

Using Generative AI to Produce Teaching Aids

Brian Butka¹, Catherine Tew², Alessia Tripadelli³, George Pozek⁴

Embry-Riddle Aeronautical University, Daytona Beach, USA^{1, 2, 3, 4}

Abstract

This study investigates the effectiveness of generative AI in developing interactive teaching aids designed to improve the student's understanding of digital logic and binary arithmetic. Unlike previous work that primarily focuses on AI-assisted content creation, this research uses a novel approach where generative AI is leveraged to produce Python programs acting as dynamic learning aids. These programs, implemented as Google Colab notebooks, guide students through step-by-step problem solving, for example, detailing the borrowing process in binary subtraction and simplifying Boolean expressions using Karnaugh maps.

A mixed methods evaluation was conducted with an 80-student classroom to measure both engagement and learning outcomes. Quantitative analysis of student performance and qualitative feedback from surveys indicate that these AI-generated aids are particularly beneficial for students in the lower performance quartile. In addition, iterative prompt engineering and program refinements, guided by specific syntax and formatting challenges, were employed to optimize the clarity and instructional value of the outputs. Our findings suggest that integrating generative AI into teaching tool development can not only reduce instructor workload but also enhance student comprehension of complex technical concepts.

By aligning technical implementation with targeted research results, this work provides a framework to advance AI-assisted education. The reviewed literature, which ranges from AI-based code generation to the evaluation of digital logic teaching methodologies, supports the design decisions made in this study and underscores the potential for further research in this domain.

Keywords: Generative AI, Python Notebook, Learning Aid

1. Introduction

The rapid evolution of generative artificial intelligence (AI) has transformed multiple domains, with education emerging as a key area of impact. In recent years, AI has gone beyond automating content creation to facilitate the development of interactive, student-centered learning tools. This study investigates a novel application of generative AI: the automated creation of programs that serve as learning aids to support classroom instruction in subjects such as arithmetic and digital logic.

Currently, relevant research focuses primarily on three areas: assessing the capabilities of generative AI in programming education, generating teaching materials, and analyzing student work. However, the use of generative AI to develop interactive programs that guide students through complex problem solving processes remains underexplored. This research addresses that gap by leveraging AI-generated Python programs, deployed as Google Colab notebooks, to illustrate key concepts, such as the mechanics of binary subtraction and the intricacies of Boolean logic simplification.

To evaluate the effectiveness of these tools, our approach involves both qualitative and quantitative assessments. Implemented in a classroom of 80 students, the study examines the impact of these learning aids on student engagement and comprehension. Student feedback and performance metrics are analyzed to determine how these interactive, AI-generated programs can enhance understanding, particularly for learners facing difficulties with foundational concepts.

The paper is organized as follows. First, we review the relevant literature to establish the current state of generative AI applications in educational contexts. Next, we detail the design and implementation of our AI-driven learning aids and describe the classroom environment in which these tools were deployed. Following this, we present an analysis of student responses and learning outcomes. Finally, we discuss the implications of our findings for future research and broader integration of AI-assisted teaching tools in educational practices.

2. Literature Overview

A review of the literature on the use of generative AI in education reveals three primary areas of research focus:

- Evaluating generative AI capabilities in programming education,
- Using generative AI to generate teaching materials,
- Analyzing student work with generative AI

2.1 Assessing Generative Ai Capabilities in Programming Education

This area investigates the use of generative AI in code generation for programming exercises. A significant focus has been on evaluating their performance in generating code and solving classroom exercises [1]–[11]. Generative AI has been found to typically match or surpass the performance of average students in tasks related to generating code [12], [13]. Research also indicates that code produced by ChatGPT can be accurately differentiated from student-generated code, yet the tools used to identify text created by generative AI often yield a significant number of false positives.

2.2 Generating Teaching Materials

Research on using generative AIs to generate teaching materials is still emerging. These efforts have ranged from generating appropriate learning objectives [14], generative feedback on responses to open-ended questions [15], generation of programming exercises and code explanations [16], producing educational content [17] and evaluating concerns about the generated content [18]. While the exercises generated by generative AIs were innovative and practical, authors have cautioned that they may require further refinement by instructors before being shared with students. The consensus among the papers is that generative AIs can significantly reduce the time instructors spend creating teaching materials.

2.3 Analyzing Student Work with Generative Ai

Research is also advancing in the use of generative AIs to assist in assessing student work [19]–[22]. Three of the papers examine paper details using generative AIs to help identify bugs and provide feedback on student work [19]–[21]. While the remaining paper examines the ethics and perceptions of using AI to assess student work [22].

3. Binary Arithmetic Challenges

Binary subtraction introduces specific challenges as students must understand base-2 place values and accurately apply binary borrowing rules. Often, students circumvent these challenges by converting binary numbers to decimal, performing the subtraction, and then converting the result back to binary. While this method produces the correct result, it bypasses a foundational understanding of binary arithmetic, which can cause difficulties as the students progress to more advanced digital logic topics.

3.1 Unsigned Binary Subtraction Example

To illustrate the challenges with binary subtraction, consider an example of subtracting 7 (111 in binary) from 40 (101000 in binary):

```
101000 (40)
-000111 (7)
-----
100001 (33)
```

In this example, the rightmost column requires a borrow because subtracting 1 from 0 is not possible without borrowing. Unlike decimal subtraction—where borrowing typically involves taking from a position valued at ten times the adjacent place (for example, borrowing from tens to ones)—binary positions are based on powers of 2. Thus, borrowing in binary entails taking from the next available “2” value and adjusting each position accordingly.

We observe that in the rightmost column, subtracting 1 from 0 requires a borrow; however, the adjacent column also contains a 0. Consequently, a 1 is borrowed from the 2^3 position, representing the 8 as $1 \times 2^2 + 1 \times 2^1 + 2 \times 2^0$ (or “112”). The problem then becomes:



$$\begin{array}{r}
 100112 \text{ (40)} \\
 -000111 \text{ (7)} \\
 \hline
 100001 \text{ (33)}
 \end{array}$$

(Here, the red digits indicate the borrow from the 8's column and the corresponding borrow representation of 8.)

An additional challenge occurs with multiple borrows. Consider subtracting 23 (10111 in binary) from 40 (101000 in binary):

$$\begin{array}{r}
 101000 \text{ (40)} \\
 -010111 \text{ (23)} \\
 \hline
 \end{array}$$

which requires 2 separate borrows.

$$\begin{array}{r}
 020112 \text{ (40)} \\
 -010111 \text{ (23)} \\
 \hline
 010001 \text{ (17)}
 \end{array}$$

3.2 Two's Complement Subtraction Example

In computer systems, two's complement notation is used to represent both positive and negative numbers, enabling efficient arithmetic operations within a fixed number of bits. In two's complement subtraction, the borrowing process can extend beyond the visible bit positions, allowing the representation of negative results even within a fixed bit-length system.

For example, consider subtracting 1 from 0 using a 6-bit two's complement representation. While this calculation would be invalid in unsigned binary (since it would produce a negative result), it is entirely valid in two's complement, where borrowing from an imaginary higher bit position allows the result to represent -1 correctly. The subtraction is illustrated as follows:

$$\begin{array}{r}
 000000 \text{ (0)} \\
 -000001 \text{ (1)} \\
 \hline
 111111 \text{ (-1)}
 \end{array}$$

This problem requires a borrow from an imaginary 7th bit position for numbers represented with 6 bits. In such cases, the effective calculation can be conceptually represented as:

$$\begin{array}{r}
 111112 \text{ (0)} \\
 -000001 \text{ (1)} \\
 \hline
 111111 \text{ (-1)}
 \end{array}$$

An important question to consider is whether generative AI work as a teacher.

To explore this, the following prompt was entered into ChatGPT-4o (the latest version at the time of writing):

Perform $10 - 3$ using 4 bit binary numbers (1010 – 0011) and explain all of the steps. Here is the AI response.



Alright! Let's break it down step-by-step in a simple way.
Imagine you have two numbers, 1010 and 0011, and you want to subtract the second number from the first.

Line up the numbers: Place the numbers one below the other so that each digit is in the correct column.

```
1010
- 0011
```

Start from the rightmost column: Just like in regular subtraction, you start with the rightmost digit (column).

Subtract each column: Let's do it step by step.

Rightmost column: $0 - 1$. You can't subtract 1 from 0, so you need to borrow from the next column.

Borrowing: Change the second 1 (second from the right) to 0 and change the rightmost 0 to 10 (in binary, this is like changing a 0 to 2 in decimal subtraction). Now subtract: 10 (which is 2 in binary) - 1 = 1. Write down 1.

```
  1
1001
- 0011
-----
```

Next column: Now, after borrowing, you have $0 - 1$. You can't do that again, so you need to borrow from the next column.

Borrowing: Change the next column's 1 to 0 and the 0 (third from the right) to 10 (which is 2 in binary).

Now subtract: $10 - 1 = 1$. Write down 1.

```
  1
1000
- 0011
-----
```

```
  011
```

Next column: Now subtract $0 - 0 = 0$. Write down 0.

```
1000
- 0011
-----
```

```
  011
```

Last column: Now subtract $1 - 0 = 1$. Write down 1.

```
1000
- 0011
-----
```

The generative AI response is confusing, and its reported result—interpreting $1010 - 0011$ as 5—is incorrect (the correct result should correspond to 7 in decimal). We need to develop tools to help the students solve problems like these.

3.3 Using Chatgpt to Write Learning Aids

To develop this teaching tool, an initial prompt was crafted to leverage ChatGPT's ability to explain the subtraction process step-by-step. This prompt enabled the system to demonstrate the subtraction process in a clear, sequential manner.

Write a python program for Google Colab that will ask the user to input two binary numbers. Display the inputs as a typical subtraction problem including displaying the final result

The prompt used was:

After copying the generated program into an empty Google Colab notebook and running it, the initial output from running the program on the operation $40 - 7$ is shown below.

```
Enter the first binary number: 101000
Enter the second binary number: 000111
  101000
- 000111
-----
  100001
```

The answer is correct, but we wanted it to display the decimal values of the input numbers. The following prompt was given.

Modify the program to display the decimal values of the inputs in parenthesis

The modified output, now displaying decimal equivalents, is shown below.

```
Enter the first binary number: 101000
Enter the second binary number: 000111
  101000 (decimal 40)
- 000111 (decimal 7)
-----
  100001 (decimal 33)
```

To remove the text “decimal” in the result and specify the desired output formatting, the following prompt was provided:

eliminate the word decimal in the output. make a second copy of the problem. Analyze this problem column by column starting at the right. If there is a 0 - 1 then look to columns to the left for a 1 to borrow from. replace that 1 with a 0. Replace the column with the initial 0 - 1 with a 2 and all other columns used for the borrow with 1's. highlight all changes in red

“After copying and running the program on Colab, a syntax error was encountered. The error message, shown below, was then provided to ChatGPT for troubleshooting.”

```
File "<ipython-input-13-aac8317e9cbd>", line 29
    bin1_list = list(bin1_p padded)
                  ^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
```

To troubleshoot, the unedited error message displayed by Colab can be copied and provided to ChatGPT for correction. By supplying the exact error message, and including the error type and line numbers, ChatGPT can offer precise adjustments to address the syntax issues and ensure smooth execution of the program. ChatGPT identified the error and offered a new version of the program that fixed the error.

After several iterations, a program that performed addition and subtraction on binary and two's complement numbers was developed. The user interface of the program is shown below.

Binary M:

Binary Q:

2's Comple...

☐ Yes, 2's Complement
 ☒ No, Binary

The program output is shown next.

Addition Problem:

```

      (carries)
    101000 (40)
+   111 (7)
-----
    101111 (47)
  
```

Subtraction Problem:

```

    101000 (40)
-   000111 (7)
-----
  
```

Problem with borrows

```

  100112 (40)
-   000111 (7)
-----
  100001 (33)
  
```

Restart

The code for this program can be found and run with the following link

https://colab.research.google.com/github/bbutka/CEC220/blob/main/Binary_arithmetic.ipynb#scrollTo=QOtNO6vCyUYx

3.4 Generative Ai for Logic Expressions

A second area where generative AI often encounters challenges is in Boolean algebra and digital logic simplification. As an example of where generative AI struggles with digital logic, ChatGPT 4o was given the following prompt.

Simplify this Boolean logic expression $AB + \bar{A}C + AC$

ChatGPT's response was

To simplify the Boolean logic expression $AB + \bar{A}C + AC$:

1. Apply the **Distributive Law** to group terms involving A :

$$AB + AC + \bar{A}C = A(B + C) + \bar{A}C$$

2. Next, observe that $B + C$ is always true when either B or C is true, so the expression simplifies further to:

$$A + \bar{A}C$$

3. Now, apply the **Distributive Law** again:

$$A + \bar{A}C = (A + \bar{A})(A + C)$$

4. Since $A + \bar{A} = 1$ (due to the **Complement Law**), the expression simplifies to:

$$A + C$$

Thus, the simplified Boolean expression is $A + C$.

ChatGPT indicates that this expression simplifies to $A+C$. However, the correct simplification is $AB+C$. An examination of the AI's process reveals an error in the second step, where it incorrectly states "Next, observe that $B + C$ is always true when either B or C is true." This misinterpretation leads to a flawed analysis and an incorrect result.

It is clear that developing teaching aids for logic expressions is also a useful area. ChatGPT was used to develop a program to simplify logic expressions and produce a Karnaugh Map for the solution. This proved to be a much more difficult task. The table below documents the time spent developing this program.

Task	Time Required
Input logic equations and produce a Karnaugh map for equations of 3 variables.	1.5 Hrs
Highlight the groups of 1's used for simplification.	1 Hr
Assign colors for overlapping groups.	0.5 Hrs
Allow for truth table or logic equation inputs.	1 Hr
Allow don't care terms in the truth table.	1 Hr
Extend operation to allow 3 or 4 variable problems.	3 Hrs
Total time required	7 Hrs

4. Using the Developed Learning Tools in Practice

The programs were made accessible to students through a Canvas page labeled “Helpful Programs.” Some of these programs were demonstrated in class to solve example problems and were designed primarily to assist the bottom 20% of the class who struggle with these concepts.

Usage statistics indicate that the programs were rarely utilized by the top 80% of the class; these students acknowledged their usefulness but felt they were unnecessary for their studies. In contrast, approximately 44% of students in the bottom 20% actively used the programs—especially the Karnaugh map tool—and 88% of these students attended in-person tutorials and exam review sessions.

Feedback revealed that students who consistently used the learning aids tended to perform better. However, lower-performing students also noted that simply viewing the solution steps was insufficient to address deeper comprehension issues, suggesting a need for future development of more interactive or guided explanations.

Summary

This research adopted an innovative approach by utilizing generative AI not for conventional content generation—such as lecture slides or tests—but for developing interactive programs that serve as dynamic teaching aids. We detailed an iterative development process that encompassed prompt engineering and program deployment, and we evaluated the impact of these tools using both performance metrics and student feedback. The study not only documented the substantial time investment required for program refinement but also highlighted that AI-driven solutions enhance student comprehension by addressing specific educational challenges.

By demonstrating that well-crafted AI prompts and iterative refinement can produce effective, subject-specific learning aids, our work illustrates the broader potential of generative AI in personalized education. These findings lay a foundation for future research on integrating AI tools into diverse educational environments, suggesting that such technologies can be tailored to support various learning levels and instructional needs, ultimately contributing to improved educational outcomes.

REFERENCES

- [1] J. Savelka, A. Agarwal, C. Bogart, Y. Song, and M. Sakr, “Can Generative Pre-trained Transformers (GPT) Pass Assessments in Higher Education Programming Courses?” in *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education*, V. 1, ser. ITICSE 2023, New York, NY, USA: Association for Computing Machinery, Jun. 2023, pp. 117–123.
- [2] B. Puryear and G. Sprint, “Github copilot in the classroom: Learning to code with AI assistance,” *J. Comput. Sci. Coll.*, vol. 38, no. 1, pp. 37–47, Nov. 2022.
- [3] V.-A. Pădurean, G. Tzannetos, and A. Singla, “Neural Task Synthesis for Visual Programming,” 2023.
- [4] R. W. Brennan and J. Lesage, “Exploring the Implications of OpenAI Codex on Education for Industry 4.0,” in *Service Oriented, Holonic and Multi-Agent Manufacturing Systems for Industry of the Future*, T. Borangiu, D. Trentesaux, and P. Leitão, Eds., Cham: Springer International Publishing, 2023, pp. 254–266.
- [5] A. Moradi Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, and Z. M. J. Jiang, “GitHub Copilot AI pair programmer: Asset or Liability?” *Journal of Systems and Software*, vol. 203, p. 111734, Sep. 2023.
- [6] R. A. Poldrack, T. Lu, and G. Beguš, “AI-assisted coding: Experiments with GPT-4,” Apr. 2023.

- [7] J. Finnie-Ansley, P. Denny, A. Luxton-Reilly, E. A. Santos, J. Prather, and B. A. Becker, "My AI Wants to Know if This Will Be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises," in *Proceedings of the 25th Australasian Computing Education Conference*, ser. ACE '23, New York, NY, USA: Association for Computing Machinery, Jan. 2023, pp. 97–104.
- [8] A. Tripaldelli, B. Butka, and C. Elder, "The Development and Evaluation of Artificial Intelligence (A.I.) Tutor for a Java Programming Class," ISSN: 2758-0962 *The Paris Conference on Education 2023: Official Conference Proceedings*, pp. 363–370, Feb. 2023.
- [9] P. Denny, V. Kumar, and N. Giacaman, "Conversing with Copilot: Exploring Prompt Engineering for Solving CS1 Problems Using Natural Language," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, V. 1, ser. SIGCSE 2023, New York, NY, USA: Association for Computing Machinery, Mar. 2023, pp. 1136–1142.
- [10] J. Finnie-Ansley, P. Denny, B. A. Becker, A. Luxton-Reilly, and J. Prather, "The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming," in *Proceedings of the 24th Australasian Computing Education Conference*, ser. ACE '22, New York, NY, USA: Association for Computing Machinery, Feb. 2022, pp. 10–19.
- [11] Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. Dal Lago, T. Hubert, P. Choy, C. de Masson d'Autume, I. Babuschkin, X. Chen, P.-S. Huang, J. Welbl, S. Goyal, A. Cherepanov, J. Molloy, D. J. Mankowitz, E. Sutherland Robson, P. Kohli, N. de Freitas, K. Kavukcuoglu, and O. Vinyals, "Competition-level code generation with AlphaCode," *Science*, vol. 378, no. 6624, pp. 1092–1097, Dec. 2022.
- [12] J. Savelka, A. Agarwal, M. An, C. Bogart, and M. Sakr, "Thrilled by Your Progress! Large Language Models (GPT-4) No Longer Struggle to Pass Assessments in Higher Education Programming Courses," in *Proceedings of the 2023 ACM Conference on International Computing Education Research – Volume 1*, ser. ICER '23, vol. 1, New York, NY, USA: Association for Computing Machinery, Sep. 2023, pp. 78–92.
- [13] C. Elder, G. Pozek, S. Horine, A. Tripaldelli, and B. Butka, "Can Artificial Intelligence Pass a Sophomore Level Digital Design Laboratory?" in *SoutheastCon 2023*, IEEE, 2023, pp. 861–868.
- [14] P. Sridhar, A. Doyle, A. Agarwal, C. Bogart, J. Savelka, and M. Sakr, "Harnessing LLMs in Curricular Design: Using GPT-4 to Support Authoring of Learning Objectives," Jun. 2023.
- [15] J. K. Matelsky, F. Parodi, T. Liu, R. D. Lange, and K. P. Kording, "A large language model-assisted education tool to provide feedback on open-ended responses," 2023.
- [16] S. Sarsa, P. Denny, A. Hellas, and J. Leinonen, "Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models," in *Proceedings of the 2022 ACM Conference on International Computing Education Research – Volume 1*, ser. ICER '22, vol. 1, New York, NY, USA: Association for Computing Machinery, Aug. 2022, pp. 27–43.
- [17] P. Denny, S. Sarsa, A. Hellas, and J. Leinonen, "Robosourcing Educational Resources – Leveraging Large Language Models for Learner-sourcing," Nov. 2022.
- [18] P. Denny, H. Khosravi, A. Hellas, J. Leinonen, and S. Sarsa, "Can We Trust AI-Generated Educational Content? Comparative Analysis of Human and AI-Generated Learning Resources," Jul. 2023.
- [19] J. Zhang, J. Cambronero, S. Gulwani, V. Le, R. Piskac, G. Soares, and G. Verbruggen, "Repairing Bugs in Python Assignments Using Large Language Models," Sep. 2022.
- [20] T. Phung, J. Cambronero, S. Gulwani, T. Kohn, R. Majumdar, A. Singla, and G. Soares, "Generating High-Precision Feedback for Programming Syntax Errors using Large Language Models," in *International Conference on Educational Data Mining*, Jul. 2023, pp. 370–377.
- [21] S. Horine, G. Pozek, and B. Butka, "Can Artificial Intelligence Be Used as a Tutor to Improve Student Performance in a Technical Writing Class?" ISSN: 2758-0962 *The Paris Conference on Education 2023: Official Conference Proceedings*, pp. 305–321, Jan. 2023.
- [22] S. Amani, L. White, T. Balart, L. Arora, D. K. J. Shryock, D. K. Brumelow, and D. K. L. Watson, "Generative AI Perceptions: A Survey to Measure the Perceptions of Faculty, Staff, and Students on Generative AI Tools in Academia," Apr. 2023.