from an in-person second-level undergraduate introductory Java programming course due to a shortened lecture schedule caused by winter weather. Following a decline in average midterm performance, the instructor reinstated the activities. A comparison of midterm and final exam results with those from the previous year's course, taught by the same instructor, revealed that role-playing activities positively impacted student learning, particularly among low-performing students, while having a minimal effect on high-performing students.

1 Introduction

Computer programming pedagogy employs a wide array of techniques to facilitate student learning. These methods range from traditional approaches, such as chalkboard-based lectures and PowerPoint presentations, to more active learning techniques like live coding (where instructors write and explain

*Copyright ©2025 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

code in real-time), in-class coding activities (encouraging students to code during lessons), gamification, pair programming, and code tracing, among others [1, 2, 3].

One commonly used active learning technique specific to object-oriented programming is role-playing, where students enact the roles of objects in a program [4]. This approach helps students conceptualize the idea that objects store data and are responsible for behavior. It provides a better understanding of classes and objects while increasing student interaction and engagement, leading to improved learning outcomes.

However, there is scant empirical evidence of the effectiveness of role-play techniques in teaching computer programming. Does incorporating role play in in-person lectures actually improve student learning? Under what conditions is it helpful, and when might it fall short? This paper aims to answer these questions through a case study involving two batches of students. For the first batch, role-playing activities were integrated throughout the semester, whereas for the second batch, these activities were introduced only after the midterm.

For an in-person undergraduate class titled Introduction to Programming II, which is the second part of a two-course sequence to introduce object-oriented programming in Java at a mid-south university in the USA, winter weather closure and absenteeism at the start of the semester required covering the same topics with less time. The instructor chose not to include role-playing activities that were conducted when they taught this class for the first time at this university, in the previous year.

While general feedback and a decade of prior teaching experience suggest that role-play activities are beneficial to students, anonymous end-of-term feedback from a separate programming class at this university indicated that students here might perceive these activities as overly simplistic or childish. In light of this feedback, the instructor assumed that students at this university might not benefit from the activities and chose to proceed with the course using all other instructional techniques unchanged. These techniques included live coding, walkthroughs of code, and drawing memory diagrams on the whiteboard [5, 6].

Additionally, the instructor recorded all in-person lectures and posted them on the online Learning Management System (LMS) for students to refer to after class. Recordings were made by starting a Zoom session during each class, sharing the instructor's screen, and recording the session. Although the recordings could not capture what was written or drawn on the physical whiteboard, they did include the code written on the instructor's computer. The textbook, the code written and explained, and the slides used were consistent across both batches of students.

The second cohort midterm exam scores were noticeably lower than the

first cohort. This could have been due to various factors, such as missed lecture time, removal of role-playing activities, unintentional differences in instruction, individual differences among the cohorts, external factors such as events on campus, tutor availability or other year-specific conditions. The low midterm scores prompted the instructor to incorporate numerous role-playing exercises in the second half of the semester, aiming to help students grasp concepts more effectively.

This study conducts a post-hoc analysis of the improvement in final exam scores relative to midterm scores for the second cohort, comparing it to the improvement observed in the first cohort. For the first cohort, role-playing activities were incorporated throughout the semester, while the second cohort had them only after the midterm. The analysis finds considerably more improvement in the second cohort's final exam over midterm, compared to the first cohort, especially for low and moderate performing students, but not for the high performing students.

As such, instructors should carefully consider the trade-offs between incorporating more active learning exercises, providing additional coding examples, or covering more advanced concepts in their lesson plans. It is crucial to take into account the varying performance levels and capabilities of students when making these choices. It is hoped that this case study can guide instructors to make better-informed decisions when teaching introductory programming classes.

2 Background

2.1 Live Coding in Programming Pedagogy

Live coding, the practice of writing and explaining code in real-time during lectures, has become an increasingly popular pedagogical approach in programming education. By allowing students to observe the thought processes of an experienced programmer, live coding provides a dynamic and engaging way to teach coding concepts. Research has shown that live coding helps students develop problem-solving skills and encourages active participation in the learning process [2].

Effective live coding requires careful preparation and adaptability on the part of the instructor, to type and debug code in front of students. In introductory courses, some students may find the pace of live coding intimidating or overwhelming. To address this, live coding should be supplemented with additional instructional methods, to enhance student engagement and ensure comprehensive coverage of course material.

2.2 Memory Diagrams in Programming Pedagogy

Memory diagrams are a widely recognized pedagogical tool for teaching programming concepts, offering a visual representation of how memory is allocated and manipulated during code execution. By mapping variables, objects, references, and function calls to a memory model, these diagrams make abstract concepts tangible and comprehensible. Research has shown that memory diagrams effectively aid in teaching challenging topics such as object-oriented programming and recursion by enhancing mental models of programming [5, 6].

In addition to illustrating key concepts such as scope, object instantiation, and references in OOP, memory diagrams also help develop students' critical thinking and debugging skills. By bridging the gap between abstract programming concepts and their practical implications in memory, they serve as an essential tool in programming education. By fostering more accurate mental models, memory diagrams significantly enhance comprehension and retention in foundational programming courses.

2.3 Role-Playing in Object-Oriented Programming Pedagogy

Role-playing has long been recognized as a valuable pedagogical technique in computer science education, particularly in teaching Object-Oriented Programming (OOP). The shift to OOP languages in the late 1990s, coupled with the growing need for effective teaching strategies, brought role-playing into prominence as an innovative method.

Role-playing in programming education involves using human beings as metaphors for objects, allowing students to better understand concepts through physical enactment [4, 7]. It often includes activities such as "be the compiler," popularized by resources like *Head First Java* [8]. These activities encourage students to actively engage with programming concepts by stepping into the roles of components within a system.

Role-playing also connects with active learning principles by fostering participation, engagement, and motivation. However, it should not be conflated with learning styles theories, which have been critiqued as neuromyths [9]. Instead, role-playing is an inclusive and effective active learning approach that can enhance comprehension and retention for a wide range of students.

It is important to distinguish role-playing from other participatory teaching methods. Unlike methods where students take on roles such as designers, users, or administrators [10], role-playing in programming focuses more on embodying the behavior of computer objects or systems rather than person-based roles. Furthermore, it differs significantly from gamification, which uses game-like elements such as points, levels, badges, and rankings to motivate students.

While gamification introduces challenges and tasks for participants [11], role-playing emphasizes the simulation of programming concepts without relying on extrinsic rewards.

3 Integrating role play activities into programming instruction

Role-playing exercises, where students act as programming components, are an effective supplement for teaching introductory object-oriented programming. These activities are incorporated alongside live coding and memory diagram walkthroughs to enhance engagement and understanding. Two types of role-playing exercises exist: experiential, where actions are vaguely defined, and scripted, where actions are explicitly outlined [4]. Due to the complexity of OOP concepts, scripted role-playing is preferred, as it provides clear guidance, helping students distinguish between classes and objects and visualizing the behaviors and data unique to each object.

3.1 Conducting Role-Playing in Classroom

A program is written by the instructor that includes a class with instance variables and methods and creates multiple objects. During the walkthrough, students are assigned roles to act as the objects. When an object is instantiated, the instructor hands a notecard (representing memory allocation) to a student and explains that they are now responsible for the object's data and behavior. The notecard contains names of instance variables of the object and space for each value, along with names of all the methods of that object.

When a method is called on an object, the student representing the object receives a marker/pen, signifying control flow, and updates their variables or outputs as instructed. The marker is then returned to the instructor, who represents the `main()` method, to show return of control. Once the role-play concludes, the program is executed once again to verify behavior, since the program is usually edited as the roleplay progresses, using input from the students as to what values to assign to variables, what methods to call, etc.

The metaphor is explicitly stated, and memory diagrams are maintained throughout to visually illustrate variable changes and differences between objects for all students to observe.

3.2 Extending Role-Playing to Other Concepts

Role-playing can be adapted to teach various programming concepts, including branching, exception handling, and data passing. Below are examples with additional details on how each adaptation works.

3.2.1 Branching (If-Else Statements)

To demonstrate branching, two students represent the if and else blocks in a program. The instructor, acting as `main()`, evaluates a Boolean condition and decides which block executes. The student corresponding to the chosen block receives the marker, representing control flow. The activity demonstrates that only one block executes during a program's run. The class observes the flow, which emphasizes how conditional decisions are made and how control always returns to `main()` after execution.

3.2.2 Exception Handling

In this adaptation, one student acts as the `try` block, and another as the `catch` block. If no exception occurs, control flows directly from the `try` block back to the instructor acting as `main()`. If an exception is raised, the marker (control flow) moves from the `try` block to the `catch` block, responsible for error handling. This visual and physical representation helps students see how exception handling redirects control flow.

3.2.3 Primitive vs. Reference Data Passing

Students representing methods can illustrate how data is passed to functions. For primitive data types, the student playing the method receives a value verbally, writes it on their notecard, and updates it. Changes are local to the method and do not affect the original variable. For reference data types (objects), the name of the student playing an object is told to the student playing the method and receiving the argument. The student playing the method interacts with the student playing the object by directly modifying the data on the object's notecard. This demonstrates how passing an object allows changes to persist beyond the method's scope.

3.2.4 Arrays

To demonstrate arrays, three or more students stand in a line, each representing an element in the array. The position of each student corresponds to their index number, while their name or assigned value represents the array element. The students are handed notecards to signify memory allocation, if demonstrating array of objects. The instructor can "read" or "write" values or call object methods for the object at specific indices, interacting with the student at that position. This activity helps students visualize how arrays store data and the difference between indices and contents. It is recommended to conduct the exercise with an array of primitive data types or `String` first, and afterwards conduct the exercise demonstrating an array of objects.

Adaptation for two-dimensional arrays uses the seating arrangement in the classroom to represent a grid. Rows and columns are treated as indices, and students identify their positions in the grid, connecting array indices to stored data (their names).

3.2.5 Method Overloading

Each student represents a method with the same name, but different numbers or types of arguments. Different-colored notecards are used to represent different argument data types, and students hold notecards corresponding to their method's parameters. When a method is called, the argument configuration (colors and number of notecards) determines which student "executes." This clearly demonstrates the concept of method overloading and how the compiler differentiates methods based on signatures.

3.2.6 Recursion

To teach recursion, students play multiple calls to a recursive method. Each recursive call is represented by a new student receiving a notecard (memory allocation) and the marker (control flow). The process continues until the base case is reached. The student representing the base case returns a value to the previous student, and this continues back through the chain until control returns to `main()`. This physical representation demystifies recursion by showing how each method call has its own memory space and how the base case resolves the recursion.

3.2.7 Linked Lists

For linked lists, three to five students play node objects, while one student plays the linked list object, which includes a variable for the head node. The linked list student's notecard contains the name of the first node, and each subsequent node student stores the name of the next node. Linked list traversal can be demonstrated by interacting with each of the nodes sequentially. Adding or removing nodes from the linked list can also be demonstrated.

3.3 Best Practices and Considerations

Almost any program can be converted into a role-playing exercise. But not all of them need to be conducted in the classroom. Activities should be selected based on the students' level of understanding and interest. To maximize engagement in a large class, each student may participate in only one role-playing activity per semester, allowing more students the opportunity to take part.

Two key elements are essential for an effective classroom role-playing exercise. First, it is important to clarify the metaphor of what each student represents (e.g., a method, object, processor, block of code or data) before, during, and after the role play. Second, maintaining memory diagrams alongside role-playing is crucial for all students to be able to see the changes in state of variables and objects.

4 Data analysis

4.1 Overview and initial analysis

Data from two on-ground cohorts from different years, of a second-level introduction to object-oriented programming class using Java, at a mid-south university in USA, are compared. Year 1 cohort had role-playing exercises throughout the semester. Year 2 cohort had role-playing exercises incorporated after the midterm exam. The two cohorts were taught by the same instructor. Students who dropped the class are not included in the analysis, as their data is not available. Students who did not take the final exam, one in each of the cohorts, are also excluded from the analysis.

The arithmetic means, standard deviations, t-statistics, degrees of freedom and p-values for midterm exam scores, final exam scores, overall assignment scores, overall quiz scores and overall final grades are summarized in Table 1. All p-values for t-tests are greater than 0.05, which indicates that the two cohorts are not statistically significantly different on any of these parameters. This is not surprising, given that both cohorts are students at the same university, within the same program, taking the same exams, assignments, and quizzes, being taught by the same instructor, with only one instructional technique differing between the two cohorts.

Assignment scores varied considerably between the cohorts. Year 2 students performed much better on the early assignments compared to Year 1. While exams depend solely on the student, since they are taken in a proctored environment, with no notes, completion of assignments is not the same. In fact, students are encouraged to get tutoring help on assignments. It is also possible that the difference is due to the fact that different teaching assistants graded the assignments in both years. Another possibility is that since Year 2 students had an extended deadline for the initial assignments, due to winter weather closures, that could also have resulted in higher scores. A myriad of factors could be responsible for these differences. Hence this difference is not a focus of this study.

Quiz scores are similar for both cohorts. There is less than one point of difference between the average quiz scores of the two cohorts. The same quizzes were assigned to both cohorts, with the Year 2 cohort completing one additional

quiz compared to Year 1, which is not included in the analysis. All quizzes were open notes, allowing students to refer to their textbook or personal notes. Of the overall grade, midterm exam, final exam, and quizzes are weighed 20% each, while assignments are 40%.

The focus of this study is primarily on exam scores, as the same exams were administered to both cohorts and graded by the same instructor. The Year 2 average midterm score is 5.61 points lower than that of Year 1, translating to a letter grade of D, while the Year 1 midterm average corresponds to a low C. The Year 2 average final exam score is 3.53 points lower than that of Year 1, with both Year 1 and Year 2 final exam averages corresponding to a letter grade of C. The average final exam score for Year 1 students is 6.98 points higher than their average midterm score, while average final exam score for Year 2 students is 9.06 points higher than their average midterm score. Since role-playing activities were the only major instructional difference, introduced after the midterm for Year 2 students, this data suggests that these activities may have contributed to a greater increase in final exam scores for Year 2.

Category	Year 1 mean	Year 2 mean	Year 1 s.d.	Year 2 s.d.	p-value (two-tailed t-test)
Midterm exam	70.20	64.59	18.84	21.50	0.31
Final exam	77.18	73.65	20.43	19.23	0.51
Assignments	87.59	92.20	23.91	10.94	0.36
Quizzes*	88.70	89.51	8.78	6.71	0.70
Overall grade	82.31	82.45	14.61	10.61	0.97
Year 1 n = 28; Year 2 n = 27; Degrees of freedom, df = 53.					
*Year 2 had an extra quiz, not included in this analysis.					

Table 1: Arithmetic means, standard deviations, and p-values for midterm exam scores, final exam scores, overall assignment scores, overall quiz scores, and overall final grades for Year 1 and Year 2 cohorts. Students who dropped and students with no final exam are excluded from analysis.

4.2 Trend analysis

Figure 1 shows midterm exam scores and final exam scores for each of the 28 students in Year 1, on the left, and each of the 27 students in Year 2 on the right. A visual review of the scores shows that while a lot of students scored higher on the final than the midterm, more students in Year 1 did worse on the final, than in Year 2.

In general, students scored higher on the final exam than the midterm in

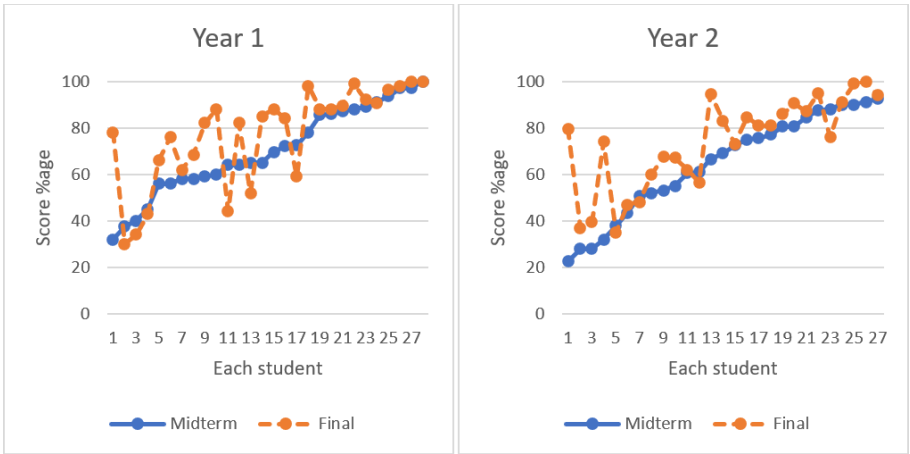


Figure 1: Midterm and Final exam scores for each student in Year 1, left, and for each student in Year 2, right.

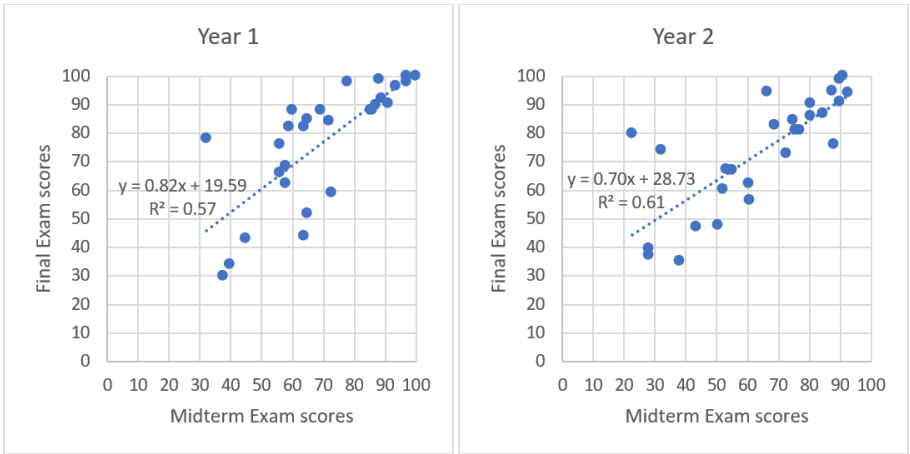


Figure 2: Plot of final exam scores over midterm exam scores of Year 1, left, and of Year 2, right, with trendline.

both cohorts. Comparing regression of final exam scores on midterm exam scores for Year 1 and Year 2 cohorts (Figure 2), while the average scores on both exams are lower for Year 2 students, the y-intercept for Year 2 is more than 9 points higher than that of Year 1. This indicates higher baseline performance

on the final exam, given the midterm grades.

Moreover, the rate of change of the regression line for Year 2 is lower, with a slope of 0.7 compared to slope of 0.82 for Year 1. This means that if two students have a difference of ten points on the midterm, they are likely to have a difference of 8.2 points on the final if they are in Year 1 cohort, and a difference of 7 points if they are Year 2 cohort. This suggests that students on the lower end of the regression line were able to close the gap with those on the higher end to a greater extent in Year 2 compared to Year 1.

4.3 Analysis of subgroups

The regression lines for Year 1 and Year 2 final exam scores over midterm scores intersect at (76.17, 82.05), with the first element being midterm score, and the second element being final exam score, in the ordered pair. Table 2 summarizes the average midterm and final exam scores for each subgroup, dividing the cohorts into three groups based on midterm scores: those below 60 (pass/fail threshold), those between 60 and 76, and those above 76.

In the low performance category, among students who scored less than 60 on the midterm exam, Year 1 students scored 8.8 points higher than Year 2 students on the midterm, but only 4.49 points higher on the final. In other words, Year 2 students in this category improved their score by 15.2 points, whereas Year 1 students in this category improved their scores by 10.89 points from midterm to final. This is a substantial improvement on the final, over the midterm, for Year 2 students. It is also 4.3 points higher than the corresponding Year 1 cohort's improvement.

In the moderate performance category, among students who scored 60 or more, but less than 76 points on the midterm, Year 2 students actually scored higher midterm average, than Year 1 students in this category, by 2.07 points. In this category, Year 2 students improved their score on the final by 7.79 points, whereas the Year 1 students had an improvement of 6.25 points over their midterm scores. The improvement in Year 2 students is 1.54 points higher than the Year 1 cohort.

In the high performance category, among students who scored 76 or more points on the midterm, Year 1 students midterm average was 4.03 points higher than Year 2 students. Year 1 students in this category scored 4.32 points higher on the final, on average. While Year 2 students in this category improved their final exam score by 3.8 points, lagging behind by 0.52 points, for a lower improvement on the final exam score, compared to Year 1 cohort.

These results suggest that role-playing activities were highly effective for low performers at midterm, moderately effective for moderate performers, and not effective, possibly even counterproductive, for high performers. However, this conclusion is drawn from a case study comparing two cohorts taught in differ-

Category	Year	Count	Avg ME	Avg FE	Avg FE - Avg ME
<60 ME	Year 1	9	49.05	59.94	10.89
	Year 2	10	40.25	55.45	15.20 (4.31)
>=60 && <76 ME	Year 1	8	66.50	72.75	6.25
	Year 2	7	68.57	76.36	7.79 (1.54)
>=76 ME	Year 1	11	90.18	94.50	4.32
	Year 2	10	86.15	89.95	3.80 (-0.52)

Table 2: Average midterm (ME) and final exam (FE) scores by subgroup category for Year 1 and Year 2 cohorts. Year 2’s difference of FE over ME, relative to Year 1’s difference, is shown in parentheses.

ent years. While the timing of role-playing activities was the only instructional technique that differed, other factors may have contributed to these outcomes. No two classes are the same, even with the same instructor. The two cohorts consist of distinct groups of students, and differences in outcomes could be a reflection of variations in their abilities and motivations. Additionally, external factors, such as events on campus, tutor availability or other year-specific conditions, may have played a role. These findings should be interpreted with caution, as causation cannot be definitively established.

5 Conclusion

For teaching introductory object-oriented programming, role play activities provide a unique viewpoint of understanding for students, and can improve engagement and understanding. The case study presented here provides evidence that role-playing has a positive impact on student learning, particularly for low and moderate performing students, when used as a supplementary tool alongside live coding and memory diagrams. However, for high-performing students, incorporating additional role-playing activities may be ineffective or even counterproductive.

Therefore, instructors must carefully weigh the trade-offs between incorporating more active learning exercises, reviewing additional coding examples, or covering more advanced concepts in their lesson plans. It is essential to consider the performance levels and capabilities of their students when making these decisions.

References

- [1] D.-M. Cordova-Esparza, J.-A. Romero-Gonzalez, K.-E. Cordova-Esparza, J. Terven, and R.-E. Lopez-Martinez. “Active learning strategies in computer science education: A systematic review”. In: *Multimodal Technol. Interact.* 8.6 (2024), p. 50. DOI: 10.3390/mti8060050.
- [2] A. G. S. Raj, J. M. Patel, R. Halverson, and E. R. Halverson. “Role of live coding in learning introductory programming”. In: *Proc. 18th Koli Calling Int. Conf. Computing Education Research*. Koli, Finland, Nov. 2018, pp. 1–8. DOI: 10.1145/3279720.3279725.
- [3] J. Waite and S. Sentance. *Teaching Programming in Schools: A Review of Approaches and Strategies*. Online. Raspberry Pi Foundation, 2021, pp. 1–53. URL: <https://www.raspberrypi.org/app/uploads/2021/11/Teaching-programming-in-schools-pedagogy-review-Raspberry-Pi-Foundation.pdf>.
- [4] S. K. Andrianoff and D. B. Levine. “Role playing in an object-oriented world”. In: *Proc. 33rd SIGCSE Tech. Symp. Computer Science Education*. 2002, pp. 121–125. DOI: 10.1145/563340.563405.
- [5] M. Holliday and D. Luginbuhl. “Using memory diagrams when teaching a Java-based CS1”. In: *Proc. 41st Annu. ACM Southeast Conf.* 2003, pp. 376–381. DOI: 10.1145/1073368.1073453.
- [6] T. Dragon and P. E. Dickson. “Memory diagrams: A consistent approach across concepts and languages”. In: *Proc. 47th ACM Tech. Symp. Computing Science Education*. 2016, pp. 546–551. DOI: 10.1145/2839509.2844587.
- [7] J. Börstler and C. Schulte. “Teaching object-oriented modelling with CRC cards and roleplaying games”. In: *Proc. WCCE*. 2005, pp. 1–5.
- [8] K. Sierra and B. Bates. *Head First Java*. 2nd. Sebastopol, CA: O’Reilly Media, 2005.
- [9] P. M. Newton. “The learning styles myth is thriving in higher education”. In: *Front. Psychol.* 6 (2015), p. 1908. DOI: 10.3389/fpsyg.2015.01908.
- [10] J. S. Jones. “Participatory teaching methods in computer science”. In: *ACM SIGCSE Bull.* 19.1 (1987), pp. 155–160.
- [11] M. R. N. Gari, G. S. Walia, and A. D. Radermacher. “Gamification in computer science education: A systematic literature review”. In: *Proc. ASEE Annu. Conf. Expo.* 2018, pp. 1–10. DOI: 10.18260/1-2--30596.

Design Guidelines for a Rigorous Deep Learning Course*

Mukulika Ghosh, Jamil Saquer
Computer Science Department
Missouri State University
Springfield, MO 65697
`mghosh, jsaquer@missouristate.edu`

Abstract

This paper presents the design and implementation of a rigorous undergraduate Deep Learning (aka Deep Neural Networks) course, developed as part of a newly approved interdisciplinary Data Science major jointly offered by the Computer Science and Mathematics departments at Missouri State University. The course is designed to provide students with a comprehensive understanding of deep learning models, their mathematical foundations, and their practical applications to real-world problems across diverse domains. Unlike approaches that either focus heavily on theoretical rigor or prioritize applied problem-solving, this course adopts a blended pedagogy that emphasizes both the mathematical intuition behind deep learning operations and the practical skills needed to select, implement, and optimize models for various applications.

1 Introduction

Inspired by the high demand for students trained in data science [1, 10], many universities are offering undergraduate majors in data science. Such a major was recently approved by Missouri State University as an interdisciplinary program between the computer science and mathematics departments. Students

*Copyright ©2025 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

complete foundational mathematics courses such as calculus, discrete mathematics, linear algebra, and statistics, alongside core computer science courses including programming and data structures. Deep learning (DL), given its significance in modern data analysis, is a required course for this major.

Despite the growing number of DL courses in higher education, few target undergraduate students specifically. Existing offerings typically either emphasize mathematical theory at the expense of application or focus on tool implementation without developing fundamental conceptual understanding. This division presents a significant teaching challenge, especially in interdisciplinary fields where both theoretical knowledge and practical skills are necessary.

To address these limitations, we present the design and implementation of a rigorous undergraduate DL course that combines theoretical principles with practical application. The course builds upon students' existing mathematical and programming knowledge while introducing them to real-world applications. While primarily designed for data science majors who have completed the prerequisite coursework, the content remains approachable for students from related STEM disciplines such as physics and engineering, with similar foundational preparation.

The main contribution of this paper is an undergraduate course design that balances mathematical intuition with practical application, preparing students for both academic and industry careers in AI and data science. By implementing a blended pedagogy that integrates project-based learning with theoretical foundations, the course empowers students to understand the mathematical underpinnings of deep learning models and apply them effectively across various domains. This approach supports the interdisciplinary nature of the data science program at Missouri State University.

2 Literature Review

Deep learning courses have been increasingly offered at universities worldwide, though primarily at the graduate level. Tirkeş et al. [9] developed undergraduate AI courses to address workforce needs in Turkey, proposing a two-course structure: Fundamentals of Machine Learning followed by Applications of Deep Learning. Their curriculum combines theoretical lectures with laboratory sessions using tools like TensorFlow to apply concepts to practical problems.

Similarly, Zhang et al. [13] noted that China's rapidly growing AI industry requires an educational approach beyond traditional theory-focused teaching. They advocate for an integrated model that combines theoretical foundations with hands-on practice to better prepare students for industry demands.

In the United States, while several universities offer deep learning courses [6, 8, 11], undergraduate offerings remain limited at institutions outside Ivy

League and top-tier research universities [4, 7]. Johnson [7] described an undergraduate deep learning course taught at the University of New Hampshire that introduced backpropagation, gradient descent, CNNs, RNNs, and briefly covered generative modeling through a project-based approach. Our course differs from Johnson’s by covering a broader range of deep learning architectures and providing more mathematical depth while maintaining the balance between theory and application.

3 Course Description and Outcomes

In the design of the course, we identified the following detailed learning outcomes for this course:

1. Fundamentals of deep learning models

- L1: Develop a strong understanding of data representation and interpretation for machine learning and deep learning tasks, incorporating fundamental concepts from linear algebra and probability.
- L2: Understand the fundamental operations in neural network including behavior of activation functions and interpretation of loss functions.
- L3: Implement and analyze algorithms such as gradient descent and back propagation on simple networks.

2. Comparative analysis of existing models

- L4: Understand the core operations of widely used neural networks, including convolution in Convolutional Neural Networks (CNNs), recurrent connections in Recurrent Neural Networks (RNNs), attention mechanisms in Transformers, and adversarial training in Generative Adversarial Networks (GANs).
- L5: Implement and analyze various neural network architectures, including Visual Geometry Group Network (VGGNet), Residual Network (ResNet), Long Short Term Memory (LSTM), and Gated Recurrent Unit (GRU). Gain a comparative understanding of selecting the appropriate architecture for specific tasks while balancing performance and computational efficiency.
- L6: Understand design principles behind popular network architectures, such as selecting the number of channels and filters across layers, utilizing residual connections in CNNs, implementing gating mechanisms in LSTMs and GRUs, and leveraging self-attention and bidirectional context in Transformers.

3. Essentials in training deep learning models

- L7: Understand the cause behind common issues found in training deep learning models such as exploding and vanishing gradients, overfitting, slow convergence, and computational complexity.
- L8: Learn common techniques for addressing these challenges, including regularization, initialization, and optimization methods. Develop a comparative understanding of selecting the appropriate technique for different task scenarios.
- L9: Optimize and fine-tune neural network performance by applying hyperparameter tuning techniques, such as learning rate adjustment, batch size selection, and regularization strategies. Develop a comparative understanding of choosing the right hyperparameters for different tasks while balancing accuracy and efficiency.

This course provides students with practical proficiency in deep learning by introducing essential Python libraries such as NumPy, PyTorch, and TensorFlow. Students develop foundational skills in data generation, loading, augmentation, custom neural network construction, and training loop optimization. They also gain expertise in using pre-trained networks (e.g., ResNet, VGGNet) as backbones for fine-tuning or feature extraction within meta-architectures like U-Net and R-CNN, for real-world applications.

Prerequisite: The course requires students to have basic programming skills especially in Python, a course in data structures, and recommended mathematical skills including discrete structures, calculus, and a course in statistics. Knowledge in linear algebra is recommended, as our students learn the basics of matrices such as matrix multiplication in the discrete structures course.

4 Course Content

Table 1 provides the schedule and topics covered in the course. The details of the content for each topics are described below.

Preliminaries: This section reviews fundamental concepts in linear algebra and probability necessary for understanding advanced topics in deep learning. It covers linear representations like scalars, vectors, matrices, and tensors, along with operations such as addition and multiplication. The probability component includes marginal, joint, and conditional probabilities, as well as the sum rule, product rule, and Bayes' rule. Beyond basic definitions, the course emphasizes interpretation to help students connect these concepts when revisiting them in the context of neural network design. For instance, students

Table 1: Tentative schedule and topics covered in the course

Week	Topics
1	Preliminaries
2-3	Linear Neural Networks
3-4	Multi-layer Perceptron
5-6	Principles in training
7-9	Convolution Neural Networks
9-10	Recurrent Neural Networks
11-12	Attention and Transformers
13-14	Generative Modeling

will later explore how probability is applied in multi-class classification tasks. Given the possibility of a diverse backgrounds of students, this foundational topic ensures they can develop a deeper understanding of deep neural networks.

Linear Network Networks: This topic covers the fundamental building blocks of neural networks, focusing on the neuron and its application in tasks such as linear regression, logistic regression, and softmax regression. It also introduces the basics of the gradient descent algorithm, explaining its role in optimization tasks, along with the loss functions used in training these simple networks. The interpretation of the update step in the gradient descent algorithm is explored, as well as the effect of the learning rate hyperparameter on convergence. This topic provides the foundational knowledge needed to understand the selection of loss functions and the basic optimization techniques used in training deep learning models.

Multi-Linear Perceptrons: This topic explains what "deep" means in deep learning models and the implications of increasing the number of layers in networks, from handling simple tasks with linearly separable data to modeling more complex functions. It also covers the details of the backpropagation algorithm, including how errors are calculated and the impact of choosing different activation functions. This topic provides an in-depth understanding of the backpropagation process, introducing students to the challenges faced in deep learning models and key design decisions.

Principles in training: This topic explores common challenges encountered when training neural network models, including the underlying causes such as artifacts in the topology of the loss function landscape and non-uniform scaling of parameters. It also covers issues like vanishing and exploding gradients,

as well as overfitting. The second part of the topic introduces techniques to address these challenges, such as batch normalization, regularization, and optimization methods. This topic provides a comprehensive understanding of the design principles behind various initialization, regularization, and optimization techniques, along with practical guidance on tuning hyperparameters.

Convolution Neural Networks (CNNs): This topic introduces the rationale behind using CNNs for image analysis tasks, where they are commonly applied, and covers basic operations such as convolution, padding, and pooling. It also explains how the shape of feature maps changes across layers in traditional CNNs, the number of parameters, and their impact on computational complexity. This foundational knowledge leads into the second part of the topic, which reviews and compares existing CNN models, including AlexNet, VGGNet, InceptionNet, ResNet, and SENet. The topic provides students with a comprehensive understanding of popular architectures used for common problems, as well as a performance analysis of the operations involved and their impact on the results achieved.

Recurrent Neural Networks(RNNs): This topic introduces “deep” neural networks in the temporal dimension through the use of recurrent connections, explaining their usefulness for text-based data. It revisits the issues of vanishing and exploding gradients, providing an interpretation of these problems in the context of memory in text data analysis. Within this framework, different types of recurrent networks, such as GRU and LSTM, along with their respective gates, are discussed. This provides the students a deep understanding of how RNNs handle temporal data in real world applications.

Attention and Transformers: This topic introduces the concept of attention mechanisms and their role in enhancing neural networks, particularly in handling sequential data in natural language processing (NLP) tasks. The topic covers the mechanics of self-attention and multi-head attention, which allow models to focus on different parts of the input sequence simultaneously. It also introduces the various Transformer architectures such as BERT, which leverages attention mechanisms to achieve high performance. The students are provided with a clear understanding of the attention mechanisms and the reasons for their success in Transformers.

Generative Modeling: This topic covers generative modeling in the context of Generative Adversarial Networks (GANs), autoencoders, and variational autoencoders (VAEs), including key concepts such as loss functions, ELBO (Evidence Lower Bound), latent space representation, and the reparameterization

trick. It also includes a comparative study of various GAN architectures and their applications in solving different generative tasks. The students are provided with comprehensive understanding of generative modeling techniques and insights into the role of latent space representation and the reparameterization trick in improving model performance.

Readings The textbooks [12] (primary), [2], [3] and [5] provide students with foundational knowledge and practical skills needed for the course. Most of the textbooks are available online for free and serve as additional resources for the students. Specific page and section references are provided on the Learning Management System for easy access to the relevant material.

5 Course Deliverables and Grading

The final grade is computed based on the following components:

1. **Quizzes 20% and Exam 15%** There are at least eleven in-class quizzes and one midterm, given midway, during the semester. We chose not to give a final exam so that students can concentrate on working on their projects. The lowest quiz grade is dropped from consideration. The quizzes and exam are included in the course deliverables to reinforce and assess students' understanding of fundamental concepts involved in the deep learning model design. Example questions include determining the output or behavior of a network with a given set of weights, finding the appropriate weights to model specific behavior, calculating the total number of parameters and the shape of feature tensors, and computing attention for a given set of inputs. Questions are designed to provide students with hands-on experience in understanding the functioning of various components and operations in neural networks before implementing them in software, thereby enhancing their ability to debug and optimize novel network designs.
2. **Homework Assignments 35%** Assignments consist primarily of programming tasks, text-based question-and-answer exercises and essay-based questions. Programming assignments evaluate the implementation of techniques learned in the class. Deliverables include report and code implementation. Text-based assignments include problems and exercises that are designed to reinforce the understanding of the concepts taught in the class. They also include reading, reviewing and critically analyzing blogs and/or a conference paper.

The homework assignments serve as a foundation for completing the final project. These assignments include tasks with straightforward objectives,

such as implementing gradient descent, the backpropagation algorithm, comparing two convolutional networks, and performing the forward and backward passes of an LSTM model. The implementation exercises offer students hands-on experience with popular libraries like PyTorch, guiding them through model creation and training from scratch.

Each assignment also includes a report section that involves experimental analysis of the software developed. A series of questions are designed around the problem, such as examining how performance (in terms of loss and accuracy) changes with different learning rates, initialization methods, and the presence or absence of regularization. This provides students with experience in hyperparameter tuning and teaches them how to design experiments for their final project.

3. **Final Project 30%** The final project is a major component of the course. Teams of 2-3 students work on a deep learning application, either from a pre-determined list of topics and datasets provided by the instructor or through an approved student proposal. Examples from this semester include “Fault Detection in Wireless Sensor Networks” and “Vegetation Classification to Prevent Robot Entanglement.”

To ensure steady progress throughout the semester, students submit a mid-term report outlining the problem statement, proposed method, implementation progress, and evaluation plan. At the end of the semester, teams deliver a final project report, their software implementation, and an oral presentation. The presentation is evaluated based on clarity, depth of background knowledge, and the overall quality of the work.

6 Observations and Recommendations

This course is being offered to undergraduate students for the first time in the Spring 2025 semester, so complete data on the effectiveness of its design is not yet available. However, we have collected data through Week 9, including students’ performance on the midterm exam, quizzes, and homework assignments. Based on this preliminary data, students demonstrate better performance on programming and homework assignments compared to their performance on exams and quizzes. This is similar to students’ performance in other courses in the program.

6.1 Discussion and recommendations

We believe that offerings of this course for undergraduate students should aim to balance foundational knowledge with reduced mathematical complexity. To achieve this, we recommend and apply the following:

- We recommend that project deliverables, such as the project proposal, be presented as an informal proof-of-concept in slide format rather than a written paper, aligning with industry expectations where most students aspire to build their careers. However, the final project report and presentation should be retained to provide research experience.
- We suggest maintaining the current assignment structure, which includes objective experimental questions, enabling students to apply these methods in their project evaluations. Including guest lecture from industry or research conducted by graduate students or faculty is also recommended.
- Many undergraduate students, particularly outside R1 universities, have little to no experience in reading technical papers from journals and conferences. To ease them into this process, we recommend reading assignments that focus on blog reports or comparative analyses of two to three papers that are less technically dense. Selecting papers from OpenReview is also encouraged, as students can access publicly available review comments, helping them better understand and analyze research work.
- We recommend inclusion of in-class practice sessions for math-based problems, such as determining weights for specific behaviors and calculating attention values for given weights in a network. Additionally, for quiz-based assessments, a review session focusing on the quiz questions is preferred. The frequency of quizzes could be reduced while increasing the number of midterm exams to provide regular checkpoints. However, we still recommend not having a final exam, allowing students to integrate their knowledge from the latter part of the semester into their projects. The percentage of final grade in quizzes and exams in consideration should be kept lower than project and homework.

7 Conclusion

In this paper, we present a comprehensive deep learning course designed for undergraduate students in an interdisciplinary data science program. The course balances theoretical foundations and practical applications to address the growing demand for skilled data science professionals. Initial performance data indicates that students perform better in programming and homework assignments compared to exams and quizzes. For future work, we plan to collect additional data to evaluate the effectiveness of this course design for undergraduate students.

References

- [1] Adam SZ Belloum et al. “Bridging the Demand and the Offer in Data Science”. In: *Concurrency and Computation: Practice and Experience* 31.17 (2019).
- [2] Christopher M. Bishop and Hugh Bishop. *Deep Learning: Foundations and Concepts*. 2023. ISBN: 978-3-031-45468-4.
- [3] François Chollet. *Deep Learning with Python*. Second. Manning, 2021. ISBN: 9781617296864.
- [4] Anthony Clark. *CS 152: Neural Networks*. Spring 2025. <https://cs.pomona.edu/classes/cs152/archive/24-25spring/>. 2025.
- [5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Book in preparation for MIT Press. MIT Press, 2016. URL: <http://www.deeplearningbook.org>.
- [6] Indiana University Indianapolis - Luddy School of Informatics, Computing, and Engineering. *CSCI-P 558 Deep Learning*. <https://luddy.indianapolis.iu.edu/degrees/courses/csci-p558>. 2023.
- [7] Jeremiah W. Johnson. “Teaching Neural Networks in the Deep Learning Era”. In: *Journal of Computing Sciences in Colleges* 34.6 (2019), pp. 16–25.
- [8] Bhiksha Raj and Rita Singh. *11-785 Introduction to Deep Learning*. Spring 2025. <https://deeplearning.cs.cmu.edu/S25/index.html>. 2025.
- [9] Güzin Tirkeş et al. “An Undergraduate Curriculum for Deep Learning”. In: *2018 3rd International Conference on Computer Science and Engineering (UBMK)*. IEEE. 2018, pp. 604–609.
- [10] U.S. Bureau of Labor Statistics. *Occupational Outlook Handbook–Data Scientists*. <https://www.bls.gov/ooh/math/data-scientists.htm>. 2023.
- [11] Richard Zemel. *COMS 4995: Neural Networks and Deep Learning*. <https://www.cs.columbia.edu/~zemel/Class/Nndl/syllabus.pdf>. 2023.
- [12] Aston Zhang et al. *Dive into Deep Learning*. <https://D2L.ai>. Cambridge University Press, 2023.
- [13] Yujie Zhang and Xia Qiu. “Exploration of Teaching Mode Combining Theory and Practice in Deep Learning Courses”. In: *Journal of Education and Educational Research* 10.2 (2024), pp. 26–29.

Gamification in Public Event Night: Evaluating the Impact of Unplugged Activity on K-12 Students' Computational Thinking in a Short Time Frame.*

Seong Yeon Cho¹, Jae Geun Lee², Paul Kim³

¹Computer Science

²Computer Science

³Computer Science Department

Bridgewater State University

Bridgewater, MA 02324

{scho,j22lee}@student.bridgew.edu

{p2kim}@bridgew.edu

Abstract

Computational thinking (CT) is a necessary skill that must be learned in today's tech-focused world, especially within K-12 education, where early learning of problem-solving skills and algorithmic thinking is beneficial in everyday life. Various ways have been used to cultivate this skill in these early education phases [6]. Our approach was to use an unplugged gamified coding activity that involved creating a learning environment that required hands-on participation and critical computational thinking. This study examines the impact of a gamified coding activity on computational thinking (CT) skills of K-12 students in a public stem event setting where each student has around 30 minutes of solving time. This study also observes the influence of the short-term intervention on algorithmic thinking and problem solving abilities by incorporating a before-and-after survey, in which students were asked to write down the

*Copyright ©2025 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

steps to brush their teeth before and after the activity. This data was used to evaluate improvements in student procedural thinking and logical structure. The findings suggest that gamification enhances engagement and helps students develop clearer and more structured algorithms. These results contribute to research on effective teaching strategies for CT in K-12 education and to improving the learning environment.

1 Introduction

Computational thinking (CT) has become a fundamental skill for students in the 21st century, with education systems worldwide integrating programming and problem solving into the K-12 curriculum [4]. Moreover, as computational literacy expands, innovative teaching methods, such as interactive coding platforms and especially gamification, have gained attention for their potential to enhance learning engagement [5]. Gamification in education has been shown to increase student participation, improve motivation, and provide immediate feedback loops that reinforce learning [7]. Unlike traditional lecture-based methods, gamification encourages students alone or in a group to actively engage with computational concepts through problem solving, experimentation, and interactive challenges. With this knowledge, this study introduces a gamified coding activity using P5.js, a JavaScript library designed for creative coding, which allows students to explore visual programming concepts interactively. The activity incorporates Optical Character Recognition (OCR) technology, allowing students to scan physical code blocks and immediately see their code results. This integration provides a tangible, hands-on coding experience, reinforcing algorithmic thinking and problem decomposition. Moreover, this project explores the effectiveness of a short-term, engaging activity where students interact with computational concepts in an informal setting. Beyond coding, this study also examines the impact of gamification on students' procedural thinking skills. To measure this, students were asked to write down the steps for brushing their teeth before and after participating in the P5.js coding activity. This comparison before and after allows us to analyze whether exposure to gamified computational exercises improves a student's ability to think algorithmically in everyday scenarios.

2 Related Works

2.1 Computational Thinking in K-12

Computational thinking (CT) has been widely recognized as an essential skill for students, encompassing problem decomposition, abstraction, algorithmic design, and debugging [8]. The integration of CT in K-12 curricula has been

widely recognized as an effective strategy to prepare students for the digital economy [7]. Research categorizes CT learning into plugged (digital) and unplugged (physical, non-digital) environments, both of which contribute to skill development [6]. Studies have found that unplugged approaches can serve as a 'priming' step, helping students grasp algorithmic concepts before engaging in coding [3]. Studies have emphasized algorithmic thinking as a core component of CT, highlighting the importance of step-by-step problem solving and the ability to build effective algorithms [1]. This study builds on this research by incorporating OCR technology to facilitate a unique gamified approach to algorithmic thinking.

2.2 Gamification in Computational Thinking Education

Gamification is increasingly used as an effective teaching strategy in computer science education. Studies have shown that interactive and game-based activities improve motivation, engagement, and retention rates among K-12 students [2]. Unplugged approaches have also been found to be effective in teaching algorithmic thinking by using hands-on and tangible experiences [9]. Incorporating gamification into CT education improves participation and engagement, which are often limited in traditional computer science classrooms [7]. This study extends this research by applying gamification to a short-term coding activity and evaluating its impact compared to traditional teaching methods.

2.3 Assessment of Computational Thinking

Assessing computational thinking remains a challenge due to the various available methods, including quizzes, hands-on projects, debugging exercises, and algorithm development [1]. Although digital tools offer automated assessments, unplugged activities often require manual evaluation through observation and student reflection [6]. This study uses a mixed-method approach, including:

- Pre- and post-activity surveys to assess the self-perceived learning outcomes of students.
- Observation of student engagement and participation during the session.
- An analysis of OCR-executed code outputs to measure the accuracy and correctness of student-constructed algorithms.

By combining qualitative and quantitative data, we aim to provide a comprehensive evaluation of the effectiveness of gamification in CT learning.

Activity Sheet

Your (your child's) age: _____

**This age information will be used to analyze data based on age and will not be used for any other purpose.*

First Task

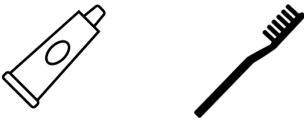
Let's assume that we have a toothbrush and toothpaste as shown below. Let's write instructions to brush your teeth using them.

Last Task

Let's try the first task which is to write instructions to brush your teeth again. Then let's talk with your parents or friends about what improvements you have in the instructions.

Instructions:

-- Hold the toothbrush with one hand



Instructions:

-- Hold the toothbrush with one hand

(a) Starting question for survey.

(b) Ending question for Survey.

Figure 1: Survey Sheet used by K-12 student for data collection.

3 Methodology

3.1 Setting and Participants

This study was conducted in an open classroom environment, where K-12 students and their families participated in an interactive coding activity using puzzle blocks of code. Students with diverse levels of prior coding experience engaged in hands-on programming. The participants' age ranged from 8 to 15, with varying levels of computational exposure. Each participant was given approximately 30 minutes to complete the activity. In Figure 1 shows the question used to test the improvement in CT of the students.

3.2 Activity Design

The study consisted of three sequential tasks:

1. Pre-Activity Task: Writing Procedural Steps to Brush Your Teeth.
Before engaging in the coding activity, students were asked to write down step-by-step instructions for brushing their teeth, as shown in Figure 1a. This served as a baseline measure of their ability to break down a familiar task into discrete logical steps.

2. P5.js Gamified Coding Activity.

The students were introduced to a Turtle Puzzle game in which they used puzzle block codes made from paper to create structured commands. The puzzle activity required students to use two primary commands:

- `Line()`: Move forward while drawing a line.
- `Rotate()`: Turn left 90 degrees.

The students had to join a sequence of these commands to create shapes such as squares. Once their puzzle was assembled, they scanned it using the OCR program, which converted their sequence into actual P5.js code that was executed visually, as shown in Figure 2.

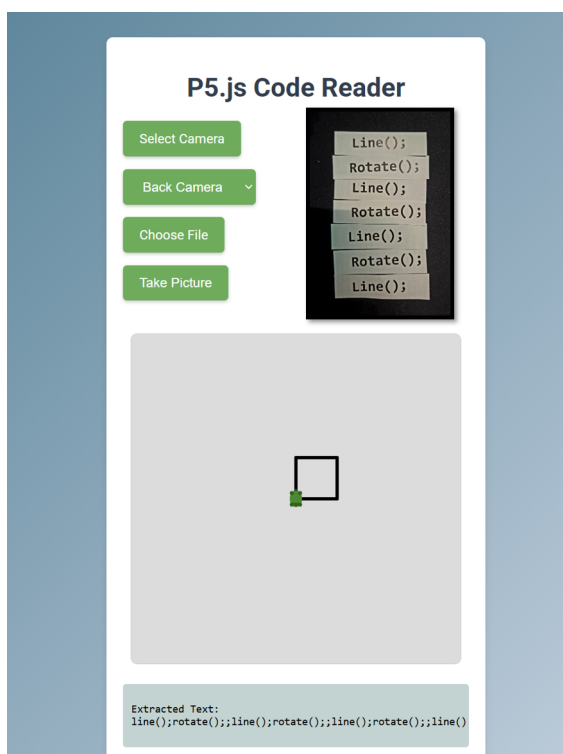


Figure 2: picture of the OCR program running the puzzle block.

3. Post-Activity Task: Rewrite the Procedural Steps.

After completing the P5.js activity, students were asked to write down the brushing steps again as shown in Figure 1b. This helped measure

any improvements in their ability to structure procedural instructions, a key aspect of computational thinking.

3.3 Data Collection and Analysis

The study collected both qualitative and quantitative data: Survey Data (Before and After Writing Tasks) Student responses were categorized into "Improved" and "No Improvement" groups based on clarity, detail, and logical structure. Examples of responses were analyzed for patterns in procedural thinking.

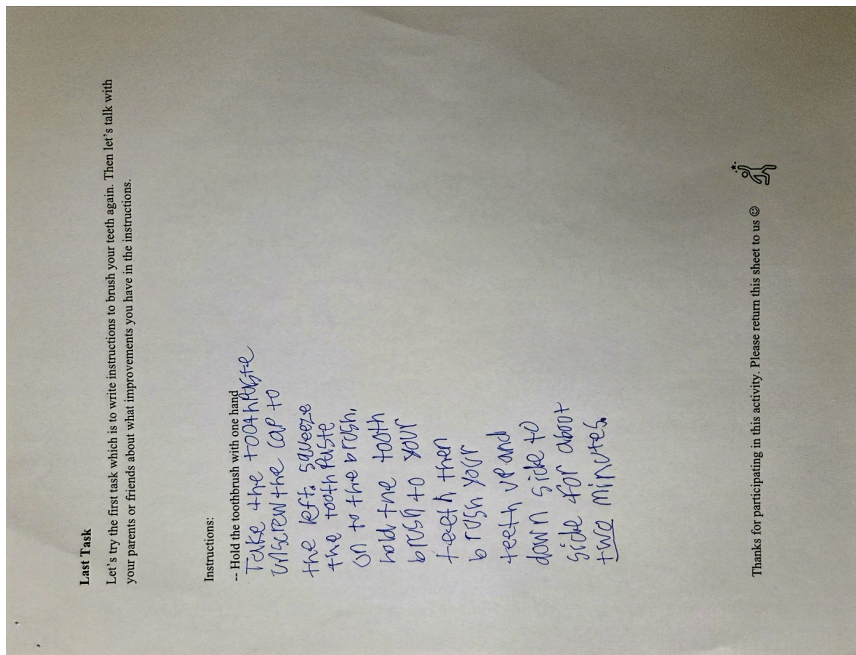
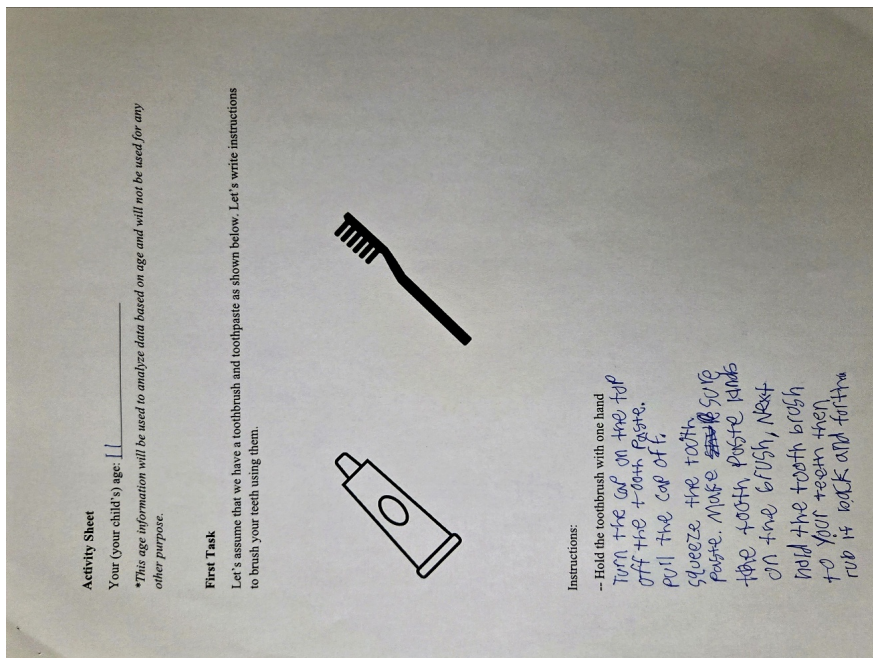
4 Results

4.1 Engagement and Motivation

Students showed high levels of engagement, particularly due to the immediate feedback loop from the OCR-based execution system. Gamification elements (e.g. progressive challenges and visual rewards) kept students motivated and persistent in problem solving.

4.2 Computational Thinking Gains (Measured via Procedural Writing Task)

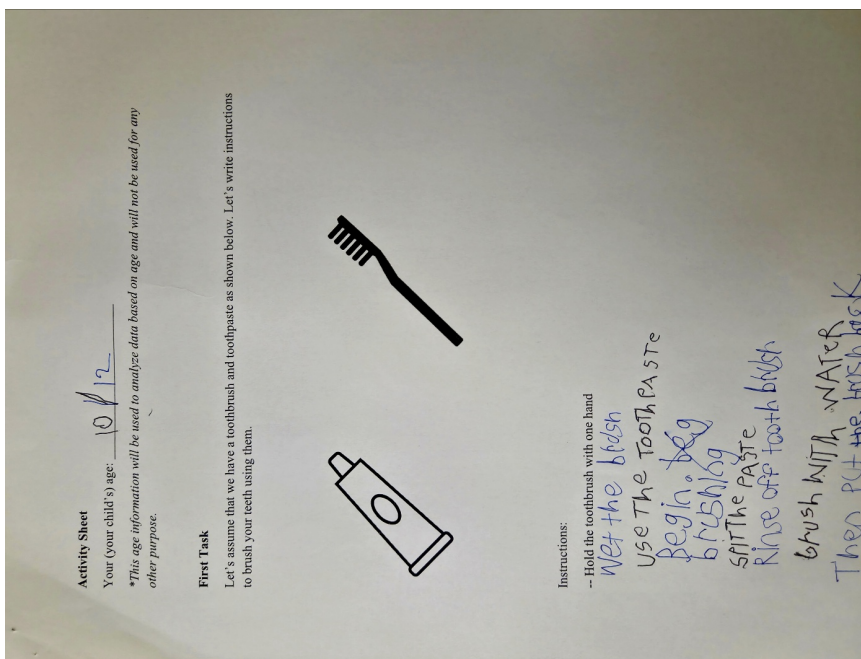
Out of 17 students, 11 demonstrated improvement in their ability to structure procedural instructions, while 6 students did not show any change. Figures 3 and Figure 4 show the difference between the students who improved and the students who did not.



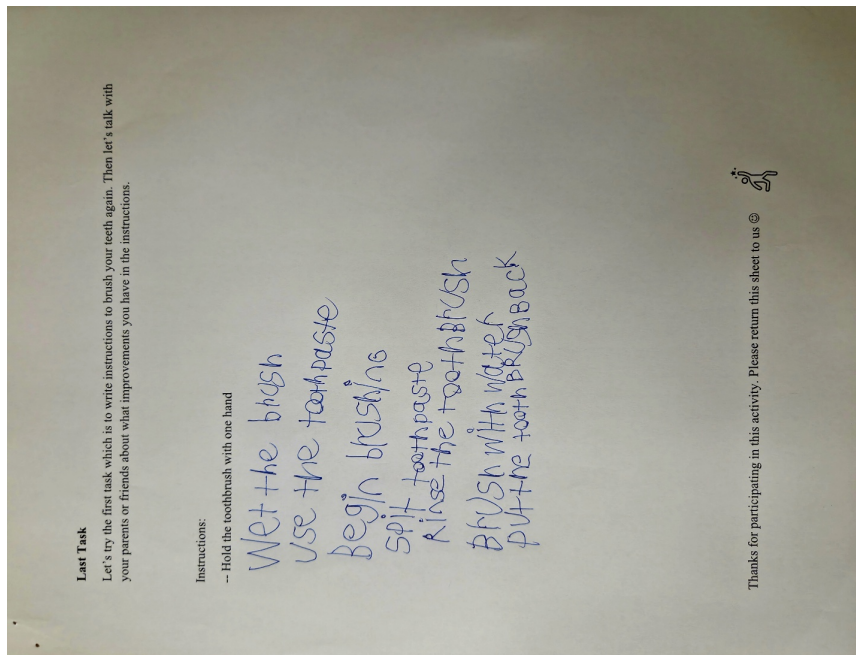
(a) Picture of one student writing the instructions before activity.

(b) Picture of the same student in writing the instructions after activity.

Figure 3: Example data of student who improved.



(a) Picture of one student writing the instructions before activity.



(b) Picture of the same student in writing the instructions after activity.

Figure 4: Example data of student who did not improved.

4.3 Challenges and Limitations

Some younger students struggled with reading and understanding code snippets, requiring more scaffolding and guided assistance. Furthermore, time constraints in a short-term intervention limited the depth of exploration of advanced CT concepts along with the completion of more shapes. Moreover, the program would sometimes not correctly output the code results due to oddly placed code blocks.

5 Discussion

5.1 Key Findings

This study aligns with previous research suggesting that gamification and hands-on activities can improve algorithmic thinking [2] [7]. Key takeaways include

- Students who improved in procedural writing showed greater clarity and logical structure in their responses.
- The gamified approach using P5.js encouraged sequential problem solving, leading to improvements in step-by-step thinking.

5.2 Limitations and Future Work

- Short-term intervention: A 30-minute session may not be sufficient for all students to show measurable gains.
- Varying levels of cognitive development: Younger students (ages 8-10) needed more help and were less likely to show improvement compared to older participants (ages 12-15).

6 Conclusion

This study highlights the potential of gamification and OCR-based execution in short-term computational learning environments. The integration of P5.js puzzle-based learning demonstrated that students could improve their ability to structure procedural instructions after a short intervention. However, longer-term participation may be required for younger students to develop sustained improvements in computational thinking.

References

- [1] Giorgia Adorni et al. “Development of algorithmic thinking skills in K-12 education: A comparative study of unplugged and digital assessment instruments”. In: *Computers in Human Behavior Reports* 15 (2024), p. 100466. ISSN: 2451-9588. DOI: <https://doi.org/10.1016/j.chbr.2024.100466>. URL: <https://www.sciencedirect.com/science/article/pii/S245195882400099X>.
- [2] Wan Chong Choi and Chi In Chang. “The Students’ Perspective on Computational Thinking through Flipped Classroom in K-12 Programming Course”. In: *Proceedings of the 2024 8th International Conference on Education and Multimedia Technology*. ICEMT ’24. Tokyo, Japan: Association for Computing Machinery, 2024, pp. 106–113. ISBN: 9798400717611. DOI: 10.1145/3678726.3678729. URL: <https://doi.org/10.1145/3678726.3678729>.
- [3] Wendy Huang and Chee-Kit Looi. “A critical review of literature on “unplugged” pedagogies in K-12 computer science and computational thinking education”. In: *Computer Science Education* 31.1 (2021), pp. 83–111. DOI: 10.1080/08993408.2020.1789411. eprint: <https://doi.org/10.1080/08993408.2020.1789411>. URL: <https://doi.org/10.1080/08993408.2020.1789411>.
- [4] Redar Ismail, Theresa A. Steinbach, and Craig S. Miller. “A Guide Towards a Definition of Computational Thinking in K-12”. In: *2022 IEEE Global Engineering Education Conference (EDUCON)*. 2022, pp. 801–810. DOI: 10.1109/EDUCON52537.2022.9766703.
- [5] Yasmin B. Kafai and Chris Proctor. “A Revaluation of Computational Thinking in K-12 Education: Moving Toward Computational Literacies”. In: *Educational Researcher* 51.2 (2022), pp. 146–151. DOI: 10.3102/0013189X211057904. eprint: <https://doi.org/10.3102/0013189X211057904>. URL: <https://doi.org/10.3102/0013189X211057904>.
- [6] Aycaan Çelik Kirçali and Nesrin Özdenler. “A Comparison of Plugged and Unplugged Tools in Teaching Algorithms at the K-12 Level for Computational Thinking Skills”. In: *Technology, Knowledge and Learning* 28.5 (2023), pp. 1485–1513. DOI: 10.1007/s10758-021-09585-4. URL: <https://link.springer.com/article/10.1007/s10758-021-09585-4>.
- [7] Sze Yee Lye and Joyce Hwee Ling Koh. “Review on teaching and learning of computational thinking through programming: What is next for K-12?”. In: *Computers in Human Behavior* 41 (2014), pp. 51–61. ISSN: 0747-5632. DOI: <https://doi.org/10.1016/j.chb.2014.09.012>. URL: <https://www.sciencedirect.com/science/article/pii/S0747563214004634>.

- [8] Belén Palop et al. “Redefining Computational Thinking: A Holistic Framework and Its Implications for K-12 Education”. In: *Education and Information Technologies* (2025). DOI: 10.1007/s10639-024-13297-4. URL: <https://link.springer.com/article/10.1007/s10639-024-13297-4>.
- [9] Amanda Peel, Troy D. Sadler, and Patricia Friedrichsen. “Algorithmic Explanations: an Unplugged Instructional Approach to Integrate Science and Computational Thinking”. In: *Journal of Science Education and Technology* 31 (2022), pp. 428–441. DOI: 10.1007/s10956-022-09965-0. URL: <https://link.springer.com/article/10.1007/s10956-022-09965-0>.

Reviewers — 2025 Midsouth Conference

Oleksiy Al-saadi	<i>Sonoma State University, Rohnert Park, CA</i>
Abbas Attarwala	<i>California State University, Chico, Chico, CA</i>
Rickey Casey	<i>Campbellsville University, Clarksville, AR</i>
Haiyan Cheng	<i>Willamette University, Salem, OR</i>
George Dimitoglou	<i>Hood College, Frederick, MD</i>
Michael Flinn	<i>Frostburg State University, Frostburg, MD</i>
David Furcy	<i>University Wisconsin Oshkosh, Oshkosh, WI</i>
Robin Ghosh	<i>Arkansas Tech University, Russellville, AR</i>
Brian K Hare	<i>University of Missouri - Kansas City, Kansas City, MO</i>
Gongbing Hong	<i>Georgia College and State University, Milledgeville, GA</i>
Charles Glen Hoot	<i>Northwest Missouri State University, Maryville, MO</i>
David Hovemeyer	<i>Johns Hopkins University, Baltimore, MD</i>
Deborah Hwang	<i>University of Evansville, IN</i>
Charles Donald Lake	<i>Coastal Alabama Community College, Bay Minette, AL</i>
David L. Largent	<i>Ball State University, Muncie, IN</i>
Ed Lindo	<i>Regis University, Denver, CO</i>
Mohamed Lotfy	<i>Utah Valley University, Orem, UT</i>
Baochuan Lu	<i>Southwest Baptist University, Bolivar, MO</i>
Sugandha Malviya	<i>Ball State University, Muncie, IN</i>
Sean McCulloch	<i>Ohio Wesleyan University, Delaware, OH</i>
David Middleton	<i>Arkansas Tech University, Russellville, AR</i>
Mika Lee Morgan	<i>Midwestern State University, Archer City, TX</i>
Muhammad Rahman	<i>Clayton State University, Atlanta, GA</i>
Kenneth Rouse	<i>LeTourneau University, Longview, TX</i>
Michael Shindler	<i>UC Irvine, Irvine, CA</i>
Scott Sigman	<i>Drury University, Springfield, MO</i>
Takako Soma	<i>Illinois College, Jacksonville, IL</i>
Maria Weber	<i>Saint Louis University, Maryland Heights, MO</i>
Karen Works	<i>Florida State University, Panama City, FL</i>
Mudasser F Wyne	<i>National University, Westland, MI</i>