Draft of

Proceedings of the 40th Annual Conference of The Pennsylvania Association of Computer Science and Information Science Educators (PACISE 2025)

April 4th – 5th 2025

Hosted by:

Faculty Articles

An Audiovisual Demo Platform as an Interactive Kiosk for Engagement in Computing

David G. Cooper
Department of Computer Science
West Chester University
West Chester, PA, USA
dcooper@wcupa.edu

ABSTRACT

Interactive Kiosks have been used in museums and rural settings as a way to introduce people to new areas of study, to allow people to learn in a self-paced way, and to connect with people who are interested in the topic that the kiosk is presenting. This paper explores an exploration into modifying an existing demo of an audiovisual recording and labeling platform to make a kiosk-like system for passers-by in a university setting to label live video clips using a mobile application downloaded from an app store. This work describes the transformation of the recording system into an interactive kiosk, the way that people engage with the kiosk, and lessons learned when designing a system that will be entirely unattended once deployed.

ACM Reference Format:
David G. Cooper. 2025. An Audiovisual Demo Platform as an Interactive Kiosk for Engagement in Computing . In *Proceedings of 40th Annual Spring conference on of the Pennsylvania Computer and Information Science Educators (PACISE '25).* ACM, New York, NY, USA, 4 pages.

1 INTRODUCTION

The Computer Science Department of a University in the Northeastern Part of the United States has a display case in the hallway where a Single Board Computer such as a Raspberry Pi is connected to a display and information related to the department is shown. This display sometimes has results of competitions or videos of student research in order to showcase what is happening in the department. A number of faculty suggested that a more interactive display be used so that people passing by the display will stay longer and engage with the items in the display case. In addition it's an opportunity for people to get a glimpse of computing through interaction rather than just passive viewing of what is in the display case.

With the goal of an interactive system in mind, the process of using a demo of an audiovisual recording and labeling system as an interactive display was initiated. The process stared by attempting to run the demo as is, and see what kind of engagement ensued. In the first couple of days of the system running the demo, the system had to be restarted many times, so it was difficult to identify how much, if any engagement was happening with the system. This led to a number of stability tests and enhancements. Then the system was tested again, and additional enhancements were made to give more transparency of the information saved by labeling the video. The rest of this paper will discuss related work in interactive kiosks for informal computational learning, and the journey from a demo system meant to work for a couple of hours to a kiosk system capable of working indefinitely with little or no maintenance or interaction needed from the people deploying the system.

PACISE '25, April 4-5, 2025, West Chester, PA

2 RELATED WORK

The related work to our proposed method discusses work that uses Raspberry Pi devices or other single board computers as different types of kiosks in museums, rural settings, and schools.

Some of the earliest work of using a Raspberry Pi as a kiosk describes a system that will automatically play and loop a slideshow provided either as images or a slideshow file in a specified location on the Raspberry Pi's SSD or network mounted drive [5]. More recent work has shown how a Raspberry Pi 4 along with an Arduino and some additional sensors including a webcam, can be used to provide an interactive touchscreen experience similar to more expensive interactive information kiosks [6]. [5] has a high likelihood of being stable, since there is no interaction, and the startup of the system is automatic, but they do not address questions of how the system handles unexpected reboots due to power failure or other software reasons. [6] does not discuss how long the system works without needing to be restarted, nor do they discuss how they achieve a "plug-and-play" system. In addition, though Raspberry Pi computers have been around for a while, and the most recent paper was published over four years prior to writing, there is not a prevalence of Raspberry Pi based kiosk systems. This suggests, that the ease with which to convert a Raspberry Pi or similar single board computer to a plug and play kiosk may not be as straightforward as suggested by these works.

When the internet was becoming popular, some experiments were done introducing ruggedized computers in rural areas so that children could learn about computers and the internet in a hands on way without any instruction [1–4]. [7] describes the BingBee kiosk designed to expose children to ideas around computational thinking, and it allows them to explore these ideas interactively through a ruggedized input device called the ticklepad. Though most college students have access to computers, there are many things not related to computation available to them while using computers. This motivates the interest of making a demo available in a kiosk style environment to allow students and other people passing by to learn about it on their own terms.

3 SYSTEM OVERVIEW

The hardware of the demo used consists of a server that includes an Orange Pi 5 with 16GB of RAM, an edimax Bluetooth adapter, a Logitech 720p web camera, and client iOS and Android devices with the labeling app. The server, shown in Figure 1 has a specially written java program that connects to the camera and Bluetooth and logs QR codes received on the camera and Bluetooth messages that come from the client in order to enable plots of the messages received from the clients as well as create video clips based on the client labels.

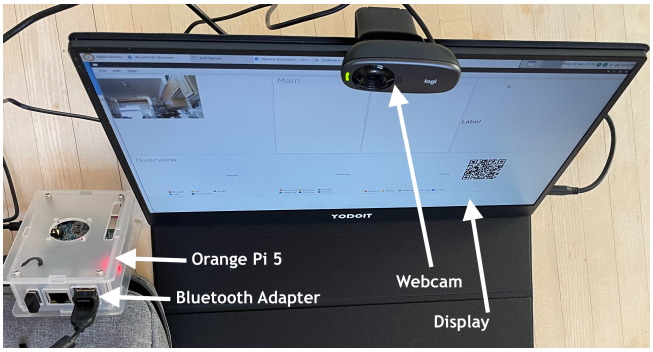


Figure 1: The Orange Pi 5 with an Edimax Bluetooth adapter and a Logitech web camera is displaying the kiosk server software.

The demo system streamlines the labeling of user activities through a series of coordinated processes. Once the server is running. A person passing by will scan the QR code to find out how to download the labeling software, then they download the software. When they run the labeler app, they first fill out the information on the calibration page as shown in Figure 2 and become a participant. Once they click 'calibrate', the app will start displaying a QR code that changes every second, Figure 3. The participant will then show the QR code to the camera, wait for a blue box to show up around the QR code in the live video part of the display (Figure 4), and then holds the QR code steady for at least one second to get the best calibration time. The Participant id then shows up in listing of the Main panel as shown in Figure 5 Then a participant can label as many Emotion, Class Action, or Move Action activity labels. Each time the participant submits a label, a ten second clip is added to the video clips display on the upper right of the display, as shown in Figure 6.

4 STEP 1: USING THE DEMO SOFTWARE AS-IS

The first attempt to use the audiovisual demo software was to copy it to the Raspberry Pi 4 with 4 GB of ram that the department already had running in the display case. This involved installing the required software onto the Raspberry Pi 4 and adding a camera. It turned out that even though the Raspberry Pi 4 has a working Bluetooth LE capability, it failed at least half of the time the program tried to use it. So, the edimax bluetooth adapter was also added to the Raspberry Pi 4 setup. The demo needed a java program to be run from the command line, and a port needed to be selected for

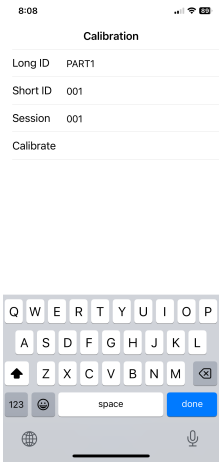


Figure 2: An alphanumeric Participant ID, a Hexadecimal Short ID, and a Hexadecimal session ID are specified before the calibrate button is pushed.

Figure 3: After calibration is pressed, the QR code begins to display and updates every second with the current time of the labeler.

the video input. In addition, since there was nothing in the demo system that described how to use it, 2 additional windows were presented on the screen in the display case. The first was a web page with a qr code for downloading the app, and the second was instructions on how to send labels to the system. This is shown in Figure 8. There were a number of difficulties with this setup. The first is that when there was a power outage, the system didn't automatically open again. The second is that the demo would get stuck after a couple of hours for various reasons. 1. If messages came in from different sources in a particular order, occasionally, this would make the display freeze. 2. Occasionally the bluetooth scanner would fail and would need the program to be reset for it to start up again. 3. Sometimes the program would just abort without explanation.

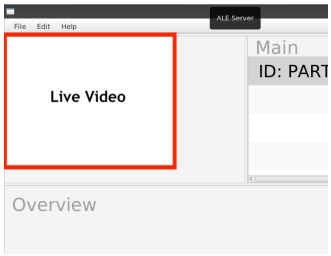


Figure 4: This is the part of the screen where live video is displayed.

Figure 5: After the QR code is detected, the participant id shows up in the Main panel.

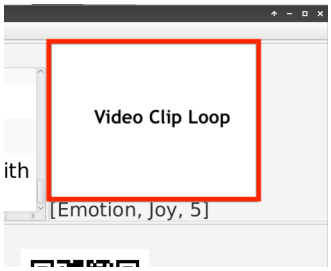


Figure 6: This is the part of the screen where video clips loop and the corresponding label is displayed below each looped video clip.

5 STEP 2: MAKING THE DEMO SOFTWARE MORE ROBUST

After having the demo "running" for about a week with a need to restart the demo more than once each day, an effort was made to try to break the system in a lab setting so that the logs and other details could be observed as the messages were coming in. There are a number of things that were found that were causing issues. 1. Sometimes when a Bluetooth message was received, the wrong thread was used when updating the GUI causing a crash or an unresponsive GUI. 2. After a long time sometimes the thread pool would run out of threads.

The demo has 5 parts that are directly connected with receiving messages. When the message is received, two tables and one chart is updated. Since Bluetooth LE Broadcast messages are handled on their own thread, when these changes happen, they notify the GUI thread to update. Some of these notifications must happen from the GUI thread. Specifically, in Java FX, the modification of GUI objects that are visible must happen on the GUI thread otherwise there can be a concurrent modification exception. Using the GUI thread is done by putting the code that changes the GUI objects into a

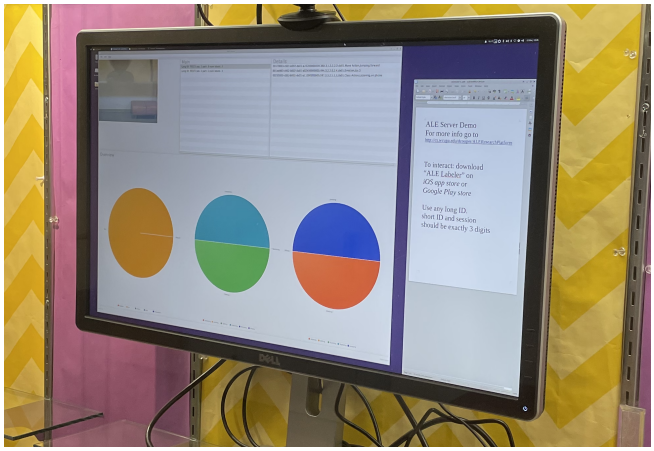


Figure 7: On the left is the demo. On the right is the text file of instructions for people to use the demo.

Runnable and passing the Runnable into the Platform.runLater() method. The particular offending call in the case of our software was a scrollTo() method that sometimes had no effect, sometimes was called when the table was not displayed yet, and sometimes had an effect and the table was visible, so the GUI thread would crash. When this happened, an exception and stack trace would be in the output log of the program, so it was easier to track down and fix the bug.

The second bug of the scanner randomly failing was much harder to track down. On the Raspberry Pi Platform the error message was just an abort with no reason, and it typically happened overnight so it was unclear. When running on the Orange Pi 5, the error didn't appear to happen overnight, so we kept the system running until it would stop. There were a number of power outages that would cause the whole system to restart, so it took many days of waiting to find out the reason for the failure. Once the failure did happen, the messages that were shown didn't throw an exception, it just failed quietly. After finding which exception was most likely the culprit and adding a stack trace, it appears that the cause of the problem is running out of threads, possibly caused by running out of memory. Since this was happening in the Bluetooth library code, there are only two potential solutions. The first is to restart the program regularly and before the error occurs. The second is to try to update the Bluetooth library code and hope that the error has been fixed in the latest version. Since the update would require a lot of additional testing, the program restart solution was chosen in the interest of getting a working kiosk in a shorter time.

6 STEP 3: ADDING MORE ENGAGING FEATURES

After having worked out what appear to be the last outstanding problems to having a working kiosk that can run unattended, a few features needed to be added for the observers to see what was being demonstrated. When a person is demonstrating the system, the demonstrator can explain how the viewer can start labeling, what is happening as the message comes in, and guide the viewer's

gaze to look at the parts that are relevant to the most recent message received. In addition, the demonstrator can describe how the message received can be converted to a saved video clip.

However, without a demonstrator, it's possible and even likely that a person passing by will look at the screen and have no idea that they can download an app and interact with the system. There are a number of additional features to add to the demo to make it more likely for interaction. The first addition is a qr code that brings the user to a web page that describes how to download the app and interact with the system. The second addition, is to add a clip playback functionality that plays the most recent clip or clips that were acquired through the labeling interaction with the system. The final version is shown in Figure **??**

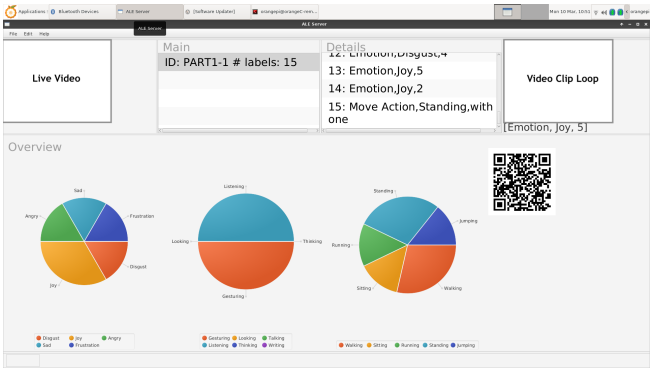


Figure 8: The final display for the kiosk. The upper left is the live video. The upper right is the video clip loop. In between are the participant labels and the detail labels for each participant. The detail labels change based on who the last participant that a label was received from. The lower part of the display has 3 pie charts. One for each type of label, Emotion, Class Action, or Move Action. And finally there is a QR code to access a description of how to use the kiosk.

Adding the QR code is straightforward with javafx, because an image element can be added as well as a link to the image file that has the QR code. If there is potential for the QR code to change, then the image element can be given an id so that it can be changed programmatically in java. The QR code then links to a web page that has directions on how to download the app, and how to use the app to label the video. There is also a description on what to enter into the calibration page to start the interaction.

The clip playback functionality requires an additional video pane in the GUI, a fixed size video buffer that is big enough to grab the clips based on the labels sent over bluetooth, and then a list of video buffers for playback into the video pane. As messages come into the system, the time of the message is matched with the video frame, and a clip is made that has the ten seconds preceding the time that the label was sent from the labeler app. This clip is then added to the list of video buffers meant for the playback video pane. The list is looped through, playing each clip in turn until the demo program is stopped.

7 FUTURE SCOPE: MAKING A KIOSK MODE

There are still a couple of steps that are yet to be implemented so that the server can work in kiosk mode. The first is that the script to run the server needs to be called, and the second is that the program needs to restart every night. An additional feature that could be nice is to add a feature that will save the video clips as files so that when the kiosk is restarted it can reload the video clips as if it had never stopped.

8 CONCLUSION

Though single board computers are an inexpensive option for creating interactive kiosks, this case study in converting a demo into an interactive kiosk describes some of the difficulties that can come up, and the process that was taken to surmount those difficulties.

9 ACKNOWLEDGEMENTS

This work is the research product of undergraduate and graduate students at Cheyney University, Drexel University, and West Chester University. I'd like to thank Salamata Bah, Ines Constant, Medeline Cooke, Isaiah DeSantis, Ramon Estevez, Olivia Harley, Eshwar Sai Ram Mamillapalli, Justin McGriff, Tashawn Patton-Taylor, Srijan Pandey, Darril Vilbrun, Erin Ward, Jake Winemiller, and Daria Washington. A portion of this work was supported by the National Science Foundation under Grant No. HRD-1912011.

REFERENCES

[1] Savita Bailur, Renee Kuriyan, Joyojeet Pal, Aishwarya Ratan, Janaki Srinivasan, Kentaro Toyama, Rajesh Veeraraghavan, Akshaya Namma Dhwani, Azim Premji, and One Roof. 2007. Review of research on rural PC Kiosks. *Microsoft Research India* (2007).

[2] Kim Gush and Ruth de Villiers. 2010. Application usage of unsupervised digital doorway computer kiosks in remote locations in South Africas. In *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*. 93–103.

[3] Parimala Inamdar. 2004. Computer skills development by children using 'hole in the wall' facilities in rural India. *Australasian Journal of Educational Technology* 20, 3 (2004).

[4] Parimala Inamdar and Arun Kulkarni. 2007. 'Hole-In-The-Wall' computer kiosks foster mathematics achievement- A comparative study. *Journal of Educational Technology & Society* 10, 2 (2007), 170–179.

[5] Raleigh Clayton Muns. 2013. Portable Display Kiosk and Signage Using the Raspberry Pi. In *Brick & Click Libraries: An Academic Library Symposium*, Vol. 13. ERIC, 68–71.

[6] Asm Mehedi Hasan Sad, Md Mashrur Sakib Choyon, Abu Hasnat Md Rhydwan, and Chowdhury Akram Hossain. 2020. An interactive low-cost smart assistant system: Information kiosk as plug & play device. In *2020 27th Conference of Open Innovations Association (FRUCT)*. IEEE, 193–199.

[7] Peter Wentworth. 2010. BingBee@ RaglanRoad-A field trial with unattended educational kiosks. In *2010 IST-africa*. IEEE, 1–8.

MAKING QUANTUM COMPUTING ACCESSIBLE: A PATH FOR CS MAJORS WITH LIMITED FOUNDATIONS

Jingnan Xie
Millersville University of Pennsylvania, Computer Sciences Department
jingnan.xie@millersville.edu

ABSTRACT

As quantum computing continues to advance, its transformative potential in fields such as cryptography, material science, and machine learning is undeniable. However, incorporating quantum computing into computer science curricula remains a significant challenge due to the steep learning curve posed by quantum mechanics and the heavy reliance on physics concepts. To address this, we propose an approach to quantum computing education tailored for computer science students, aimed at making complex concepts accessible through a focus on linear algebra. In this paper, we summarize nine key linear algebra concepts essential for understanding quantum computing and express the four postulates of quantum mechanics from a linear algebra perspective. Additionally, we provide practical materials and hands-on resources that educators can easily adapt for their own courses, fostering broader adoption of quantum education. By lowering barriers to entry, this work empowers both students and educators to engage with quantum computing, helping to prepare a capable workforce for this transformative technology.

1 Introduction

Quantum computing leverages the principles of quantum mechanics, such as superposition and entanglement, to process information in fundamentally different ways compared to classical computing (see [1] for a detailed introduction). In recent years, the field has advanced rapidly, with significant progress in both theoretical research and practical applications. In December 2024, Google's parent company, Alphabet, introduced the Willow quantum computing chip, which claims to solve a problem in five minutes, a task that would take classical computers an impractically long time to complete. Similarly, in November 2024, IBM unveiled its most advanced quantum computers, suggesting enhanced computational capabilities and potential breakthroughs in fields such as cryptography, material science, and machine learning. Beyond research laboratories, industries are investing heavily in quantum technologies, with applications in cryptography, material science, and machine learning driving this momentum. As quantum computing moves from theoretical research into practical applications, the need to educate the next generation of computer scientists has become a strategic priority (see [2] and [3]).

Despite its significance, quantum education remains in its infancy (see [3]). While some universities have introduced quantum courses, the subject has yet to be broadly integrated into standard computer science curricula. This lack of widespread adoption is partly due to the steep learning curve associated with quantum mechanics, a discipline traditionally grounded in physics. Concepts such as the spin of electrons, the behavior of particles at quantum scales, and wave-particle duality require a level of familiarity with physics that most computer science undergraduates lack. These challenges are compounded by the use of specialized notations and terminologies that can overwhelm students without prior exposure.

Yet, the urgency to overcome these barriers cannot be overstated. As quantum technologies continue to mature, the demand for a workforce capable of designing, programming, and applying quantum systems will only grow. The democratization of quantum education is essential to meet this need and ensure that a diverse array of students can participate in shaping this transformative field.

By taking a computer science perspective, this approach makes quantum computing concepts more accessible, significantly reducing the need for extensive physics knowledge. This paper not only highlights key foundational concepts, but also provides hands-on materials and resources that educators can easily adapt to create their own quantum computing courses. With this framework, we aim to lower barriers and empower educators to initiate quantum computing education within the computer science community. This effort represents a crucial step toward making quantum computing both practical and inclusive, ensuring that it reaches a broader audience of students and educators.

2 Related Work

Quantum computing education is an evolving field that is gaining considerable attention in academic settings, particularly in computer science programs. The recent literature emphasizes innovative teaching approaches and curricula designed to make quantum concepts more accessible to students.

Several studies have focused on the integration of quantum

computing into existing computer science programs. For instance, the development of modular and scaffolded learning frameworks helps students progressively understand complex topics, reducing the intimidation often associated with quantum mechanics. These frameworks facilitate the incremental acquisition of knowledge, allowing students to seamlessly connect classical computer science principles with quantum theories [4], [5].

Moreover, hands-on experience with real quantum computing platforms has become an essential component of effective instruction. Recent education initiatives encourage students to use cloud-based quantum computing platforms like IBM Q and Microsoft Quantum Development Kit, where they can run experiments and manipulate quantum circuits. This experiential learning not only reinforces theoretical concepts, but also prepares students to tackle real-world challenges posed by quantum technologies [6].

Game-based learning and simulations have been shown to improve participation among students learning about quantum computing. Studies illustrate that integrating playful elements into the curriculum, such as serious games designed to teach quantum principles, can significantly improve the understanding and retention of students of complex concepts while fostering enthusiasm for the subject matter [7], [8].

Furthermore, research emphasizes the importance of interdisciplinary approaches in quantum computing education. By engaging students from various academic backgrounds, including physics, mathematics, and computer science, educators can create a richer learning environment that promotes collaborative techniques to understand quantum computing [9].

Lastly, there is a growing focus on equity and inclusivity within quantum computing education. Recent articles highlight initiatives aimed at attracting underrepresented groups into STEM fields, specifically quantum technology. Tailored outreach programs and mentorship opportunities are essential to build a diverse pipeline of future quantum computing professionals and to ensure that the field benefits from a wide range of perspectives [10].

The findings from these studies collectively illustrate that by employing diverse pedagogical strategies, promoting practical experiences, and emphasizing inclusivity, educators can effectively prepare students for the future of quantum technologies.

3 Pedagogical Approach and Essential Concepts

Traditional quantum computing courses often introduce fundamental concepts like superposition and entanglement through physical examples, such as photon polarization or electron spin. These approaches, rooted in physics, require students to grasp additional concepts, such as electromagnetic waves and quantum measurements, which can be challenging for computer science students without a solid background in physics (for example, see [11]).

In contrast, this paper takes a linear algebra-based approach, similar to [4], drawing on concepts familiar to most computer science majors. By presenting quantum computing as a generalization of classical probabilistic computing and abstracting physical phenomena through linear algebra, the course minimizes the need for specialized physics knowledge. As described in [4], concepts like superposition and entanglement are framed as properties of unit vectors in Hilbert spaces, and quantum gates are introduced as simple operations on these vectors, avoiding the need for complex number manipulation.

One key distinguishing feature of our approach is its focus on computational theory rather than quantum programming. Given the limited commercial success of quantum programming languages and their experimental nature, the course emphasizes the computational parallels between classical and quantum devices [12], [13] and [14]. This approach not only deepens understanding of quantum complexity classes but also enhances the connection to classical computation theory. By emphasizing these core theoretical concepts, the course helps learners appreciate the foundational principles of classical computation and recognize its computational limits. This perspective allows for a deeper comprehension of both classical and quantum computing, reinforcing their interrelationship. Focusing on theory, rather than experimental programming languages, provides a more practical and meaningful introduction to quantum computing.

In the following, we outline some of the most fundamental topics covered in our approach. These concepts are designed to be accessible, requiring only a minimal mathematical background, especially in the early stages. As learners progress, they gradually build upon this foundation, enhancing both their understanding of quantum computing and their mathematical skills. This approach allows readers to easily adapt these materials for their own educational efforts, even with limited prior experience.

3.1 Linear Algebra

1. Complex Numbers

   Let $\mathbb{R}$ and $\mathbb{C}$ denote the set of real numbers and the set of complex numbers, respectively. A *complex number* $c \in \mathbb{C}$ is written in its standard form as

   $$c = a + bi,$$

   where $a, b \in \mathbb{R}$, and i is the imaginary unit satisfying $i^2 = -1$.

   The *conjugate* of c is denoted by c^* and is given by

   $$c^* = a - bi,$$

   The *magnitude* or length or modulus of $c \in \mathbb{C}$ is

   $$|c| = \sqrt{c \cdot c^*} = \sqrt{a^2 + b^2}$$

2. Complex Vectors

   Complex vectors shall be crucial since they represent quantum states (more details are discussed in section 3.2). Let $|\psi\rangle \in \mathbb{C}^d$ denote a complex (column) *vector*:

$$|\psi\rangle = \begin{pmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_d \end{pmatrix},$$

   where each entry $\psi_i \in \mathbb{C}$ for $i = 1, 2, \ldots, d$.

   The notation $|\cdot\rangle$ is referred to as *Dirac* notation, and is read as "ket". The dual representation of $|\psi\rangle$, denoted as $\langle\psi|$ and read as "bra," is the conjugate transpose (Hermitian conjugate) of $|\psi\rangle$. It is represented as a row vector:

$$\langle\psi| = (\psi_1^*, \psi_2^*, \ldots, \psi_d^*).$$

   Exercise 3.1. Given $|\psi\rangle = \begin{pmatrix} 1 \\ i \end{pmatrix}$, what is $\langle\psi|$?

   Solution: The conjugate of 1 is $1^* = 1$, and the conjugate of i is $i^* = -i$. Therefore, $\langle\psi| = \begin{pmatrix} 1 & -i \end{pmatrix}$.

3. Matrix Operations

   Since we require some essential matrix operations, let us review them briefly. Let A be a matrix, and $A(i, j)$ represent the entry of A in the i-th row and j-th column. The following operations are defined as:

   - *Conjugate of A:* $A^*(i, j) = (A(i, j))^*$.
   - *Transpose of A:* $A^T(i, j) = A(j, i)$.
   - *Adjoint (also called conjugate transpose, Hermitian conjugate, or dagger) of A:* $A^\dagger = (A^*)^T$.

   Example 3.1. Let $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$. Then, the transpose of A is given by:

$$A^T = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}.$$

   Let A and B be two matrices. The *dot product* (multiplication) of A and B is given by

$$A \cdot B(i, j) = \sum_{k=1}^{d} A(i, k) \cdot B(k, j).$$

   Example 3.2. Let

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad B = \begin{pmatrix} 4 & 3 \\ 2 & 1 \end{pmatrix},$$

   then

$$A \cdot B = \begin{pmatrix} 1 \cdot 4 + 2 \cdot 2 & 1 \cdot 3 + 2 \cdot 1 \\ 3 \cdot 4 + 4 \cdot 2 & 3 \cdot 3 + 4 \cdot 1 \end{pmatrix} = \begin{pmatrix} 8 & 5 \\ 20 & 13 \end{pmatrix}.$$

   Exercise 3.2. Let

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \quad B = \begin{pmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{pmatrix}.$$

   What is $A \cdot B$?

4. Inner Product

   Given two vectors $|\psi\rangle$ and $|\phi\rangle$, the *inner product* of them is denoted by $\langle\psi|\phi\rangle$, and is defined as

$$\langle\psi|\phi\rangle = \sum_{i=1}^{d} \psi_i^* \cdot \phi_i.$$

   The inner product measures the "overlap" (or similarity) between the two vectors. If $\langle\psi|\phi\rangle = 0$, then the vectors are orthogonal, and if $\langle\psi|\phi\rangle = 1$, then the vectors are aligned in the same direction.

   Exercise 3.3. Let $|\psi\rangle = \begin{pmatrix} 1 \\ i \end{pmatrix}$ and $|\phi\rangle = \begin{pmatrix} 2 \\ 3i \end{pmatrix}$. Compute their inner product.

   Solution:

$$\langle\psi|\phi\rangle = (1 - i) \cdot \begin{pmatrix} 2 \\ 3i \end{pmatrix} = 1 \cdot 2 + (-i) \cdot (3i) = 2 + 3 = 5.$$

   Note that the inner product returns a scalar, and we have

$$(\langle\psi|\phi\rangle)^* = \langle\phi|\psi\rangle.$$

5. Euclidean Norm

   The *Euclidean norm (2-norm)* of a vector $|\psi\rangle$ is denoted by $\||\psi\rangle\|_2$, and is given by

$$\||\psi\rangle\|_2 = \sqrt{\langle\psi|\psi\rangle} = \sqrt{\sum_{i=1}^{d} \psi_i^* \psi_i} = \sqrt{\sum_{i=1}^{d} |\psi_i|^2}.$$

   Exercise 3.4. Let $|\psi\rangle = \begin{pmatrix} 3 + 4i \\ 1 - i \end{pmatrix}$. Compute its Euclidean norm.

   Solution:

$$\||\psi\rangle\|_2 = \sqrt{(3 + 4i)^*(3 + 4i) + (1 - i)^*(1 - i)} = 3\sqrt{3}.$$

6. Outer Product

   For two vectors $|\psi\rangle, |\phi\rangle \in \mathbb{C}^d$, the *outer product* $|\psi\rangle\langle\phi|$ yields a $d \times d$ matrix.

   Example 3.3. Let $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

$$|0\rangle\langle 0| = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

$$|1\rangle\langle 0| = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$$

   Exercise 3.5. Exercise: Let $|\psi\rangle = \begin{pmatrix} 3 \\ 2 - i \end{pmatrix}$. What is $|\psi\rangle\langle\psi|$? **Solution**:

$$|\psi\rangle\langle\psi| = \begin{pmatrix} 3 \\ 2 - i \end{pmatrix} \begin{pmatrix} 3 & 2 + i \end{pmatrix}$$

$$= \begin{pmatrix} 9 & 6 + 3i \\ 6 - 3i & 4 - (i)^2 \end{pmatrix} = \begin{pmatrix} 9 & 6 + 3i \\ 6 - 3i & 5 \end{pmatrix}$$

7. Linear Operators

A *linear operator* A is a $d\times d$ matrix that maps $\mathbb{C}^d \to \mathbb{C}^d$ with the following linear property:

$$A\left(\sum_i a_i|\psi_i\rangle\right) = \sum_i a_i A|\psi_i\rangle,$$

where $a_i \in \mathbb{C}$ and $|\psi_i\rangle \in \mathbb{C}^d$.

The *matrix element* is a scalar quantity that provides information about how the operator A acts on the state $|\psi\rangle$ and how the resulting state overlaps with the state $|\phi\rangle$.

$$\langle\phi|A|\psi\rangle = \langle\phi|(A|\psi\rangle) = \sum_{i,j} \phi_i^*(A(i,j)\psi_j),$$

where ϕ_i^* is the complex conjugate of the i-th component of $|\phi\rangle$, $A(i,j)$ is the (i,j)-th entry of the operator A, and ψ_j is the j-th component of $|\psi\rangle$.

Example 3.4. Given $|\psi\rangle = \begin{pmatrix}1\\0\end{pmatrix}, |\phi\rangle = \begin{pmatrix}0\\1\end{pmatrix}, A = \begin{pmatrix}2 & 3\\1 & 4\end{pmatrix}$,

$$\langle\phi|A|\psi\rangle = \langle\phi|\begin{pmatrix}2 & 3\\1 & 4\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0 & 1\end{pmatrix}\begin{pmatrix}2\\1\end{pmatrix} = 1.$$

Thus, $\langle\phi|A|\psi\rangle = 1$.

8. Orthonormal Bases

A set of vectors $\{|\psi_i\rangle\} \subseteq \mathbb{C}^d$ is said to be *orthogonal* if for all $i \neq j$, $\langle\psi_i|\psi_j\rangle = 0$. The set is *orthonormal* if

$$\langle\psi_i|\psi_j\rangle = \begin{cases}0, & \text{if } i \neq j,\\ 1, & \text{if } i = j.\end{cases}$$

Hence, each vector satisfies $\||\psi_i\rangle\|_2 = 1$, and every distinct pair of vectors is orthogonal.

For every vector in $\mathbb{C}^d$, it can be expressed as a linear combination of an orthonormal basis. For example, in $\mathbb{C}^2$, the most common basis is $|0\rangle = \begin{pmatrix}1\\0\end{pmatrix}$ and $|1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$. Any vector $|\psi\rangle \in \mathbb{C}^2$ can be written as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where $\alpha, \beta \in \mathbb{C}$. We say that $|\psi\rangle$ is *normalized* if $\||\psi\rangle\|_2 = 1$, which is equivalent to

$$|\alpha|^2 + |\beta|^2 = 1.$$

Exercise 3.6. Why $\||\psi\rangle\|_2 = 1$ is equivalent to $|\alpha|^2 + |\beta|^2 = 1$?

Solution: $\||\psi\rangle\|_2 = \sqrt{\langle\psi|\psi\rangle} = \langle\alpha|0\rangle + \beta|1\rangle, \alpha|0\rangle + \beta|1\rangle = (\alpha^*\langle 0| + \beta^*\langle 1|)(\alpha|0\rangle + \beta|1\rangle) = |\alpha|^2\langle 0|0\rangle + \alpha^*\beta\langle 0|1\rangle + \beta^*\alpha\langle 1|0\rangle + |\beta|^2\langle 1|1\rangle = |\alpha|^2 + |\beta|^2 = 1$

9. Eigenvalues and Eigenvectors

Given a matrix A, an eigenvector $|\psi\rangle$ is a non-zero vector that satisfies the equation

$$A \cdot |\psi\rangle = \lambda|\psi\rangle$$

for some scalar $\lambda \in \mathbb{C}$. We call λ the corresponding eigenvalue of A.

Example 3.5. Show that $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ is an eigenvector of $A = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}$.

$$A|+\rangle = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}\frac{1}{\sqrt{2}}(|1\rangle + |0\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}\begin{pmatrix}1\\1\end{pmatrix}$$
$$= \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} = |+\rangle$$

Thus, $|+\rangle$ is an eigenvector of A with eigenvalue $\lambda = 1$.

Exercise 3.7. Show that $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ is also an eigenvector of $A = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}$ and find the corresponding eigenvalue.

Solution:

$$A|-\rangle = \begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}\begin{pmatrix}1\\-1\end{pmatrix}$$
$$= \frac{1}{\sqrt{2}}\begin{pmatrix}-1\\1\end{pmatrix} = -\frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} = -|-\rangle$$

Thus, $|-\rangle$ is an eigenvector of A with eigenvalue $\lambda = -1$.

We can find the eigenvalues of a matrix A without knowing the eigenvectors. The eigenvalues satisfy the characteristic equation:

$$\det(A - \lambda I) = 0$$

where det denotes the determinant and I is the identity matrix. Let us first review how to compute the determinant with some examples.

Example 3.6.

$$\begin{vmatrix}1 & 2\\2 & 4\end{vmatrix} = 1\cdot 4 - 2\cdot 2 = 0$$

$$\begin{vmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{vmatrix} = 1\cdot5\cdot9+2\cdot6\cdot7+3\cdot4\cdot8-3\cdot5\cdot7-1\cdot6\cdot8-2\cdot4\cdot9$$
$$= 45 + 84 + 96 - 105 - 48 - 72 = 0$$

Example 3.7. Let $A = \begin{pmatrix}0 & -i\\i & 0\end{pmatrix}$. Compute its eigenvalues.

Solution:

$$\det(A - \lambda I) = 0$$
$$A - \lambda I = \begin{pmatrix}-\lambda & -i\\i & -\lambda\end{pmatrix}$$
$$\det(A - \lambda I) = (-\lambda)(-\lambda) - (-i)(i) = \lambda^2 - 1 = 0$$
$$\lambda^2 = 1$$
$$\lambda = \pm 1$$

3.2 Quantum Mechanics

With linear algebra at hand, the four postulates of quantum mechanics can be stated, addressing the following questions: How to represent a single quantum system, how to perform operations on a quantum system, how to describe multiple quantum systems, and how to measure classical information from a quantum system.

Postulate 1: Individual Quantum Systems

Recall that in classical computing, a bit is either 0 or 1. In the quantum world, a quantum bit, or *qubit*, can take on not just 0 or 1, but a state that reflects the possibility of being both 0 and 1 simultaneously. Let us formalize this phenomenon.

First, we encode the bits 0 and 1 via the standard orthonormal basis vectors $|0\rangle$ and $|1\rangle$ in $\mathbb{C}^2$. Then, to denote a qubit in states $|0\rangle$ and $|1\rangle$ simultaneously, we write:

$$|0\rangle + |1\rangle.$$

This is called a *superposition*.

More generally, the contribution of $|0\rangle$ and $|1\rangle$ is controlled by the *amplitudes* $\alpha, \beta \in \mathbb{C}$, i.e.,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle.$$

The only restriction is that $|\psi\rangle$ must be normalized (a unit vector), i.e.,

$$|\alpha|^2 + |\beta|^2 = 1.$$

In summary, any unit vector in $\mathbb{C}^2$ describes the state of a single qubit.

Example 3.8. $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ are two widely mentioned single-qubit states.

Postulate 2: Quantum Operations

Recall that a linear operator acts on the state vector and is represented by a matrix. The matrix must be with dimensions $d \times d$ to operate on a state in $\mathbb{C}^d$. A quantum operation is represented by a linear operator, which must be a *unitary* matrix. A matrix U is *unitary* if it satisfies the condition

$$UU^\dagger = U^\dagger U = I,$$

where $U^\dagger$ is the Hermitian conjugate (or adjoint) of U, and I is the identity matrix. Therefore, $U^\dagger$ is the inverse of U.

Example 3.9. *Pauli-X gate:*

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$
$$X^\dagger = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad XX^\dagger = X^\dagger X = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Pauli-Y gate:

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$
$$Y^\dagger = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad YY^\dagger = Y^\dagger Y = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Pauli-Z gate:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$
$$Z^\dagger = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad ZZ^\dagger = Z^\dagger Z = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Hadamard gate:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$
$$H^\dagger = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad HH^\dagger = H^\dagger H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Exercise 3.8.

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle,$$
$$X|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle.$$

Hence, X is also called the quantum OR gate.

$$Y|0\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ i \end{pmatrix} = i|1\rangle,$$
$$Y|1\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -i \\ 0 \end{pmatrix} = -i|0\rangle.$$

$$Z|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle,$$
$$Z|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix} = -|1\rangle.$$

$$Z|+\rangle = Z\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) = \frac{1}{\sqrt{2}}(Z|0\rangle + Z|1\rangle)$$
$$= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle.$$

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = |+\rangle,$$
$$H|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = |-\rangle.$$

$$H|+\rangle = H\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\right) = \frac{1}{\sqrt{2}}(H|0\rangle + H|1\rangle)$$
$$= \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) + \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) = |0\rangle.$$

$$H|-\rangle = H\left(\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) = \frac{1}{\sqrt{2}}(H|0\rangle - H|1\rangle)$$
$$= \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) - \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right) = |1\rangle.$$

Postulate 3: Composite Quantum Systems

So far, we have only considered single-qubit systems. However, a computer with just a single qubit is not very useful. To perform more complex computations, we need to combine multiple qubits. The tool for this task is the *tensor product*, denoted by $\otimes$. Formally, for two vectors $|\psi\rangle$ and $|\phi\rangle$ in $\mathbb{C}^2$, the tensor product $|\psi\rangle \otimes |\phi\rangle$ is a vector in $\mathbb{C}^4$, with

$$(\psi \otimes \phi)_{ij} = \psi_i \phi_j.$$

This expresses that the (i, j)-entry of the tensor product $|\psi\rangle \otimes |\phi\rangle$ is the product of the i-th component of $|\psi\rangle$ and the j-th component of $|\phi\rangle$.

Example 3.10.

$$|0\rangle \otimes |0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}$$

$$|0\rangle \otimes |1\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}$$

$$|1\rangle \otimes |0\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}$$

$$|1\rangle \otimes |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

Note, that a 2-qubit system can exist in a superposition of 4 classical basis states, and an n-qubit system can exist in a superposition of 2^n classical basis states. Although it does not hold 2^n bits of information, its ability to exist in a superposition of states, combined with *entanglement*, is why a quantum computer might **potentially** outperform classical computers.

Now, let us look at a 2-qubit state that troubled Einstein until the end of his days, one of the *Bell* states:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\0\\0\\1\end{pmatrix}$$

Note that $|00\rangle$ is $|0\rangle \otimes |0\rangle$. This state demonstrates a quantum phenomenon known as *entanglement*. Intuitively, it means that if a pair of qubits q_0 and q_1 are entangled, then they are bound regardless of the distance between them, and one cannot describe the state of q_0 or q_1 alone. This means that there do not exist two states $|\psi_1\rangle$ and $|\psi_2\rangle$ in $\mathbb{C}^2$ such that

$$|\Phi^+\rangle = |\psi_1\rangle \otimes |\psi_2\rangle.$$

For $|\Phi^+\rangle$, when we measure it, the two qubits are either both $|00\rangle$ or both $|11\rangle$ since they are entangled (quantum measurement is discussed later in detail).

The other three Bell states are:

$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle),$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle),$$

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle).$$

For a linear operator to qualify as a quantum gate, it must be a unitary operator. Moreover, for an n-qubit quantum system, the quantum gates are $2^n \times 2^n$ matrices. For example, we have seen that 2-qubit quantum states are described by unit vectors in $\mathbb{C}^4$. Accordingly, we can discuss 2-qubit quantum gates, which are unitary operators (matrices) of dimension 4×4. There are two types of such gates: tensor products of single-qubit gates and genuinely 2-qubit gates.

For a $d_1 \times d_1$ matrix A and a $d_2 \times d_2$ matrix B, the tensor product $A \otimes B$ results in a $d_1 d_2 \times d_1 d_2$ matrix.

Example 3.11. Let $A = \begin{pmatrix}a_1 & a_2\\a_3 & a_4\end{pmatrix}$ and $B = \begin{pmatrix}b_1 & b_2\\b_3 & b_4\end{pmatrix}$.

The tensor product $A \otimes B$ is given by:

$$A \otimes B = \begin{pmatrix}a_1 B & a_2 B\\a_3 B & a_4 B\end{pmatrix} = \begin{pmatrix}a_1b_1 & a_1b_2 & a_2b_1 & a_2b_2\\a_1b_3 & a_1b_4 & a_2b_3 & a_2b_4\\a_3b_1 & a_3b_2 & a_4b_1 & a_4b_2\\a_3b_3 & a_3b_4 & a_4b_3 & a_4b_4\end{pmatrix}.$$

Example 3.12.

$$X \otimes Z = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix},$$

$$H \otimes H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

Exercise 3.9. Compute $X \otimes I$, $Z \otimes H$.

Solution:

$$X \otimes I = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

$$Z \otimes H = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}.$$

Genuinely 2-qubit gates are not tensor products of single-qubit gates. An important example of such a gate is the controlled-NOT (CNOT) gate. The CNOT gate treats the first qubit as the control qubit and the second as the target qubit. It applies the Pauli X gate (NOT gate) to the target qubit only if the control qubit is $|1\rangle$; otherwise, it does nothing. More precisely:

The CNOT gate is represented by the matrix:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & X \end{bmatrix},$$

The action of the CNOT gate on computational basis states is as follows:

$$\text{CNOT}\,|00\rangle = |00\rangle, \quad \text{CNOT}\,|01\rangle = |01\rangle,$$
$$\text{CNOT}\,|10\rangle = |11\rangle, \quad \text{CNOT}\,|11\rangle = |10\rangle.$$

Now, we can do our first interesting computation: we can prepare the Bell state $|\Phi^+\rangle$ starting from an initial state of two qubits $|0\rangle|0\rangle$ (or $|0\rangle \otimes |0\rangle$, denoted as $|00\rangle$).

Example 3.13.

$$\begin{aligned} \text{CNOT}\,(H \otimes I)|00\rangle &= \text{CNOT}\,(H|0\rangle \otimes I|0\rangle) \\ &= \text{CNOT}\,(|+\rangle \otimes |0\rangle) \\ &= \text{CNOT}\left(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle\right) \\ &= \text{CNOT}\,\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) \\ &= \frac{1}{\sqrt{2}}(\text{CNOT}\,|00\rangle + \text{CNOT}\,|10\rangle) \\ &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \end{aligned}$$

Thus, the resulting state is the Bell state:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

Exercise 3.10. Try using the initial states $|01\rangle$, $|10\rangle$, and $|11\rangle$ with the same gates as in Exercise 3.13 to construct the other Bell states.

Postulate 4: Measurement

The measurement of a quantum state involves three classes of linear operators: Hermitian operators, positive semi-definite operators, and orthogonal projection operators. Readers can decide whether to cover these topics in their courses depending on the depth and audience of the course.

Without mentioning these operators, the measurement can be simplified as follows:

For a single-qubit state $\alpha|0\rangle + \beta|1\rangle$, the probability of measuring the outcome $|0\rangle$ is $|\alpha|^2$, and the probability of measuring the outcome $|1\rangle$ is $|\beta|^2$. After the measurement, the state collapses to the measured basis state, either $|0\rangle$ or $|1\rangle$.

For a two-qubit system in the state

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle,$$

the probabilities of measuring the outcomes $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$ are $|\alpha|^2$, $|\beta|^2$, $|\gamma|^2$, and $|\delta|^2$, respectively. Upon measurement, the state collapses to the measured basis state corresponding to the outcome.

In general, for an n-qubit quantum system, the measurement outcomes are determined by the probabilities associated with the amplitudes of each computational basis state. These probabilities always sum to 1, ensuring the state is properly normalized.

4 Challenges and Solutions

With Sections 3.1 and 3.2, we have reduced the mathematics and physics barriers for students to understand quantum computing. Readers can proceed with the rest of their courses,

exploring desired quantum "tricks" or algorithms. In this section, we summarize some of the challenges students may face when first learning quantum computing and offer potential solutions.

Terminologies, Notations, and Meanings

The first challenge students might encounter is that quantum computing, being a multidisciplinary topic, often uses multiple names for the same concept. For example, for a complex number c, $|c|$ is referred to as the length, magnitude, or modulus. Additionally, many distinct concepts may have similar names, such as dot product, inner product, outer product, and tensor product.

In fact, we have observed that some materials on quantum computing—and occasionally ChatGPT—claim that the outer product and tensor product for vectors are the same, which is incorrect. This can be very confusing for students.

To address this challenge, in Section 3, we provide distinct names for the same content during its definition, ensuring clarity. Furthermore, we include examples and exercises to help students become more familiar with these concepts and their differences.

Circuit Diagrams

Understanding the circuit diagrams of quantum operations can be confusing for students, even though they are similar to classical circuit diagrams. Here, we highlight some potentially confusing aspects for students and provide a famous example to clarify these concepts: quantum teleportation (first demonstrated in [15]. In a quantum circuit diagram:

1. A single wire represents a single qubit state.
2. A single wire with no gate can be interpreted as applying the identity matrix I to the single qubit state.
3. Multiple single wires represent the tensor product of individual single-qubit states.
4. A double wire following a measurement indicates that the output of the measurement is a classical bit string.

What is quantum teleportation? Suppose there is a single-qubit system given by:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where the values of α and β are unknown. How can this state be transmitted to a friend?

Quantum teleportation utilizes an entangled Bell state, specifically:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

This state consists of two entangled qubits. Suppose the first qubit is held by you, and the second by your friend. Using the following quantum circuit, the state $|\psi\rangle$ can be 'teleported' from you to your friend.

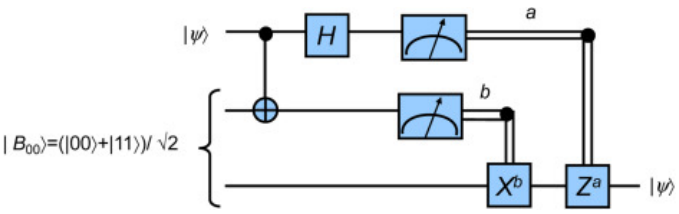


Figure 1: Quantum teleportation circuit diagram.

This means we initially start with the state:

$$|\psi\rangle \otimes |\Phi^+\rangle$$

where $|\psi\rangle$ is the state we want to teleport, and $|\Phi^+\rangle$ is a Bell state. Explicitly, this can be written as:

$$|\psi\rangle \otimes |\Phi^+\rangle = (\alpha|0\rangle + \beta|1\rangle) \otimes \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

$$= \frac{1}{\sqrt{2}}\Big(\alpha|000\rangle + \alpha|011\rangle + \beta|100\rangle + \beta|111\rangle\Big)$$

We first apply the CNOT gate on the first qubits, and get

$$\frac{1}{\sqrt{2}}\Big(\alpha|000\rangle + \alpha|011\rangle + \beta|110\rangle + \beta|101\rangle\Big)$$

Then, the H gate is applied on the first qubit. So after applying the H gate, we have

$$\frac{1}{\sqrt{2}}\Big(\alpha|+\rangle \otimes |00\rangle + \alpha|+\rangle \otimes |11\rangle + \beta|-\rangle \otimes |10\rangle + \beta|-\rangle \otimes |01\rangle\Big)$$

$$= \frac{1}{2}\Big(\alpha(|0\rangle + |1\rangle) \otimes |00\rangle + \alpha(|0\rangle + |1\rangle) \otimes |11\rangle + \beta(|0\rangle - |1\rangle) \otimes |10\rangle + \beta(|0\rangle - |1\rangle) \otimes |01\rangle\Big)$$

$$= \frac{1}{2}\Big(|00\rangle(\alpha|0\rangle + \beta|1\rangle) + |01\rangle(\alpha|1\rangle + \beta|0\rangle) + |10\rangle(\alpha|0\rangle - \beta|1\rangle) + |11\rangle(\alpha|1\rangle - \beta|0\rangle)\Big)$$

This shows that the resulting 3-qubit state has 4 components, and when measured, each component has a probability of

$(\frac{1}{2})^2 = 25\%$ of being observed. The remarkable part is that the measurement of the first two qubits (held by you) determines the state of the third qubit (held by your friend).

If the first two qubits are measured as $|00\rangle$, then the third qubit held by your friend must be $|\psi\rangle$, completing the teleportation.

For other possible measurements:

1. If the first two qubits are measured as $|01\rangle$, your friend can apply an X gate to their qubit:
$$X(\alpha|1\rangle + \beta|0\rangle) = |\psi\rangle$$
2. If the first two qubits are measured as $|10\rangle$, your friend can apply a Z gate:
$$Z(\alpha|0\rangle - \beta|1\rangle) = |\psi\rangle$$
3. If the first two qubits are measured as $|11\rangle$, your friend can apply ZX gates:
$$ZX(\alpha|1\rangle - \beta|0\rangle) = |\psi\rangle$$

Thus, based on your measurement, your friend can always reconstruct the quantum state $|\psi\rangle$ instantaneously, regardless of the distance between you and your friend. However, this does not imply that information can be transmitted faster than the speed of light, as the result of your measurement must still be communicated to your friend through a classical channel.

Algebraic Rules

The final challenge addressed in this paper is the unfamiliarity with many algebraic rules used in quantum computing for students. As a result, even simple computations can cause hesitation.

To address this, we summarize some fundamental algebraic rules in quantum computing and demonstrate their simplicity and utility through a proof of the famous no-cloning theorem.

Let A, B, C, D be matrices and $|a\rangle$, $|b\rangle$, $|c\rangle$, $|d\rangle$ be vectors.

$$(AB)^\dagger = B^\dagger A^\dagger \tag{1}$$
$$(AB)^T = B^T A^T \tag{2}$$
$$\langle(\alpha|0\rangle + \beta|1\rangle)| = \alpha^*\langle 0| + \beta^*\langle 1| \tag{3}$$
$$(|a\rangle + |b\rangle) \otimes |c\rangle = |a\rangle \otimes |c\rangle + |b\rangle \otimes |c\rangle \tag{4}$$
$$|a\rangle \otimes (|b\rangle + |c\rangle) = |a\rangle \otimes |b\rangle + |a\rangle \otimes |c\rangle \tag{5}$$
$$\alpha(|a\rangle \otimes |b\rangle) = (\alpha|a\rangle) \otimes |b\rangle = |a\rangle \otimes (\alpha|b\rangle) \tag{6}$$
$$(|a\rangle \otimes |b\rangle)^\dagger = |a\rangle^\dagger \otimes |b\rangle^\dagger = \langle a| \otimes \langle b| \tag{7}$$
$$(\langle a| \otimes \langle c|)(|b\rangle \otimes |d\rangle) = \langle a|b\rangle\langle c|d\rangle \tag{8}$$
$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD) \tag{9}$$
$$\mathrm{Tr}(A \otimes B) = \mathrm{Tr}(A)\,\mathrm{Tr}(B) \tag{10}$$

where Tr represents the trace of a matrix.

Example 4.1.

$$\begin{aligned}
|1\rangle \otimes |-\rangle &= |1\rangle \otimes \left(\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle\right) \\
&= \frac{1}{\sqrt{2}}|1\rangle \otimes |0\rangle - \frac{1}{\sqrt{2}}|1\rangle \otimes |1\rangle \\
&= \frac{1}{\sqrt{2}}\begin{pmatrix}0\\0\\1\\0\end{pmatrix} - \frac{1}{\sqrt{2}}\begin{pmatrix}0\\0\\0\\1\end{pmatrix} \\
&= \frac{1}{\sqrt{2}}\begin{pmatrix}0\\0\\0\\-1\end{pmatrix}.
\end{aligned}$$

With these rules at hand, let us prove the no-cloning theorem for quantum states.

Theorem 4.1. [16] Given an arbitrary quantum state $|\psi\rangle \in \mathbb{C}^2$, there is no quantum circuit capable of creating an exact copy of $|\psi\rangle$.

Proof: Suppose there exists a unitary operator U of dimension 2×2 such that for any quantum state $|\psi\rangle \in \mathbb{C}^2$, U maps $|\psi\rangle \otimes |0\rangle$ to $|\psi\rangle \otimes |\psi\rangle$, i.e., U creates a copy of $|\psi\rangle$.

Then, for two arbitrary states $|\psi_1\rangle$, $|\psi_2\rangle$ in $\mathbb{C}^2$, we have:
$$|\phi\rangle = U(|\psi_1\rangle \otimes |0\rangle) = |\psi_1\rangle \otimes |\psi_1\rangle$$
$$|\phi'\rangle = U(|\psi_2\rangle \otimes |0\rangle) = |\psi_2\rangle \otimes |\psi_2\rangle$$

Now, consider the inner product $\langle\phi|\phi'\rangle$:
$$\begin{aligned}
\langle\phi|\phi'\rangle &= (|\phi\rangle)^\dagger|\phi'\rangle \\
&= (U(|\psi_1\rangle \otimes |0\rangle))^\dagger\, U(|\psi_2\rangle \otimes |0\rangle) \\
&= (|\psi_1\rangle \otimes |0\rangle)^\dagger U^\dagger U(|\psi_2\rangle \otimes |0\rangle) \\
&= (\langle\psi_1| \otimes \langle 0|)(|\psi_2\rangle \otimes |0\rangle) \\
&= \langle\psi_1|\psi_2\rangle\langle 0|0\rangle \\
&= \langle\psi_1|\psi_2\rangle
\end{aligned}$$

Also,
$$\begin{aligned}
\langle\phi|\phi'\rangle &= (|\psi_1\rangle \otimes |\psi_1\rangle)^\dagger(|\psi_2\rangle \otimes |\psi_2\rangle) \\
&= (\langle\psi_1| \otimes \langle\psi_1|)(|\psi_2\rangle \otimes |\psi_2\rangle) \\
&= \langle\psi_1|\psi_2\rangle\langle\psi_1|\psi_2\rangle \\
&= \langle\psi_1|\psi_2\rangle^2.
\end{aligned}$$

This implies $\langle\psi_1|\psi_2\rangle = \langle\psi_1|\psi_2\rangle^2$. Hence, $\langle\psi_1|\psi_2\rangle$ is either 0 or 1.

Therefore, $|\psi_1\rangle$ and $|\psi_2\rangle$ are either orthogonal or in the same direction. This leads to a contradiction. □

5 Evaluation and Conclusion

In this paper, we introduced an approach to teaching introductory quantum computing from a computer science per-

spective, aiming to lower the mathematical and physical barriers for students. In Section 3, we summarized nine key concepts in linear algebra and reformulated the four postulates of quantum mechanics using linear algebra, making the material more accessible for computer science students. While the course has not yet been offered, and its evaluation and analysis remain as future work, the framework presented here provides a practical and adaptable starting point. We hope that educators can use this approach to initiate their own efforts in quantum computing education, fostering broader accessibility and interest in this emerging field.

References

[1] M. A. Nielsen, I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).

[2] S. Seegerer, T. Michaeli, R. Romeike, Quantum computing as a topic in computer science education, in *Proceedings of the 16th Workshop in Primary and Secondary Computing Education*, pages 1–6 (2021).

[3] M. Kaur, A. Venegas-Gomez, Defining the quantum workforce landscape: a review of global quantum education initiatives, *Optical Engineering*, *61*(8):(2022), 081806–081806.

[4] Ö. Salehi, Z. Seskir, I. Tepe, A computer science-oriented approach to introduce quantum computing to a new audience, *IEEE Transactions on Education*, *65*(1):(2021), 1–8.

[5] S. Goorney, J. Bley, S. Heusler, J. Sherson, A framework for curriculum transformation in quantum information science and technology education, *European Journal of Physics*, *45*(6):(2024), 065702.

[6] S.-Y. Hou, G. Feng, Z. Wu, H. Zou, W. Shi, J. Zeng, C. Cao, S. Yu, Z. Sheng, X. Rao, et al., Spinq gemini: a desktop quantum computing platform for education and research, *EPJ Quantum Technology*, *8*(1):(2021), 1–23.

[7] J. D. Weisz, M. Ashoori, Z. Ashktorab, Entanglion: A board game for teaching the principles of quantum computing, in *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*, pages 523–534 (2018).

[8] D. Escanez-Exposito, J. Correa-Marichal, P. Caballero-Gil, Using game-based learning and quantum computing to enhance steam competencies in k-16 education, *IEEE Transactions on Education*.

[9] J. C. Meyer, G. Passante, S. J. Pollock, B. R. Wilcox, Today's interdisciplinary quantum information classroom: Themes from a survey of quantum information science instructors, *Physical Review Physics Education Research*, *18*(1):(2022), 010150.

[10] J. C. Meyer, G. Passante, B. Wilcox, Disparities in access to us quantum information education, *Physical Review Physics Education Research*, *20*(1):(2024), 010131.

[11] K. Svozil, *Quantum logic* (Springer Science & Business Media, 1998).

[12] J. Xie, H. B. Hunt, III, On the undecidability and descriptional complexity of synchronized regular expressions, *Acta Informatica*, *60*(3):(2023), 257–278, ISSN 0001-5903, doi:10.1007/s00236-023-00439-3.

[13] J. Xie, H. B. Hunt, R. E. Stearns, On the computational and descriptional complexity of multi-pattern languages, *Theoretical Computer Science*, *1030*:(2025), 115063, ISSN 0304-3975, doi:https://doi.org/10.1016/j.tcs.2025.115063.

[14] J. Xie, H. B. Hunt III, R. E. Stearns, Pumping lemmas can be "harmful", *Theory of Computing Systems*, *68*:(2024), 1339–1352, doi:https://doi.org/10.1007/s00224-024-10169-9.

[15] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, W. K. Wootters, Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels, *Physical review letters*, *70*(13):(1993), 1895.

[16] W. K. Wootters, W. H. Zurek, A single quantum cannot be cloned, *Nature*, *299*(5886):(1982), 802–803.

Effective Approaches for Teaching Introductory Programming Without Coding Assignments

Jongwook Kim
West Chester University of Pennsylvania
jkim2@wcupa.edu

ABSTRACT

There is a noticeable gap between students' actual programming abilities and their evaluation results in introductory programming courses. Despite well-designed assignments, many students fail to develop essential programming skills because they rely on external help rather than writing code independently. We observed this discrepancy in prerequisite exams for 400-level courses, where senior students are asked to solve programming problems from earlier course assignments. To address this issue, we removed programming assignments from the grading criteria and introduced in-class programming quizzes, where students write code by hand under the supervision of instructors. This approach aims to ensure independent problem-solving and provide a more accurate assessment of students' coding capabilities. In this paper, we discuss our experience with the no-assignment approach, its impact on student learning, and how it leads to a more accurate reflection of their programming abilities, ultimately preparing them more effectively for advanced courses and real-world programming challenges.

1 Introduction

With years of experience teaching introductory programming courses in computer science, we (instructors) learned that even well-designed programming assignments, which require significant effort from instructors to create and grade, may not be as effective as intended. Despite the complexity and thoughtfulness put into these assignments, many students can progress to upper-level courses without acquiring the essential programming skills they should have gained, as they often do not write the code on their own. This creates a clear gap between students' programming abilities and the evaluation results from introductory programming courses. Over the years, this discrepancy became apparent in the results of prerequisite exams in our 400-level classes, where senior students' programming skills do not align with the letter grades they achieved in introductory programming courses, raising concerns about whether the assignments truly assess or improve students' programming skills.

To address this issue, several years ago, we removed programming assignments from our grading criteria entirely. Instead, we began giving students programming quizzes during each class, where they were required to write code by hand in front of us. This shift aims to achieve several goals: ensuring students write programs independently and allowing us to directly assess their coding abilities in real time. However, this change also required us to develop new logistics and structures for the course to make the approach feasible. In this paper, we discuss our own approaches for teaching introductory programming courses and their effectiveness in the context of emerging technologies like Large Language Models (LLMs), which are influencing how we teach programming today.

2 Background

Programming is a fundamental skill for students pursuing a degree in computer science, and it is hard to imagine a computer science education without it. However, through years of observation at colleges, we noticed a troubling trend: many computer science students lack confidence in their programming abilities, and as a result, they tend to avoid programming-intensive courses when possible. This issue of low confidence led us to explore different teaching methods in an attempt to discover the most effective way to teach programming.

In our early approach, we assigned students a challenging programming assignment each week, with the expectation that these assignments take the entire week to complete. Our goal was for students to invest enough time to deeply understand the course material while working independently on these difficult assignments. However, this expectation proved to be incorrect. Many students submitted incomplete assignments or cheated. We found numerous copies of our assignments available on online tutoring services like Chegg [1], where students pay a monthly subscription fee of $15 to access tutor-provided solutions. In more recent years, students have increasingly relied on LLMs like ChatGPT [2] to generate solutions for their programming assignments. This means that, technically, computer science freshmen can move on to upper-level courses without ever writing non-trivial programs independently, unless required to do so on exams, which is uncommon.

Through these experiences, we realized that traditional programming assignments often fail to achieve their intended

learning outcomes. Students can easily circumvent the challenges of these assignments by hiring freelance programmers or utilizing tutoring services, creating an unfair advantage over those who struggled to complete assignments on their own. While it is possible to use plagiarism detection software to check for copied code, such tools are limited in their ability to identify all forms of cheating. For example, it remains difficult to detect when students submit code written by a sibling or someone else. Code similarity alone cannot definitively prove academic dishonesty, especially when administrators (i.e., non computer science faculty) in charge of reviewing cases lack a technical background (e.g., renaming variables makes a program appear completely different to them.)

As a result, we decided to stop assigning programming assignments in our class. Instead, we implemented a new approach in which students take programming quizzes during each class session, requiring them to write code under our direct supervision. This method ensures that students cannot cheat, as they must write the program independently in class within a set time limit. Over the past few years, we have refined this approach in our introductory programming courses. Interestingly, freshmen in these courses outperformed senior students, solving more problems on average in a department-wide programming contest.

3 Grading Criteria

To address the limitations associated with assignment-based assessments, we adopted a grading scheme that placed 90% of the overall grade on in-class exams and quizzes. This approach ensures a more direct and consistent assessment of students' programming abilities, while also addressing concerns related to cheating and reliance on external help. The grading breakdown is structured as follows:

- **Quizzes** – Each class session includes a programming quiz that makes up 40% of the final grade. These quizzes are designed to evaluate students' fundamental understanding of the relevant concepts and their ability to apply them in real-time problem-solving scenarios, which require logical thinking and coding within a limited time.
- **Midterm exam** – The midterm exam (20%) provides an additional opportunity to evaluate students' understanding of the material covered up to that point in the course, along with their programming skills under exam conditions. The midterm consists solely of programming questions that require students to write more complex code.
- **Final exam** – The final exam (30%), like the midterm, is comprehensive and aims to assess the cumulative knowledge and skills students acquired during the semester.
- **Participation** – Students are required to follow several class policies, with violations leading to a deduction of up to 10% from their final score. Below are some of these policies to maintain focus and attention during class:
  - Stay for the entire duration of the class
  - Do not use electronic devices
  - Do not eat during class
  - Do not engage in private conversations
- **Typing test** – Students are required to pass a typing test [3] by achieving 100% accuracy within 90 seconds by the end of the semester. Failure to do so results in the student being ineligible to receive a final letter grade, regardless of their performance in other evaluations (e.g., exams and quizzes).

This grading scheme is designed to prioritize in-class assessments, providing a more consistent and reliable measure of students' programming abilities. By eliminating assignment-based grading, students are unable to seek external help or use AI-assisted tools—such as peer assistance or code-generating programs—that compromises the fairness of the assessment.

This approach not only helps maintain academic integrity but also encourages students to take ownership of their learning process and improve their coding skills through practice and self-reliance. Learning programming is comparable to solving math problems in that when students receive help from others, they lose the opportunity to solve the problem on their own and become more reliant on external assistance, which can greatly hinder their problem-solving abilities. In the worst case, this dependency during the early stages of learning programming deprives students of the chance to ever fully learn programming. It is crucial that students attempt to write the code and solve the problems independently.

4 Course Structure and Resources

To make the exam-only (or assignment-free) course effective, we revised the way students engage with course materials and prepare for quizzes and exams. Here are some of the key approaches we implemented:

4.1 Printed Textbook

Although digital course materials are widely used today, we require students to purchase a printed (rather than digital) textbook for our programming courses. This decision is based on several key reasons, all aimed at improving students' learning experiences and promoting long-term academic success. Mangen et al. [4] found that students who read printed textbooks performed better on reading comprehension tests than those who read digital versions. Many educators still prefer printed textbooks when deep learning and focus are needed. Freshmen, in particular, need to develop good study habits, and reading the textbook is a key part of their self-driven learning process, as they will be required to consult textbooks more deeply in the upcoming academic years.

At the beginning of the semester, we emphasize the importance of reading the textbook, advising students to read each chapter at least three times. We make it clear that our lectures do not cover every detail and that the textbook is the most comprehensive resource for learning. To prepare for quizzes, students also need a textbook with exercises similar to quiz questions (more details in Sections 4.2 and 4.3). We suggest students purchase affordable used textbooks, which are available for $10 to $30 on platforms like eBay or Amazon.

4.2 Exercise Problems in the Textbook

Students are expected to master all the exercise problems in each chapter to succeed in the course. This is a time-intensive task, requiring students to dedicate at least three hours[1] per day to practicing coding. Since quiz questions are directly based on these exercises, completing them helps students better prepare for the quizzes. The clear link between textbook exercises and quiz content creates a focused and intentional learning path, ensuring that students are well-prepared for assessments and build their skills over time.

Below are an example exercise problem and the related quiz question:

> Write a Java program that generates the following output. Use for-loop(s) to eliminate code redundancy. You must declare and use a variable in the for-loop header, set its initial value to 1, and increment it by 1 after each iteration.

For the same question above, students write a program that produces the exercise output (left) shown below for practice. During the quiz, they are asked to generate a more complex output (right):

Exercise Output	Quiz Output
....1 ...2. ..3.. .4... 5....	99999999977777777555553331 88888888666666444422 7777777555553333 6666664444 55555

By solving the exercises, students bridge the gap between theoretical knowledge and practical application. The more they practice, the more they refine their coding skills, which is the only path for most students to becoming proficient in programming. Through consistent practice, students also develop good programming habits necessary for success in programming-intensive courses and real-world programming tasks. In short, solving the exercises helps students gain confidence in their coding skills, succeed in the course, and ultimately prepare for their future career.

[1]For students planning to pursue internship or full-time programmer positions within the next year or few years, spending three hours a day coding is not excessive, even with their other coursework. The programming skills acquired in this course are crucial for their future careers, making the additional effort a worthwhile investment.

4.3 In-class Programming Quizzes

In each class session, students are given two quiz questions, each worth 15 points, with a 20-minute time limit. Since students are expected to have already practiced the exercises in the textbook, this time constraint is considered appropriate, making the in-class assessment less intimidating while still providing a meaningful challenge. Unlike typical programming assignments, students are encouraged to collaborate and discuss exercise solutions with their peers, which helps them explore different approaches and deepens their understanding of the concepts.

Writing code by hand, without the use of machines, is an effective method for improving students' programming abilities, especially for beginners. It requires students to slow down and engage in deeper, more deliberate thought processes. This approach encourages students to thoughtfully examine each line of code they write, helping them to internalize the meaning of every word (i.e., lexeme). Unlike typing code in an Integrated Development Environment (IDE), where syntax errors are immediately flagged, handwritten coding demands a deeper understanding of both syntax and logic.

To ensure that students follow best coding practices, the grading criteria must be strict. Even minor mistakes, such as inconsistent indentation, poor variable naming, redundant code, or unreadable code, result in point deductions. We learned from experience that senior students struggle to break long-held bad programming habits because they did not lose (many) points for these habits when they first started learning programming as freshmen. For example, a student in our class loses 4 points (26%) due to incorrect indentation in an executable Java program below:

```
public class Welcome {
   public static void main(String[] args)
      {
System.out.println("Welcome to Java!"); }
   }
```

These grading policies reinforce the idea that programming is not solely about producing the correct output but also about writing clean, maintainable code. This emphasis encourages students to pay attention to code hygiene such as formatting, naming conventions, and code-level efficiency, while also considering how other programmers will interact with their code in a professional setting. To keep students motivated, they are offered a make-up quiz that replaces the lowest score from the last three quizzes.

4.4 Unofficial Programming Assignments

The quizzes focus on short coding tasks, which, by nature, cannot require students to write hundreds of lines of code within a 20-minute time. To help students develop their ability to write more complex programs, we assign unofficial programming assignments that challenge them to solve intricate problems, requiring advanced critical thinking, problem-solving, and logical reasoning.

These assignments are not included in the course assessment and are intended to give students the opportunity to explore programming in more depth without the pressure of final grades. This approach is also intentional to discourage students from submitting code they copied from other sources. Since the assignments do not affect the final grade, there is no immediate incentive to cheat.

The programming assignments are introduced after the course withdrawal deadline (and midterm exam). Only students who are qualified to complete the course and have the required knowledge and skills (such as loops, branches, interactive programming, etc.) are given these challenging tasks. To further support student growth, we provide detailed feedback on these assignments, which help them improve their coding abilities in a meaningful way.

5 Challenges and Adjustments

5.1 Low Pass Rate

The course is designed to pass students who are truly capable of writing code and understanding a programming language, which makes it challenging for those who do not follow our guidance, such as reading the textbook multiple times and completing all the exercises. Over the last three semesters, the course enrolled 62 students, with an average pass rate of 37%, and 52% of students dropping out before the midterm. While this high dropout rate is not ideal, it reflects the course's demanding nature and the high standards it maintains. Additionally, there was an 11% failure rate after the midterm.

The low pass rate aligns with broader trends observed in similar courses at other colleges [5; 6], where introductory programming courses often experience withdrawal or failure rates in the range of 30–50%. Despite this, the decision to maintain a rigorous grading structure is intentional. We believe that it is more important to uphold high standards and rigor to better prepare students for the challenges they will face in advanced programming courses or real-world technical roles, even if this results in fewer students passing the course.

Maintaining high standards has been an ongoing debate in academia, especially in non-top-tier colleges, particularly in courses that act as gateways to technical disciplines. While some argue that high attrition rates reflect a lack of accessibility or support for students, we believe it is important to consider both the challenges students face and the long-term benefits of a high bar for success in introductory programming courses. This approach helps graduate quality students and preserves the reputation of the academic program.

5.2 Manual Grading and Verbal Feedback

Grading quizzes and exams by hand is a laborious task for instructors, especially in large classes at public universities. However, manually grading code provides significant benefits for beginner programmers, who need detailed and personalized feedback to improve. Beyond fixing compile- or run-time errors, these students need to learn how to refine their current algorithms (i.e., rewrite the code in a more efficient way). While manual grading is overwhelming in large classes, it becomes more manageable as class sizes typically decrease after the midterm in gatekeeping courses. Although not intentional, with fewer students, instructors can provide more in-depth feedback while also reducing the grading workload.

We recommend using office hours to provide verbal feedback on quizzes and exams. This gives students a direct and more efficient opportunity to ask questions, clarify any misunderstandings, and receive personalized guidance on improving their coding skills. Additionally, it helps freshmen develop good learning habits by emphasizing that office hours are a valuable resource for efficient learning.

Office hours also give instructors the opportunity to track each student's progress over time. This ongoing interaction enables instructors to identify patterns in common mistakes and adjust their teaching strategies to address recurring challenges, benefiting the entire class. Since this process is time-consuming and labor-intensive for instructors, department-wide support, such as teaching assistants, should be provided.

6 Discussion

As evidence of the effectiveness of the exam-only introductory programming course, freshmen who enrolled in the course we developed solved more problems than seniors who did not take our course in a department-wide programming competition [7], which included a total of 36 computer science students. The five freshmen averaged 2.25 problems[2] solved per competitor, while the ten seniors averaged 2.07 problems. Although the small number of participants means these results may not fully represent the broader group of freshmen and senior students, the findings are still noteworthy, as most participants are motivated and have good programming skills.

[2]The problems do not require advanced knowledge of data structures and algorithms and can be solved by anyone with basic programming skills.

Out of 23 students who completed the course and received a final letter grade (including F) during the 2023-24 academic year, the course received high ratings (91%) from students. However, we are cautious about placing too much weight on their evaluations at this stage. First, the evaluations from students who dropped the course before the midterm are not included. Second, as freshmen, the students do not yet have the perspective to fully understand the long-term benefits of a rigorous, hands-on programming education. Since our focus is on their long-term learning outcomes rather than immediate satisfaction, we believe their feedback will be more meaningful in two years or after graduation, when they are better able to appreciate the full value of our approach. Third, regardless of students' evaluations, we, as instructors, sometimes need to maintain teaching methods that may not be favored by many students. For example, although most students dislike the quizzes in each class, if this approach effectively encourages daily coding practice, we should persist in continuing it.

7 Related Work

Krusche and Berrezueta-Guzman [8] presented an interactive learning methodology aimed at improving programming education for first-year computer science students. Their approach combines real-time feedback, communication platforms, and automated assessments to support continuous learning and enhance student outcomes. Also, it includes in-class activities, group tutorials, and independent homework assignments to strengthen coding proficiency and problem-solving abilities.

Robinson and Carroll [9] proposed an open-source online learning platform to improve the teaching and assessment of programming in large classes. Their approach utilizes modern web technologies and automated assessment tools to provide real-time feedback, personalized instructor guidance, and a development environment that mimics real-world programming. The platform supports both formative and summative assessments, promoting self-paced learning, enhancing feedback, and reducing administrative workload for instructors.

Vihavainen et al. [10] conducted a systematic review of teaching interventions designed to enhance pass rates in introductory programming courses. The review analyzed 60 pre- and post-intervention pass rates from 13 different teaching methods, revealing that, on average, interventions boosted pass rates by nearly one third compared to traditional lecture-lab approaches. Although no statistically significant differences were observed between the various methods, courses that included relatable content and collaborative elements, such as pair programming, achieved the best outcomes. The study emphasizes that while no single teaching approach stands out as the best, intentional changes typically lead to improved pass rates.

8 Conclusion

The exam-only approach we developed over the years represents a shift away from traditional programming assignments towards a more hands-on, real-time method that emphasizes skill development through continuous practice and direct supervision. By abandoning traditional assignments prone to cheating and reliance on external help, and instead focusing on in-class coding exams, students are encouraged to engage more deeply with the course materials in order to succeed, thereby building their confidence and ability to write code under pressure. The positive outcomes observed in our students, including their stronger performance in a department-wide programming contest, suggest that we are on the right track in fostering a deeper understanding of programming and helping students develop the confidence necessary to succeed in more challenging programming-intensive courses. Ultimately, this experience underscores the importance of adapting teaching methods to address the challenges posed by modern educational tools, ensuring that students are not only learning but also truly mastering the skills required in the field of computer science.

References

[1] Chegg, https://www.chatgpt.com.

[2] chatGPT, https://www.chegg.com.

[3] Typing Club, https://www.typingclub.com/sportal/program-3/8832.play.

[4] A. Mangen, B. R. Walgermo, K. Brønnick, Reading linear texts on paper versus computer screen: Effects on reading comprehension, *International Journal of Educational Research*, *58*:(2013), 61–68.

[5] J. Bennedsen, M. E. Caspersen, Failure rates in introductory programming, *SIGCSE Bull.*, *39*(2):(2007), 32–36.

[6] J. Bennedsen, M. E. Caspersen, Failure rates in introductory programming: 12 years later, *ACM Inroads*, *10*(2):(2019), 30–36.

[7] West Chester University Programming Competition, https://sites.google.com/view/wcpc.

[8] S. Krusche, J. Berrezueta-Guzman, Introduction to Programming using Interactive Learning, in *2023 IEEE 35th International Conference on Software Engineering Education and Training*, pages 178–182 (2023).

[9] P. E. Robinson, J. Carroll, An online learning platform for teaching, learning, and assessment of programming, in *2017 IEEE Global Engineering Education Conference*, pages 547–556 (2017).

[10] A. Vihavainen, J. Airaksinen, C. Watson, A systematic review of approaches for teaching introductory programming and their influence on success, in *Proceedings of the Tenth Annual Conference on International Computing Education Research*, page 19–26 (2014).

INTEGRATING AI INTO CS/IS EDUCATION: PRACTICAL APPLICATIONS AND ABET COMPLIANCE

Pratibha Menon
Pennsylvania Western University
menon@pennwest.edu

ABSTRACT
Artificial Intelligence (AI) transforms computing education by providing students with creative tools that make learning more engaging and effective. This study investigates how the use of AI tools in Computer Science (CS) and Information Systems (IS) programs aligns with the standards set by ABET accreditation. A systematic literature review (SLR) identifies key AI-driven technologies, including coding assistants, intelligent tutoring systems, automated assessment, and academic-industry collaborations. The analysis highlights AI's role in personalized learning, scalable assessments, and curriculum alignment with industry needs while mapping these innovations to ABET criteria. The study emphasizes responsible implementation and recommends curriculum updates, faculty training, and stakeholder collaboration. It guides educators and program developers on leveraging AI for enriched student learning and outlines future research directions for sustainable AI-driven curriculum innovation.

KEY WORDS
AI, CS, IS, ABET, Curriculum

1. Introduction

Advancements in AI technologies are redefining the competencies needed for CS and IS graduates [1][2]. As AI-driven roles and workflows become more prevalent, integrating AI into education has become imperative [3][4]. Artificial intelligence tools provide features like personalized learning, automatic code creation, and feedback based on data analysis, addressing ongoing issues in CS and IS education [5]. For example, large class sizes and diverse student backgrounds make personalized instruction difficult. Still, AI-driven tutoring systems are capable of offering customized assistance to a large number of students simultaneously. Likewise, grading programming assignments is time-intensive; Automated feedback systems powered by AI enable educators to focus more on in-depth mentoring by handling routine evaluations. AI's can significantly enhance learning outcomes—a recent meta-analysis reported a substantial positive effect size of 1.36 for AI interventions in CS education [5].

However, educators looking to incorporate AI into curricula must ensure that AI assistants, such as code generators like GitHub Copilot, do not compromise foundational learning or academic integrity [6][7][8]. Clear strategies are needed to integrate AI in ways that support effective pedagogy. Additionally, in accredited CS and IS programs, curriculum changes are guided by standards like ABET's criteria, which require programs to achieve specific student outcomes and undergo continuous improvement based on assessment [9]. ABET expects graduates to develop competencies in areas such as problem analysis, solution design, ethics, teamwork, and the application of modern tools. The current list of ABET student outcomes and criteria that are common to both CS and IS programs and criteria are listed in Appendix A.

Furthermore, programs must involve key stakeholders, such as industry advisors, in curriculum updates and ensure students can access necessary computing resources. As a result, integrating AI-based tools must enhance, and not hinder, the achievement of learning outcomes and be supported by rigorous evaluation. Aligning AI adoption with ABET standards is critical for two reasons: it upholds educational quality by providing a well-structured framework to assess the impact of AI interventions on student learning.

Given these opportunities and constraints, this research explores how CS and IS education can effectively incorporate AI while adhering to accreditation standards. A key gap in existing research is the lack of comprehensive guidance on integrating AI educational tools within formal curricula; while previous studies have examined individual AI applications, few have analyzed their systematic implementation in accredited programs. To address this gap, the study investigates the following research questions:

1. **RQ1:** What are the primary practical applications of AI in CS and IS education based on recent literature?
2. **RQ2:** How do these AI applications enhance teaching and learning outcomes in computing courses, and what challenges arise in their implementation?

3. **RQ3:** How can AI-driven tools be integrated into curricula while ensuring compliance with ABET accreditation criteria and student learning outcomes?

This paper makes a unique contribution by addressing these research questions: it synthesizes AI-driven educational innovations and aligns them with accreditation requirements. Findings from various studies are analyzed to inform actionable approaches for curriculum design that can enable educators to utilize AI's benefits, such as personalized learning, enhanced efficiency, and industry relevance while maintaining accreditation standards. The subsequent sections present a review of related work, the literature review methodology details, an analysis of AI application areas, and recommendations for integrating these innovations within ABET-aligned curricula.

2. Existing Research on AI in CS/IS Education

For many years, researchers have explored AI's role in education, resulting in the creation of intelligent tutoring systems, automated assessment tools, and adaptive learning environments [10][11]. Within CS education, intelligent tutoring systems (ITSs) have been extensively studied for their ability to provide personalized, real-time feedback in programming courses [12]. Research shows that adapting to real-world student mistakes helps AI tutors improve understanding and academic outcomes. Early examples, such as the Python Tutor, and more advanced AI-based tutors now use machine learning algorithms to guide students through coding exercises, offering hints and correcting misconceptions [12][13].

In IS and related fields, education often involves decision-making, data analysis, and the use of information technology in organizational contexts. AI tools that aid decision support or predictive analytics can be embedded into IS coursework to enrich these topics. For example, an AI tool could be embedded in a data mining or business analytics course to help students analyze datasets or visualize patterns. Even though research explicitly focusing on AI in undergraduate IS curricula is less abundant than in CS, parallels can be drawn from case studies. For example, in an exploratory study in a database administration course, which is a subject bridging CS and IS, students were encouraged to use ChatGPT as a support tool [15].

Prior studies have explored automated assessment of programming assignments. Various tools and techniques—such as unit testing frameworks, code stylometry, and static analyzers—have been designed to automatically evaluate student code and provide immediate feedback [16]. A systematic review categorized 121 automated grading tools, highlighting that most offer near-instant feedback and allow multiple submission attempts, which enhances student satisfaction and reinforces learning through iterative improvement [16].

Beyond tutoring and grading, AI has been applied in learning analytics to detect students who may be at risk by analyzing educational data, adaptive content delivery, and virtual teaching assistants that respond to routine student inquiries in discussion forums [17][18]. All these advancements demonstrate AI's potential to improve engagement and learning outcomes in CS/IS education.

While AI-driven tools like tutors and automated graders have proven effective, comprehensive research is lacking on their systematic integration into the curricula of accredited programs. Existing studies often focus on the benefits of AI tools within individual courses or specific learning contexts but do not address broader pedagogical and administrative challenges. A structured approach is essential for embedding AI tools across various courses in a CS/IS program, ensuring alignment with learning objectives and accreditation standards. This paper addresses this gap by connecting micro-level findings from prior research (e.g., performance improvements from AI tutors in a programming course) with macro-level considerations, such as curriculum design and accreditation requirements. Furthermore, issues such as the ethical implications of AI (e.g., concerns over academic integrity and student over-reliance on AI-generated solutions) and faculty readiness to adopt AI tools will also be addressed in this study. This study is guided by three research questions, previously introduced in the Introduction section. By answering these questions, this research seeks to provide institutions with a clear roadmap for integrating AI-driven innovations into their programs while ensuring adherence to accreditation standards.

This research integrates insights from educational technology and accreditation policies. Unlike previous studies that either focus exclusively on the technological effectiveness of AI tools or discuss accreditation requirements in broad terms, this study bridges the two domains. By synthesizing research across CS education, IS education, and higher education policy, we provide actionable recommendations for educators.

Instead of focusing only on the implementation of AI tools like auto-graders and coding assistants, this study recommends revising course outcomes, refining assessment methods, and enhancing continuous improvement processes. Doing so ensures that AI-driven innovations are aligned with ABET's student learning outcomes and accreditation criteria. For example, when AI tools are utilized to improve education, this paper demonstrates how the data they produce can evaluate student performance, thus supporting adherence to accreditation standards. The article presents an organized

framework for the conscientious and efficient incorporation of AI into Computer Science and Information Systems curricula. Results from previous research on various AI-tools back this approach and align with accreditation criteria, ensuring that AI enhances student learning outcomes while adhering to program accreditation standards.

3. Methodology

A systematic review of recent academic publications was performed, concentrating on AI-based educational tools and their effects on learning outcomes, and the review results were used to investigate their conformity with accreditation criteria. Major academic databases were searched, including IEEE Xplore, ACM Digital Library, ERIC, Springer, and Elsevier, using keywords like “AI in education,” “CS curriculum AI,” “intelligent tutoring,” and “automated grading.” To ensure relevance, we focused on studies from the last decade (2012–2023) and prioritized those related to undergraduate CS and IS programs.

Out of an initial selection of more than 200 papers, the subsequent inclusion criteria were utilized: studies needed to address AI's impact on teaching, curriculum development, or accreditation within computer science/information systems education. Any purely theoretical works, K-12 studies without transferable insights, and non-peer-reviewed articles were excluded. After screening abstracts, the selection was narrowed to 60 papers, and a detailed full-text analysis left us with 36 key studies that provided strong evidence of AI's educational applications.

For each study, the following contents were extracted:

- **AI applications used** (e.g., coding assistants, tutoring systems, automated grading).
- **Educational context** (course type, student level, class size).
- **Measured outcomes** (impact on student performance, engagement).
- **Curriculum accreditation considerations** (alignment with ABET standards, ethical concerns, and practical challenges).

A diverse range of studies, from controlled experiments to faculty perceptions, made it infeasible to perform a statistical meta-analysis. Instead, a qualitative meta-synthesis was performed by identifying common themes across the literature. Thematic analysis identified four main AI application areas: coding assistants, intelligent tutoring systems, automated assessment, and industry collaboration on AI projects. Emphasis was placed on the abilities of the AI tools to enhance student learning, teaching effectiveness, and goals related to ABET accreditation. This organized strategy facilitated the collection of robust evidence to discuss the practical uses of AI in CS and IS education. The following sections present these insights, examining how each AI tool works, its impact, and how it can be integrated into accredited curricula.

4. Discussion: How AI is Transforming CS/IS Education

4.1 AI-Powered Coding Assistants

Tools like GitHub Copilot offer instantaneous coding recommendations to students, functioning similarly to an available 'pair programmer' [7]. Such code-assistance tools help students by reducing frustration with syntax, exposing them to multiple solution approaches, and accelerating their coding progress [19][20]. For example, learners may seek help while stuck on a programming problem using a code-assistance tool that generates multiple options for solutions [21][22]. Coding assistants' possible advantages include decreasing student frustration with syntax issues, offering template code, and enabling more advanced problem-solving. Code assistants expose students to multiple approaches to solving the problem, which could be tapped as a learning opportunity to develop program design and critical thinking skills.

Challenges: Despite its benefits, using AI coding assistants raises concerns about academic integrity and over-reliance [23][24]. If students copy AI-generated code without understanding it, their learning suffers [25]. Instead of banning AI tools outright—an impractical approach as they become widely available—educators are shifting towards responsible AI use [26]. Responsible use of AI includes teaching students how to engage critically with AI outputs, verify suggestions, and explain their thought processes. By doing so, students learn critical thinking and debugging skills alongside coding.

Alignment with Accreditation: ABET's curriculum criteria expect graduates to utilize contemporary tools in their computing activities. The use of AI tools supports this need, and the integration of AI tools may necessitate an adaptation of the course outcomes and evaluations. For instance, asking students to explain or improve upon AI-generated code guarantees that they cultivate problem-solving and debugging abilities instead of merely reproducing answers.

With ABET's outcome and program criteria that emphasise ethics and lifelong learning, it is important to make students aware of how coding assistants work and some of their limitations. For example, coding assistants backed by large language models could occasionally give errored responses or incorrect advice. Students need to learn to verify the answers provided by the tool, which requires critical evaluation of the coding solutions, which ties into professional responsibilities (ABET outcome of ethics and judgment).

4.2 Intelligent Tutoring Systems

AI-driven Intelligent Tutoring Systems (ITSs) offer personalized, interactive learning experiences, especially in programming courses [27]. In CS and IS education, ITS often uses virtual programming tutors. These tutors analyze student inputs, provide tailored hints, and help clarify misconceptions, like a virtual teaching assistant [27]. For example, an ITS for learning Java may ask a student to write a loop and, upon errors, give targeted hints, such as asking to check the initial value. Recent developments include the integration of Large Language Models in ITS, enabling interfaces with conversational interactions [12]. In large classes with limited one-on-one instructor time, ITSs ensure that every student gets instant feedback and can learn independently. Research consistently shows that adaptive tutoring improves problem-solving skills and student success rates. By tracking student progress, ITSs provide instructors valuable insights into common errors and areas needing reinforcement [28].

Challenges: The biggest concern with using AI-driven ITS is ensuring students do not become too dependent on AI tutors and instead develop independent problem-solving abilities [10][27]. Studies have proposed balancing AI assistance with reflective assignments where students must explicitly document how they used AI tutors and explain their problem-solving rationale [29]. However, another study recommends combining AI-based tools with traditional teaching approaches like instructor-led tutorials and peer mentoring to help students avoid becoming overly reliant on AI tutors [23].

Alignment with Accreditation: ITS can help students effectively grasp key learning concepts. These systems also provide instructors with practical insights into how students learn. For instance, by using AI tutors, instructors can track the number of attempts students typically need to solve problems or see how frequently they use the hints. Gathering data on learners' behaviors helps instructors identify areas where students might struggle. When combined with traditional assessment methods, data from ITS can provide a fine-grained understanding of student achievement. However, faculty will need to be involved in interpreting data collected from the tutors so that they can adjust the tutoring content accordingly.

4.3 Automated Assessment

AI-based grading systems can reliably and quickly handle routine programming assessments, which lets instructors spend less time on repetitive grading tasks and more time on personalized teaching activities. These tools provide instant feedback, allowing students to learn iteratively by refining their work based on feedback before final submission [30][31]. For educators, automated grading improves scalability, essential for large classes, and generates analytics highlighting everyday student struggles. This data can guide curriculum improvements and pinpoint topics that need additional attention.

Challenges: While auto-graders efficiently check code correctness, they may struggle with assessing creativity, style, or documentation quality [31][32]. Therefore, many courses combine automated grading with manual evaluation to ensure students meet professional coding standards. Another concern of using auto graders is the integrity of the assessment. The question banks and solutions might circulate among student cohorts if the same auto grader is used for tests each time. Instructors need to change the question banks to maintain assessment integrity continuously.

Alignment with Accreditation: Automated assessment tools contribute to ABET's continuous improvement requirement by providing direct evidence of student performance. Automated graders could provide a wealth of direct assessment data aggregated at course and program levels to assess learning outcomes. For example, if an outcome is related to programming proficiency, the auto grader could record how many problems a student solved and how many attempts they took to measure programming proficiency. However, to maintain rigor, assessments should ensure students engage in problem-solving rather than just "coding to the test," which could be mitigated by using a blend of automated and manual grading to ensure that students gain sound problem-solving skills. For example, the auto grader could check a program's functionality, while an instructor can manually check the design approach and coding style.

4.4 Industry Collaboration

Partnering with the industry ensures CS and IS programs stay aligned with real-world AI applications. Industry-academia collaborations can be guest lectures, capstone projects, internships, or AI-focused course modules co-developed with industry professionals. Such collaborations expose students to the latest AI tools and practices while giving them hands-on experience with real-world challenges [33]. Industry collaborations help students develop technical and professional skills, from teamwork and communication to handling real datasets and working within development constraints [34]. Programs that integrate industry insights into their curriculum ensure that graduates are job ready [35]. For CS and IS programs, industry collaboration can provide practical data sets, problem statements, and exposure to software platforms and project workflows that could enrich student projects. For example, an IS program could collaborate with a local company to prototype a machine learning model to solve a business problem as part of a class project, with company employees giving feedback to student teams. Such experiences could expose students to professional practices (such as agile methods and code

versioning) and the real-world constraints of deploying machine learning models in production environments.

Challenges: Not all students have equal access to industry projects, so ensuring broad participation is key. Academia must also balance foundational learning with industry-specific tools—focusing on relevant principles despite changing technologies [36].

Alignment with Accreditation: From a curriculum perspective, industry input is formally incorporated via advisory boards that review program objectives and outcomes. ABET program objectives criteria emphasizes involving stakeholders like industry partners when updating curricula. Collaborating with industry professionals directly supports accreditation requirements for responsiveness to stakeholder input while also providing practical benefits and real-world experience for students. Industry partners can help identify which AI skills are most needed (for example, proficiency in data analytics, cloud services, etc.) and help programs adjust course contents.

AI-focused projects also map to ABET's teamwork and communication outcomes, ensuring students gain the skills necessary for professional practice. When students work on an AI project with industry mentors, they learn how to explain technical content to non-academic stakeholders and document their work professionally. Moreover, capstone projects in collaboration with industry often serve as a culminating assessment of all student outcomes. If such capstone projects involve AI tools and methods, then the program must ensure that students gain the prerequisite knowledge to work with them, which might require curriculum restructuring.

ABET's program educational objectives criteria embodies the philosophy of preparing graduates for real-world practice and lifelong learning. While companies might be keen on students learning specific proprietary tools, academia must focus on foundational principles. Balancing these two goals requires an ongoing dialogue between industry and academia. Nevertheless, incorporating AI education helps students meet program outcomes with a clear employment path. By seeing how AI is being used in the industry, students gain the context that could spark the motivation to master the theory behind these tools and adapt to new tools in the future.

4.5 Final Thoughts: AI as a Learning Tool, not a Shortcut

AI tools can be transformative for CS and IS courses. However, effective use of AI tools that enrich student learning and don't diminish students' critical thinking and problem-solving skills requires thoughtful integration. Through careful curriculum design and assessment methods and with clear guidelines for AI use, institutions can effectively utilize AI's advantages while preserving academic standards and meeting accreditation expectations.

This review identified four key areas where AI tools are making a tangible impact: coding assistants, intelligent tutors, automated grading, and industry collaborations. When leveraged thoughtfully, these tools help address persistent challenges by providing personalized learning, instant feedback, and real-world applications. Rather than conflicting with accreditation standards, AI can support and strengthen ABET compliance when integrated thoughtfully, offering new forms of assessment data and ensuring curricula stay updated with industry advancements. However, AI integration must be accompanied by curriculum adaptations, clear guidelines, and ethical training to maximize its benefits and avoid pitfalls like over-reliance or misuse.

5. Key Recommendations for AI Integration

To help CS/IS programs incorporate AI while maintaining ABET accreditation, we suggest the following strategies:

1. Align AI Tool Use with Learning Outcomes—Update course objectives to explicitly include AI competencies, such as evaluating AI-generated code or using AI tools effectively in problem-solving. Aligning AI tool use with learning outcomes ensures that AI adoption is intentional and measurable and supports accreditation requirements.

2. Teach Responsible AI Use – Introduce ethics and professional guidelines around AI, covering issues like plagiarism, bias, and intellectual property. Faculty training is also crucial; instructors should understand AI tools to set expectations, guide responsible use, and detect misuse.

3. Leverage AI for Continuous Improvement – Use data from AI tutors and automated grading systems to track student performance trends, identify learning gaps, and inform curriculum adjustments. Using AI for continuous improvement supports ABET's emphasis on evidence-based program assessment.

4. Maintain Human Oversight – AI should enhance, not replace, traditional teaching methods. A balanced approach—combining automated tools with instructor-led discussions, manual grading, and peer feedback—ensures students develop deep, independent problem-solving skills.

5. Strengthen Industry Collaboration – Partnering with companies on AI-driven provides students with hands-on experience with real-world

applications while ensuring curricula stay relevant to workforce demands. Industry involvement also aligns with ABET's requirement for stakeholder engagement and curriculum responsiveness.

By following these recommendations, institutions can integrate AI thoughtfully, creating continuous innovation while maintaining accreditation integrity. Table 1 summarizes the educational benefits of AI based tools such as coding assistants, intelligent tutoring systems, auto graders and align with ABET outcomes and criteria. This table also show the use of AI tools align with ABET outcomes and criteria for CS and IS programs that are listed in Appendix A.

Table 1. Key AI Applications in CS/IS Education and Alignment with ABET Standards

AI Application	Educational Benefits (Practical Impact)	Alignment with ABET Standards (Criteria/SO)
AI-Powered Coding Assistants (e.g., GitHub Copilot)	*Increased coding productivity*: helps students write and complete code faster, with fewer syntax errors. *Learning by example*: expose students to standard coding patterns and multiple solution approaches. *Just-in-time support*: reduce frustration and keep students engaged through hints and code suggestions.	Supports the use of **modern tools** in computing practice (ABET Curriculum criteria). It frees time to focus on design and address higher-level outcomes like problem analysis and solution design (Student Outcomes 1 and 2). Requires teaching ethical use, tying into **professional and ethical responsibility** outcome (Student Outcome 4).
Intelligent Tutoring Systems (AI-driven personalized tutors)	*Personalized learning*: adaptive feedback tailored to each student's mistakes and pace, improving understanding. *Scalability*: provides one-on-one style help in large classes, ensuring no student is left behind. *Mastery and retention*: students can practice until mastery with guidance at each step, leading to better retention of concepts.	Enhances attainment of **student outcomes** by ensuring fundamental knowledge is solid for all students. Provides rich **assessment data** on student performance for continuous improvement (Criterion 4). Encourages independent learning skills, aligning with the outcome of **lifelong learning** (implied in ABET criteria via staying current).
Automated Assessment Tools (auto grading and feedback systems)	*Immediate feedback*: students learn from mistakes in real time and can improve their solutions through multiple attempts. *Consistency*: objective and uniform grading criteria improve fairness. *Efficiency*: instructors save time on grading, allowing more focus on teaching and mentoring.	Directly supports **assessment of outcomes** – auto-grader results serve as evidence of student proficiency (Criterion 4). Ensures curriculum has **appropriate evaluation processes** (Criterion 3 & 4) with quantitative and qualitative measures. Using such tools demonstrates the integration of **current educational practice** in program delivery, which is relevant to facilities and support criteria.
Industry Collaboration (AI-focused projects, internships, curriculum input)	*Relevance*: students work on real-world AI problems, making learning applied and up-to-date with industry trends. *Skills*: teamwork, communication, and project management skills develop alongside technical AI skills. *Networking*: students gain mentorship and connections that can lead to career opportunities.	It satisfies ABET's call for **constituency involvement** in the curriculum (Criterion 2), ensuring that program objectives meet industry needs. Reinforces **professional skills** like teamwork and communication (Student Outcomes 3 and 5). Keeps program educational objectives aligned with evolving technology in the industry, aiding continuous curriculum relevance (Criterion 4 and Program Educational Objectives).

6. Conclusion

Even though this paper provides a structured approach for incorporating AI into CS and IS curricula, ongoing research is essential to keep pace with evolving technologies. Future studies could empirically explore the effectiveness of AI tutors vs. traditional instruction in improving long-term learning. For example, controlled experiments could study how much coding assistants improve learning outcomes in a programming course or how AI-backed tutors compare with human tutoring regarding long-term retention of learning content. Another area for investigation is developing frameworks for AI literacy in CS/IS education—defining what every student should understand about AI beyond just using AI tools. Such a framework could inform standardized curriculum guidelines in the future. More work needs to be done on developing and implementing assessment strategies that adapt to AI to ensure students are evaluated based on their understanding rather than AI-generated outputs. Assignments and exams may require rethinking, and research could explore new forms of assessment that require human creativity and critical thinking that cannot be developed by relying on AI tools alone. From an accreditation standpoint, it will be helpful to develop Case studies of programs that have successfully integrated AI and went through re-accreditation. Such case studies could offer best practices for other institutions, as AI tools and their implications for access and equity could become issues. Future studies should consider how all students, not just the ones from well-resourced institutions, could correctly benefit from AI-backed education.

AI is reshaping how computing is taught, and its role will only grow in the coming years. The challenge for educators is not whether to adopt AI but how to do so responsibly, ensuring it enriches learning without compromising academic integrity or accreditation standards. With careful integration of AI tools, thoughtful updates to curricula, and consistent ethical guidance, educational institutions can produce graduates who are proficient with modern AI technologies and prepared to navigate future technological changes ethically and thoughtfully. Achieving this balance between embracing innovation and maintaining accountability is essential for effectively reshaping CS and IS education in today's rapidly evolving digital world.

References

[1] E. Eaton and S. L. Epstein, "Artificial Intelligence in the CS2023 Undergraduate Computer Science Curriculum: Rationale and Challenges," *Proc. 38th AAAI Conf. on Artificial Intelligence (AAAI-24)*, AAAI Press, 2024, pp. 23078–23083.

[2] S. Stoykova and N. Shakev, "Artificial Intelligence for Management Information Systems: Opportunities, Challenges, and Future Directions," *Algorithms*, vol. 16, 2023, p. 357. DOI: 10.3390/a16080357.

[3] A. Tlili, "Can artificial intelligence (AI) help in computer science education? A meta-analysis approach," *Revista de Pedagogía*, vol. 82, 2024, pp. 131–154.

[4] C. Van Slyke, R. D. Johnson, and J. Sarabadani, "Generative Artificial Intelligence in Information Systems Education: Challenges, Consequences, and Responses," *Communications of the Association for Information Systems*, vol. 53, 2023, pp. 1–21.

[5] M. Binhammad, A. Othman, L. Abuljadayel, H. Mheiri, M. Alkaabi, and M. Almarri, "Investigating How Generative AI Can Create Personalized Learning Materials Tailored to Individual Student Needs," *Creative Education*, vol. 15, 2024, pp. 1499–1523. DOI: 10.4236/ce.2024.157091.

[6] OpenAI, *ChatGPT*, 2021. Available: https://openai.com.

[7] GitHub, *GitHub Copilot*, 2021. Available: https://copilot.github.com.

[8] S. Jalil, S. Rafi, T. D. LaToza, K. Moran, and W. Lam, "ChatGPT and Software Testing Education: Promises & Perils," *arXiv preprint arXiv:2302.03287*, 2023.

[9] ABET, *Criteria for Accrediting Computing Programs, 2024–2025*, Computing Accreditation Commission, 2024. Available: 2024-2025_CAC_Criteria.pdf

[10] C. Lin, A. Y. Huang, and O. H. Lu, "Artificial intelligence in intelligent tutoring systems toward sustainable education: A systematic review," *Smart Learning Environments*, vol. 10, 2023, pp. 1–22.

[11] P. Wangdi, "Integrating Artificial Intelligence in Education," *Int. J. of Research in STEM Education*, 2024.

[12] C.-H. Lai and C.-Y. Lin, "Analysis of learning behaviors and outcomes for students with different knowledge levels: A case study of intelligent tutoring system for coding and learning (ITS-CAL)," *Applied Sciences*, vol. 15, 2025, Article 1922.

[13] Python Tutor, "Visualize Python, Java, C, C++, JavaScript, TypeScript, and Ruby code execution," *Python Tutor*. Available: https://pythontutor.com/visualize.html#mode=edit. [Accessed: Mar. 6, 2025].

[14] AlgoCademy, "Python Coding Practice with AI," *AlgoCademy*. Available: https://algocademy.com/uses/python-coding-practice-with-ai/. [Accessed: Mar. 6, 2025].

[15] D. López-Fernández and R. Vergaz, "ChatGPT in Computer Science Education: A Case Study on a Database Administration Course," *Applied Sciences*, vol. 15, no. 2, p. 985, 2025. doi: 10.3390/app15020985.

[16] M. Messer, N. C. C. Brown, M. Kölling, and M. Shi, "Automated grading and feedback tools for programming education: A systematic review," *ACM Transactions on Computing Education*, vol. 24, 2024, pp. 1–43. DOI: 10.1145/3636515.

[17] Z. Liu, H. Tseng, and O. H. T. Lu, "The Feasibility of Utilizing ChatGPT in Learning Analytics for the Identification of At-Risk Students," *Proc. CEUR Workshop*, vol. 3667, 2024.

[18] Learning Analytics, "Exploring the Role of GPT as Virtual Teaching Assistants in Online Discussion Forums," *University of Pennsylvania*.

[19] M. Liffiton, B. Sheese, J. Savelka, and P. Denny, "CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes," *arXiv preprint arXiv:2308.0692*, 2023.

[20] D. López-Fernández and R. Vergaz, "ChatGPT in Computer Science Education: A Case Study on a Database Administration Course," *Applied Sciences*, vol. 15, 2025, Article 985. DOI: 10.3390/app15020985.

[21] T. Wang et al., "Exploring the Role of AI Assistants in Computer Science Education: Methods, Implications, and Instructor Perspectives," in *Proceedings of the 2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Washington, DC, USA, 2023, pp. 92-102. doi: 10.1109/VL-HCC57772.2023.00018.

[22] P. Menon, "Exploring GitHub Copilot Assistance for Working with Classes in a Programming Course," *Issues in Information Systems*, vol. 24, 2023, pp. 66–81.

[23] R. Zviel-Girshin, "The Good and Bad of AI Tools in Novice Programming Education," *Education Sciences*, vol. 14, 2024, p. 1089. DOI: 10.3390/educsci14101089.

[24] 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," *Proc. 24th Australasian Computing Education Conf.*, 2022, pp. 10–19.

[25] B. A. Becker, P. Denny, J. Finnie-Ansley, A. Luxton-Reilly, J. Prather, and E. A. Santos, "Programming is hard—or at least it used to be: Educational opportunities and challenges of code generation," *Proc. 54th ACM Technical Symposium on Computer Science Education V. 1*, ACM, New York, NY, USA, 2023, pp. 500–506.

[26] S. Lau and P. J. Guo, "From 'Ban It Till We Understand It' to 'Resistance Is Futile': How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools," *Proc. ACM Int. Computing Education Research Conf. (ICER '23)*, ACM, 2023, pp. 20–31. DOI: 10.1145/3568813.3600138.

[27] S. Feng, A. J. Magana, and D. Kao, "A Systematic Review of Literature on the Effectiveness of Intelligent Tutoring Systems in STEM," *Proc. IEEE Frontiers in Education Conf. (FIE)*, 2021, pp. 1–9. DOI: 10.1109/FIE49875.2021.9637240.

[28] M. Maniktala, M. Chi, and T. Barnes, "Enhancing a student productivity model for adaptive problem-solving assistance," *User Modeling and User-Adapted Interaction*, vol. 33, 2023, pp. 159–188. DOI: 10.1007/s11257-022-09338-7.

[29] B. Yuan and J. Hu, "Generative AI as a Tool for Enhancing Reflective Learning in Students," *arXiv preprint arXiv:2412.02603*, 2024.

[30] G. Nagakalyani, S. Chaudhary, V. Apte, G. Ramakrishnan, and S. Tamilselvam, "Design and Evaluation of an AI-Assisted Grading Tool for Introductory Programming Assignments: An Experience Report," in *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSETS 2025)*, New York, NY, USA, 2025, pp. 805–811. doi: 10.1145/3641554.3701913.

[31] M. Messer, N. C. C. Brown, M. Kölling, and M. Shi, "Automated grading and feedback tools for programming education: A systematic review," *ACM Transactions on Computing Education*, vol. 24, no. 1, pp. 1–43, 2024.

[32] Z. Li and W. Hsu, "Am I Wrong, or Is the Autograder Wrong? Effects of AI Grading Errors on Learning," in *Proceedings of the ACM International Computing Education Research Conference (ICER)*, Chicago, IL, USA, Aug. 2023.

[33] A. Mohammadi and S. Babaei, *Enhancing AI Technology Progression through Industry-Academia Collaborative Partnerships*, Journal of Academy of Strategic Management, 2024.

[34] A. Alzahrani, "A Comparative Study of Machine Learning Algorithms for Predicting Student Performance," in *Proceedings of the International Conference on Artificial Intelligence and Robotics (ICAIR)*, 2025.

[35] D. Satterfield and J. Chang, "Industry and Academia Collaborative Learning: The CSULB and ISSIP AI Collab Pilot Program," in *The Human Side of Service Engineering*, C. Leitner, R. Nägele, C. Bassano, and D.

Satterfield, Eds., AHFE International Conference, vol. 143, AHFE Open Access, USA, 2024. doi: 10.54941/ahfe1005101.

[36] A. Author et al., *Tool Learning with Foundation Models*, ACM Computing Surveys, vol. 56, 2024. Available: https://dl.acm.org/doi/full/10.1145/3704435.

APPENDIX A

Student Outcomes(2024-25): The ABET Computing Accreditation Commission (CAC) has outlined specific Student Outcomes . These are general outcomes common to both CS and IS programs.

Outcome Number	Description
1	Analyze a complex computing problem and apply computing principles and other relevant disciplines to identify solutions.
2	Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.
3	Communicate effectively in a variety of professional contexts.
4	Recognize professional responsibilities and make informed judgments based on legal and ethical principles in computing practice.
5	Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.

Summary of the latest ABET CAC criteria (2024-2025):

Criterion	Title	Description
1	Students	Student performance must be evaluated. Student progress must be monitored to foster success in attaining student outcomes, thereby enabling graduates to attain program educational objectives. Students must be advised regarding curriculum and career matters. The program must have and enforce policies for accepting both new and transfer students, awarding appropriate academic credit for courses taken at other institutions, and awarding appropriate academic credit for work in lieu of courses taken at the institution. The program must have and enforce procedures to ensure and document that students who graduate meet all graduation requirements.
2	Program Educational Objectives	The program must have published program educational objectives that are consistent with the mission of the institution, the needs of the program's various constituencies, and these criteria. There must be a documented, systematically utilized, and effective process, involving program constituencies, for the periodic review of these program educational objectives that ensures they remain consistent with the institutional mission, the program's constituents' needs, and these criteria.
3	Student Outcomes	Programs must define and document outcomes, ensuring students can analyze complex problems, design computing-based solutions, communicate effectively, recognize ethical responsibilities, and function in teams. See Appendix A for a list of general student outcomes common to CS and IS programs.
4	Continuous Improvement	The program must regularly use appropriate, documented processes for assessing and evaluating the extent to which the student outcomes are being attained. The results of these evaluations must be systematically utilized as input for the program's continuous improvement actions. Other available information may also be used to assist in the continuous improvement of the program.
5	Curriculum	The program's requirements must be consistent with its program educational objectives and designed in such a way that each of the student outcomes can be attained. The curriculum must combine technical, professional, and general education components to prepare students for a career, further study, and lifelong professional development in the computing discipline associated with the program. Must offer a comprehensive curriculum combining technical, professional, and general education with at least 30 credits of computing topics covering both fundamentals and advanced content. The computing topics must include: 1. Techniques, skills, and tools necessary for computing practice. 2. Principles and practices of security and privacy in computing. 3. Local and global impacts of computing solutions on individuals, organizations, and society
6	Faculty	Faculty members must have qualifications and expertise aligned with the program's requirements, maintain professional competence, and be sufficient in number to ensure continuity and effective instruction.
7	Facilities	Facilities must include adequate, maintained classrooms, labs, offices, and modern computing resources to effectively support student learning and faculty activities.

These criteria are detailed in the ABET 2024-2025 Criteria for Accrediting Computing Programs document. For more details on each criteria refer to the ABET CAC documentation: 2024-2025_CAC_Criteria.pdf

TRAINING A NEURAL NETWORK TO KICK A BALL IN ROCKET LEAGUE

Dr. Girard
cdgira@ship.edu
Shippensburg University
Achraf Jamjama
simoachk@gmail.com

ABSTRACT
Rocket League is a soccer-like game played with cars. A few years ago, a framework was released to allow for the development of bots. Using this framework, neural networks were trained to kick a ball in different directions from different starting locations. This paper looks at how temporal information affected the accuracy of the training. Initial results showed no gain in accuracy.

KEY WORDS
Neural Network, Path Following, Rocket League

1 Introduction

Neural networks learn by adjusting weights so that they correctly predict the output from a given input. The design of the neural network has four key aspects: how many nodes in each layer, how the nodes are connected, how many layers, and how values are computed. As such there are many kinds of neural networks, each designed for the problem it seeks to solve. At the core a neural network tries to find patterns, so they perform best when inputs do not generate conflicting outputs. One example of a neural network is shown in figure 1 [3,7,8,10,11,15].

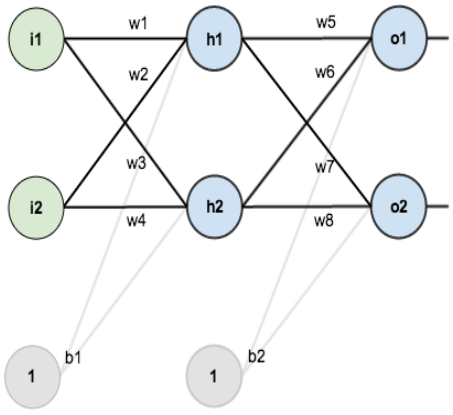


Figure 1: Connected Neural Network [11]

2 Background

In the game of Rocket League, see Figure 2, one of the primary goals of the game is to hit a large ball in a specific direction with your car. When a ball is struck, what direction the ball goes and how far it goes depends on the angle the car hits the ball from and the speed of the car. Key information available at a point in time to help with this is: location of the ball, location of car, direction of car, and speed of car. For more complex scenarios additional information would be: speed of ball, direction of ball, location of other car, etc. For this work it was assumed the ball was stationary and no other cars on the field [1].

Figure 2: Rocket League [13]

2.1 RLBot

RLBot is a framework for creating bots able to play in Rocket League using a wide range of languages. The framework provides a tool called RLBotGUI for easily creating a base bot in Python, Rust, or Scratch. Additionally, with a bit more work it is possible to build bots in Java, C#, C++, and JavaScript. This work used Python as the development language [1].

Through the bot framework it is possible to access a wide range of information using a “Game Tick Packet” or GTP for short. The GTP provides information on the ball, each car, etc… Location information is stored as three floats representing the x, y, and z position. Velocity is also stored

as three floats representing speed in the x, y and z directions. In Rocket League the z axis represents the vertical axis. Additionally, commands are sent back using a "ControllerState" or CS for short. Some key commands are throttle and steer, see table 2 for a more complete list [2]. Based on the data from experiments here, a GTP was issued, and a CS could be issued about 27 times a second.

Command	Type	Description
throttle	float	-1 for full reverse, 1 for full forward
steer	float	-1 for full left, 1 for full right
pitch	float	-1 for nose down, 1 for nose up
yaw	float	-1 for full left, 1 for full right
roll	float	-1 for roll left, 1 for roll right
jump	bool	true/false press the jump button
boost	bool	true/false press the boost button
handbrake	bool	true/false press the handbrake button
use_item	bool	true if you want to use a rumble item

Table 1: ControllerState Options [2]

2.2 RLBot Training

While RLBotGUI provides tools to create a bot and deploy it in a game, it does not provide a way to train a bot. For this research RLBotTraining is used to gather data for training a neural network and to evaluate the results of said training. This framework provides support to setup one or more scenarios in Rocket League with one or more bots and a fixed time limit. It is then possible to gather data either from the bots directly or from RLBotTraining. However, one shortcoming of this framework is there is only limited communication between the training framework and the bots in the scenario [12].

To create a scenario in RLBotTrainer one or more exercises are created. A list is then created of each exercise and that list is played out in order by RLBotTrainer in an infinite loop until stopped. Each exercise can be tweaked using input parameters. For example, an exercise to kick a ball can be tweaked such that the car or ball start in different locations. Additionally, RLBotTrainer allows for graders to be designed to determine if the bot succeeded or not. The graders can check for things such as: was a goal scored or if the car travelled a certain distance. The grader also determines how long each exercise is allowed to run before automatically failing [12].

2.3 Time Dependent Neural Networks

Neural networks take an input and produce an output. In some cases the correct output is influenced not just by the present state, but also a previous state or states. For problems of this type there are multiple approaches to implementing a time dependent neural network. One approach is the use of a long-short term memory network, or LSTM. Another approach is by simply adding additional input nodes to represent multiple time slices.

Long-Short Term Memory (LSTM) networks utilize special memory nodes as shown in figure 3. These nodes compute their output by combining the present input (E) with what they have seen recently (STM) and over a long period of time (LTM). Key to their approach is large changes in what is occurring (e.g. car completely changes direction) will cause the LTM to reset and start over. Thus the LSTM takes the approach that something completely different has started and as such the previous long-term past is no longer of use [9].

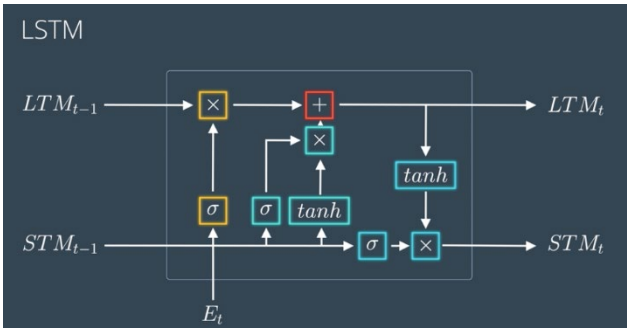


Figure 3: LSTM Memory Node [9]

Encoding the time slices in the input layer allows the use of a basic feed forward neural network. It will function similar to an LSTM, but with some downsides. First, it takes a lot of input nodes to allow it to follow long term trends. As such it tends to behave more like an LSTM without the LTM part, so more of just a STM. As such, it doesn't track long term trends well, nor does it have the built-in structure to realize when sudden shifts occur. The upside is it can be implemented with a basic feed forward neural network without any changes.

The work here only uses additional input nodes to provide information about previous time periods. Use of an LSTM was not tested. See section 6 for how this approach might be utilized in later work.

2.4 Representing Location

Representing location in a neural network can be a challenge. Due to how neural networks respond to changes it is not possible to simply encode the coordinate locations directly [14]. The approach here uses direction to and distance of objects to convey location information to the neural network. Using this approach will allow for the environment to be represented with a small number of nodes, while hopefully producing accurate results [5,14,16].

3 Experiment Design

Each neural network will learn how to "kick" a ball a specified direction in Rocket League. To improve learning the car will have different starting points.

The input to the neural network is broken up into three approaches: Just the immediate information, immediate plus previous, and immediate plus four-steps ago previous. For all three approaches the output layer will be the same.

3.1 Hypotheses

Because movement is usually about changes over time, the assumption is temporal information will help the neural network perform better. So, the hypotheses are:

H1: The four-step approach will do better than immediate or immediate and previous.

H2: The immediate and previous approach will do better than immediate by itself.

3.2 Scenario Setup

The ball always starts at roughly the 0,0 location. Because of limits of RLBotTrainer, to inform the bot of which run was taking place the ball's starting x position ranged from 0.1 to 1.5 and starting y from 1 to 3. As the ball has a diameter of 184, this small variance in starting position should have a negligible effect on learning performance.

The car starts in three locations shown in table 2. These start locations were chosen by drawing a circle of radius 2000 around the ball. Then the start, end, and middle points for one quarter of that circle were selected. The dimensions of the arena used were: -4,100 to 4,100 along the X-Axis and -5,100 to 5,100 along the Y-Axis. The Z-Axis was ignored as the car stayed on the ground. The car always drove with throttle set to 1.

	X Coordinate	Y Coordinate
Position 1	0	2000
Position 2	765	1847
Position 3	1414	1414

Table 2: Starting Points for the Car

To generate training data, from each start point, the car was told to drive to one of the three first locations shown in table 3. Once the car reached the specified location it would then steer directly to the ball. Once the ball was hit the direction the ball went was recorded. This produced 9 different training scenarios.

	X Coordinate	Y Coordinate
Destination 1	0	1250
Destination 2	478	1154
Destination 3	883	883
Destination 4	625	1082
Destination 5	323	1207

Table 3: Destination Points for the Car

To evaluate of how well the neural networks generalized the training, six additional scenarios were created from the same three starting points. From each starting point the car would instead first head to one of the last two destinations in table 3. As before the direction the ball went was recorded.

3.3 Neural Network Setup

The immediate only method has a total of 8 input nodes: 2 for direction to the ball, 1 for distance to the ball, 1 for direction car is travelling, 2 for speed car is traveling in the x,y plane, and 2 for direction to kick the ball. The other two methods added 6 additional nodes that included all the previous information except the direction to kick the ball, as that stayed constant. In all cases there was only one output node that represented what to set the steer value to for the ControllerState.

After some test runs the number of hidden nodes for all methods was set to 20. For the activation function a ReLU type approach was used, see equation 1.

$$(1)\ ACT(x) = \begin{cases} \max(2, 1 + (x - 1) * 0.01 & \text{if } x > 1 \\ x\ if\ -1 \le x \le 1 & \\ \min(-2, -1 + (x + 1) * 0.01 & \text{if } x < 1 \end{cases}$$

The basic ReLU activation function converts any value less than 0 to 0, and leaves unchanged any value greater than 0. However, having a hard cutoff can have unintended effects. A traditional leaky ReLU allows for slow growth in values when the input is less than 0 [4,6]. However, this still allows growth to infinity which created unwanted side effects in testing. As such the neural network uses the approach shown in equation 1. This allowed for both positive and negative values, no immediate hard cutoff, but also no growth to infinity [5].

3.4 Neural Network Training

Initial values for weights ranged from -1 to 1. The learning rate started at 0.05 and then gradually reduced until it reached 0.0099. A neural network was trained on the 9 scenarios until 200 training attempts had occurred without additional improvement. The number of training iterations ranged from 344 to 2214. Each training iteration involved showing the neural network one of the nine training scenarios. Each training scenario consisted, on average, of 76 items. Where each item was the state of the game and what action the car should take next.

After each training iteration the neural network's level of error was computed using all nine scenarios. During each training pass the neural network would train on the nine scenarios in random order, plus the scenarios it performed the worst on the previous pass. This allowed for scenarios with more complex driving to get more training time. The neural network during training with the lowest error was saved. For each setup being tested 10 neural networks were

trained for evaluation, creating a total of 30 neural networks.

For evaluation each trained neural network was measured on the original 9 scenarios and the 6 additional scenarios detailed in section 3.2. The absolute error in both the x direction and y direction was recorded or if the car completely failed to hit the ball.

Because of fails two levels of comparison were required. First, for each network the number of fails was recorded. These fails were averaged across all 10 neural networks for a given setup. The other level of comparison involved first averaging for each scenario the error in the x and y direction. Then all of these were averaged together to give an overall error for a given setup. This approach was taken as due to fails, not all networks would have data for all scenarios. This ensured each scenario was given equal weight in the final computation of overall error.

4 Solution Description

The Python programming language was utilized to implement a neural network with one hidden layer and to create the bot for Rocket League. It did not utilize a bias and used the specialized ReLU discussed in section 3.3 as the activation function. RLBotTraining was used to craft and run all the scenarios. Because of the design of the framework each network had to be run individually, rather than one large batch. Data was recorded in csv format and then transferred to an Excel spreadsheet for final analysis.

	15 Scenarios
Immediate	X
Plus Previous	X
Four-Step	X

Table 4: Block Design

In Table 4, Immediate stands for only using the immediate input approach, Plus Previous for using the immediate plus previous data approach, and Four-Step for using the immediate plus four-steps before. The accuracy of the network is measured based on how far off the resulting ball direction unit vector was from the correct value.

Training a single neural network took approximately one minute, though this varied greatly depending on how many iterations it trained. To gather evaluation data, it took about 3 minutes for each neural network. Each scenario had a cutoff time of 4.5 seconds. This time was selected based on ensuring the simulation stopped before a kicked ball would hit something. As hitting something would change the appearance of what direction the ball headed.

5 Results

Table 5 shows the average error for Immediate, Plus Previous, and Four-Step from running the 15 scenarios. As mentioned, fails are not included in the error computed here. From the standard deviation it can be seen there is a fairly wide range of accuracy across the neural networks.

15 Scenarios	Avg Error Rise (Y)	Std Dev Rise (Y)	Avg Error Run (X)	Std Dev Run (X)
Immediate	0.0986	0.111	0.0964	0.0873
Plus Previous	0.0915	0.0712	0.113	0.0773
Four-Step	0.0949	0.0681	0.125	0.0528

Table 5: Accuracy Results – 15 Scenarios

Table 6 shows the results of applying a paired t-test to compare the averages in table 5. Based on the T-scores we can say with confidence that none of the approaches did any better than the others.

Approach A	Approach B	Rise (Y)	Run (X)
Immediate	Plus Previous	0.160	0.436
Immediate	Four-Step	0.0845	0.837
Plus Previous	Four-Step	0.103	0.370

Table 6: T-test Results – 15 Scenarios

Table 7 shows the average number of fails per approach.

Fails	Avg Fails	Std Dev Fails
Immediate	1.5	1.65
Plus Previous	0.8	1.23
Four-Step	1.1	1.29

Table 7: Accuracy Results – Fails

Table 8 shows the results of running a paired t-test to compare the average number of fails. For the fails we see the same result as from the 15 scenario error differences, in that there is no statistical difference between the results. As additional support for this, all three approaches had five neural networks that recorded failures and five that did not.

Approach A	Approach B	Fails
Immediate	Plus Previous	1.02
Immediate	Four-Step	0.574
Plus Previous	Four-Step	0.506

Table 8: T-test Results – Fails

Some other notes from the results, the Plus Previous approach had the best overall neural network with zero fails and an x error of 0.0756 and y error of 0.0499. The

best Immediate neural network had zero fails and an x error of 0.0892 and y error of 0.0565. Lastly, the best Four-Step neural network had zero fails and an x error of 0.104 and a y error of 0.0734.

6 Conclusion

First, each approach was able to train at least one neural network that could effectively kick a ball the approximate direction requested. Second, none of the hypotheses were supported. However, from watching the training there are obviously challenges to the neural networks learning the correct paths, likely because some paths were very similar at times, but ended with the ball heading in different directions.

Future work, need to find which similar paths were causing the most conflict and determine how to model them as different to the neural network. Related to that, investigate if an LSTM is able to overcome the shortfalls this simplified time slice approach took. Also, it could be the data used to train the neural networks had design flaws that need to be addressed. Lastly, once this problem is solved the goal would be to train the car to strike a moving ball to go in a specified direction.

References:

[1] RLBot. Dec 15th, 2024, from https://rlbot.org/

[2] Redd, etc al. Input and Output Data. RLBot GitHub, Nov 25th 2022. Input and Output Data · RLBot/RLBotPythonExample Wiki · GitHub

[3] Perry, Steven J., Create an artificial neural network using the Neuroph Java framework, *IBM Developer*. Jan 8, 2018. [Online]. Available: https://developer.ibm.com/tutorials/cc-artificial-neural-networks-neuroph-machine-learning

[4] P. Baheti, Activation Functions in Neural Networks [12 Types & Use Cases], v7 Labs, Feb. 02, 2023. https://www.v7labs.com/blog/neural-networks-activation-functions

[5] Girard, C. Dudley, How to Position vs Direction Affects Learning to Pass a Ball, 39th Annual Conference of PACISE, 2024.

[6] V. Gupta, Understanding Feedforward Neural Networks, *LearnOpenCV*, 20-Apr-2021. [Online]. Available: https://learnopencv.com/understanding-feedforward-neural-networks/

[7] M. Ibrahim, M. Louie, C. Modarres, and J. Paisley, Global Explanations of Neural Networks, *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, 2019.

[8] D. M. F. Izidio, A. P. D. A. Ferreira, and E. N. D. S. Barros, Towards better generalization in WLAN positioning systems with genetic algorithms and neural networks, *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019.

[9] Moghar, Adil and Mhamed Hamiche, Stock Market Prediction Using LSTM Recurrent Neural Network, Procedia Computer Science, Vol. 170, 2020, pp 1168-1173.

[10] Tesauro, Gerald, Temporal Difference Learning and TD-Gammon, Communications of the ACM, Vol. 38 No. 3, March 1995.

[11] Mazur, A Step by Step Backpropagation Example, Matt Mazur, 15-Feb-2022. [Online]. Available: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

[12] RLBot, RLBotTraining, Youtube, 2019. RLBotTraining Tutorial - YouTube

[13] Evample, Rocket League Highlights – 77, Youtube, Oct 9th, 2023. 2024 TREMFYA NOW APPROVED BUMPER

[14] Lewis, Joshua and C. Dudley Girard, Position Information with Neural Networks, *38th Annual Conference of PACISE*, 2023.

[15] Janji, Salim and Adrian Kliks, Neural Networks for Path Planning, arxiv.org, 2022. [2207.00874] Neural Networks for Path Planning

[16] Gomes, Gilzamir, Creto A. Vidal, Joaquim B. Cavalcante-Neto, and Yuri LB Nogueira, Two level control of non-player characters for navigation in 3d games scenes: A deep reinforcement learning approach, 20th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), pp. 182-190. IEEE, 2021

FROM KNUTT'S AXIOMS TO THE ART GALLERY PROBLEM

Zhongxiu Yang
Millersville University of Pennsylvania
zhongxiu.yang@millersville.edu

ABSTRACT

Art gallery problem is a well-studied visibility problem in computational geometry, one variant can be placing the minimum number of guards that can observe the entire simple polygon, where the points can be placed anywhere inside the polygon. In this paper, we analyze this problem starting from Knutt's axioms and show structural lemmas that can be used to prove the VC-dimension of visibility on the simple polygon.

KEY WORDS

Art Gallery Problem · VC-Dimension · Visibility

1 Introduction

The set-cover problem is a classic NP-Hard problem. Given a set U of n elements and a collection $S = \{S_1, S_2, S_3, \ldots, S_m\}$ of m subsets of U such that $\bigcup_i S_i = U$, the set-cover problem is to select as few subsets as possible from S such that their union covers U. It does not have a polynomial-time solution available, and the greedy algorithm provides a $\Theta(logn)$ approximation. The interesting thing is that, when it comes to geometry, how much can we do it better?

1.1 Problem Statement

A simple polygon P is defined by a set of points $V = \{p_1, p_2, \ldots, p_n\}$, and the connecting edges $E = \{\overline{p_1p_2}, \overline{p_2p_3}, \ldots, \overline{p_{n-1}p_n}, \overline{p_np_1}\}$ that do not intersect each other. We define two points p and q "see" each other if the line segment $\overline{pq}$ fully inside polygon, and one point p "sees" a line segment $\overline{qr}$ if p sees q and p sees r. See Figure 1 for the illustration.

We also define (p, q) to be a sequence of points on the boundary of the polygon from p to q in counterclockwise order excluding p and q, and $[p, q]$ to be the same sequence of points but including p and q. And we also say that a point $\overline{pq}$ is pierced if $\overline{pq}$ leaves the polygon, that is, p sees q if and only if $\overline{pq}$ is not pierced. If every line segment at the boundary of the polygon was seen by some point, then the simple polygon is "covered" by such a set of points. One variant of the art gallery problem is to place the minimum number of "guard" points G such that the entire polygon P is covered by some guard $g \in G$, the basic version of this problem is called vertex guarding, where guards only can be placed on the boundary of the polygon; we are interested in another version of this problem, which is called point guarding, which means that the guards can be placed anywhere inside the polygon. There are many applications based on the geometric model of art gallery problems, such as placing surveillance cameras or indoor motion detectors.

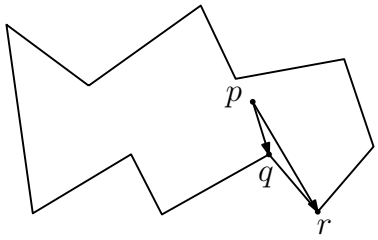


Figure 1: Given $\overline{qr}$ on the boundary of polygon, p sees q and p sees r implies p sees qr

1.2 VC-Dimension

VC-Dimension is an important measure of the complexity of the set system. We say that a guard set G is shattered if every subset of G can be seen by some "viewpoint" $v \in V$. The VC-Dimension is the largest d such that there exists a simple polygon P and a guard set G of size d can be shattered. The motivation of finding VC-Dimension is that Brönnimann and Goodrich [2] give a polynomial-time $O(logOPT)$ approximation algorithm for any set system with constant VC-Dimension, where OPT is the size of an optimal cover. Clearly, the VC-Dimension for the classic set-cover problem is infinity, and we hope to find a constant VC-Dimension for this problem.

1.3 Order Claim

One of the key properties about the visibility of simple polygons is the order claim.

Claim 1. Let p, q, r, s, t, u be six points on the boundary of a simple polygon in counterclockwise order, if (1) p sees r, (2) s sees q, (3) p sees t, and (4) s sees u, then p sees s.

See Figure 2 for the illustration. Without loss of generality, let $y(p) = y(s)$. Since p sees r and $y(r) < y(p)$, then there is no point in (p, r) can be placed above $\overline{pr}$, similarly, there

is no point in (q, s) can be placed above $\overline{qs}$. Since p sees t and s sees u we know that there is no point in (t, p) and (s, u) can be placed below $\overline{tp}$ and $\overline{su}$ respectively. That implies that there is no point in (p, s) that can be placed above $\overline{ps}$ and there is no point in (s, p) that can be placed below $\overline{ps}$, therefore p sees s.

For the vertex guarding version of the problem, we can see

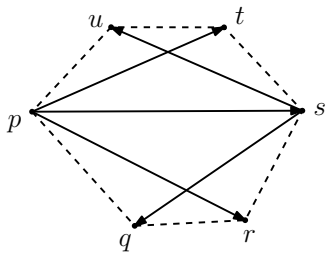


Figure 2: Order Claim

that the order claim is very useful since we can discretize the points by sorting the guards and viewpoints along the boundary in counterclockwise order, then we can potentially reject some cases where the configuration of the points violates the order claim. But when it comes to the point guarding version of the problem, it would be difficult to discretize the points since every point can be placed anywhere inside the polygon, so we need a better approach to categorize every placement of the points in terms of visibility.

1.4 Knutt's Axioms

One such way to obtain such a discrete set of points is through the use of counterclockwise/clockwise systems as described by Knuth [1]. Let p, q, r be three points in the polygon, we say that pqr is a triple such that pqr is $true$ (or T) if p, q, r are traversed in counterclockwise order; otherwise, we say that pqr is $false$ (or F). See Figure 3 for the illustration. The following five axioms are properties that a set of triples has to satisfy in order to be placed on a 2D plane.

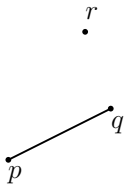


Figure 3: r is on the counterclockwise side of $\overline{pq}$, thus $pqr = true$

Axiom 1. $pqr \implies qrp$
Axiom 2. $pqr \implies \neg prq$
Axiom 3. $pqr \vee prq$
Axiom 4. $tqr \wedge ptr \wedge pqt \implies pqr$
Axiom 5. $tsp \wedge tsq \wedge tsr \wedge tpq \wedge tqr \implies tpr$

2 Structural Lemmas

The following lemmas are the main contribution to the problem, they show how to discretize every placement of points to find the upper bound of the VC-Dimension.

Lemma 2.1. Let four points p, q, r, t in the 2D plane, if $pqr = true$, then pqt, qrt and rpt cannot all be $false$; and if $pqr = false$, then pqt, qrt and rpt cannot all be $true$.

Proof. See Figure 4 for the illustration. Let $pqr = true$, then the line $\overleftrightarrow{pq}$, $\overleftrightarrow{qr}$, and $\overleftrightarrow{rp}$ divide the entire 2D plane into eight regions, and the values of the tuple (pqt, qrt, rpt) can be (T,T,T), (T,T,F), (T,F,T), (T,F,F), (F,T,T), (F,T,F), (F,F,T). Similarly, if $pqr = false$, then there is no (T,T,T) region on the plane. □

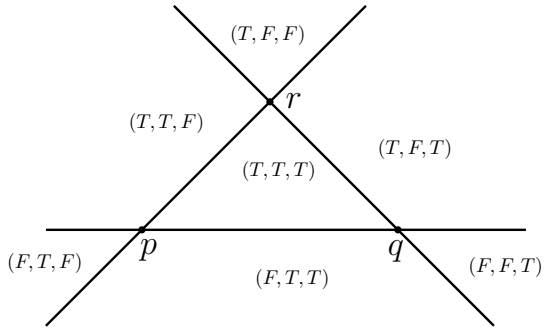


Figure 4: $pqr = false$ implies there is no (F, F, F) region on the plane

Lemma 2.2. Let four points a, b, c, d in the 2D plane and $\overline{ab}$ intersect $\overline{cd}$. If $abc = true$ and $abd = false$, then placing a point e such that $(abe, cde, bde) = (T, F, T)$ is infeasible.

Proof. See Figure 5 for the illustration. Since $\overline{ab}$ intersect $\overline{cd}$, $abc = true$ and $abd = false$, we have $cda = false$ and $cdb = true$, otherwise $\overline{cd}$ does not intersect $\overline{ab}$. Consider the placement on point e. Essentially, $\overleftrightarrow{ab}$ and $\overleftrightarrow{cd}$ divide the entire 2D plane into four regions, and e must be in one of the regions. Let us say e is in the "northwest" region as shown without loss of generality, and name the intersection point of $\overline{ab}$ and $\overline{cd}$ to be o, then the entire plane angle $\angle aoc$ is on the clockwise side of $\overline{bd}$. Since e is in $\angle aoc$, then bde cannot be $true$, as well as bda and bdc. □

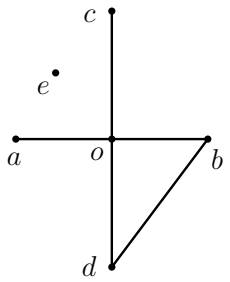


Figure 5: bde must be $false$ since e is in $\angle aoc$

Lemma 2.3. Let five points a, b, c, d, e in the 2D plane, let $abc = true$, $abd = true$, $dca = false$, $dcb = true$, and $abe = true$. If $dce = true$, then $ace = true$ and $ade =$

$true$. Symmetrically, if $dce = false$, then $bce = true$ and $bde = true$.

Proof. See Figure 6 for the illustration. Since $abc = true$ and $abd = true$, both c and d are on the counterclockwise side of $\overline{ab}$. Since $dca = false$, $dcb = true$, then a is on the clockwise side of $\overrightarrow{dc}$ and b is on the counterclockwise side of $\overrightarrow{dc}$. Since $abe = true$, e has to be in the third or fourth "quadrant" defined by $\overrightarrow{ab}$ and $\overrightarrow{cd}$. Clearly, any point in the fourth "quadrant" is on the clockwise side of $\overline{ac}$ and $\overline{ad}$. That implies if $dce = true$, then $ace = true$ and $ade = true$. Symmetrically, $dce = false$ implies that the point e is on the clockwise side of $\overline{dc}$, so $bce = true$ and $bde = true$. □

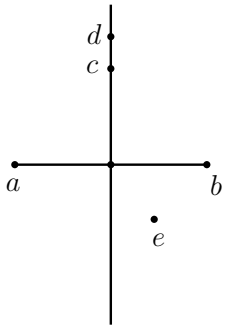


Figure 6: e is on the clockwise side of $\overline{ac}$ and $\overline{ad}$

Lemma 2.4. Let four points a, b, c, d in the 2D plane and $\overleftrightarrow{ab}$ intersect with $\overleftrightarrow{cd}$. Let $abc = false$, $abd = false$, $dca = false$, $dcb = false$. If there exists a point e such that $abe = true$ and $dce = true$, then if $ace = false$, then every point f such that $abf = true$ and $dcf = true$ must satisfy $acf = false$ and $bdf = false$. Symmetrically, if $ace = true$, every point f such that $abf = true$ and $dcf = true$ must satisfy $acf = true$ and $bdf = true$.

Proof. Since $abc = false$ and $abd = false$, both c and d must be on the same clockwise side of $\overline{ab}$. Then, $dca = false$ and $dcb = false$ guarantee that the intersection point is not between $\overline{ab}$ or $\overline{cd}$. $abe = true$ and $dce = true$ implies that there will be two cases in terms of the relative position of e, see Figure 7 and Figure 8 for illustration, but both cases cannot occur at once since two lines intersect at most once. That implies if $ace = false$, then e is on the clockwise side of $\overline{ac}$ and $\overline{bd}$, then any point f that satisfies $abf = true$ and $dcf = true$ must be in the same wedge where e is, and vice versa. □

Lemma 2.5. Suppose p_1, p_2, p_3 are three points inside polygon P and they can see each other. Then any pair of points in $\triangle p_1p_2p_3$ must see each other.

Proof. Given two points p, q, we know that p does not see q if and only if $\overline{pq}$ is not fully inside the polygon. See Figure 9 for the illustration. Suppose p does not see q, then one of the edges of $\overline{p_1p_2}$, $\overline{p_2p_3}$, $\overline{p_1p_3}$ must not be fully inside of the polygon, therefore p sees q. We can extend this lemma by finding a triangle form with intersections, and note that the enclosed polygon must be a triangle, see Figures 10, 11, 12 for the illustration. □

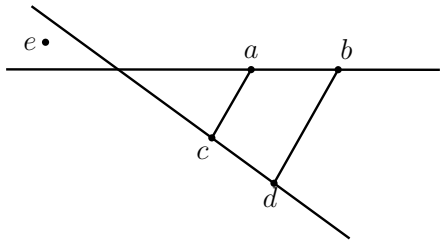


Figure 7: e is to the left, then all other points that are over $\overline{ab}$ and below $\overline{cd}$ also must be to the left

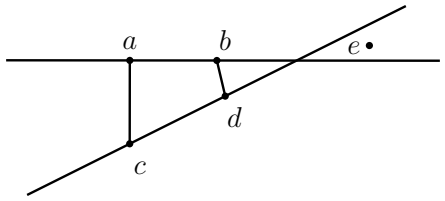


Figure 8: Symmetric picture

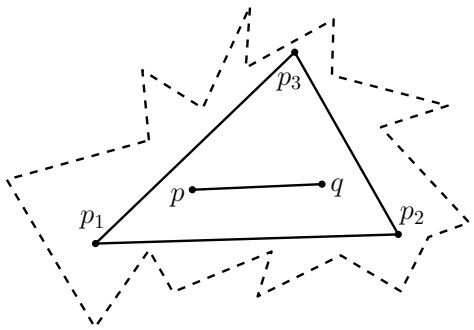


Figure 9: p must see q

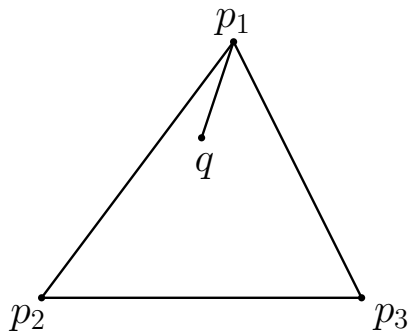


Figure 10: If $p_1p_2q = true$, $p_2p_3q = true$, $p_3p_1q = true$ and $p1$, p_2, p_3 see each other, then p_1 sees q

Lemma 2.6. Let $(p_{m-1}, p_m, p_{m+1}, \dots, p_n, \dots)$ is an enclosed polygon bounded in counterclockwise order. If $\overline{p_mp_n}$

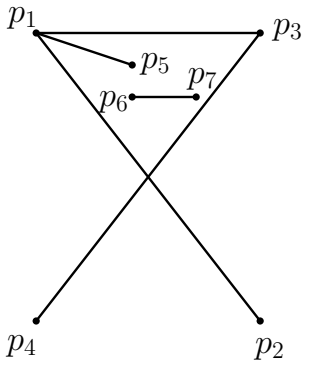


Figure 11: If p_1 sees $p2$, p_3 sees p_4, and $\overline{p_1p_2}$ intersects $\overline{p_3p_4}$, for any point p_5, p_6, p_7 counterclockwise to $\overline{p_1p_2}$ and clockwise to $\overline{p_3p_4}$ and $\overline{p_1p_3}$, then p_1 must see p_5, and p_6 must see p_7

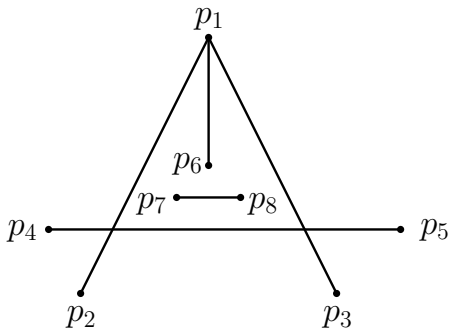


Figure 12: If p_1 sees p_2 and p_3, and p_4 sees p_5, and if $\overline{p_4p_5}$ intersects $\overline{p_1p_2}$ and $\overline{p_1p_3}$, then for any point p_6, p_7, p_8 counterclockwise to $\overline{p_1p_2}$ and $\overline{p_4p_5}$ and clockwise to $\overline{p_1p_3}$, p_1 must see p_6, and p_7 must see p_8

is fully outside this region and never intersects with any line segment in this region, then $p_{m-1}p_mp_{m+1} = true$ and $p_mp_np_{m+1} = true$.

Proof. See Figure 13 for the illustration. Without loss of generality, let $y(p_m) = y(p_n)$ and $y(p_{m-1}) > y(p_m)$. Since the entire region was bounded in counterclockwise order, every point in (p_m, p_n) must be above $\overline{p_mp_n}$, otherwise $\overline{p_mp_n}$ is not fully outside of the region, which implies $p_mp_np_{m+1} = true$. Since $y(p_{m-1}) > y(p_m)$ and the region is bounded in counterclockwise order, we have $p_{m-1}p_mp_{m+1} = true$. □

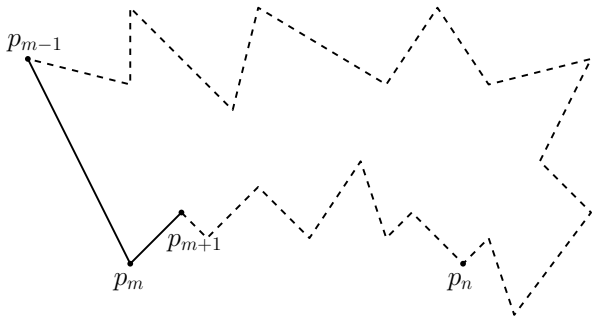


Figure 13: $p_{m-1}p_mp_{m+1} = true$ and $p_mp_np_{m+1} = true$

Lemma 2.7. If $(p, q, q_1, q_2, \ldots, q_m)$ forms an enclosed polygon in counterclockwise order and $(q, p, p_1, p_2, \ldots, p_n)$ forms an enclosed polygon in counterclockwise order, then p sees q.

Proof. This lemma is an extension of Claim 1. See Figure 14 for the illustration. Without loss of generality, let $y(p) = y(q)$, and $\overline{pq}$ cannot be pierced above and below by any points on the boundary of an enclosed polygon. □

The main difference of Lemma 2.7 is that we can determine two enclosed polygons by the truth value of triples, and Figures 15, 16 show the extension cases from which we can form the enclosed polygon.

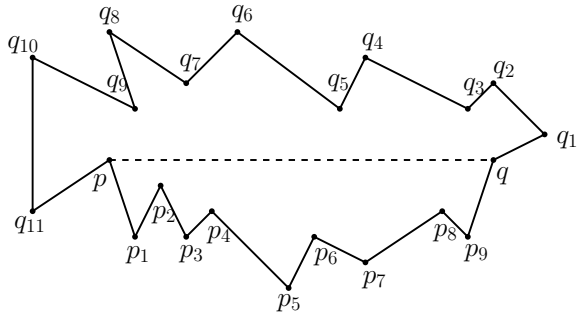


Figure 14: p must see q

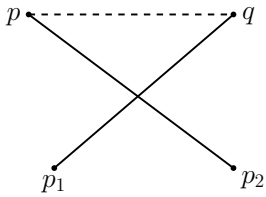


Figure 15: If $\overline{pp_2}$ and $\overline{p_1q}$ intersects, and if $p1$ and p_2 both on the clockwise side of $\overline{pq}$, then there is an enclosed polygon on the clockwise side of $\overline{pq}$

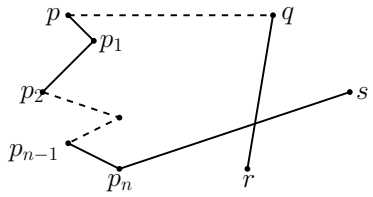


Figure 16: If $(p, p_1, p_2, \ldots, p_n)$ is a sequence of points in counterclockwise order, and every point is on the clockwise side of $\overline{pq}$, and if $\overline{p_ns}$ intersects $\overline{qr}$, then there is an enclosed polygon on the clockwise side of $\overline{pq}$

3 Procedure

We would like to find the VC-Dimension of point guarding simple polygon from shattering one guard. Figures 17 and 18 show the realization of shattering one guard and two guards,

which indicate that the VC-Dimension is at least that many. Starting from a three-guard configuration, we introduce the counterclockwise/clockwise system for the guard set such that, for every triple, we want to consider every combination of the truth value for triples. In other words, given any set of points in the 2D plane, there is one case that can describe the placement of those points.

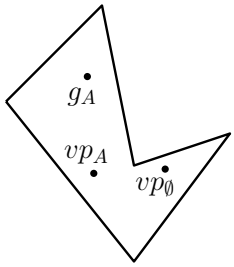


Figure 17: Realization of shattering one guard. $G = \{g_A\}$, $V = \{vp_\emptyset, vp_A\}$

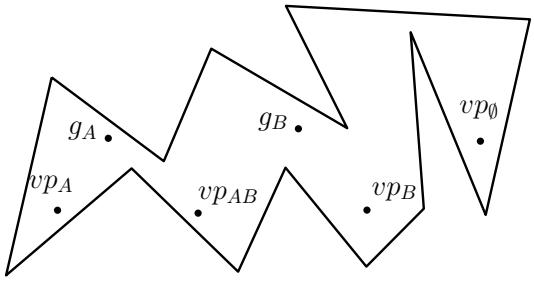


Figure 18: Realization of shattering two guards. $G = \{g_A, g_B\}$, $V = \{vp_\emptyset, vp_A, vp_B, vp_{AB}\}$

3.1 Removal of Redundancy

Suppose we have n points on the 2D plane, if we brute force the truth value in terms of counterclockwise or clockwise for every triple, then we will have $2^{\binom{n}{3}}$ cases, and adding one more point will increase the number of cases exponentially although most of cases can be immediately rejected by the Knutt's axioms without using our structural lemmas, still the number of cases is unbounded as number of guards increases.

To do that, we define a convex hull decomposition to produce an ordering for a set of points. Given a set of points that can be placed anywhere inside a simple polygon, we arbitrarily find a starting point a where a is any point on the convex hull of those points. We label the next point as b, where all other points are counterclockwise to $\overline{ab}$ and keep labeling until every point on the convex hull is labeled. Then we remove from consideration every point on the convex hull except for a. If a is the only point left, then we are done, but if there is at least one point left, we continue to label the points from the convex hull of the remaining points in counterclockwise order starting from a. See Figures 19 and 20 for the illustration. We continue until all points have received their spot in the order. This allows us to find all orderings that satisfy Knuth's axioms without brute force.

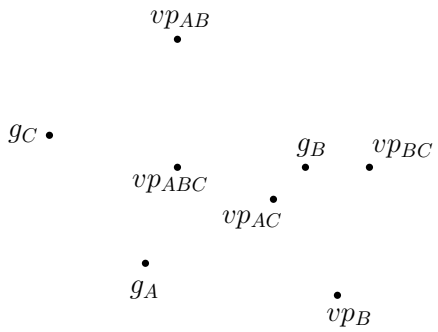


Figure 19: Original Case

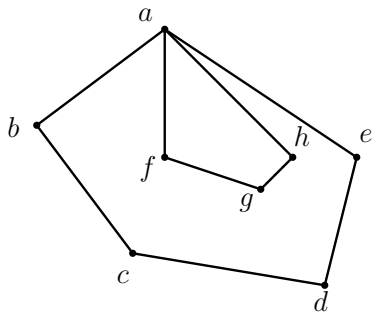


Figure 20: Same case after convex hull decomposition

3.2 Algorithms

We first generate every combination of the truth value of triples after removing the redundancy, for the $(n + m)$ point system that includes n guards and m distinct viewpoints. Clearly, if $m = 2^n$, then the VC-Dimension is at least n and we should consider if $(n + 1)$ guards can be shattered.

If we did not detect any violation on our lemmas as shown in Algorithm 1, then we will add a new viewpoint in this ordering, consider every assignment of truth value with respect to this viewpoint, and recompute the decomposition of the convex hull for the entire set of points to remove redundancy. Note that a different assignment of the truth value could lead this viewpoint to a different position, and the label of this viewpoint and all other points could be different. If we find any lemma violation for this ordering, then we can terminate this subcase, and all extensions of this ordering is not feasible.

4 Conclusion

In conclusion, we show structural lemmas to find the VC-Dimension of point guarding simple polygon problem, relying on the clockwise/counterclockwise system defined by Knuth and discretizing the set of possible polygons. We also introduced the dynamic programming approach to find a certain way to place guards and viewpoints in a simple polygon efficiently.

Algorithm 1 isFeasibleOrdering(G, V, $triples$)

for $(p, q) \in G \cup V$ **do**
  if $p \in G$ **and** $q \in V$ **and** p sees q **then**
    $visibility[(p, q)] = unknown$
  else if $p \in G$ **and** $q \in V$ **and** p does not see q **then**
    $visibility[(p, q)] = false$
  else
    $visibility[(p, q)] = unknown$
evaluate Lemma 2.1
evaluate Lemma 2.2
evaluate Lemma 2.3
evaluate Lemma 2.4
evaluate Lemma 2.6
while $visibility$ has not converged **do**
  evaluate Lemma 2.5
  evaluate Lemma 2.7
return true

References

[1] *Axioms and Hulls*. Springer Berlin Heidelberg, 1992.

[2] Hervé Brönnimann and Michael T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete & Computational Geometry*, 14(4):463–479, 1995.

Graduate Student Articles

SURVEY OF THE HISTORY OF COMPUTER MALWARE

Benjamin Imwald, Daniel Gehman, Kaitlyn Zeigler, Jeonghwa Lee
Department of Computer Science, Shippensburg University
bi7059@ship.edu, dg5957@ship.edu, kz2818@ship.edu , jlee@ship.edu

ABSTRACT
This paper gives a concise yet comprehensive survey of the history of computer malware, starting with the early computing and interconnected devices and continuing through each decade up to the present, concluding with the implications for the future. The introduction covers the early definitions of the first computing systems, malware, and key developments in computer hardware and architecture. The 1940s-1960s laid the foundation for viruses and worms. The 1970s saw multiple proof-of-concepts, and the ARPANET laid the foundation for the Internet. The 1980s demonstrated the first logic bomb, personal computing viruses, worm, denial-of-service attacks, and ransomware. The 1990s saw more instances of malware that are built on top of the DoS attack concepts and demonstrate the first Distributed-Denial-of-Service attacks. The 2000s malware built on top of the DDoS attacks as well as displayed the damage worms can do in the 21st century. The 2010s showed improvements with ransomware, DDoS attacks, phishing type attacks, large scale Zero-Day Vulnerabilities and more. Now, in the present and future, as generative AI continues to advance, its ability to produce malicious code is evolving and increasing at rates beyond human ability. Thus, modern cybersecurity concerns are intensified, and the fight against malware becomes more complex.

KEY WORDS
Computer viruses, computer worms, computer malware, malware evolution, malware detection, cybersecurity threats

1. Introduction

The Analytical Engine, created by Charles Babbage in the 19th Century, is often referred to as the first computer, which was designed with the end goal of completing any mathematical computation. This was with help of Ada Lovelace, of whom is noted as being the first computer programmer [1]. While the Analytical Engine was limited to the number of digits it was able to hold, this was a foundational piece of history to what we know today.
The modern computer was theorized by John von Neumann in the first half of the 20th century. The von Neumann architecture, as it is referred to as, consists of four primary components: a central processing unit (CPU), main memory, and input and output devices [2].
The history of modern computing, based on the evolution of hardware, can be categorized into generations. Table I outlines the generation number, the approximate decade in which they were used, and the technology employed to build these computers. The Electronic Numerical Integrator and Computer (ENIAC), influenced by John von Neumann and utilizing vacuum tubes, is considered the first electronic computer [2].

Table I. Evolution of Computer Hardware by Generations [2].

Generation	Decade	Technology
1	1950s	Vacuum Tubes
2	1960s	Transistors
3	1970s	Integrated Circuits (IC)
4	1980s	Very Large Scale ICs (VLSI)
5	2000s	Ultra Large Scale ICs (ULSI)

With many human inventions such as gun powder, chemical agents, cars, etcetera, while their original use may have been to further science and benefit humans, they have been shown in the past to cause undue harm to the human race throughout history. Computers are one such invention, while originally being created to compute mathematical calculations, which have shifted to that of inciting harm to other people.

The term malware (malicious software), first cited in 1990 [3], can be defined as “a class of software designed to cause harm” [4]. While this term was not coined or recorded until the 1990s, the concept of malicious programs was seen demonstrated a couple decades prior and theorized even earlier in recent history.

2. The 1940s-1960s

The *Theory of Self-Reproducing Automata* was a book originally written by John von Neumann in the 1940s until his death in 1957, and it was later edited and published by Arthur W. Burks in 1966. This starts out by comparing the complexity of the human nervous system to the ENIAC and other computers of the time, giving comparisons such as that the human nervous system “is roughly a million times

more complicated" [5]. He goes on to propose automata, not as the stagnant or hardwired processes that were thought of at the time, but rather a version of them more like cells in the human nervous system. Furthering this, and getting to his main takeaway, he proposes automata or cells which contain information, have ties or connections to the other cells around them, and have functions that allow them to self-replicate or self-reproduce [5].

Albeit not defined in this paper, we can see the theoretical definitions for computer viruses and computer worms in these cellular descriptions. Both viruses and worms can carry payloads; however, the primary difference between the two is that viruses require another program to attach themselves to, in order to replicate or transfer to other system, whereas worms can replicate onto other systems on their own [4]. The term computer virus was officially coined much later by Frederick Cohen in 1984 [6].

In addition to this, he defined the cells in ways that had functions in which to deploy their payloads or contents like viruses. Further, these cells had functions to adapt to their surroundings, which is similar to the ways viruses or worms evade detection in systems [5]. This work of the self-replicating automata became the foundation for malware which the world would not see come to fruition until decades after his death.

In the late 1960s, the Advanced Research Projects Agency (ARPA) under the U.S. Department of Defense (DoD) created the ARPANET. It was an early network of computers mainly limited to certain academic and research organizations who had contracts with the Defense Department. The ARPANET paved the way for computers to connect to one another seamlessly and eventually became a means in which malicious software was spread [7].

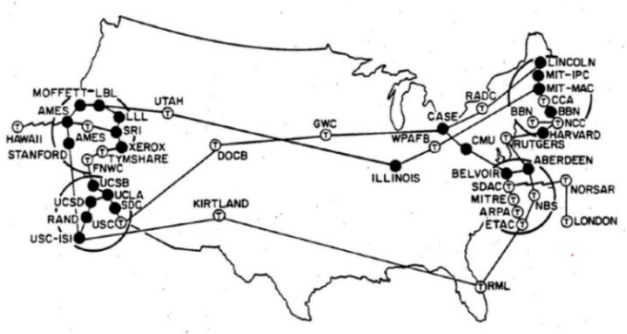


Fig 1. Map of "ARPANET access points in the 1970s" [8].

3. The 1970s

The first proof-of-concept (PoC) multimachine programs were seen in the early 1970s using the ARPANET. In 1971, Bob Thomas created one of the first PoCs for what that we know today as a worm. His program, called the "Creeper," infected DEC PDP-10 computers that were running the TENEX operating system, and it moved from computer to computer simply displaying the message, "I'm the creeper, catch me if you can!" [9]. This process worked as follows: "start to print a file, but then stop, find another Tenex, open a connection, pick itself up and transfer to the other machine (along with its external state, files, etc.), and then start running on the new machine" [10]. Following this, colleagues furthered his work by allowing it to replicate rather than just traverse the network as well as creating a variant that identified copies of the Creeper on the network and log them out [10].

A trojan horse, also referred to as a trojan, is a computer virus which appears to be doing the desired function while also carrying out a malicious or unwanted function as well [4, 11, 12]. In 1975, John Walker created a simple, innocuous trojan called ANIMAL which was designed to look like other animal guessing games of the time, but it also had a subprocess called PERVADE which scanned directories and copied itself [11].

4. The 1980s

Up until the 1980s, the majority of malware was on the theoretical or PoC side in a lab type environment, with no real immediate impact on the surrounding world. This changed in the 1980s. With the last phase of the cold war, tensions were at a high. The U.S. created one of the first instances of a logic bomb [13]. A logic bomb is a type of malicious code that remains dormant until a specific event or trigger condition is met. One specific type of this is a time bomb, which goes off when a specific time is hit [4, 11, 12]. The logic bomb in 1982 was hidden in Canadian code that was set up by the CIA in its efforts against Siberia, resulting in the explosion of a soviet gas pipeline that was able to be seen from space [13]. The reason that these types of attacks can be so dangerous is the difficulty of systems and engineers to find and counter them. Logic bombs rely on specific conditions such as running with a specific machine or software, time zone, location, or environmental variables present [12, 14]. This is still a current threat that systems and people have to account for.

Table II. Elk Cloner Notable Manifestations, subset from [6].

Boot Number	Behavior
15th	Modified the video mode so that the text on the screen was inverted.
20th	Wrote to the speaker, causing a brief click to be heard.
25th	Modified the video mode so that the text on the screen flashed.
35th	Modified the value that represented CONTROL-D so that DOS commands invoked from Applesoft BASIC are printed instead of executed.
40th	Overwrote the reset vector with an instruction that pointed to the reset vector so that pressing

	CONTROL-RESET caused the machine to enter an infinite loop.
50th	Modified the reset vector so that pressing CONTROL-RESET caused the Elk Cloner poem to be displayed.
55th	Modified a constant in the diskette calibration code, causing the sound the disk calibration process made during the boot process to change.
75th-78th	Unconditionally branched to the first instruction that is executed when a disk booted, leading to four consecutive reboots.
79th	Reset the boot counter.

Further, in 1982 with the Apple II series, otherwise known as the Apple][, we see the advent of the first Macintosh malware—the first personal computer virus, specifically one that had the ability to self-replicate, called the Elk-Cloner. This was a boot sector virus which is a virus that infects the boot system and loads itself whenever the computer is booted [4, 11, 15]. The Elk-Cloner Virus's process had 3 main sections: Boot Loading where it loaded itself from the infected boot disk into memory, Replication where it infected new disks, and Manifestation where it displayed signs of the virus. A notable element of the elk-cloner virus is that it had a boot counter which changed the action down by the virus every increment or time it booted, see Table II for a couple of the more notable manifestations [6].

A transition away from the ARPANET came in 1983. The problem prior to this was that there was not a standard way for computers to communicate with one another internationally, or globally. This change came with the introduction of Transfer Control Protocol / Internet Protocol (TCP/IP), which gave us established protocols for computers to use across the globe. Here we see a shift from what was formerly the ARPANET to what we know today as the Internet [7].

In 1986, the Brain Virus was released, much to a similar light to that of the Elk Cloner Virus. The Brain Virus was also a boot sector virus and was notable for being the first virus to affect MS-DOS personal computers and specifically IBM computers. The Brain Virus was created with the intent to annoy the people that pirated the author's software, and it distributed contact information for users to pay for the original software. The glaring difference in this from the Elk Cloner Virus was the number of computers infected. By 1988, 100,000 floppy disks were infected with the virus [15].

Also in 1988, the Morris Worm was released into the world. Notably, this was the first known worm with malicious intent, and it is sometimes further called the first computer worm [9,16]. This worm was developed by Robert Tappen Morris to show the vulnerabilities in the Interconnected Network (Internet) and connected devices of the time. Note that this Internet of the time only linked university, governmental, and military computers around the country. Morris had 4 primary vulnerabilities that the worm used to break into computers: a bug in the "SEND MAIL" program, a bug in the "finger daemon" program, through the "trusted hosts" feature, and through a brute-force-based password guessing program. In addition to exploiting these vulnerabilities, he used methods to avoid detection. The main method was to check if the program was already running on the target computer; if the response was "no," then the computer become infected, otherwise every seventh computer became re-infected. This resulted in more than one instance of the worm running on the target computer. The underlying problem with the worm was that the rate at which computers were asked if they were infected again was so high that computers started running many instances of the worm to the point that it resulted in Denial-of-Service (DoS) attacks [9]. This "rendered several thousand computers unusable" [11].

DoS attacks refer to the flooding of a computer's resources and communications to prevent individuals from accessing the system, data, or network [4]. This was exhibited by the Morris Worm where processes spawned until their computers became unusable. DoS attacks largely became a thing following the creation of the ARPANET and Internet due to the ease of communication between devices, used for both good and malicious intent.

The first ransomware attack was in 1989. Ransomware can be defined as: "Malware that encrypts sensitive files and demands their return for a ransom" [4]. This first ransomware attack was distributed via sending 20,000 floppy disks labelled "AIDS Information Introductory Diskette, Version 2.0" by mail, which were received by attendees of the AIDS conference [18]. The program encrypted file names (not files) on the C:\ drive with a symmetric encryption algorithm, demanding a ransom of $198 for limited use or $398 for lifetime use of your hard disk [17, 18]. After the initial attacks, researchers were able to break the encryption and distribute software to decrypt the file names and remove the ransomware [17]. This attack demonstrates the blurred lines between categories of malware: It was ransomware, as it encrypted data and demanded a ransom, but it also was a trojan horse since the program masqueraded as a desired program. Figure 2 shows one of the messages received by computers infected with the AIDs Trojan.

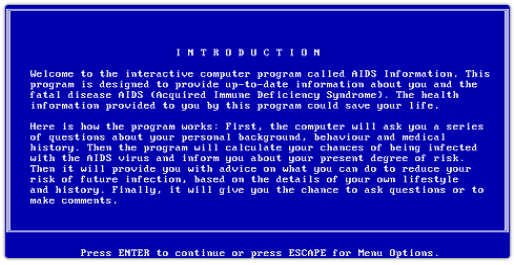


Fig 2. AIDs Trojan Message [19].

5. The 1990s

Intrusion Detection Systems (IDS), which were first desired by the US Government in the 1970s and later theorized by the NSA's Dorothy Dennin in the 1980s, are "system[s] that detects inappropriate or malicious activity on a computer network" [4, 11]. There are four main goals that these systems have: "Detect a wide variety of intrusions, detect intrusions in a timely fashion, present the analysis in a simple, easy-to-understand format, [and] be accurate" [11]. IDS systems are like a "burglar alarm" for unusual activity and to have an audit trail or list that can be examined as to what has happened [20]. This is used in conjunction with other methods in stopping attacks.

In the early 1990s, a new type of DoS attack was detected, called a SYN Flood Attack, which takes advantage of how the TCP/IP network protocol functions. The typical TCP Handshake is reliant on three steps: a client sends a server a SYN packet, the server sends back a SYN/ACK packet, and the client then sends back the ACK packet to allow communication between the devices [12]. When the server does not receive the ACK package back, it gets stuck waiting for it. The key with a SYN Flood attack is that the client sends the original SYN packet using a spoofed IP address; then the server sends the SYN/ACK packet to the spoofed IP but will get no response back. With this attack, the client sends many requests to the server in sequence to bog the server down while it waits for packets that it will never get back, potentially leading to a denial-of-service [4]. Figure 3 below displays this attack, where the attacker, potentially with several bot computers, which then makes it a DDoS Attack, sends spoofed SYN Packets, and the target sends the spoofed SYN/ACK packets to be left waiting [4].

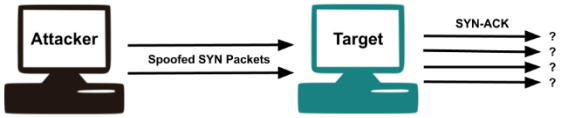


Fig 3. SYN Flood Attack, modified from [21].

The mid to late 1990s also saw the introduction of ping of death (POD) attacks, which is a type of DoS attack that leverages the Internet Control Message Protocol (ICMP). The attack sends an ICMP packet of greater than or equal to the size of 64 KB, which is the limit of ICMP packets. The problem is when systems, which are not designed to deal with larger packet sizes, hang or crash, resulting in a DoS. Most modern systems are built to handle this, but certain ones may be vulnerable to this problem [4].

Distributed Denial-of-Service (DDoS) attacks are a type of DoS attack where the attacker uses (usually) unwilling systems to launch a "many-against-one" attack. A many-against-one attack involves several attacker (i.e., bot) devices that attack a single target. The way hackers accumulate the numbers of devices necessary to run these attacks is through the use of botnets. A botnet is a collection of (generally) unwilling devices that are controlled by malicious code. These devices, called bots, can be used for pooling their computing resources for common goals like mining cryptocurrency, or, in this case, to send a multitude of spoofed SYN packets to the target. With a good botnet, the goal is for the users of the bot computers not to notice the resource usage, which may lead to detection. Botnets and DDoS attacks have become prevalent today [4]. Figure 4 gives a diagram for a DDoS attack, specifically one using multiple DNS Resolvers to attack the victim's device.

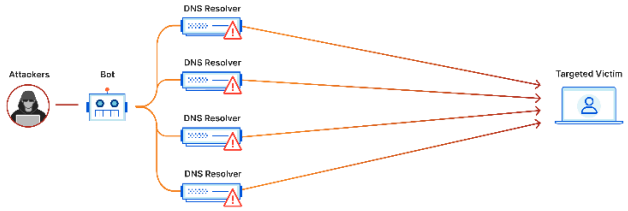


Fig 4. Diagram of a DDoS Attack [22].

DDoS attacks have been shown to cause serious damage for businesses and organizations since they lead to system crashes due to the inundation of fake requests, but, more importantly, they cause down time that customers and partners, who are left unable to access their resources [4]. The larger DDoS attacks go for large hosting entities rather than go for small targets, for example the attack against Google in 2020. This attack lasted 6 months, came from several Chinese Internet Service Providers (ISPs), spoofed 167 Mpps (millions of packets per second), and hit 180,000 CLDAP, DNS, and SMTP Servers [23]. Ignoring the downtime resultant from this, the amount of data and power required for such an attack is extremely large for all parties involved and further shows the scale of these modern attacks. DoS and DDoS attacks have grown exponentially since their inception.

A specific type of DDoS attack, which first emerged near the end of the 1990s, was called the Email Bomb. This is a type of DDoS attack sends a target email address or domain between hundreds of emails per hour and hundreds of emails per minute with the end goal of rendering the mail system unusable, unable to receive outside mail. An example of one of the early attacks was in 1998 when the Tamil Tigers swamped Sri Lankan embassies with 800 emails per day for two weeks [23].

6. The 2000s

The 2000s started off with rampant fears of potential problems with the Y2K "bug." This was due to predictions that systems would have issues with changing dates from

1999.12.31 to 2000.01.01. This turned out to be a problem with some systems, but ultimately the vast majority of systems were unaffected [24].

Following this, on May 4th, the U.S. Army was affected by the "ILOVEYOU" Virus. The virus was distributed through an email attachment and spread automatically by users who opened the attachment, which overloaded the email servers. This virus ended up infecting 2258 workstations and costing around $79,200 [24].

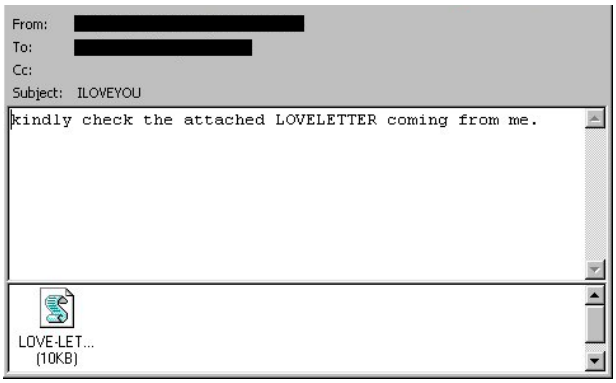


Fig 5. Screenshot of ILOVEYOU Virus email and attachment [25].

In 2007, the Zeus Virus, also known as the Zeus crimeware toolkit, was released. This was a virus that created a botnet, which was used to execute massive DDoS attacks while also spying on the bot computers' keystrokes and activity in order to gain access to financial systems. This originally only affected Windows devices, but it spread to various Android-based OSs too. Devices became infected by either spam messages or drive-by-downloads, which occur whenever you visit a website and it automatically downloads some content (such as another virus) [26, 27].

7. The 2010s

Countries and threat actors have been fighting against public infrastructure digitally since the cold war. This was still the case for the 2010 Stuxnet attack, and it is still a common threat today. The Stuxnet worm was meant to target Iran's nuclear facilities, and it reportedly ended up damaging around 1,000 of their computers, but also it infected 44,000 other computers in Indonesia, India, Russia, Cuba, the U.S., and beyond [13, 28]. Stuxnet, as also seen with other attacks, is a collection of a couple different types of malware including logic bombs and worms. The Stuxnet was described as "the most technologically sophisticated malicious program developed for a targeted attack to date" [28], and it leaves people with the question of what is at risk with everything being digitized in our daily lives now [13].

The CryptoLocker Ransomware from 2013 was innovative in the way it processed payments in addition to its advancements in its cryptography. It was the first ransomware that demanded the ransom in Bitcoin (or another cryptocurrency). This was advantageous to CryptoLocker since there were no geographical restrictions with Bitcoin, it was not subject to local laws, it protected anonymity, and it allowed for the transfer of large sums of money. Cryptographically, CryptoLocker used 2048-bit RSA encryption, which was superior to other encryption-based attacks prior. In all aspects, it was ahead of its time [29].

Fig 6. Screenshot of a CryptoLocker Ransomware Message [30].

In 2016, The Mirai botnet debuted as not the first Internet of Things (IoT) botnet, but rather the first high-profile DDoS threat. IoT devices include but are not limited to the following: smart doorbells, DVRs, routers, printers, security cameras, smart ovens and ranges, smart thermostats, robot vacuums, Wi-Fi enabled cars, and many more. This botnet was able to spread to so many of these IoT devices due to them using default passwords and low security that it had a peak of 600,000 devices and ultimately a stable population of 200,000 to 300,000 connected devices. The Mirai botnet launched about 15,000 attacks for about a year and a half, performing HTTP flood attacks, UDP flood attacks, SYN flood attacks, DNS flood attacks, and more [31].

WannaCry was known as the worst cybercrime of 2017. This was another ransomware attack, affecting more than 250,000 systems and using AES encryption to encrypt each file with individual keys; then the keys were encrypted with 2048-bit RSA encryption, adding an additional layer than that of CryptoLocker [29].

Fig 7. Screenshot of a WannaCry Ransomware Message [32].

8. The Present

In the current times of 2020 to 2025, malware-based attacks and cyber attacks are still extremely prevalent, and it is important for individuals, businesses, and organizations to be on the lookout. According to Petrosyan [33], from 2020 to 2023, there were between 5.4 and 6.06 billion malware-based attacks worldwide. A large number of the largest cyberattacks today occur with large businesses and cloud providers like Microsoft, Amazon, Google, Apple, and Wells Fargo [34]. In addition to this, other large targets are government and public infrastructure entities, seen in the following attacks from 2020-2022: Finnish Parliament attack, Montenegro government attack, Estonian government attack, Belgian government attack, Lithuanian energy company attack, the Colonial Pipeline attack, an attack against Paris' public hospital system, Argentinian government attack, the WHO attack, and many more [35]. Public entities and especially public infrastructure have been the target of government sponsored attacks and publicly sponsored attacks since before the 1980s which has not changed in the over 40 years of history there. In addition to these, a larger number of attacks where hackers leak sensitive user information stolen or obtained from entities have occurred in recent years [35, 36]. There exist websites that list recent and large data breaches and even have tools to show if you have been affected via your passwords contained, personal information, and account usernames [36].

Another important type of malicious attack involves Zero-Day Vulnerabilities. Zero-Day Vulnerabilities, or Zero-Days, are vulnerabilities for which there is no previous knowledge of them [4]. The Common Vulnerabilities and Exposures (CVE) program is a collection of vulnerabilities that have been identified in operating systems and applications "which is maintained by the MITRE corporation and sponsored by the U.S. Department of Homeland Security (DHS) and the Cybersecurity and Infrastructure Security Agency (CISA)" [37]. The list gives information of the vulnerabilities, scores the severities, tracks progress, and lists fixes. This is used by many security personnel, hackers, and companies alike to see the current and past threats to infrastructure. Hackers find and exploit Zero-Days as there is no current fix for the problems, leaving systems open for attack. Typically there is a rather quick turnaround time between where Zero-Day vulnerabilities are announced and when they are patched; otherwise, there can be even worsened or prolonged attack surfaces. Back in 2021, the Log4j vulnerability, also called Log4J or Log4Shell, was a vulnerability found in the Apache Log4j logging library [38]. This was a logging library used for a wide range of applications including Adobe, Cisco, AWS, Broadcom, Fortinet, FortiGuard, IBM, Okta, VMware and many more, and it was noted as being one of the most widely used logging libraries in the world [38, 39]. This attack allowed for remote code execution by hackers, who sent malicious commands through publicly accessible forms and chatbots, allowing the communications to hit other services on the target devices using Log4j [38]. The associated CVE identifier for this vulnerability is CVE-2021-44228. This vulnerability was publicly disclosed on December 10, 2021, and several patches were released within the following week, and ultimately, the patch 2.17.1 on December 18 fixed the vulnerability in entirety [38]. Even though the vulnerability was patched with 8 days, Log4j was used in many applications and software that all had to be patched themselves, in most instances by the company but also by the end user of the software to make sure it was up to date. Some companies took months or even more than a year to update their applications and devices, leaving vulnerability to become one of the most commonly used vulnerabilities by threat actors and attacks still in 2023 [38]. With this slow process for remediation, one thing that this did improve was the addition of rules for firewalls and intrusion detection to be able to block these attacks themselves rather than at the application level [38].

Social Engineering is the practice of "deceiving another person so that they reveal confidential information" [4]. This exploits the human trust aspect of computers, has many facets involved, and is one of the most prevalent forms of cyber-attacks [4, 34]. Phishing refers to deceiving someone over email, Vishing refers to the deception over voice calls, Smishing refers to this practice over Short Message Service (SMS), also called text [4]. Two other specific phishing attacks are Spear Phishing where specific groups are targeted rather than sending the messages to anyone and everyone, and Whaling refers to Spear Phishing high value targets like CEOs, CTOs, and other executives or celebrities [4]. These attacks occur against individuals via communication messages while trying to impersonate or befriend the user, and attack individuals at companies, especially Amazon, Google, and Microsoft [34]. Spoofing is the technique of impersonating another user or computer; examples include MAC or IP Address spoofing, website Uniform Resource Locator (URL) spoofing, and more [4].

Again, the intent with spoofing is to deceive the user to make them believe that you are someone or some company that they can trust in order to gain information like demographics, login credentials, social security numbers, security questions, and more. These attacks can be combined with other attacks including installing malware such as keyloggers (which harvest user credentials), botnet software, and more. Another technique used in this is the Virus Hoax, which attempts to deceive the user into doing something like giving user credentials and such with the threat that there is a virus on their computer [4]. Another specific type of phishing is pharming, this is where threat hackers use a fake website to steal credentials. They use a culmination of steps to make these websites look at real as possibly including typo squatting, domain spoofing, previously listed techniques and more with the goal to obtain real user credentials from users [4]. Figure 8 shows a simple example, specifically depicting two common signs that a website may be fake: the website is unsecured and there are typos in the URL. These are not always the case, but signs like these are good things to watch out for in cases like these. These phishing based social engineering threats can be found across many website, forms, and functions.

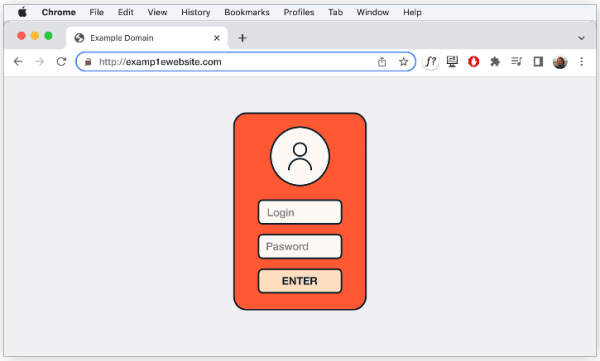


Fig 8. Example of a fake login page [40].

9. The Future of Malware and Cybersecurity

Innovation in computer penetration likewise breeds innovation in the field of computer security. Without the cat-and-mouse game from hackers and security professionals, there would be no need for security to improve for computers. Different computer attacks showcase different vulnerabilities and lead to improvements in their respective attack spheres. Take, for example, the Morris Worm back in the 1980s, which led to the creation of a government entity called the Computer Emergency Response Team (CERT) to deal with emerging threats, led to the creation of the firewall and more [41]. With the improvements in security that are of benefit, there are also conversely improvements to malware or cryptology. Another example of this was the AIDS Trojan in 1989. Researchers at the time were examining the malware in effects to both find how to decrypt users' data, but in doing so they also discovered the flaws that ended up allowing them to decrypt this so easily. With this knowledge, they were able to come up with a proof-of-concept for improving the encryption algorithm, using public key cryptography rather than its private key cryptography; this improvement led to advancements made to later forms of ransomware [17].

A modern and ongoing example is the involvement of artificial intelligence (AI) in both the proliferation of malware as well as its detection. The evolution of generative AI is heightening existing cybersecurity threats and creating new challenges in the fight against malware. Phishing, for example, can become more easily targeted to a specific victim when generated by AI. Additionally, AI is able to produce malicious code at rates much faster than human capabilities [42]. In response, developers have made strides to use AI for preventative detection. GPTZero and similar software have been created to recognize AI-generated material while other generative AI programs detect and report suspicious patterns that are common of malware and phishing [42]. As O'Neill notes, AI is reshaping cybersecurity by enhancing both attacks and defenses, leading to a new era where offense and defense are increasingly automated. This forces cybersecurity professionals to adapt to both more complex threats and the rising role of AI in defense [43].

Another example of technology that has been up and coming for more than a quarter of a century is quantum computing, which is expected to have a large impact on cyber security in the coming years. Quantum computing utilizes quantum mechanics as opposed to the traditional way computers have worked [44]. The excitement regarding quantum computing in the modern age is its improved efficiency and speed compared to traditional computing, yet the technology is not developed to make full use of it [44, 45]. Shor's Quantum Factoring Algorithm, which is meant to efficiently factor large integers, in theory threatens our traditional encryption methods but is unable to beat our current forms of encryption in a reasonable amount of time and resources [46, 47]. Beyond cracking, quantum computing is also being researched for use in creating more secure forms of encryption as well as being used in tandem with machine learning against malware analysis for its power and efficiency [44, 47].

This strive toward innovation and improvement in security is a perpetual process with the constant improvement of technology. For example, the ever-growing and rapidly changing field of machine learning has provided several modern solutions for detecting malware. Decision trees, Support Vector Machines, and similar algorithms have

been trained to reliably classify and distinguish malware from "benign" data [48].

10. Conclusion

Throughout the history of computing and malware, there have been some clear patterns that provide valuable insights and lessons for the future.

The original types of malwares were conceived in theoretical terms, which later transitioned into proof-of-concept (PoC) demonstrations in labs, evolving into harmless pranks or educational tools. Over time, these early forms of malware grew into the complex cyberattacks and the ongoing cyber warfare we see today. This shift has contributed to the rise of a multi-billion-dollar industry focused on computer security and penetration testing [49]. Most attacks historically have started with smaller PoC attacks which were later taken and improved on to increase their attack complexity, payload, and connections [50].

There has been a shift over the years in the types of people who have created malware, and those who partake in cyber-attacks. Some categories for attacks can be defined: hackers (although commonly used for a nefarious person) can be defined as someone hired by corporations or governments for the good of benefiting their security landscape, crackers can be defined as people on their own or with a coalition of people for breaking into systems for nefarious purposes, script kiddies are defined as those without the computer knowledge to write attacks but rather just use publicly available scripts from the Internet [4]. As with any of these, there can be levels or a range of skillset that people have, but the increase of the spread of knowledge on the Internet has led to the ease of people getting into the field with little to no know-how prior.

Appendix A, titled "The Impact Table," provides a detailed overview of the evolution of the malware discussed in this paper, including information of the hardware and operating system, significance, and the damage caused by each type of malware.

References:

[1] W. Isaacson, The Innovators. New York, NY: Simon & Schuster, 2014.

[2] W. Stallings, *Computer Organization and Architecture*, 11th ed. Hoboken, NJ: Pearson, 2020

[3] "Malware, N. Meanings, etymology and more," Oxford University Press, https://www.oed.com/dictionary/malware_n (accessed Mar. 5, 2025).

[4] W. A. Conklin, G. B. White, C. Cothren, R. L. Davis, and D. Williams, Principles of computer security: comptia security+ and beyond, (Exam SY0-601), 6th ed. (New York, New York: McGraw-Hill, 2022).

[5] J. V. Neumann and A. W. Burks, *Theory of Self-Reproducing Automata* (Champaign, IL: University of Illinois Press, 1966).

[6] S. Levy and J. R. Crandall, The program with a personality: analysis of elk cloner, the first personal computer virus, *Arxiv 1*(1), 2021. https://arxiv.org/abs/20

[7] A Brief History of the Internet, *University System of Georgia,* https://www.usg.edu/galileo/skills/unit07/internet07_02.phtml#:~:text=This%20allowed%20different%20kinds%20of,the%20birth%20of%20the%20Internet (accessed Mar. 5, 2025).

[8] "ARPANET," Wikipedia, https://en.wikipedia.org/wiki/ARPANET (accessed Mar. 5, 2025.

[9] F. Syed, Understanding worms, their behavior and containing them, https://www.cse.wustl.edu/~jain/cse571-09/ftp/worms/, 2009.

[10] J. F. Shoch and J. A. Hupp, The 'worm' programs—early experience with a distributed computation," *Communications of the ACM, 25*(3), 1982, 172–180. doi:10.1145/358453.358455

[11] M. Bishop, *Computer Security: Art and Science* (Upper Saddle River, NJ: Pearson Education, 2003)

[12] W. Stalling, L.Brown, *Computer security: principles and practice* (Hoboken, New Jersey: Pearson Education, 2018).

[13] S. J. Shackelford and R. B. Andres, State responsibility for cyber attacks: competing standards for a growing problem, *Conference on Cyber Conflict,* Cambridge, UK, 2010, 197–208.

[14] M. Botacin and A. Grégio, Malware multiverse: From Automatic Logic Bomb Identification to automatic patching and tracing, *Arxiv 1*(1), 2021, https://arxiv.org/abs/2109.06127.

[15] C. Miles, Early history of the computer virus, University of Louisiana at Lafayette, http://craigmil.es/pubs/History_of_Computer_Virus.pdf (accessed March 6, 2025).

[16] U.S. v. Morris, 928 F.2d 504 (2d Cir. 1991).

[17] H. Oz, A. Aris, A. Levi, and A. S. Uluagac, A survey on ransomware: evolution, taxonomy, and defense solutions, *ACM Computing Surveys 54*(11), 2022, 1–37, 2022.

[18] M. Alexander, "Trojan horse sneaks in with AIDS program," Computerworld, p. 4, Dec. 18, 1989.

[19] Gdatasoftware, https://www.gdatasoftware.com/fileadmin/web/general/images/blog/2019/12_2019/Eddy_1.png (accessed Mar. 5, 2025).

[20] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, 13(2), 1987, 222-232.

[21] How to prevent a SYN flood attack. PurpleSec, https://purplesec.us/learn/prevent-syn-flood-attack/ (accessed Mar. 5, 2025).

[22] C. Houle and R. Pandey, A layered approach to defending against list-linking email bombs *APWG Symposium on Electronic Crime Research (eCrime)*, San Diego, CA, 2018, 1-9.

[23] What is a DDoS attack?, Cloudflare, https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/ (accessed Mar. 5, 2025).

[24] J. Burns, 'Iloveyou' virus lessons learned report, *Defense Technical Information Center,* Fort McPherson, GA, 2003.

[25] "ILOVEYOU," Wikipedia, https://en.wikipedia.org/wiki/ILOVEYOU (accessed Mar. 5, 2025).

[26] H. Binsalleeh et al., On the Analysis of the Zeus Botnet Crimeware Toolkit, *Eighth Annual International Conference on Privacy, Security and Trust,* 2010, 31-38. doi:10.1109/pst16766.2010

[27] Zeus virus, Kaspersky Lab, https://usa.kaspersky.com/resource-center/threats/zeus-virus (accessed Mar. 5, 2025).

[28] A. Matrosov, E. Rodionov, D. Harley, and J. Malcho, "Stuxnet under the Microscope," *ESET*, https://icscsi.org/library/Documents/Cyber_Events/ESET%20-%20Stuxnet%20Under%20the%20Microscope%20v1.31.pdf (accessed Mar. 5, 2025).

[29] H. Oz, A. Aris, A. Levi, and A. S. Uluagac, A survey on ransomware: evolution, taxonomy, and defense solutions, *ACM Computing Surveys 54*(11), 2022, 1–37. doi:10.1145/3514229

[30] M. Buckbee, CryptoLocker: Everything You Need to Know, *Varonis*, https://www.varonis.com/blog/cryptolocker (accessed Mar. 5, 2025).

[31] M. Antonakakis et al., Understanding the Mirai Botnet, *Proceedings of the 26th USENIX Conference on Security Symposium,* Berkeley, CA, 2017, 1092–1110.

[32] "US Publicly Blames North Korea for WannaCry Ransomware Attack," MarketWatch, Dec. 19, 2017. [Online]. Available: https://www.marketwatch.com/story/us-publicly-blames-north-korea-for-wannacry-ransomware-attack-2017-12-18 (accessed Mar. 6, 2025).

[33] A. Petrosyan, Number of malware attacks per year 2023, *Statista*, https://www.statista.com/statistics/873097/malware-attacks-per-year-worldwide/ (accessed Mar. 21, 2024).

[34] M. St. John, Cybersecurity stats: Facts and figures you should know, *Forbes*, https://www.forbes.com/advisor/education/it-and-tech/cybersecurity-statistics/ (accessed Mar. 5, 2025.

[35] Recent cyber attacks, *Fortinet*, https://www.fortinet.com/resources/cyberglossary/recent-cyber-attacks (accessed Mar. 5, 2025).

[36] T. Hunt, Have I been pwned: Check if your email has been compromised in a data breach, *HaveIBeenPwned,* https://haveibeenpwned.com/ (accessed Jun. 3, 2024).

[37] CVEs and the NVD Process, *National Institute of Standards and Technology*, https://nvd.nist.gov/general/cve-process#:~:text=Founded%20in%201999%2C%20the%20CVE,Infrastructure%20Security%20Agency%20(CISA). (accessed March 21, 2025).

[38] What is the LOG4J vulnerability?, *IBM*, https://www.ibm.com/topics/log4j (accessed March 5, 2025).

[39] O. Nath, Log4j Flaw: Top 10 Affected Vendors and Best Solutions to Mitigate Exploitations, *Spiceworks*, https://www.spiceworks.com/it-security/vulnerability-management/articles/log4j-flaw-top-10-affected-vendors-and-best-solutions-to-mitigate-exploitations/ (accessed March 5, 2025).

[40] D. Bodnar, What is pharming and how to protect against it, *Avast,* https://www.avast.com/c-pharming (accessed March 5, 2025).

[41] H. Orman, The Morris Worm: A Fifteen-Year Perspective, *IEEE Security & Privacy, 1*(5), 2003, 35–43. doi:10.1109/MSECP.2003.1236233

[42] T. Munson, V. Tao, & J.J. Mohr, The Double-Edged Sword of Generative AI, *CPA Journal*, *94*(7/8), 2024, 35-40

[43] M. O'Neil, AI in Cybersecurity: The Future of Attack and Defense. Security Tech Review, 15(3), 22-34., 2021

[44] F. Mercaldo, G. Ciaramella, G. Iadarola, M. Storto, F. Martinelli, & A. Santone, Towards explainable quantum machine learning for mobile malware detection and classification, Applied Sciences, 12(23), 2022, 12025. https://doi.org/10.3390/app122312025.

[45] U. Ahmed, T. Sipola, & J. Hautamäki, Cyber protection applications of quantum computing: A review, European Conference on Cyber Warfare and Security, 23(1), 2024, 10–17. http://dx.doi.org/10.34190/eccws.23.1.2182.

[46] D. Willsch, M. Willsch, F. Jin, H. De Raedt, & K. Michielsen, Large-scale simulation of Shor's quantum factoring algorithm, Mathematics, 11(19), 2023, 4222. https://doi.org/10.3390/math11194222.

[47] A. Riani, "The Quantum Cybersecurity Revolution: Arguably the Biggest Startup Opportunity in 2025," forbes.com. [Online]. Available: https://www.forbes.com/sites/abdoriani/2024/12/30/t

he-quantum-cybersecurity-revolution-arguably-the-biggest-startup-opportunity-in-2025/ (accessed Mar. 6, 2025).
[48] J. Ferdous, R. Islam, A. Mahboubi, & M.Z. Islam, A Survey on ML Techniques for Multi-Platform Malware Detection: Securing PC, Mobile Devices, IoT, and Cloud Environments, *Sensors, 25*(4), 2025, 1153
[49] P.W. Singer & A. Friedman, Cybersecurity and Cyberwar: What Everyone Needs to Know. New York, NY: Oxford University Press, 2014.
[50] M.L. Cohn, *The Malware Handbook.* 2020
[51] "A brief history of computer viruses & what the future holds," Kaspersky Lab, https://usa.kaspersky.com/resource-center/threats/a-brief-history-of-computer-viruses-and-what-the-future-holds (accessed Mar. 5, 2025).
[52] U.S. leads multi-national action against 'Gameover zeus' botnet and 'cryptolocker' ransomware, charges botnet administrator," *United States Department of Justice*, https://www.justice.gov/opa/pr/us-leads-multi-national-action-against-gameover-zeus-botnet-and-cryptolocker-ransomware (accessed March 5, 2025).

Appendix A: The Impact Table

Year	Name	Hardware / OS	Significance	Damage / Cost
1971	Creeper Worm	DEC PDP-10 / TENEX OS [9,10]	First PoC worm type program [9,10]	No damage to the infected system, negligible cost
1975	Animal Trojan	UNIVAC 1108 [6]	First trojan horse [11]	Potential data loss or system malfunction, expensive for the organization affected
1982	Elk Cloner	Apple][Plus / 6502 [6]	One of first boot sector viruses, affecting the Macintosh [6]	No significant damage to systems, operational costs for users and downtime
1986	Brain Virus	IBM / MS-DOS [15]	First boot sector virus to affect MS-DOS [15]	Greater than 100,000 floppy disks [15]
1988	Morris Worm	DEC VAX and SUN / BSD UNIX OS [9]	First worm with malicious intent [9,16]	"Over 10 million USD" [9], and "rendered several thousand computers unusable" [11].
1989	AIDs Trojan	Windows [29]	First Ransomware attack	20,000 Floppy Disks
2000	ILOVEYOU Virus	Windows, Microsoft Outlook [9]	First worm to inflict billions of USD worth of damage [9]	"5.5 to 10 billion USD [9]."
2007	Zeus Botnet	Windows [26,27]	Notable for its many variants and scale of infections.	"Infected over 3.6 million computers in the US [26]."
2010	Stuxnet	Windows [51]	Used to attack Iran's nuclear facilities but ended up infecting many more computers [51].	Damaged around 1,000 Iranian Centrifuges but affected more than 44,000 computers worldwide [51].
2013	CryptoLocker	Windows [29]	First ransomware to use bitcoin for payment [29]	More than 234,000 computers infected [52]
2016	Mirai Botnet	IoT Devices	Notable IoT-based botnet for its size and length of use [31]	Peak of ~ 600,000 devices infected [31]
2017	WannaCry	Windows [29]	Scale and using both AES and RSA in sequence for encryption [29]	250,000 Computers / 150 Countries [29]

INTERGENERATIONAL CLASSIFICATION OF REDDIT COMMENTS BASED ON SLANG AND EMOJI USAGE

James Dracup, Dr. Richard Burns
West Chester University
jd926102@wcupa.edu, rburns@wcupa.edu

ABSTRACT

The rapid evolution of language, driven by technological advancements, has created notable cultural gaps between generations, particularly in how they communicate. This gap is most apparent in the growing use of slang and emojis among younger generations. This study aims to explore whether Reddit comments can be classified by generation based on the usage of slang and emojis, the frequency of their use across generations, and how such features might influence the meaning of traditional language. Using Reddit's API, we collected comments from four generational subreddits and applied various machine learning models, such as Naïve Bayes, Neural Networks, and Decision Trees to identify the most effective classification method. We compared both standard models and improved models that focus on selective features—specifically slang and emojis—using both imbalanced and balanced datasets. Through this research, we seek to determine if machine learning models can effectively classify social media comments by generation based on certain linguistic features. Our results show that Neural Network models are the ideal candidates for further work on generational classification, going from an accuracy of 77% with just balanced, to 91% with balanced and selective features applied.

KEY WORDS

Natural Language Processing (NLP), Machine Learning (ML), Generational classification, slang, emojis, social media

1. Introduction

With the rise of the Internet came the eventual rapid evolution of society. Through social media platforms, we can express our thoughts, share our beliefs, and even make friends all with the help of computers allowing us to communicate with each other [1]. With these platforms came their unique groups, which share what they like, talk about events, and even create the newest trends and linguistic framework of the current age for the cultures of one's society. This creates challenges, for example, older generations critiquing the younger ones, and the younger ones blaming and not understanding the worldview and knowledge their elders have.

One common example is the usage of such emojis such as the skull emoji, 💀 [2]. This to older generations can be perceived as a threat, or actual death of someone, while for younger generations, the skull emoji is a form of expressing one's extreme form of laughter about what they have heard or seen [3]. This is one example of such a standard form of communication changing with the younger generations. In the form of slang, some common examples would be 'Ohio' which for older generations would mean *the state of Ohio in the United States*, while for younger generations it would mean *weird, bad,* or *cringe*. This brings a lot of nuances since slang and emojis have potentially different meanings. Our research focuses on the ability to understand relationships and differences between slang and emoji usage in intergeneration text. In this paper, we aim to classify Reddit comments to which generation wrote that comment. This is useful as it can bridge the gap of understanding between different eras.

We posit that Reddit comments can be classified by generation based on word usage, as well as the presence of slang and emojis. Using Reddit's API, we gathered data from four generational subreddits which were: `r/BabyBoomers`, `r/GenX`, `r/Millennials`, and `r/GenZ`, and applied various models to these datasets. Such models were: Naïve Bayes, Neural Networks, and Decision Trees to identify the most effective classification method. Both a Standard model and an Improved model (with slang and emoji features) will be tested on both imbalanced and balanced datasets. With these models, we seek to determine if we can classify comments from Reddit by generation.

2. Background

Natural Language Processing is a field of artificial intelligence and computer science that works on creating machines to understand, interpret, and generate human language. It involves the interaction between computers and human (natural) languages, and it uses techniques from machine learning, linguistics, and statistics to process and analyze large amounts of natural language data.

To do this we will gather data from the four different generation subreddits, `r/BabyBoomers`, `r/GenX`, `r/Millennials`, and `r/GenZ`. Preprocessing will check for emojis and the presence of slang or not. This is a challenge that many are focusing on since slang is always changing and has different meanings as the decades go by [4].

The paper, 'Slang or Not' [4], focuses on introducing a binary classification system designed to identify if a sentence contains slang or not. They conduct investigations of several

machine learning models such as Random Forest (RF), logistic regression (LR), Support Vector Machines (SVM), Adaptive Boosting, and Category Boosting with all trained using default parameters with TF-IDF as one feature used [4]. The paper also introduced a new corpus with annotated slang and non-lang labels, for the binary classification, and used traditional machine learning models (ML), deep learning models (DL), fine-tuning of language models (LMs), and large language models (LLMs) [4]. The dataset they tried to use is not publicly accessible, comes with notable limitations, and does not differentiate words that have slang and non-slang meanings in the sentences. For their examples, they used sources like the Clean Corpus of Historical American English (CCOHA) for its detailed documentation and utilized data from X (Twitter) for a rich current source of contemporary slang and informal language [4].

(Lynch, Conor, et al) paper [8], focused on their sentiment analysis classifications of text from X (Twitter) with work on neural network—based frameworks like TensorFlow and Keras as benchmarks for bespoke sentiment analysis algorithms. Their dataset was four publicly available and manually annotated sets. They also mention using Bernoulli Naïve Bayes, Decision Tree, Gaussian Naïve Bayes, Logistic Regression, Linear Support Vector Classification, k-nearest neighbors (KNN), Passive-Aggressive, Perceptron, Random Forest and SVC algorithms using scikit-learn [8]. Convolutional Neural Networks (CNN), Long-Short-Term-Memory (LSTM) and Gated Recurrent Unit (GRU) models were implemented using Keras [8].

Compared to (Slang or Not) and (Lynch, Conor, et al) my work differs and contributes to the field of text classification in NLP for generational classification based on select features (Slang and Emojis) in a few ways:

1. For (Slang or Not) they use sites like Urban-Dictionary [12] and Online Slang Dictionary [4] while I use the built-in Unix standard words file to cross-check all words that aren't standard by our operating system's dictionary.

2. Instead of X (Twitter) my focus is on Reddit for the collected dataset as it's easy to gather said data if you are limited on X's restricted access to scrapping tons of comments. I do collect four datasets as well and have each comment labeled by their respective generation subreddit they were scrapped from.

3. Though I use models that are talked about in their papers I'm more focused on multiclass generational classification instead of predicting binary classification results [4] if slang exists or not in a sentence, or if a text is positive or negative [8].

Overall, while I am using similar models such as Naïve Bayes, Neural Networks, and Decision Trees, my focus is using select features to help build a model that can classify comments made publicly online into a generation based on linguistic pattern usage by users. In the next section, we will present and analyze our scraped dataset.

3. Dataset

This section discusses our dataset collection, preparation, and analysis. The creation of the complete dataset used in this research is split into three phases: (1) the scraping of four Reddit generations datasets: BabyBoomer, GenX, Millennial, GenZ, (2) the preprocessing and creation of two forms (*Standard* and *Selective Features*) for Multinomial Naïve Bayes, Neural Network, and Decision Tree models, and (3) the evaluation of each models' classification capabilities.

3.1 Reddit Generational Dataset Creation

The four generations datasets (Baby Boomer, Gen X, Millennial, and Gen Z) were sourced from four separate subreddits. We assume that the poster or commenter in a specific subreddit is made by an individual of that generation. The sources for each one are `r/BabyBoomers`, `r/GenX`, `r/Millennials`, and `r/GenZ`.

3.1.1 Data Collection

We scraped data from the four locations using the Reddit API [5], which allowed us to gather from the top and new post options that appear on each of the subreddits. Using basic Python scripts, we traversed the subreddits and scraped all the comments that existed in each post from the subreddits. This resulted in us having four separate structured JSON files of: the time, the title of the post, and the comments made under each post. We also accounted for instances when a comment was deleted. We appended into a comments list if a comment had a valid author or ignored it if the comment was deleted since the commenter was removed.

We set up our scripts on a department server with crontab executions that would vary times in the day to activate scrapping comments from Reddit to avoid its rate limit of 100 queries per minute. The scraping focused on gathering all current posts through November 21 and December 11 last year, 2024. Table 1 shows the distribution of our dataset.

Labels	Comments	Words	Emojis	Slang Words
Baby Boomer	513	29,620	28	4,180
Gen X	261,926	7,262,445	22,564	87,419
Millennial	370,488	13,067,792	27,062	103,804
Gen Z	326,434	10,685,833	26,451	94,233
Total:	959,361	31,045,690	76,105	289,636

Table 1. Total amount of comments, words, emojis, and slang in data before preparing it for use.

The size of Baby Boomers is very small in numbers compared to the other three generations. This is due to many of the Baby

Boomer generation not being on the platform.

3.1.2 Dataset Preparation

In this section, we will describe the preprocessing steps that were conducted on the dataset. Figure 1 shows a brief overview of the steps taken to acquire and prepare the data.

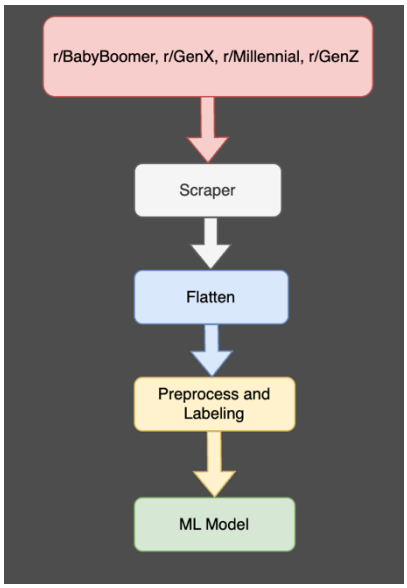


Figure 1. Overview of Data Preparation

We scraped the comments, storing them in separate JSON files containing all comments in their separate posts, that they were gathered. Then we flattened the datasets into one complete comment section. This is done for all four generation datasets. Once flattened we preprocess by tokenizing, filtering out stop words, and eventually labeling each comment with their respective generation name. After that, the datasets were ready for analysis and eventual use in our ML models. Each sentence is a concatenation of words, which can be viewed as features. Stop words, were filtered out using Natural Language Tool Kit (NLTK) [6]. Table 2 shows the total comments that each dataset contains for each generation following the conclusion of the preprocessing.

Labels	Comments	Words	Emojis	Slang Words
Baby Boomer	508	15,208	28	300
Gen X	261,128	3,877,192	22,564	46,546
Millennial	369,648	6,811,040	27,062	61,564
Gen Z	325,421	5,696,290	26,451	54,511
Total:	956,705	16,399,730	76,105	162,921

Table 2. Total amount of comments, words, emojis, and slang in data after preparing it for use.

Lastly, we divide the complete dataset into an 80/20 split ratio for later training and testing sets respectively.

3.2 Dataset Analysis

We are also interested in understanding the prevalence of slang in our dataset. Our approach checks if words are slang or not by using the built-in dictionary called the Unix standard words file. This iterates through each word to check if it exists or not in the standard words file. Those that do not appear are stored in separate JSON files of their respective generation and then compared with each other to find unique slang words that only appear in one generation instead of all. The result is a single file that contains a list of unique slang words that do not appear in the other three generations. Table 3 shows a sample captured from two generation lists of what words were detected as possible slang in our slang word-checking script.

Generation Snippets	Slang
Baby Boomer	[concealers, ceta, sufferagettes, undereye, admundane, subte, …]
Gen X	[entiry, aleast, ravy, rockso, …]

Table 3. A sample captured unique words considered slang from our method.

The other feature of our focus is emojis which have universal Unicode used across all platforms and systems. We ignore the top layer of the Unicode that gives emojis different appearances depending on the platform (e.g. The waffle emoji 🧇 on iPhone vs.  Android). Initially, we have a snippet of features based on frequency per generation as shown below in Table 4.

Emoji Frequencies and Most Common Generation:
👍 - Generation Boomer: 1 times
👍 - Generation GenX: 529 times
👍 - Generation Millennial: 246 times
👍 - Generation GenZ: 872 times
▨ - Generation Boomer: 2 times
▨ - Generation GenX: 520 times
▨ - Generation Millennial: 677 times
▨ - Generation GenZ: 404 times
🧿 - Generation Boomer: 1 times
🧿 - Generation GenX: 4 times
🧿 - Generation Millennial: 9 times
🧿 - Generation GenZ: 1 times
🤓 - Generation Boomer: 1 times
🤓 - Generation GenX: 328 times
🤓 - Generation Millennial: 434 times
🤓 - Generation GenZ: 323 times.

Table 4. Emoji frequency per generation snippet

While all generations use emojis in Table 4, they are not with the same frequency as other generations. To check how frequently each generation uses an emoji over others in our total dataset we have a counter as shown in Table 4 above, for each emoji for when a Unicode is matched in the text. Figures 2 – 4 show an ordering of emojis, by least used to most used emoji by generation.

Boomer:

GenX:

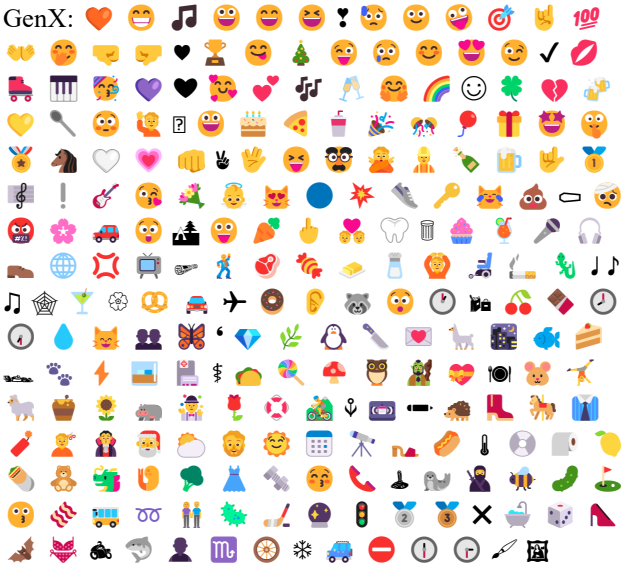


Figure 2. The frequency amount of emojis usage used by BabyBoomers, followed by GenX, the third largest.

Millennial:

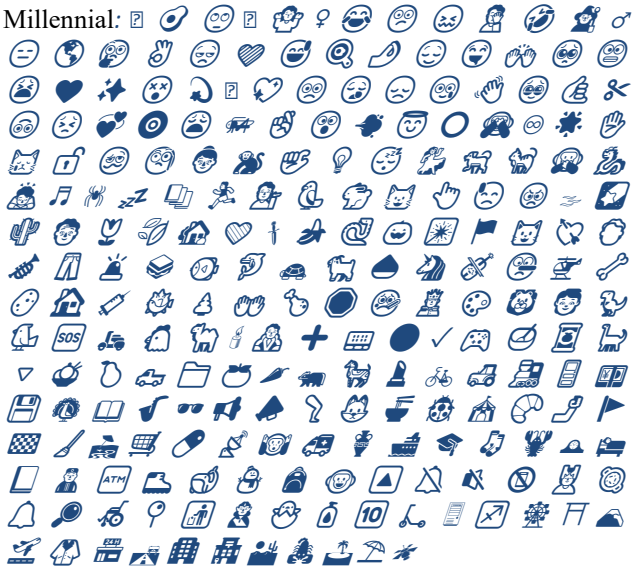
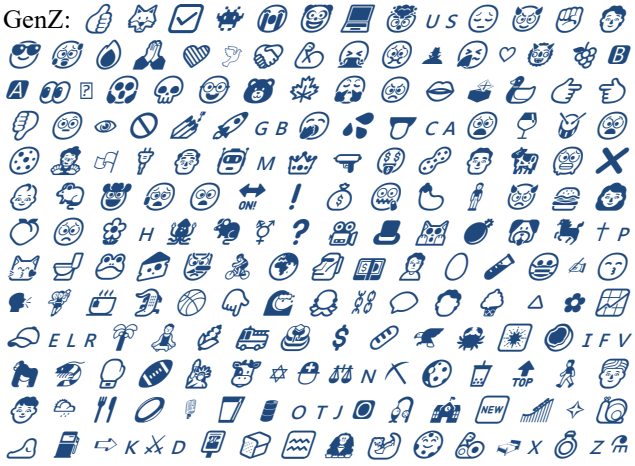


Figure 3. Second largest frequency of emoji usage by Millennials.

GenZ:



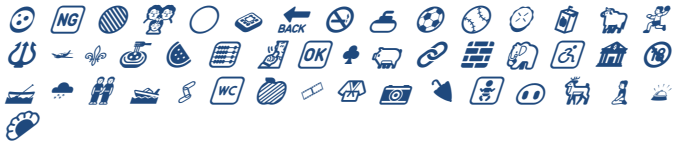


Figure 4. Largest frequency usage of emojis used by GenZ.

Emojis make up about 0.069% of all characters in our dataset based on the total amount of emojis over all the characters as seen in Table 5.

Labels	Characters	Emojis
Baby Boomer	106,440	28
Gen X	24,892,772	22,564
Millennial	46,839,035	27,062
Gen Z	38,262,283	26,451
Total:	110,100,530	76,105

Table 5. Total characters and emojis per generation and combined.

The most common generation on Reddit that uses emojis based on total characters for respective generation data is GenX (0.09%), followed by GenZ (0.07%), Millennials (0.06%), and Baby Boomers (0.03%) from Table 5. Gen Z has the highest amount of emojis by word count with that generation's data (0.46%) since they use features more in their communication on platforms based on the data in Table 2.

In the next section, we will focus on the models chosen to see the classification capabilities for unseen instances for each generation.

4. Model

Once again, we are interested in building a model that is trained on our dataset and seeing if we can classify unseen instances by generation. We chose to experiment with three machine learning techniques: (1) Multinomial Naïve Bayes, (2) Neural Network, and (3) Decision Tree. Using our training set we wished to see if the ML models would classify the comments better in a Standard Form (where all features are treated equally) or in a Selective Features Form (where slang words of best capability and emojis were presented to the model as additional features).

4.1 Multinomial Naïve Bayes

Naïve Bayes applied to the training dataset, treats all features as independent and equally important. When classifying the comments, the model computes posterior probabilities. Slang and emojis are treated the same as other features since the model assumes all are independent. NLTK is used for text preprocessing, including tokenization, stop word removal, and punctuation stripping. TF-IDF Vectorization converts the text into numerical features, emphasizing word importance. Naïve Bayes then calculates posterior probabilities to predict the most likely generation for each comment, considering all features equally.

4.2 Neural Network

The Neural Network (NN) model encodes input data text comments, as numerical features using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer. TF-IDF was chosen because it emphasizes the importance of words that are significant to specific comments, while reducing the weight of more common words, helping the model focus on distinctive language patterns relevant to the classification task.

The architecture of the neural network is a feedforward design with dense layers. The first hidden layer contains 128 neurons, followed by a second hidden layer with 64 neurons. Both layers use the ReLU (Rectified Linear Unit) activation function, which introduces non-linearity to the model, enabling it to capture complex patterns in the data [7]. Dropout layers were incorporated after each hidden layer with a 30% dropout rate to help prevent overfitting. By randomly disabling a portion of the neurons during training, dropout forces the network to generalize better and not over-rely on specific features. The output layer consists of 4 neurons corresponding to the four generations with a SoftMax activation function applied. This function converts the raw output of the network into probabilities, with each value representing the likelihood that a given comment belongs to one of the four categories. The model was trained using the Adam optimizer, which adjusts the learning rate during training for more efficient convergence. The loss function used was sparse categorical cross-entropy, suitable for multi-class classification when the labels are integer-encoded. The model was trained for 5 epochs with a batch size of 32, from the training data.

We also created a second NN model that uses emoji and slang features in addition to the TF-IDF. This separate JSON file containing generation-specific emoji lists were used with each comment, having a binary vector created indicating the presence or absence of each emoji relevant to that generation. This feature was processed and padded to ensure that all vectors for emojis had the same length, making them compatible for integration into the model. To create both neural network models, I used TensorFlow with the Keras API for building the neural networks [8]. The first model was built using a feedforward architecture with TF-IDF vectorization for text feature extraction. The second model incorporated additional features of emoji and slang data, extracted from JSON files. Both models were trained with the Adam optimizer and sparse categorical cross-entropy loss function for multi-class classification.

4.3 Decision Trees

We also built a Decision Tree (DT) classifier, using the Classification and Regression Tree (CART) algorithm [9]. CART uses a binary tree structure where each internal node represents a feature, and each leaf node represents a class label. The tree's splitting criterion was the Gini index, a measure of impurity that helps identify the best split by minimizing misclassifications. The use of slang and emojis is important because they allow the model to focus on the relevant information in the data, which helps in the accuracy. The decision-making options with slang features use TF-IDF vectorization to capture the importance of words in the comments and incorporate generational slang counts by counting slang words specific to each generation. These slang features are derived from predefined lists of slang words for each generation. The TF-IDF and slang features are combined into a single feature matrix, which is then used to train a Decision Tree classifier.

The next section will go over the results of our current models and their working versions to see the classification outputs when running on the test data.

5. Results

In this section, we present the results of running our models against our unseen test set of standard datasets plus selective feature datasets that contain in separate files the unique slang words, and emojis. The randomness of the training and testing set being made makes these results fluctuate a bit with each execution of the models but should not be far off from what is shown. We also compare performances of balanced versus imbalanced data. Using measures of Precision, Recall, F1-Score, Support, and Area Under the Precision-Recall Curve (AUC-PR). Precision, recall, and F1-score are key metrics for evaluating classification models.

Precision measures how many of the predicted positive cases are actually correct, focusing on reducing false positives. Recall measures how many actual positive cases the model successfully identifies, aiming to minimize false negatives. F1-score is the harmonic mean of precision and recall, providing a balanced measure when both are important. A high precision means fewer incorrect positive predictions, while high recall ensures most actual positives are detected. F1-score is useful when there's a trade-off between precision and recall, such as in spam detection or medical diagnoses [10]. AUC-PR is a performance metric for evaluating classification models, especially when dealing with imbalanced datasets. It measures the trade-off between precision (how many predicted positives are actually correct) and recall (how many actual positives the model captures). A high AUC-PR score indicates better performance, meaning the model effectively balances identifying positive cases while minimizing false positives.

5.1 Multinomial Naïve Bayes

In Table 6 we see our accuracy rate for an imbalanced data set using the Multinomial Naïve Bayes model.

Label	Precision	Recall	F1-Score	Support	AUC-PR

					(%)
Baby Boomer	0.00	0.00	0.00	106	0.05
GenX	0.65	0.37	0.47	52386	60.47
Millennial	0.52	0.72	0.61	73848	62.77
GenZ	0.64	0.59	0.57	65001	68.41
Accuracy:	0.58			191341	

Table 6. Imbalanced dataset classification using MNB.

Our NB model has a 58% accuracy in classifying the imbalanced dataset, compared to a baseline of 38.6%, which represents the frequency of the most common class, Millennial. This baseline is calculated by dividing the data from one generation by the total amount of data. The AUC-PR also evaluates the model's performance in classifying comments by their respective generation, with scores in the 60-68% range for GenX, Millennial, and GenZ.

The difference between Accuracy and AUC-PR is that Accuracy measures the proportion of correct predictions (true positives and true negatives) out of all predictions, while the AUC-PR provides a more detailed assessment by considering precision and recall [11]. This is especially valuable in imbalanced datasets where AUC-PR focuses on the model's ability to predict the positive class, making it more effective when class distribution is skewed [11]. Support in the table refers to the number of actual occurrences of each class in the dataset. This offers context for metrics like precision, recall, and F1-score, which can be influenced by the class distribution.

If we retrain this model with balanced data sets, by under-sampling the majority and larger classes to be the same amount as the minority class, results are better with accuracy improving from 58% to 77%. The results of improvement are shown in Table 7.

Label	Precision	Recall	F1-Score	Support	AUC-PR (%)
Baby Boomer	0.74	0.73	0.73	102	83.03
GenX	0.77	0.76	0.77	97	89.32
Millennial	0.85	0.79	0.82	104	93.66
GenZ	0.75	0.82	0.78	104	88.53
Accuracy:	0.77			407	

Table 7. Balanced dataset classification of MNB.

5.2 Neural Networks

Our NN model, when trained on the original imbalanced dataset has comparable accuracy to NB as seen in Table 8. It gets around 59% accuracy, which is comparable to the accuracy of the imbalanced MNB model of 58%. But it improves somewhat for Baby Boomers in the AUC-PR (up to 0.12%) score while being similar to the other generations.

Label	Precision	Recall	F1-Score	Support	AUC-PR (%)
Baby Boomer	0.00	0.00	0.00	106	0.12
GenX	0.58	0.52	0.55	52386	61.42
Millennial	0.57	0.58	0.58	73848	63.23
GenZ	0.61	0.66	0.63	65001	65.94
Accuracy:	0.59			191341	

Table 8. Imbalanced classification with standard NN.

Retraining the model on a balanced dataset as seen in Table 9 also yields comparable results to NB balanced dataset. Both models have the same accuracy results with the balanced dataset with some fluctuation in the AUC-PR scores.

Label	Precision	Recall	F1-Score	Support	AUC-PR (%)
Baby Boomer	0.76	0.75	0.76	102	83.40
GenX	0.76	0.75	0.76	97	86.23
Millennial	0.80	0.83	0.81	104	90.88
GenZ	0.77	0.76	0.77	104	86.92
Accuracy:	0.77			407	

Table 9. Balanced classification with standard NN.

However, when adding slang and emoji features, the performance of the NN dramatically improves to 91% accuracy as seen in Table 10.

Label	Precision	Recall	F1-Score	Support	AUC-PR (%)
Baby Boomer	0.82	0.92	0.87	102	94.19
GenX	0.92	0.89	0.91	97	95.63
Millennial	0.96	0.91	0.94	104	97.65
GenZ	0.93	0.90	0.92	104	97.51
Accuracy:	0.91			407	

Table 10. Balanced classification with selective features NN.

5.3 Decision Trees

We only evaluated the DT model on a balanced dataset with slang features incorporated. The model yields an accuracy of 59%. For the balanced classification with the slang words only we get these types of outcomes below in Tables 11 and 12.

Label	Precision	Recall	F1-Score	Support	AUC-PR (%)
Baby Boomer	0.46	0.59	0.52	161	55.85
GenX	0.78	0.58	0.67	156	71.50
Millennial	0.65	0.60	0.62	147	67.12
GenZ	0.60	0.62	0.61	146	65.79
Accuracy:	0.60			610	

Table 11. Balanced classification standard DT.

Label	Precision	Recall	F1-Score	Support	AUC-PR (%)
Baby Boomer	0.63	0.71	0.67	161	71.30
GenX	0.56	0.55	0.56	156	61.41
Millennial	0.60	0.57	0.59	147	63.95
GenZ	0.54	0.51	0.53	146	58.40
Accuracy:	0.59			610	

Table 12. Balanced classification with slang-only DT.

5.4 Analysis

In this section, we will discuss the current performance of each model implementation. The DT, MNB, and NN models do not have results for certain versions of the data due to storage limits on the current server they run on.

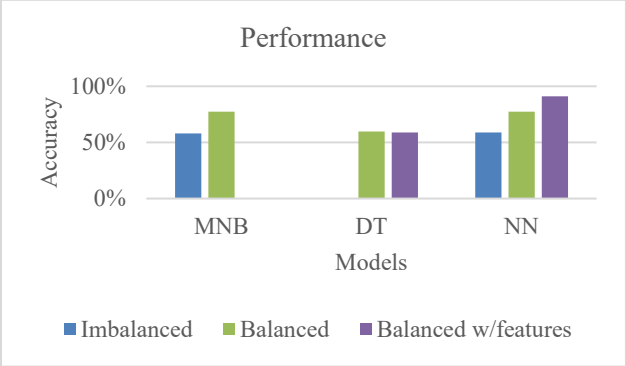


Figure 5. All current ML model performances.

As shown in Figure 5, we present at least two versions for each mode. For MNB we can see improvement from imbalanced to balanced showing that when all data is balanced it improves in its independence calculations for each generation based on features. For the DT there is a small decrease in accuracy between the standard-balanced to a feature-balanced version. For the NN model, accuracy increases from imbalanced to balanced; the addition of both features seems to show improvement in accuracy for classifying unseen instances for each generation. Overall, our NN models show the best performance in improving generational classifications based on emojis and slang.

6. Discussion

Based on the results that are achieved for current implementations of the standard and selective features models, the results are very interesting. The NN model showed the most drastic improvement in accuracy for generations based on selective features. The DT model barely showed improvements even with only half of the selective features added in the model. This can be due to the DT models not having the addition of dropping a fraction of nodes in its path by summing up all the weights for the features it has towards a classification. The MNB model also shows improvement in the balanced dataset but not as much as the balanced data NN model. Our results show promising capabilities for classifying unseen comments based on generations using both slang and emojis.

7. Conclusion

Our NN model performed the best for generational classification based on select features, which included slang and emojis. Some limitations of our results are that slang is very ambiguous and constantly changing. Future improvements for slang, preprocessing, and stemming could be incorporated into this work, as well as other social media platforms. To better improve our understanding of how generations use such features since it is constantly changing over time, we can expand our check of words with other locations such as Urban Dictionary, and Merriam-Webster [12,13]. Also, we would wish to account for n-gram instances where more than one slang word is combined [14]. With better slang detection our long-term goal is to eventually improve results with a plan of creating a cultural translator that can translate specific slang and emojis into one's own emoji and slang usage to better understand each other.

References:

[1] Choi, D., et al. “Characterizing conversation patterns in reddit: From the perspectives of content properties and user participation behaviors." *Proceedings of the 2015 acm on conference on online social networks*. 2015.

[2] Kostadinovska-Stojchevska, B., and E. Shalevska. “THE SKULL EMOJI IN GEN-Z INTERNET SLANG: A STUDY OF ITS USE AS TONE TAG AND PUNCTUATION”. *International Journal of Education Teacher, vol. 27*, May 2024, pp. 124-30.

[3] Wu D, Zhang X and Zhang X (2024) Is there an intergenerational discrepancy in the comprehension and aesthetic preference regarding emoji usage? Evidence from WeChat. *Front. Psychol.* 15 (2024): 1424728.

[4] Anonymous ACL submission, Slang or Not? Exploring NLP Techniques for Slang Detection using the SlangTrack Dataset. *ACL ARR* 2024 December Submission 1763 Authors, 16 Dec. 2024.

[5] Praw-Dev. “GitHub - Praw-dev/Praw: PRAW, an Acronym for ‘Python Reddit API Wrapper’, Is a Python Package That Allows for Simple Access to Reddit’s API.” GitHub, github.com/praw-dev/praw.

[6] Bird, Steven, Ewan Klein, and Edward Loper. *natural language processing with python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[7] Kulathunga, Nalinda, et al. *"Effects of nonlinearity and network architecture on the performance of supervised neural networks."* Algorithms 14.2 (2021): 51.

[8] Lynch, Conor, et al. "A review of open-source machine learning algorithms for twitter text sentiment analysis and image classification." 2020 *International Joint Conference on Neural Networks (IJCNN). IEEE*, 2020.

[9] Priyam, Anuja, et al. "Comparative analysis of decision tree classification algorithms." *International Journal of current engineering and technology 3.2* (2013): 334-337.

[10] Filali, Adnane, and Mostafa Merras. "Enhancing Spam Detection with GANs and BERT Embeddings: A Novel Approach to Imbalanced Datasets." *Procedia Computer Science 236* (2024): 420-427.

[11] Keilwagen, Jens, Ivo Grosse, and Jan Grau. "Area under precision-recall curves for weighted and unweighted data." PloS one 9.3 (2014): e92209.

[12] Urban Dictionary, March 9: yawnsense. (n.d.). In Urban Dictionary. https://www.urbandictionary.com/

[13] Merriam-Webster. (n.d.). Dictionary by Merriam-Webster. In Merriam-Webster. https://www.merriam-webster.com/

[14] Allam, Hesham, et al. *"Text Classification: How Machine Learning Is Revolutionizing Text Categorization."* Information 16.2 (2025): 130.

Undergraduate Articles

THE FUTURE OF TEACHING: AI ASSISTED LEARNING PLATFORM

Mya Bishop, Adam Faust, Alexander Furst, Si Chen, Liu Cui
West Chester University
mb994977@wcupa.edu, af991408@wcupa.edu, af1005095@wcupa.edu, schen@wcupa.edu, lcui@wcupa.edu

ABSTRACT
AI has increasingly been integrated into learning platforms to personalize educational experiences. Despite its potential, fully adaptive and personalized learning systems remain elusive due to limitations in educational resources, interactive models, analytical capabilities, and adaptability. In this paper, we propose an architecture designed to augment AI's capabilities in education. This architecture not only assists instructors in providing personalized materials and tailored assessment questions but also offers learners individualized learning experiences. Additionally, it provides valuable research opportunities for scholars in the field of AI-driven education.

KEY WORDS
AI Assisted Learning, Knowledge Tracing

1. Introduction
Educators routinely encounter the challenge of varying student skills and learning styles, which significantly influences their knowledge acquisition [1]. Traditional teaching methods often fall short in catering to the diverse needs within a classroom. For instance, sensing learners thrive on concrete information such as examples and data, whereas intuitive learners favor abstract concepts like theories and flowcharts [2]. Furthermore, students with a solid grasp of prerequisite knowledge may wish to advance quickly, in contrast to peers who struggle with foundational concepts and require a more measured pace.

To address these disparities, there has been a surge in the development of adaptive e-learning systems over the past decade [3]. Nevertheless, a fully personalized and adaptive learning tool remains elusive, hindered by constraints in educational resources, interactive models, analytical capabilities, and adaptability [4].

The integration of Large Language Models (LLMs) within Artificial Intelligence (AI) presents an unprecedented opportunity to overcome these limitations [5]. By leveraging LLMs, we can develop an AI-driven educational assistant tool that is both instructor-guided and student-personalized, catering to individual learning preferences and proficiencies [6].

The objectives of the AI assisted learning platform are threefold:

1. Develop a Dynamic Learning Platform: Create an AI-assisted adaptive learning environment that recognizes and adapts to each student's unique learning style and proficiency, using advanced AI to deliver a customized educational experience [7].
2. Enhance Instructional Design and Delivery: Provide educators with sophisticated tools to curate and administer computer science curricula, allowing for a versatile range of examples, teaching methods, and assessment techniques [8].
3. Foster Research Opportunities in AI: Offer student workers hands-on research experiences within the fast-evolving domain of AI, promoting practical skills and a comprehensive understanding of AI's role in educational strategies.

This paper focuses on (1) prototype of the AI assisted Learning platform and (2) knowledge tracing that predict students' future performance based on their past responses.

The organization of the paper is as follows. Section 2 introduces the AI assisted learning environment Users. Section 3 illustrates the system architecture. Section 4 is the prototype of the system. Section 5 describes the preliminary knowledge tracing model implemented. Section 6 develops the knowledge tracing for Computer Science Education. Section 7 concludes the paper and highlights directions for future research.

2. AI Assisted Learning Environment Users
The AI-Assisted Learning Environment is designed around the interaction of four key user groups, crucial for the development and success of the project:

1. Students: They interact with course content to learn and provide essential feedback through assessments and discussions, which inform the adaptive learning process [5].

2. Educators: They curate course content and evaluate both student performance and the overall effectiveness of the learning environment, contributing to the refinement of AI models.

3. Technicians: These specialists develop and fine-tune the AI algorithms, ensuring the system performs optimally and aligns with educational goals.

4. Student Researchers: Working with technicians, they aid in the development and implementation of

AI algorithms, gaining hands-on experience in the field.

Each group's contributions are vital to achieving the project's goal of creating a personalized learning experience.

3. System Architecture

As shown in Figure 1, our system is structured around the Content Design and Adaptation Engine, with three main components:

1. Learner Model Engine: Identifies students' learning styles from interaction logs, feeding this information into the AI Engine.
2. Content Design & Presentation Engine: Merges educator input and online resources to create adaptable content, customized by the LLM to fit different learning styles, such as visual or reflective [9].
3. AI Engine: Predicts learning preferences using advanced algorithms and guides the LLM to personalize the content delivery, ensuring that each student's educational experience is as effective as possible [10].

The architecture of this system is central to our objective of enhancing computer science education with AI, facilitating a dynamic and responsive learning environment that adapts to individual student needs and preferences.

The methodologies are tailored to our objectives of improving academic outcomes through personalized learning and providing educators with robust tools for course delivery. We aim to measure the success of these methodologies through improved student engagement and learning outcomes, as well as positive feedback from educators on the utility of the AI tools.

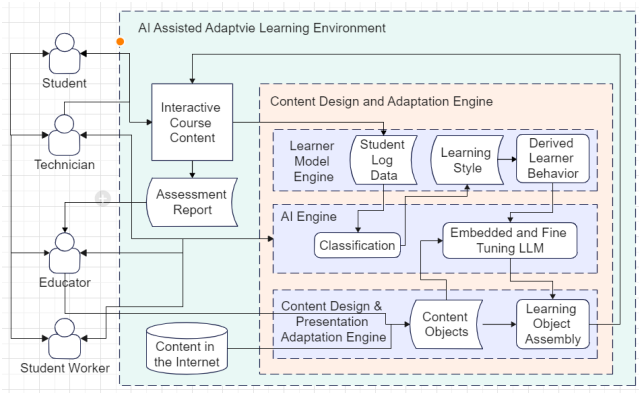


Figure 1. AI Assisted Adaptive Learning Environment System Architecture

4. Prototype

Galaxymentor System (galaxymentor.com, in Figure 2) is the pilot platform for our concept in action. It currently supports subjects from computer programming to math basics, illustrating the AI system's versatility. The backend is powered by Go language, and the frontend by NodeJS, integrating AI technologies like GPT-4 and GPT-3.5-Turbo, with upcoming plans to incorporate LLAMA2 model. The system adeptly tailors' content to user preferences, offering an engaging and interactive learning experience.

4.1 Services Provided in the Platform

There are four components of the AI assisted learning platform, namely Learn, Example, Quiz, and Review. Currently, only Learn is available. While users try each of the component, they could choose the depth of learning from following categories: Elementary, Middle School, High School, Undergrad, Graduate, Ph.D, and Professor. Based on the level that the user chooses, different content will be generated automatically by AI. In addition, Galaxymentor support multiple languages, such as English, Spanish, Franch, Chinese, Korean, etc.

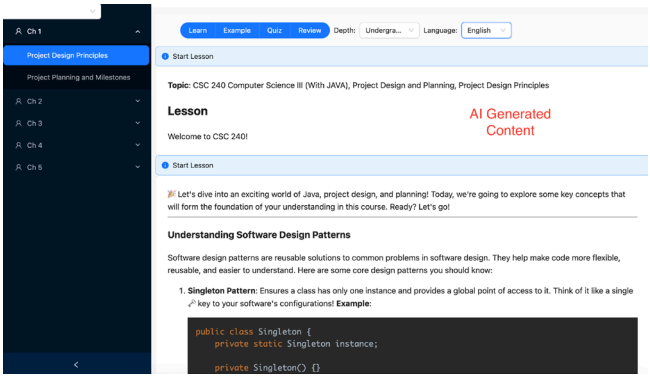


Figure 2. Interface Snapshot: Galaxymentor's AI-Generated Content Tailored to User-Selected Topics

4.2 Generate Learning Content

Each of the course on the AI assisted learning platform has a json file for course template. On one extreme, it can be as simple as only has the course name and chapter name. On the other extreme, the instructor can provide detailed examples that they want to use for each of the knowledge point. Here is a sample course template for CSC 240 Computer Science III.


```
{
  "CourseID": "21",
  "CourseTitle": "CSC 240 Computer Science III (With JAVA)",
  "Chapters": [
    {
      "chapterTitle": "Project Design and Planning",
      "subItems": [
        {
          "subItemTitle": "Project Design Principles",
          "knowledgePoints": [
            "Understanding Software Design Patterns",
            "UML and Class Diagrams",
            "Identifying Project Requirements"
          ]
        },
```

```
      {
       "subItemTitle": "Project Planning and Milestones",
       "knowledge Points": [
         "Setting Project Milestones",
         "Creating Development Checklists",
         "Iterative Development and Feedback Loops"
       ]
      }
     ]
    },
```

5. Knowledge Tracing

In addition to generate course content, another important function for AI is to generate assessment questions. The goal of generating assessment questions is to provide students with enough exercise problems while pushing them quickly from their current difficult level to a higher one. Therefore, the AI engine needs to predict whether the student will answer the next question correctly or not. If the probability of answering the next question in the same topic and same difficult level is high, we want to provide questions in a higher difficult level. If the probability of answering the next question in the same topic and same difficult level is low, we want to provide question in the same difficult level.

5.1 Related Work in Knowledge Tracing

The technique that we use to predict students' capability of answering the next question correct is knowledge tracing. Knowledge tracing refers to the problem of predicting students' future performance by estimating students' time-varying concept/skill mastery level from their past responses to questions.

Knowledge tracing model can be broadly categorized as (1) traditional knowledge tracing models; and (2) deep knowledge tracing (DKT) models. There are two main methods under traditional knowledge tracing models, namely Bayesian knowledge tracing (BKT) and factor analysis models. BKT models often use probabilistic model such as Hidden Markov Model and Bayesian Belief Network to trace students' changing knowledge master level. One of the key models in factor analysis model is Additive Factor Model (AFM), which is a logistic regression model. It calculates the probability of answering a question correctly as proportion to an additive combination of the ability of the student, the difficulty of skills involved in the item, and the amount of the learning gained from each attempt. The three assumption of ATM is: (1) students have different prior knowledge; (2) students learn at the same rate; (3) some skills are easier to master than others [11]. DKT uses deep learning for knowledge tracing. It employs Recurrent Neural Network (RNN) to predict the probability of correctly answering an exercise at each time step [11, 12]. The Dynamic Key-Value Memory Networks for Knowledge Tracing (DKVMN) is one of the DKT.

5.2 Implementing Dynamic Key-Value Memory Networks for Knowledge Tracing

This implementation[1] provides a practical approach to knowledge tracing using neural networks inspired by DKVMN. The code creates a system that can predict whether a student will answer a question correctly based on their history of interactions with different skills or knowledge components. The reason why we chose this algorithm to implement is because in addition to the topics, it also considers students' learning capability change.

In the following sections, we will introduce the knowledge tracing algorithm implemented in this paper, data processing, neural network architecture, results, and customization. Section 6.1 will discuss how to modify this implementation to the assessment for Computer Science education.

5.3 Knowledge Tracing Algorithm

The model starts from (1) determine students' learning ability. There are two steps to determine students' learning ability. First, the correct rate and error rate of questions answered is calculated. Second, a K-means Clustering algorithm is used to group students into three groups. After the clustering is completed, the average learning ability for each group is calculated. (2) A crossover feature that combines student's learning ability, group average learning ability, whether the student asks for help, and number of attempts are determined then fed into the DKVMN model. (3) The DKVMN model understands the similarities of exercise and track the knowledge level that students have.

5.3.1 Data Processing

The implementation works with synthetic datasets from the DeepKnowledgeTracing repository, which represent student-skill interactions in a matrix format:

1. Data Extraction: The data loader downloads the repository and extracts the synthetic datasets, where each row represents a student, and each column represents a skill/question.
2. Data Transformation: The implementation converts this matrix format into a sequence format where each row represents a single interaction (user_id, skill_id, correct), which is the standard format for knowledge tracing.
3. Sequence Creation: For each student, we create sliding windows of interactions to capture the sequential nature of learning. Each window consists of previous skill interactions and correctness values, with the next interaction serving as the prediction target.

[1] Github repository: https://github.com/MyaBishop22/DKVMN-for-KT/blob/main/README.md

5.3.2 Model Architecture

The implementation uses a simplified but effective neural network architecture:

1. Embedding Layers: Separate embedding layers for skills and correctness values convert categorical identifiers into continuous vector representations, allowing the model to learn relationships between skills.
2. Flattened Representation: The embeddings are flattened to create a fixed-length representation of the student's knowledge state based on their interaction history.
3. Dense Layers: Multiple dense layers with dropout for regularization process this representation to make the prediction.
4. Output Layer: A sigmoid activation function produces a probability that the student will answer the next question correctly.

5.3.3 Training and Evaluation

The model is trained and evaluated with a focus on standard knowledge tracing metrics:

1. Training Process: The model is trained on 80% of the data, with 20% of that set aside for validation to monitor for overfitting.
2. Evaluation Metrics: The model is evaluated on Area Under the ROC Curve (AUC) and accuracy, the standard metrics in the knowledge tracing literature and used in Table 1 of the paper.
3. Results Presentation: Results are presented in a table format.

5.4 Results

The result we got is the same as the authors shown in the paper [13], which verify the correctness of implementation. The algorithm will be customized to the input we have for the computer science related questions.

```
Using synthetic dataset: naive_c2_q50_s4000_v9
Processed data shape: (199950, 3)
Number of skills: 50
Created 179955 sequences with window size 5
Train set: 143964 examples
Test set: 35991 examples
Model built successfully
Training model...
Epoch 1/3
1800/1800 ━━━━━━━━━━━━━━━━━━━━ 2s 951us/step - accuracy: 0.7539
Epoch 2/3
1800/1800 ━━━━━━━━━━━━━━━━━━━━ 2s 921us/step - accuracy: 0.7763
Epoch 3/3
1800/1800 ━━━━━━━━━━━━━━━━━━━━ 2s 908us/step - accuracy: 0.7792
```

Figure 3. Results for DKVMN implementation

5.5 Customization

The implementation can be modified in several ways:

1. Change the window_size parameter to experiment with different history lengths
2. Adjust the model architecture by modifying the build_kt_model function
3. Change the number of training epochs for more refined results
4. Try different synthetic datasets by modifying the dataset selection logic

6. Knowledge Tracing in Computer Science Education

There are two types of assessments in Computer Science education, open-ended questions such as short answers and programming and binary-valued questions where the answers are evaluated as right or wrong. The knowledge tracing that we implemented could only capture the binary-valued questions.

6.1 Binary-Valued Questions

In order to quickly push students from their current learning levels (low, medium, or high) to higher level while give them enough exercises. Here is the design of assessment input data that needed for binary-valued questions. In the next stage of the research, we will use AI to automatically generate assessment questions that provide such data.

Majority of the knowledge tracing model use ASSISTmentData as the training dataset. While the public dataset helps researchers compare different knowledge tracing algorithms, it does not tailor to Computer Science problems. Here are the inputs needed for each of the assessment questions we will get for a more accurate prediction.

- user_id: the ID of the student doing the problem
- prior_problem_count_topic_diff: total number of problems tried by this students in topic m and difficult level n
- priori_correct_topic_diff: the number of problems the student had answered correctly in topic m and difficult level n
- problem_id: the ID of the problem
- problem_id_tried: total number of attempts for problem_id n
- problem_id_correct: total number of correct attempts for problem id n
- problem_tag: topic of the problem
- correct:
    - 1 = Correct
    - 0 = Incorrect
- first_action: whether the student requested help during the first exercise
- difficulty_id: high, medium, low

The design logic is that in addition to difficult level of different topics, for example, define variable is easier than loop and decision structure. Each topic also contains questions with different difficult levels. Students' learning capabilities will be applied to determine the starting difficult level, and the changing rate of difficult level. Also, it is possible that a conceptually easy questions is difficult for students because of related knowledge/skill required. Therefore, the difficulty level will be adjusted by the correct

answer rate accordingly.

6.2 Open-Ended Questions

The only paper we found on open-ended questions is Open-ended knowledge tracing for computer science education [14]. The difficult part for open-ended questions is that the knowledge tracing model need to predict students' precise open-ended responses to programming questions. Then, it integrates programming synthesis techniques using language models with student knowledge tracing methods to determine whether students will answer this question correctly or not. We will start from this method and further adjust it based on the data we got.

7. Evaluation of the AI Assisted Learning Platform

The effectiveness and educational impact of the AI assisted and adaptive learning environment will be evaluated through a blend of objective and subjective meausres.

Objective Measures:

- Performance Improvement: Statistical comparison of test/quiz grades on Galaxymentor system and completion rates pre- and post-implementation using paired t-tests to evaluate significant differences.
- Engagement Analysis: Evaluation of interaction logs of Galaxymentor system through descriptive statistics to quantify system usage patterns and time on task.
- Learning Gains: Measurement of concept mastery via pre- and post-tests, analyzed using Analysis of variance (ANOVA) to discern learning advancements attributable to the system.

Subjective Measures:

- User Satisfaction: Survey data analyzed using a Likert scale to determine user satisfaction levels and perceived ease of use.
- Qualitative Feedback: Thematic analysis of open-ended survey responses and focus group discussions to extract prevailing opinions and suggestions for system improvement.

This integrated approach will provide a thorough assessment of the system, ensuring that it meets both technical standards and educational needs.

8. Conclusion

Our mission is to create an inclusive educational tool that transcends disciplinary boundaries. We plan to collaborate with faculty across various departments, including STEM fields, humanities, and social sciences, to adapt our platform's content for their specific academic requirements. This will not only ensure the pedagogical validity of our platform but also harness the collective expertise of our faculty, thus enhancing the educational value for a diverse student body.

In the future, we have two stages to further develop this research. First, create codes that automatically generate assessment questions. We will start from binary-value questions, such as multiple choice, fill-in the blank, and True/False, and then open-ended questions. Topics will be limited to loop, decision structure, and array. So, all students in major can try it out. Also, questions will be generated in three different difficult levels. Second, after collecting the data, we could implement the knowledge tracing with the data we have to predict students' knowledge state, then feed this information to the AI model, which will further improve students' personalized learning experience.

References

[1] G. C. Magulod Jr., "Learning styles, study habits and academic performance of Filipino University students in applied science courses: Implications for instruction," JOTSE: Journal of Technology and Science Education, vol. 9, no. 2, pp. 184–198, 2019.

[2] R. R. Maaliw III, "Adaptive virtual learning environment based on learning styles for personalizing e-learning system: Design and implementation," Online Submission, vol. 8, no. 6, pp. 3398–3406, 2020.

[3] A. Ahadi, R. Lister, H. Haapala, and A. Vihavainen, "Exploring machine learning methods to automatically identify students in need of assistance," in Proc. 11th Annu. Int. Conf. Int. Comput. Educ. Res., 2015, pp. 121–130.

[4] G. Kaymakci and S. Can, "Investigation of the effects of some variables on middle school students' problem-solving skills, science process skills and learning styles," Educational Policy Analysis and Strategic Research, vol. 16, no. 1, pp. 394–426, 2021.

[5] P. Denny, J. Prather, B. A. Becker, C. Mooney, J. Homer, Z. C. Albrecht, and G. B. Powell, "On designing programming error messages for novices: Readability and its constituent factors," presented at the ACM Conference on Programming Language Design and Implementation (PLDI), May 2021.

[6] M. A. Cardona, R. J. Rodríguez, and K. Ishmael, Artificial intelligence and the future of teaching and learning. Department of Education, 2023.

[7] B. Pardamean, T. Suparyanto, T. W. Cenggoro, D. Sudigyo, and A. Anugrahana, "AI-based learning style prediction in online learning for primary education," IEEE Access, vol. 10, pp. 1–1, 2022.

[8] A. Hellas, J. Leinonen, S. Sarsa, C. Koutcheme, L. Kujanpää, and J. Sorva, "Exploring the responses of large language models to beginner programmers' help requests," in Proc. 2023 ACM Conf. Int. Comput. Educ. Res. (ICER '23), vol. 1, pp. 93–105, Association for Computing Machinery, 2023.

[9] E. Kasneci et al., "ChatGPT for good? On opportunities and challenges of large language models for education," Learning and Individual Differences, vol. 103, p. 102274, 2023.

[10] O. El Aissaoui, Y. E. A. El Madani, L. Oughdir, and Y. El Allioui, "Combining supervised and unsupervised machine learning algorithms to predict the learners' learning styles," Procedia Computer Science, vol. 148, pp. 87–96, 2019.

[11] G. Abdelrahman, Q. Wang, and B. Nunes, "Knowledge tracing: A survey," ACM Computing Surveys, vol. 55, no. 11, pp. 1–37, 2023.

[12] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," in Advances in Neural Information Processing Systems, vol. 28, 2015.

[13] J. Zhang, X. Shi, I. King, and D.-Y. Yeung, "Dynamic key-value memory networks for knowledge tracing," in Proc. 26th Int. Conf. World Wide Web (WWW), 2017, pp. 765–774.

[14] N. Liu, Z. Wang, R. Baraniuk, and A. Lan, "Open-ended knowledge tracing for computer science education," in Proc. 2022 Conf. Empirical Methods in Natural Language Processing (EMNLP), 2022.

ON PRODUCTIVENESS AND COMPLEXITY IN REAL FUNCTION ANALYSIS THROUGH HILBERT'S 10TH PROBLEM

Emily Riley, Jingnan Xie
Millersville University of Pennsylvania, Computer Science Department

ABSTRACT

In this paper, we introduce the concept of productiveness, which is a stronger form of non-recursive enumerability, and extend the undecidability results of Hilbert's Tenth Problem and the equivalence to the identically zero function problem to the context of productiveness, demonstrating that both problems are not only undecidable but also unprovable. We also show that many predicates within the theory of real functions, such as determining the continuity or differentiability of an arbitrary function, are as hard as the equivalence to the identically zero function problem through highly efficient many-one reductions. Hence, for some classes of elementary functions, these problems are also productive. Furthermore, these efficient reductions preserve almost any level of complexity classes, allowing us to potentially prove undecidability results and hardness results simultaneously.

KEY WORDS

Undecidability · $\overline{K}$-Productiveness · Computable Analysis · Computational Complexity

1 Introduction

Computable analysis is the study of mathematical structures, such as real numbers or functions, in the context of computability or algorithms. It attempts to translate continuous structures to computable discrete structures that can be examined and analyzed on a computer. For example, by representing a function in some discrete way on a computer, we can then take the derivative of the function or ask questions about the domain and range, as we will look at later in this paper. One of the major tasks that computable analysis focuses on is the computability of real numbers, which involves searching for algorithms that can approximate real numbers to any degree. For example, a common approximation method is encoding a real number as a Cauchy sequence of rational numbers [1]. Despite real numbers being mathematically simple, most real numbers are not computable. Another computable analysis topic of study is the representation of real numbers. Since most real numbers do not have finite representations, computable analysis works to find different methods of representing real numbers. These representations of real numbers can translate to computable functions, which involve determining whether it is possible to algorithmically compute a function to any precision similar to computing or approximating a real number to some precision. For all measures of precision in this area, we compare the error and attempt to get it below some threshold. In addition, computable analysis also focuses on determining the complexity of certain mathematical tasks, such as determining the continuity or differentiability of a function. In computable analysis, the word 'complexity' refers to the existence and hardness of a computation or algorithm for a given problem. Some recent research in this area can be found in [2] and [3].

Computational complexity is an area that studies the amount of resources which are needed for an algorithm to solve a problem. While both computable analysis and computational complexity are rooted in theoretical computer science, they approach computation, real number representation, and precision very differently. While computable analysis focuses on what can be computed and will compute real numbers and functions to as many digits of precision as needed, computational complexity usually works with finite, discrete inputs, focusing on the resources needed to do computations. In computational complexity, precision is typically fixed, and the input size changes while keeping precision the same. The focus is on resource bounds, such as time complexity and space complexity. So, in computational complexity, the word 'complexity' refers to the efficiency of a computation or algorithm. However, using computational complexity theory to classify these mathematical tasks is challenging. The continuous nature of problems studied in computable analysis does not always lend itself neatly to the finite discrete framework employed by computational complexity. A significant amount of research is being done to try to bridge these two gaps between the continuous nature of computable analysis and discrete computational complexity, and this remains an open research area. Some of this research can be found in [4] and [5].

Our work will focus on showing that certain mathematical tasks cannot be proved, as previous work has shown that an algorithm cannot solve these tasks. A Turing machine is a theoretical mathematical model of computation. It represents an abstract computing device, which operates based on a set of rules and states. It illustrates how a simple set of rules can model any algorithmic process, given infinite time and memory. In computable analysis, a Turing machine serves as a mechanism for computing or analyzing various problems. For example, a set is *recursively enumerable* if an algorithm

or Turing machine exists that can list its elements over time. Additionally, A *decision problem* (yes/no problem) is *decidable* if there exists a Turing machine that halts on all inputs and correctly decides the problem. A stronger form of undecidability is *productiveness*, which implies that the problem is not only undecidable but also unprovable. A *productive set*, P, is too large to be recursively enumerable, and given any recursively enumerable subset of P, you can always find an element in P that is not in the subset.

Our work focuses on extending the undecidability results of Hilbert's Tenth Problem (HTP) and the equivalence to the identically equal to zero function problem (denoted by "$\equiv 0$") problems. HTP refers to the problem David Hilbert posed in 1900. He asked for a general algorithm that could find an integer solution of a given Diophantine equation, with any number of unknown quantities and with rational integral numerical coefficients. In [6], Yuri Matiyasevich, building on work by Martin Davis, Hilary Putnam, and Julia Robinson in [7], proved that HTP is undecidable, therefore, there is no general computational method to determine whether arbitrary Diophantine equations have integer solutions.

Building on the undecidability results of HTP, Richardson and Caviness show that the "$\equiv 0$" problem is also undecidable for some classes of elementary functions over $\mathbb{R}$ in [8] and [9]. This means there is no general algorithm to show a certain elementary function is identically equal to 0 over $\mathbb{R}$. We examine the proofs of these results and improve undecidability results to productive results (a stronger form of non-recursive enumerability).

Demonstrating that these problems are unprovable reveals fundamental limits of algorithms in real function analysis. Even with rigorous mathematical methods, we cannot devise a general algorithm to decide these problems across certain elementary functions, highlighting the boundaries of computational classification. Additionally, in mathematical logic, productive sets provide a framework for exploring what can be defined and computed within a formal system.

Furthermore, we show that the "$\equiv 0$" problem is many-one reducible to many mathematical tasks, such as testing continuity certain elementary functions. This illustrates the deep interconnections among the complexities of these problems. Moreover, these issues' productiveness suggests that many real-world applications involving functions may resist straightforward algorithmic solutions.

2 Definitions and Preliminaries

In this section, we review many crucial formal definitions, such as productiveness and Diophantine equations. Several preliminary definitions and notations are also explained. This will provide an overview of the definitions and notations needed to develop our results. The reader is referred to [10] for more information on productiveness and [6] for Diophantine equations.

First, note that for any sets A and B, if A is *many-one reducible* to B, we write $A \leq_m B$. Many-one reducibility is a way to compare the complexity of two problems by showing that one can be transformed into the other using a specific kind of reduction. $A \leq_m B$ if there exists a computable function f such that for every input x, $x \in A$ if and only if $f(x) \in B$.

We first introduce the definition and importance of productiveness, a stronger form of non-recursive enumerability. As our work focuses on productive results, it is important to introduce this idea first. Productive sets and their properties are a standard topic in mathematical logic/recursion theory textbooks such as [11] and [12].

Definition 2.1 recalls the definition of a productive set on $\mathbb{N}$, as developed in [11] and [10].

Definition 2.1. Let W be an effective G*ö*del numbering of the recursively enumerable sets. A set A of natural numbers is called *productive* if there exists a total recursive function f so that all $i \in \mathbb{N}$ if $W_i \subseteq A$, then $f(i) \in A - W_i$. The function f is called the *productive function* for A. □

From this definition, we can see that no productive set is recursively enumerable. It is well-known that the set of all provable sentences in an effective axiomatic system is always recursively enumerable. So for any effective axiomatic system, F is a set A of G*ö*del numbers of true sentences in F is productive, then there is at least one element in A which is true but cannot be proven in F. Moreover, there is an effective procedure to produce such an element.

Let W be an effective G*ö*del numbering of the recursively enumerable sets. K denotes the set $\{i \in \mathbb{N} \mid i \in W_i\}$. $\overline{K}$ denotes the set $\{i \in \mathbb{N} \mid i \notin W_i\}$.

Two well-known facts of productive sets (see [11]) that are necessary for the development of our research are as follows:

Proposition 2.1. 1. $\overline{K}$ is productive.

2. For all $A \subseteq \mathbb{N}$, A is productive if and only if $\overline{K} \leq_m A$.

□

Let Σ, Δ be two different finite alphabets such that both $A \subseteq \Sigma^*$ and $A \subseteq \Delta^*$. It is easily seen that

- There exists a total recursive function $F : \mathbb{N} \longrightarrow \Sigma^*$ such that $\overline{K} \leq_m A$ (via F) if and only if there exists a total recursive function $G : \mathbb{N} \longrightarrow \Delta^*$ such that $\overline{K} \leq_m A$ (via G).

Proposition 2.2. [10] Let $A \subseteq \Sigma^*$, $B \subseteq \Delta^*$ and $A \leq_m B$. Then the following holds:

1. If A is productive, then so is B.
2. If A is productive, then there exists a total recursive function Ψ: $\Sigma^* \to \Sigma^*$, called a productive function for A, such that for all $x \in \Sigma^*$,
   $\mathcal{L}(M_x) \subseteq A \implies \Psi(x) \in A - \mathcal{L}(M_x)$, where $\{M_x \mid x \in \Sigma^*\}$ is some G*ö*del-numbering of Turing machines over alphabet Σ.

□

Additionally, we will introduce the definitions and theorems needed that are related to Diophantine equations. These definitions will allow us to examine the proof of the undecidability of HTP and "$\equiv 0$" to extend the results to productiveness.

Definition 2.2. A *Diophantine equation* is an equation of the form

$$D(x_1, \ldots, x_m) = 0,$$

where D is a polynomial with integer coefficients. □

Definition 2.3. A *family* of Diophantine equations is defined by an equation of the form

$$D(a_1, \ldots .a_n, x_1, \ldots .x_m) = 0$$

where D is a polynomial with integer coefficients, the variables of which are split into two groups:

- the *parameters* $a_1, \ldots .a_n$;
- the *unknowns* $x_1, \ldots .x_m$.

□

Additionally, we can consider the set $\mathfrak{M}$ of all the n-tuples $< a_1, \ldots, a_n >$ for which our parametric equation (2.3) has a solution, that is

$$< a_1, \ldots, a_n > \in \mathfrak{M}$$
$$\Longleftrightarrow$$
$$\exists x_1 \ldots x_m \{D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0\}.$$

An equivalence of the form (2) is called a *Diophantine representation* of the set $\mathfrak{M}$.
We can also present the formal theorem showing the set of false instances of HTP is undecidable as proven by Davis, Putnam, Robinson, and Matiyasevich in [6].

Theorem 2.1. [6] **DPRM-THEOREM** Every recursively enumerable set $\mathfrak{M}$ of n-tuples of nonnegative integers has a Diophantine representation, that is

$$< a_1, \ldots, a_n >$$
$$\Longleftrightarrow$$
$$\exists x_1 \ldots x_m \{D(a_1, \ldots, a_n, x_1, \ldots, x_m) = 0\}$$

for some polynomial D with integer coefficients. □

Corollary 2.1 follows from the previous definition.

Corollary 2.1.

$\overline{K} \leq_m \{D \mid D(x_1, \ldots, x_n)$ is a Diophantine equation, and $D(x_1, \ldots, x_n)$ has no solution in the natural numbers$\}$.

Hence, the set of false instances of HTP is productive.

□

We will also introduce the notation of expressions representing functions. This provides a clear notation to represent our expressions for the various real functions present in this work. Let R be a set of expressions representing functions. All the expressions are assumed to represent real, single-valued elementary functions of one or more real variables over some domain D. If $F \in R$, then $F(x_1, \ldots., x_k)$ for $k \geq 1$ is the function represented by F. It is supposed that given $A, B \in R$, there is an efficient procedure for finding expressions in R to represent the functions

(i) $A(x_1, \ldots., x_m) + B(x_1, \ldots., x_n)$

(ii) $A(x_1, \ldots., x_m) - B(x_1, \ldots., x_n)$

(iii) $A(x_1, \ldots., x_m) * B(x_1, \ldots., x_n)$

(iv) $A(\ldots., B(x_1, \ldots., x_n), \ldots.)$.

Additionally, $A(x_1, \ldots., x_k) \equiv B(x_1, \ldots., x_k)$ means that $A(x_1, \ldots., x_k)$ and $B(x_1, \ldots., x_k)$ are defined over the same domain and equal wherever they are defined.
Finally, we can define the "$\equiv 0$" problem.

Definition 2.4. Let R be a set of expressions. The *equivalence to the identically zero function problem* (denoted by "$\equiv 0$") is the problem of determining, given $F \in R$, over domain D, whether $F(x_1, x_2, \ldots, x_n) = 0$ for all $x_1, x_2, \ldots, x_n \in D$. □

With these definitions and preliminaries in place, we can begin to develop our results in the context of HTP and the "$\equiv 0$" problem.

3 Equivalence to Identically 0 Function Problem

Now, we know that the set of false instances of HTP is productive. We will use this to prove that the set of functions identically equal to 0 is also productive, through a series of efficient many-one reductions.
First, we will define a subset class of expressions that will allow us to prove our intended result. Note that these proofs follow a very similar structure to the proofs in [8] and [9].

Theorem 3.1. Let $\mathfrak{R}_2$ consist of the class of expressions generated by

(i) the rational numbers and π

(ii) the variables $x_1, \ldots, x_n$

(iii) the operations of addition, multiplication, and composition, and

(iv) the sine function

Then,

$$\overline{K} \leq_m \{F \in \mathfrak{R}_2 \mid F(x_1, \ldots., x_n) \geq 0 \text{ over } [0, +\infty]\}.$$

Proof. First, we know that for any polynomial $P(x_1, \ldots, x_n)$ with integral coefficients, deciding if there exists integers $a_1, \ldots., a_n$ such that $P(a_1, \ldots., a_n) = 0$ is productive.

So, we will construct a function $F(x_1, \ldots., x_n)$ in $\mathfrak{R}_2$ such that there exists integers $a_1, \ldots., a_n$ such that $P(a_1, \ldots., a_n) = 0$ if and only if there exist real numbers $b_1, \ldots., b_n$ such that $F(b_1, \ldots., b_n) < 0$.
Construct F such that

$$F(x_1, \ldots, x_n) = (n+1)^2[P^2(x_1, \ldots .x_n) + \sum_{i=1}^{n} \sin^2(\pi x_i) K_i^2(x_1, \ldots., x_n)] - 1$$

Obviously with this equation, if there exists $P(a_1, \ldots, a_n) = 0$ with integers a_i for $1 \leq i \leq n$ then $F(a_1, \ldots, a_n) < 0$. This is due to the fact that $\sin^2(\pi x_i) = 0$ when x_i is an integer. So $F(a_1, \ldots, a_n) = -1 < 0$.

To show the converse, let there exist some $b_1, \ldots., b_n$ such that $F(b_1, \ldots .b_n) < 0$. Also, choose a_i to be the smallest integer such that $|a_i - b_i| \leq 1/2$. So,

$$(n+1)^2[P^2(b_1, \ldots .b_n) + \sum_{i=1}^{n} \sin^2(\pi b_i) K_i^2(b_1, \ldots., b_n)] - 1 < 0$$

That is,

$$P^2(b_1,\ldots.b_n)+\sum_{i=1}^{n}\sin^2(\pi b_i)K_i^2(b_1,\ldots.,b_n) < 1/(n+1)^2$$

Hence, $P^2(b_1,\ldots.b_n) < 1/(n+1)^2 < 1/(n+1)$ and $|\sin(\pi b_i)|K_i(b_1,\ldots.,b_n) < 1/(n+1)$ for all $1 \le i \le n$.

Recall the mean value theorem says for the continuous interval $[a,b]$ and differentiable interval (a,b), there exists some c such that

$$f'(c) = \frac{f(b)-f(a)}{b-a}.$$

Extending this to n dimensions, we can say

$$P^2(a_1,\ldots.,a_n) \le P^2(b_1,\ldots.,b_n) + \sum_{i=1}^{n}|a_i-b_i|\frac{\delta}{\delta x_i}P^2(c_1,\ldots.,c_n)$$

where c_i is between a_i and b_i. From the definition of K_i, which is the dominating function, we can rewrite the equation as

$$P^2(a_1,\ldots.,a_n) \le P^2(b_1,\ldots.,b_n) + \sum_{i=1}^{n}|a_i-b_i|K_i(b_1,\ldots.,b_n)$$

Now, we must prove that $|a_i-b_i| \le |\sin(\pi b_i)|$, knowing $|a_i-b_i| \le 1/2$.

Consider $b_i \in [k, k+1/2]$ where k is any integer. Then our choice of a_i (where a_i should be the smallest integer such that $|a_i-b_i| \le 1/2$), will always be integer k. Assume $b_i = k+j$ where $0 \le j \le 1/2$.

We want to prove $|a_i-b_i| \le |\sin(\pi b_i)|$. So the following statements must be true:

$$|k-(k+j)| \le |\sin(\pi(k+j))|$$
$$|-j| \le |\sin(\pi k+\pi j)|$$
$$j \le |\sin(\pi k)\cos(\pi j)+\sin(\pi j)\cos(\pi k)|$$

Since k is an integer, $\sin(\pi k)=0$. Similarly, $|\cos(\pi k)|$ will always be 1. So,

$$j \le |\sin(\pi j)|$$

Now we will prove $j \le \sin(\pi j)$. That is $0 \le \sin(\pi j) - j$ for $0 \le j \le 1/2$.

Define a function $f(x) = \sin(\pi x) - x$. We must show $f(x) \ge 0$ for all $0 \le x \le 1/2$.

$$f'(x) = \pi\cos(\pi x) - 1$$

Note that $f'(x)$ has one solution in the interval $[0, 1/2]$, which is approximately 0.397. Also, $f'(0) = \pi - 1 > 0$ and $f'(1/2) = 0 - 1 < 0$. So, the function $f(x)$ first increases and then decreases, meaning the one solution is a maximum. Checking the endpoints of the function $f(0) = 0$ and $f(1/2) = 1/2$. So the function is positive in the entire interval $[0, 1/2]$. So we have proved $|a_i-b_i| \le |\sin(\pi b_i)|$, for any $b_i \in [k, k+1/2]$ where k is any integer.

Now consider the case where $b_i \in (k+1/2, k+1]$ for some integer k. In this case, our choice a_i will be the integer $k+1$. Assume $b_i = k+j$ where $1/2 < j \le 1$.

We want to prove $|a_i-b_i| \le |\sin(\pi b_i)|$. So the following statements must be true:

$$|k+1-(k+j)| \le |\sin(\pi(k+j))|$$
$$|-j+1| \le |\sin(\pi k+\pi j)|$$
$$j \le |\sin(\pi k)\cos(\pi j)+\sin(\pi j)\cos(\pi k)|$$

Since k is an integer, $\sin(\pi k)=0$. Similarly, $|\cos(\pi k)|$ will always be 1. So,

$$|-j+1| \le |\sin(\pi j)|$$

So now we will prove $-j+1 \le \sin(\pi j)$. That is $0 \le \sin(\pi j) + j - 1$ for $1/2 < j \le 1$.

So define a function $f(x) = \sin(\pi x) + x - 1$. We must show $f(x) \ge 0$ for all $1/2 < x \le 1$.

$$f'(x) = \pi\cos(\pi x) + 1$$

Note that $f'(x)$ has one solution in the interval $[1/2, 1]$, which is approximately 0.603. Also, $f'(1/2) = 1 > 0$ and $f'(1) = -\pi + 1 < 0$. So, the function $f(x)$ first increases and then decreases, which means that the one solution is a maximum. Checking the endpoints of the function $f(1/2) = 1/2$ and $f(1) = 0$. So the function is positive in the entire interval $[1/2, 1]$. So we have proved $|a_i-b_i| \le |\sin(\pi b_i)|$, for any $b_i \in (k+1/2, k+1]$ where k is any integer.

So we have successfully proved $|a_i-b_i| \le |\sin(\pi b_i)|$ for our choice of a_i

Since $|a_i-b_i| \le |\sin(\pi b_i)|$,

$$P^2(a_1,\ldots.,a_n) \le P^2(b_1,\ldots.,b_n) + \sum_{i=1}^{n}|\sin(\pi b_i)|K_i(b_1,\ldots.,b_n)$$

We know $P^2(b_1,\ldots.,b_n) < 1/(n+1)$ and $|\sin(\pi b_i)|K_i(b_1,\ldots.,b_n) < 1/(n+1)$. Extending the second term to match our equation, we have

$$\sum_{i=1}^{n}|\sin(\pi b_i)|K_i(b_1,\ldots.,b_n) < \sum_{i=1}^{n}1/(n+1)$$
$$\sum_{i=1}^{n}|\sin(\pi b_i)|K_i(b_1,\ldots.,b_n) < n/(n+1)$$

Therefore,

$$P^2(a_1,\ldots.,a_n) \le P^2(b_1,\ldots.,b_n) + \sum_{i=1}^{n}|\sin(\pi b_i)|K_i(b_1,\ldots.,b_n) < 1/(n+1) + n/(n+1) < 1$$

If $P^2(a_1,\ldots.,a_n) < 1$, then $P(a_1,\ldots.,a_n) = 0$. So there exists n natural numbers $a_1,\ldots.,a_n$ such that $P(a_1,\ldots.,a_n) = 0$. We have shown that if there exist real numbers $b_1,\ldots.,b_n$ such that $F(b_1,\ldots.,b_n) < 0$ then exists integers $a_1,\ldots.,a_n$ such that $P(a_1,\ldots.,a_n) = 0$.

Therefore, because we have shown both sides of the implication, we can say

$$\overline{K} \le_m \{F \in \mathfrak{R}_2 \mid F(x_1,\ldots.,x_n) \ge 0 \text{ over } [0,+\infty]\}.$$

□

Now, we must prove a stronger result, where the function F is only of one variable. To do this, we will need a lemma that allows us to reduce a function of multiple variables to a function of one variable, proved in [8].

Lemma 3.1. [8] Let $h(x) = x\sin(x)$ and $g(x) = x\sin(x^3)$. Then for any real numbers $a_1, \ldots, a_n$, and any $0 < \epsilon < 1$, there exists $b > 0$ such that

$$|h(b) - a_1| < \epsilon, |h(g(b)) - a_2| < \epsilon, \ldots, \\ |h(g(\ldots(g(b))\ldots)) - a_n| < \epsilon$$

□

Now, with this lemma defined, we will define the class of expressions that we wish to prove the result for and prove our result.

Theorem 3.2. Let $\mathfrak{R}_1$ consist of the class of expressions generated by

(i) the rational numbers and π

(ii) the variable x

(iii) the operations of addition, multiplication, and composition, and

(iv) the sine and absolute value functions.

Then,

$$\overline{K} \leq_m \{F \in \mathfrak{R}_1 \mid F(x) \equiv 0 \text{ over } (-\infty, +\infty)\}.$$

Hence, the problem "$\equiv 0$" for $\mathfrak{R}_1$ over $\mathbb{R}$ is undecidable. Moreover, the set of true instances of the problem "$\equiv 0$" is productive.

Proof. So, from 3.1, we have

$$\overline{K} \leq_m \{F \in \mathfrak{R}_2 \mid F(x_1, \ldots, x_n) \geq 0 \text{ over } [0, +\infty]\}.$$

Now for any $F \in R$, we can effectively construct an expression $G \in R$ such that

$$G(x_1, \ldots, x_n) = F(x_1^2, \ldots, x_n^2).$$

If there exist n nonnegative real numbers $b_1, \ldots, b_n$ such that $F(b_1, \ldots, b_n) < 0$, then there exist n real numbers $\sqrt{b_1}, \ldots, \sqrt{b_n}$ such that $G(\sqrt{b_1}, \ldots, \sqrt{b_n}) < 0$. On the other hand, if there are n real numbers $b_1, \ldots, b_n$ such that $G(b_1, \ldots, b_n) < 0$, clearly, there exist n nonnegative real numbers $b_1^2, \ldots, b_n^2$ such that $F(b_1^2, \ldots, b_n^2) < 0$.
That means we have

$$\{F \in \mathfrak{R}_2 \mid F(x_1, \ldots, x_n) \geq 0 \text{ over } [0, +\infty)\} \leq_m \\ \{F \in \mathfrak{R}_2 \mid F(x_1, \ldots, x_n) \geq 0 \text{ over } (-\infty, +\infty)\}.$$

For any function $F(x_1, \ldots, x_n)$ represented by an expression in R, we can effectively construct an expression $G \in R$ such that

$$G(x) = F(h(x), h(g(x)), \ldots, h(g(\ldots(g(x))\ldots)))$$

where $h(x) = x\sin(x)$ and $g(x) = x\sin(x^3)$.
We know there exists a real number b such that $G(b) < 0$ if and only if there exists n real numbers $a_1, \ldots, a_n$ such that $F(a_1, \ldots, a_n) < 0$ by 3.1. So we have

$$\{F \in \mathfrak{R}_2 \mid F(x_1, \ldots, x_n) \geq 0 \text{ over } (-\infty, +\infty)\} \leq_m \\ \{F \in \mathfrak{R}_2 \mid F(x) \geq 0 \text{ over } (-\infty, +\infty)\}. \quad (1)$$

For any function $F(x)$ represented by an expression in R, we can effectively construct an expression $G \in \mathfrak{R}_1$ such that

$$G(x) = |F(x)| - F(x)$$

If $F(x) \geq 0$ for all $x \in (-\infty, +\infty)$, then $G(x) \equiv 0$ over $(-\infty, +\infty)$. Otherwise, there exists a real number a such that $F(a) < 0$. So $G(a) = 2F(a) \neq 0$. Then we have

$$\{F \in \mathfrak{R}_2 \mid F(x) \geq 0 \text{ over } (-\infty, +\infty)\} \leq_m \\ \{F \in \mathfrak{R}_1 \mid F(x) \equiv 0 \text{ over } (-\infty, +\infty)\}.$$

By the transitivity of the many-one reduction, we have

$$\overline{K} \leq_m \{F \in \mathfrak{R}_1 \mid F(x) \equiv 0 \text{ over } (-\infty, +\infty)\}.$$

□

From a mathematical standpoint, this result truly captures the limitations of computation and algorithms in real function analysis. The problem of identifying "$\equiv 0$" is not only undecidable but productive. We can systematically generate unprovable functions, highlighting the sheer difficulty of verification.
These findings are incredibly significant as they address the boundaries of algorithmic solvability in continuous mathematics. It provides a glimpse into the structure and properties of real functions, showcasing that even fundamental ideas like equivalence to zero are resistant to algorithmic classification. We will continue to examine these fundamental properties in the next section, where we examine continuity and differentiability.

4 Differentiation and Continuity Problem

We now extend our previous result to show that the differentiability and continuity problem is at least as hard as the "$\equiv 0$" problem. This connection highlights the deep complexity present in real-valued functions. By leveraging function constructions, we can establish an efficient reduction that links these two problems. First, we will introduce some important functions and definitions.

Definition 4.1. A *Dirichlet function* is defined as

$$D(x) = \begin{cases} 1 & \text{if } x \in \mathbb{Q} \\ 0 & \text{if } x \notin \mathbb{Q} \end{cases} \quad (2)$$

where $\mathbb{Q}$ is the set of rational numbers. □

$D(x)$ is nowhere continuous and, therefore, nowhere differentiable. Despite its simple definition, $D(x)$ exhibits extremely unstable behavior, making it a power tool for constructing complex arguments in the concept of continuity and/or differentiability results.
Construct some function $G(x)$ such that

$$G(x) = f(x) * D(x)$$

where $D(x)$ is the Dirichlet function and $f(x)$ is any equation from some class of expressions R that includes multiplication and one variable.
Then the "$\equiv 0$" problem is many-one reducible to the continuity and differentiability problem for real value functions.

Theorem 4.1. Let $\mathfrak{R}_3$ consist of the class of expressions generated by

(i) the rational numbers and π

(ii) the variable x

(iii) the operations of addition, multiplication, and composition, and

(iv) the sine, absolute value, and Dirichlet function

Then,

$$\{F \in \mathfrak{R}_1 \mid F(x) \equiv 0 \text{ over } (-\infty, +\infty)\} \leq_m \{F \in \mathfrak{R}_3 \mid F(x) \text{ is continuous over } (-\infty, +\infty)\}$$

Hence, determining the continuity of a function in $\mathfrak{R}_3$ over $\mathbb{R}$ is undecidable. Moreover, the set of true instances of the continuity problem is productive.
Also,

$$\{F \in \mathfrak{R}_1 \mid F(x) \equiv 0 \text{ over } (-\infty, +\infty)\} \leq_m \{F \in \mathfrak{R}_3 \mid F(x) \text{ is differentiable over } (-\infty, +\infty)\}$$

Hence, determining the differentiability of a function in $\mathfrak{R}_3$ over $\mathbb{R}$ is undecidable. Moreover, the set of true instances of the differentiable problem is productive.

Proof. We know from the previous proof that

$$\overline{K} \leq_m \{F \in \mathfrak{R}_1 \mid F(x) \equiv 0 \text{ over } (-\infty, +\infty)\}.$$

Now for any function $F(x)$ represented by an expression in $\mathfrak{R}_1$, we can effectively construct an expression $G \in \mathfrak{R}_3$ such that

$$G(x) = D(x) * F(x)$$

where $D(x)$ is the Dirichlet function.
If $F(x) \equiv 0$, then $G(x) \equiv 0$ and, therefore, is continuous and differentiable. On the other hand, if $F(x) \not\equiv 0$, then at some point, $F(x) \neq 0$. At that point, $G(x)$ will not be continuous or differentiable as $D(x)$ is nowhere continuous and differentiable. So we have

$$\{F \in \mathfrak{R}_3 \mid F(x) \equiv 0 \text{ over } (-\infty, +\infty)\} \leq_m \{F \in \mathfrak{R}_3 \mid F(x) \text{ is continuous over } (-\infty, +\infty)\}$$

and

$$\{F \in \mathfrak{R}_3 \mid F(x) \equiv 0 \text{ over } (-\infty, +\infty)\} \leq_m \{F \in \mathfrak{R}_3 \mid F(x) \text{ is differentiable over } (-\infty, +\infty)\}$$

□

Similarly to the "$\equiv 0$" problem, this result has significant implications in the world of real-value functions. It again highlights the limitations of algorithms and computational approaches to function analysis and reinforces the complexity of real analysis in logic. This further pushes forward our argument that even fundamental properties of simple mathematical functions cannot be algorithmically determined, emphasizing the need for alternate mathematical systems to classify functions.

5 Other Real Function Problems

The result of productiveness for the "$\equiv 0$" problem is an extremely strong result, as a function being identically equal to 0 allows the creation and solution of many other real function problems. We can create connections between many different real function problems that may seem unrelated. In this section, we investigate two of these problems: determining whether a function takes on zero and determining whether a function is surjective. By exploring these problems, we show that they are as hard as the "$\equiv 0$" problem and demonstrate the interconnection between the complexities of real-value function problems.
We begin proving that determining whether a function ever takes on the value zero is as hard as the identically equal to zero problem.
We can construct some function $G(x)$ such that

$$G(x) = 1 - \frac{f(x)}{f(x)}$$

where $f(x)$ is any equation from the class of expressions defined in 3.2.

Theorem 5.1. Let $\mathfrak{R}_4$ consist of the class of expressions generated by

(i) the rational numbers and π

(ii) the variable x

(iii) the operations of addition, multiplication, division, and composition, and

(iv) the sine and absolute value functions

Then,

$$\{F \in \mathfrak{R}_1 \mid F(x) \equiv 0 \text{ over } (-\infty, +\infty)\} \leq_m \{F \in \mathfrak{R}_4 \mid F(x) \text{ is nowhere defined over } (-\infty, +\infty)\}.$$

and

$$\{F \in \mathfrak{R}_1 \mid F(x) \equiv 0 \text{ over } (-\infty, +\infty)\} \leq_m \{F \in \mathfrak{R}_4 \mid F(x) \text{ never takes on the value zero over } (-\infty, +\infty)\}.$$

Proof. We know from the previous proof that

$$\overline{K} \leq_m \{F \in \mathfrak{R}_1 \mid F(x) \equiv 0 \text{ over } (-\infty, +\infty)\}.$$

Now for any function $F(x)$ represented by an expression in $\mathfrak{R}_1$, we can effectively construct an expression $G \in \mathfrak{R}_4$ such that

$$G(x) = 1 - \frac{F(x)}{F(x)}.$$

If $F(x) \equiv 0$, then $G(x)$ is nowhere defined. Also, $G(x)$ never takes on the value zero. If $F(x) \not\equiv 0$, then at some point $F(x) \neq 0$. Therefore, at that point, $G(x)$ will be defined. Also, $G(x) = 0$ as $\frac{F(x)}{F(x)} = 1$. So we have

$$\{F \in \mathfrak{R}_1 \mid F(x) \equiv 0 \text{ over } (-\infty, +\infty)\} \leq_m \{F \in \mathfrak{R}_4 \mid F(x) \text{ is nowhere defined over } (-\infty, +\infty)\}.$$

and

$$\{F \in \mathfrak{R}_1 \mid F(x) \equiv 0 \text{ over } (-\infty, +\infty)\} \leq_m \{F \in \mathfrak{R}_4 \mid F(x) \text{ never takes on the value zero over } (-\infty, +\infty)\}.$$

□

This problem of determining where a function takes on the value zero is not only computationally difficult but also important in many areas of mathematics and computer science. Areas such as optimization and numerical analysis, where finding a solution to an equation involves checking whether a function ever attains the value 0. The complexity of this problem suggests that many real-world applications involving functions may not be easily solved using algorithms.
We can now prove that determining whether a function is surjective is many-one reducible to the "$\equiv 0$" problem.
We can construct some function $G(x, y)$ such that

$$G(x, y) = f(x) * y + \sin(y)$$

where $f(x)$ is any equation from the class of expressions defined in 3.2.
Reducing this function to one variable will allow us to relate the identically equal to zero problem to the surjective problem through many-one reductions.
In order to prove our desired result, first recall a theorem proved in [8],

Theorem 5.2. Let $h(w) = w\sin(w)$, $g(w) = w\sin(w^3)$. For any x_1 and x_2 and any $\delta > 0$, there is a $w > 0$ so that

$$|h(w) - x_1| < \delta, \quad g(w) = x_2.$$

Now, we can prove our result.

Theorem 5.3. For the class of expressions $\mathfrak{R}_1$ (defined in 3.2), then

$$\{F \in \mathfrak{R}_1 \mid F(x) \equiv 0 \text{ over } (-\infty, +\infty)\} \leq_m \{F \in \mathfrak{R}_1 \mid F(x) \text{ is not surjective over } (-\infty, +\infty)\}$$

Proof. We know from the previous proof that

$$\overline{K} \leq_m \{F \in \mathfrak{R}_1 \mid F(x) \equiv 0 \text{ over } (-\infty, +\infty)\}.$$

Now for any function $F(x)$ represented by an expression in R, we can effectively construct an expression $G \in \mathfrak{R}_2$ such that

$$G(x, y) = F(x) * y + \sin(y)$$

If $F(x) \equiv 0$, then $G(x, y)$ is not surjective. If $F(x) \not\equiv 0$, then there is somewhere where $F(x) \neq 0$ so $G(x, y)$ is surjective. That is

$$\{F \in \mathfrak{R}_1 \mid F(x) \equiv 0 \text{ over } (-\infty, +\infty)\} \leq_m \{F \in \mathfrak{R}_2 \mid F(x, y) \text{ is not surjective over } (-\infty, +\infty)\}$$

Now, to obtain $F(x, y)$ in terms of one variable, we can use Theorem 5.2.
So, in our function $G(x, y)$, we let $x = h(w)$ and $y = g(w)$. This allows us to have a function of only one variable.

$$G(x, y) = F(w) = F(h(w)) * g(w) + \sin(g(w)).$$

$$\{F \in \mathfrak{R}_1 \mid F(x) \equiv 0 \text{ over } (-\infty, +\infty)\} \leq_m \{F \in \mathfrak{R}_1 \mid F(x) \text{ is not surjective over } (-\infty, +\infty)\}$$

□

The complexity of the surjective problem further emphasizes the difficulty in computationally determining properties of real-value functions and how interconnected different concepts in mathematics are. These results truly emphasize the need for a careful mathematical understanding of problems to find solutions rather than forceful computation.

6 Conclusion

We introduced the concept of productiveness, proving the set of false instances of HTP is productive. Building on the prior work of [8] and [9], we also showed that "$\equiv 0$" problem is productive for some classes of elementary functions over $\mathbb{R}$. Additionally we proved that "$\equiv 0$" problem is many-one reducible to many predicates in the theory of real functions such as determining continuity, differentiability and surjectivity of a function. Due to the highly efficient reductions, these relative complexity results preserve almost every level of complexity in computation complexity theory. In the future, we can consider extending the results of the "$\equiv 0$" problem to different domains, as currently it is productive in the set of all real numbers. We may also consider looking at the hardness of these problems, not just the computability.

References

[1] K. Weihrauch, *Computable Analysis: An Introduction*. Berlin, Heidelberg: Springer, 2000.

[2] F. Steinberg, L. Théry, and H. Thies, "Computable analysis and notions of continuity in coq," *Log. Methods Comput. Sci.*, vol. 17, no. 2, 2019.

[3] W. Calvert, K. Kramer, and R. Miller, "Noncomputable functions in the blum-shub-smale model," *Logical Methods in Computer Science*, vol. 7, no. 2, 2011.

[4] U. Berger, J. N. Franklin, F. Manea, and A. Pauly, *Revolutions and Revelations in Computability: 18th Conference on Computability in Europe, CiE 2022, Swansea, UK, July 11–15, 2022, Proceedings*. Springer Nature, 2022, vol. 13359.

[5] P. H. Vasco Brattka, Ed., *Handbook of Computability and Complexity in Analysis*, ser. Theory and Applications of Computability. Springer Cham, 2021.

[6] Y. Matiyasevich, "Hilbert's tenth problem: What was done and what is to be done," *American Mathematical Society*, 2000.

[7] M. Davis, *Computability and Unsolvability*. New York: Dover Publications, Inc., 1982, originally published by McGraw-Hill Book Company, New York, 1958. Dover edition includes a new preface and an appendix, "Hilbert's Tenth Problem Is Unsolvable," from The American Mathematical Monthly, 1973.

[8] D. Richardson, "Some undecidable problems involving elementary functions of a real variable," *The Journal of Symbolic Logic*, vol. 33, no. 4, 1968.

[9] B. Caviness, "On canonical forms and simplification," *Journal of the Association for Computing Machinery*, vol. 17, no. 2, 1970.

[10] J. Xie and H. B. H. III, "On the undecidability and descriptional complexity of synchronized regular expressions," *Acta Informatica*, vol. 60, 2023.

[11] H. Rogers, Jr., *Theory of Recursive Functions and Effective Computability*. Cambridge, MA, USA: MIT Press, 1987.

[12] R. I. Soare, *Recursively Enumerable Sets and Degrees*. New York, NY, USA: Springer-Verlag New York, Inc., 1987.

CONSOLIDATING LECTURE NOTES AND COMPUTING ENVIRONMENT THROUGH CONTAINERIZATION TO SUPPORT PARALLEL COMPUTING EDUCATION

Huy D. Nguyen[1], Bao G. Ngo[1], Tejas Karusala[2], Michael Collins[3], and Linh B. Ngo[3]
[1]Oberlin College, [2]Downingtown East Highschool, [3]West Chester University
{hnguye15,bngo}@oberlin.edu,tkarusala01@student.dasd.org,{mc965348,lngo}@wcupa.edu

ABSTRACT

Teaching parallel and distributed computing (PDC) requires consistent HPC resources, which can be challenging for institutions with limited on-site hardware. We present PDCP (Parallel and Distributed Computing Platform), a container-based system built on Docker and Docker Compose that provides a standardized environment for shared- and distributed-memory programming. PDCP consolidates lecture materials, course assignments, and HPC infrastructure into an easily deployable package, eliminating the overhead of switching between multiple computational platforms. Through multi-architecture images and resource allocation best practices, PDCP supports a variety of hardware, including ARM-based devices. Its flexible Docker Compose configuration enables the quick creation of multi-node clusters that demonstrate scalability concepts, even on personal laptops. Experiments show near-linear speedups for MPI-based trapezoid computations up to available cores, and real-world classroom deployments confirm that students can focus on learning rather than configuring HPC environments. Future directions include integrating scheduling frameworks and migrating PDCP to Kubernetes for broader deployment.

1 Introduction

Teaching parallel and distributed computing (PDC) without onsite hardware resources is a significant challenge, especially when actual coding activities in assignment or in-class examples are involved. At university A, in the past three years, we have offered our PDC course every Spring semester. Each time, a new computing platform was introduced to students. In the first offering, the course focused primarily on shared-memory programming, and our department server was sufficient. In the second offering, Message Passing Interface (MPI) contents were added, and a multi-node cluster was necessary to demonstrate the communication overhead and performance improvement. We switched to XSEDE, a federal computing resource that was available all U.S. academic research and education [1]. One limitation of XSEDE is the additional required knowledge on to interact with a shared high-performance computing resource. This includes learning to write submit script and to become relatively well-versed in working with Linux, which were not typically required for students in our PDC class. Furthermore, XSEDE was heavily utilized, and students often had to wait to get on a resource allocation. This made in-class demonstration and hands-on activities not possible. In the third offering, we switched to another cloud-based federal resource called CloudLab [2]. Using previous work in dynamically launching small high-performance computing (HPC) cluster [3], each student now had the ability to launch their own HPC environment. No wait time for resource was needed, and students were able to run their programs directly. This time, the limitation was due to the long set up time (45 minutes to one hour to completely deploy an HPC) and to the ephemeral nature of these environments (they only lasted at most 16 hours and all data were removed afterward). Having to change from one computing environment to another creates overhead in modifying lecture materials, assignment descriptions, and grading scripts. It also introduces potential new technical issues into the mix. In this case, without the financial and technical mean to have an on-site HPC cluster, it is important to identify a long-term solution for providing both instructor and students with a stable computing environment for teaching PDC.

In this work, we propose to address this issue by developing a container-based platform called PDCP (Parallel and Distributed Computing Platform) to support PDC education. This platform provides a virtual environment in which students can access the same learning environment for shared and distributed memory programming model. The platform can be deployed on a small 4-core laptop, but can also be scaled up across 16-core desktops or high-end laptops. Initial test deployment shows that the infrastructure works for a class of 24 students with no additional technical issues for the instructor. The remainder of the paper is organized as follows. Section 2 reviews previous work on the topic of creating a suitable computational learning environment for PDC topics. Section 3 describes the infrastructure in detail. Section 4 discusses the experimental deployment and the actual in-class deployment of the infrastructure. Finally, Section 5 concludes the paper and discusses future work.

2 Literature Survey

There have been different approaches in developing and deploying local computing clusters that can be used to teach basic PDC concepts. An early solution is to boot a temporary networked computer lab into a pre-configured distributed computing environment [4]. This solution was further extended in combination with a small-scale multi-processor build-out hardware kit to create inexpensive and portable mini clusters for education [5]. In the early 2010s, advances in virtualization technologies led to solutions that support the creation of virtual computing clusters within existing computer laboratories. These clusters can scale across all resources [6] or contain many mini clusters for learning at individual levels [7]. Without leveraging existing on-premise resources, reduction in hardware costs leads to approaches that lean toward the development of personal- and classroom-scale computing clusters. The costs of these cluster can range from approximately $3,000 [8] to $200 [9]. In both scenarios, they also require additional administrative effort, which could either be facilitated by student teams, supported by technical staff or require time effort from the instructors. This presents challenges to institutions with limited personnel resources, teaching responsibilities are high, and typical students are not prepared to take up advanced Linux system administration tasks.

In 2015, the availability of CloudLab [2], a federal large-scale cloud computing resource, provided more options to address the above issues. It is possible to automate the deployment of large-scale computing cluster for individual students on CloudLab to teach PDC topics such as high-performance computing [3], cloud computing [10], and big data engineering [11]. However, we has observed one issue with these approaches, which is the gradual increases of CloudLab utilization and the eventual degrading of hardware on this 10-year-old resource, leading to significantly longer wait time for students. Another observation is that students will be cut off from institutional and federal resources when they finish taking the class or graduate. This limits the ability to self study beyond the scope and duration of the class.

As centralized institutional and federal resources face the issues of hardware degrading and increased utilization through participation, individual computers (laptop and desktop) have also seen gradual improvement in performance while pricing remain relatively consistent. In the 2010s, a low-end laptop ($200 to $400 range) would have had a dual-core CPU and 2-3GB of RAM. Nowaway, a similarly priced laptop would boast 8GB of RAM and a 4-core CPU (or even 8 if we count Intel's hyper-threading). This has made the building of a PDC computing environment inside containers and deploying it on personal computing devices a reality. Containerization of PDC applications is not a new concept. This has been done to support the growing complexity of scientific applications that can interfere with the aspects of system administration of PDC infrastructures [12; 13; 14]. The need to support on-demand and cloud-bursting workloads leads to solutions that dynamically deploy entire systems including their schedulers on external cloud resources [15]. For educational purposes, containerization of an entire PDC infrastructure remains limited. The most notable setup is the HPC Toolset Tutorial, developed by University of Buffalo as part of an introductory workshop to various tools for HPC system administration [16]. The HPC Toolset provided a Docker Compose recipe to build and deploy a PDC infrastructure that includes a login node with scheduler, two compute nodes, a LDAP node that hosts user login information, a XDMoD (Metrics on Demand) node that supports performance evaluation dashboards [17], and a ColdFront node that enables research groups to manage their own resource allocation [18]. While HPC Toolset provides a comprehensive environment for HPC system management and administration study, the complexity of this toolset makes it unsuitable for college-level PDC concepts. The entire infrastructure takes up approximately 20GB of storage when deployed and requires closer to 8 physical CPU cores to ensure smooth operation. It also lacks both application software (GCC compilers for OpenMP and the MPI library itself) and a shared directory to support parallel programming communication.

Leveraging HPC Toolset, we setup a new environment, the PDC Platform (PDCP), to support teaching of PDC topics, more specifically parallel programming for shared and distributed memory. PDCP is equipped with all necessary software (gcc, MPI) and utilities (shared storage, passwordless SSH connection, and browser-based code editor and terminal access). At the same time, PDCP is also optimized to ensure reasonable performance and minimal storage consumption on smaller personal computing devices.

3 Development

The PDCP is developed using Docker images and deployed using Docker Compose. The architectural and deployment diagram is shown in Figure 1. Two Docker images form the basis of all the running containers in a deployment: base and master.

3.1 Building Images

The base image is built on top of a RockyLinux 9.3 image layer and contains the common software and dependencies that are needed for a compute node in a traditional HPC setting. These include basic gcc/++ compilers, OpenMPI, Python, and SSH server. The installation and configuration of the software packages are automated through an external script. This script was copied into the image and run during build time. The compilers, Python, and SSH are installed using RockyLinux's dnf tool. However, OpenMPI was manually compiled from source and installed into a custom location. This is to ensure that OpenMPI can take advantage of Docker's ability to fully expose the compute containers to the host machine's CPU architecture. A user account called *student* is created. This is so that a placeholder directory inside

the image for */home/student* is created. An actual */home/student* directory containing pre-generated SSH keys and configuration, extensions, and password for *code server* are copied into the base image at */opt/home/student*.

The master image is built upon a layer of base image. The master image contains additional software packages that facilitate means for users (faculty and students) to easily access PDCP via browser (*code-server*) and libraries to enable faculty to edit and build interactive lectures using *jupyter-book* [19]. An *entrypoint.sh* script is added to */usr/local/bin* and set up as an entrypoint for the master container.

The image building process is laid out in a Docker Compose YAML file. It specifies the subdirectories containing the relevant Dockerfiles and additional support files as well as the tags for the images.

3.2 Deploying Platform

Once the images are built, PCDP can be deployed using Docker Compose. The *head* container, launched from the *master* image, is to be deployed first. When *head* is deployed, an external directory is used on the host machine to mount and map to the */home/student* directory inside the container. This directory was originally created and remained empty during the initial build. The external host directory will override this empty path whenever the container is run, ensuring that the data is seamlessly shared between the container and the host. This allows users to easily access the content created by work done inside PDCP.

Both *base* and *master* carry their own *entrypoint.sh* scripts with different contents. The *entrypoint.sh* script on *base* will start the SSH server to allow SSH connection to all compute containers. In addition, the *entrypoint.sh* script on *image* will do the following:

- Copy the content of */opt/home/student* into the actual home directory of the *student* account. As this directory is mapped to a directory on host storage, which is then mounted on all other containers, this effectively creates a shared home directory and enables passwordless SSH connections among all the containers.
- Launch a code server with a predefined port and password, as stored in the configuration file. The port is exposed and assigned to a set of ports on the host computer, specified in the *docker-compose.yaml* file.

By default, the *docker-compose.yml* file contains the build and deploy instructions for one head node (container) and two compute nodes (containers). Each container can be configured with the number of computing cores and the size of memory available. The number of compute containers can be arbitrarily increased simply by copying an existing compute segment and modifying (increasing) the counting identifiers in the segment. On a single computing device, the number of compute containers should be carefully calculated to match with the total *available* physical cores, after accounting for cores allocated to support the host machine's operating system. The *docker-compose.yml* can be used to deploy the platform across multiple nodes by leveraging Docker Swarm [20].

A successful deployment provides users with browser access to PDCP in the form of a code server interface. This includes file editing and terminal access similar to a normal environment with a traditional high-performance environment. For instructors, there are available configurations within the *docker-compose.yml* file to mount lecture materials. This ensures that all code examples and assignments listed for the course can be tested within PDCP prior to being distributed to students. While it is possible to make the lecture materials part of PDCP itself, we have decided not to out of consideration for potential copyrighted materials that can only be presented to students indirectly via lectures.

4 Discussion

This section addresses the challenges encountered during the deployment of the PDCP system and offers recommendations based on our experiences. We focus on cross-platform compatibility issues and best practices for containerized PDC environments.

4.1 Cross-Platform Compatibility

The initial version of PCDP uses a default x86 setting for Docker image building. This creates a problem when PCDP is built on different platforms. In a preliminary test on the MacBook with Apple Silicon, the platform did not work well with the ARM64 architecture, which caused constant warnings. The building process was very slow and often failed for no specific reason. The recovery process frequently requires deletion of the cache data and restart. Even in the case of a successful build, containers can randomly fail during deployment. To address this issue, we update the Dockerfiles with the following:

```
FROM --platform=linux/amd64 \
    rockylinux:9.3-minimal AS stage-amd64

FROM --platform=linux/arm64/v8 \
    rockylinux:9.3-minimal AS stage-arm64
```

Without the *–platform* flag, Docker tries to use a predefined base image for the x86 architecture, thus creating compatibility problems when it runs on ARM64 systems such the MacOS on Apple Silicon. By specifying both architectures, Docker now has a choice can choose a base image that is suitable to the host system. As a result, such conflicts are avoided.

The deployment to Linux-based and Windows-based ma-

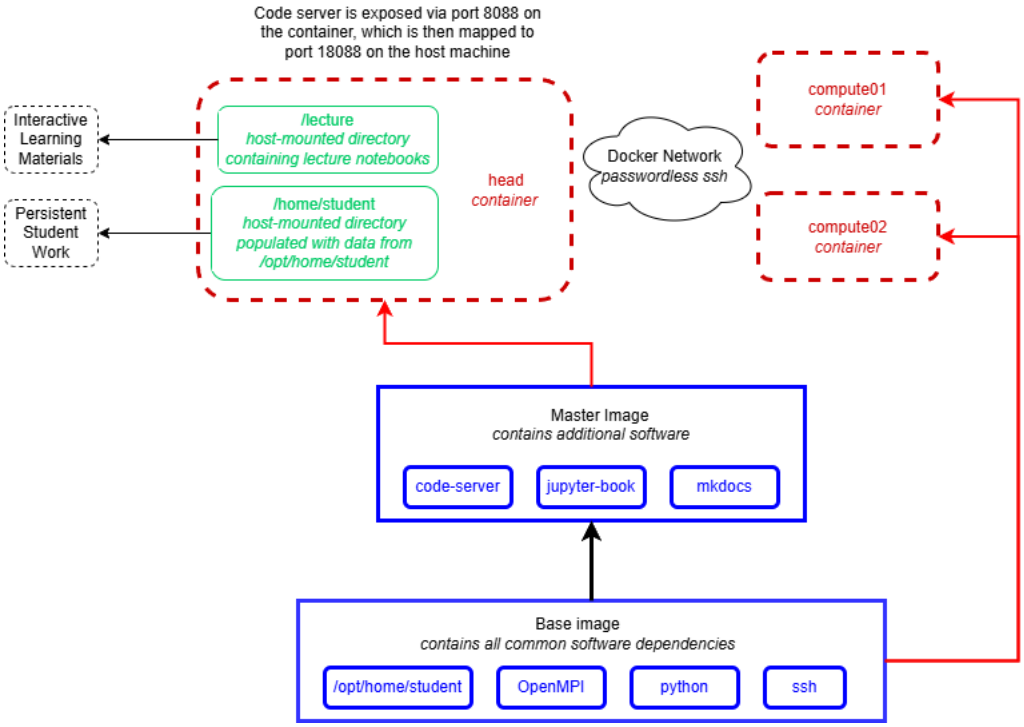


Figure 1: Architectural diagram of the PDCE platform

chines was straight forward with no compatibility issues for both versions of the base Dockerfiles. Linux natively supported Docker without the extra virtualization layers found in MacOS and Windows, providing better resource management and quicker builds. These experiences demonstrated that PDCP can be deployed across platforms once architecture-specific considerations are put in place. in containerized educational environments, particularly when deploying on ARM64-based systems.

4.2 Deployment Best Practices

PDCP has been deployed and utilized in both experimental and in-class testing. From an experimental perspective, the platform was deployed on the authors' personal computing devices including one Windows desktop, one low-end Windows laptop, one Linux laptop, one high-end Windows laptop, one Macbook Air, and one Macbook Pro. At the beginning of Spring 2025, PDCP is used in a Parallel and Distributed Programming class with 25 students. All students use their personal laptop to deploy the platform with no deployment or execution issue. Based on this experience, we identified the following best practices to implement and deploy PDCP in educational settings. All code examples from the lectures up to this point in the semester have been usable by students as is.

- **Multi-architecture Image Building:** Define platform targets explicitly in Dockerfiles to accommodate multiple architectures. This provides compatibility with the wide variety of hardware instructors students will most likely use.
- **Resource Allocation:** Set Docker's resource constraints appropriately prior to image building. This prevents unwanted utilization spikes that could interfere with the host machine's normal operations, leading to freezes and eventually crashes.
- **Documentation:** Keep platform-specific setup guides that deal with the unique challenges of each operating system. It reduces support costs drastically by documenting known problems and fixes.

These best practices greatly improve the usability and consistency of containerized PDC teaching environments. By resolving architecture compatibility, resource management, data persistence, and documentation requirements in advance, instructors can expose students to a cleaner experience that concentrates on learning parallel computing concepts instead of debugging environment problems. These guidelines provide a starting point for institutions and individuals wanting to pursue comparable containerization strategies to PDC education.

4.3 Performance Evaluation

It is clear that a containerized HPC environment running on a single personal computing device will not have the same performance as its production counterparts. However, we want to demonstrate that even on a single laptop, this environment

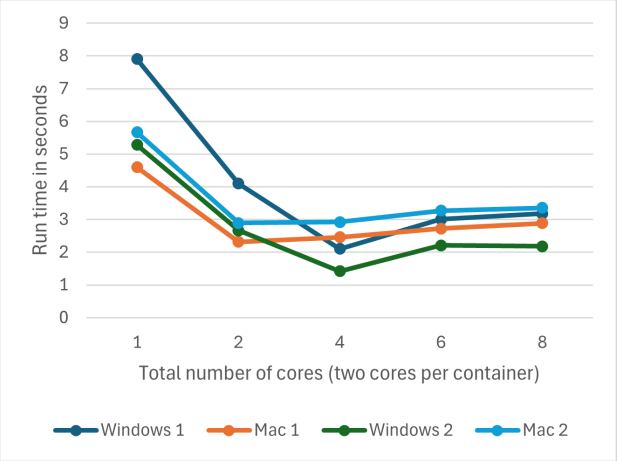


Figure 2: Runtime scalability of trapezoid calculation on high-end computing devices

Cores	Windows 1	Mac 1	Windows 2	Mac 2
1	7.9	4.599	5.278	5.669
2	4.1	2.319	2.663	2.891
4	2.1	2.458	1.42	2.923
6	3.01	2.726	2.21	3.269
8	3.18	2.89	2.18	3.357

Table 1: Scaling of runtimes in seconds over increased total core counts across different personal computing devices

can exhibit scalability, an important attribute that we wish students to observe when they learn about PDC concepts. In this evaluation, we use the following personal computing devices.

- *Windows 1*: Desktop PC with 32 GB RAM, 8-Core AMD Ryzen 7 3.6 GHz
- *Windows 2*: Laptop PC with 16 GB RAM, 14-core Intel i7-12700H 2.30GHz
- *Mac 1*: Macbook Pro with 24 GB RAM, 10-core Apple M4 (4 performance cores at 4.5 GHz and 6 efficiency cores at 2.9 GHz).
- *Mac 2*: Macbook Air with 8 GB RAM, 8-core Apple M1 (4 performance cores at 3.2 GHz and 4 efficiency cores at 2.0 GHz).

The experiment implements a simple integral estimation in C/MPI using the trapezoid methods. The calculated area under the curve is for the function $y = x^4$ over the range of 0 to 10. This area is estimated using $10,000,000,000$ trapezoids. Using PDCP, we deploy a 5-node infrastructure: one head container and four compute containers. Each container is configured with 2 GB of memory and 2 cores. Prior to the experiment, all non-critical applications are terminated, and it is confirmed that the platforms are running with a CPU utilization below 10%. Table 1 shows the runtime measurements of the experiment. On the Mac 1 platform, almost perfect scalability can be seen as the number of cores increase from 1 to 2. However, the runtimes plateau out and grow slightly worse as the number of cores increases from 2 to 4, 6, and then 8. This is likely due to the higher single-core performance of the Mac M4's performance cores, therefore making the communication overhead more noticeable faster. On the Windows 1 platform, almost perfect scalability can be seen with the core count sequence of 1, 2, and 4. The same performance stagnation is observed only for 6 and 8 cores. This is consistent with the hardware configuration (physical core counts) of the devices used in the experiment. The performance values of Windows 2 and Mac 2 devices demonstrate the consistency in scaling behaviors of PDCP across two different CPU architectures: Mac ARM and x86_64 (AMD and Intel). This is consistent with the visualization shown in Figure 2.

5 Conclusion

The deployment experience and performance evaluation discussed in the previous chapter have demonstrated the practicality and usability of PDCP in practical classroom settings. This is possible through advances in virtualization technologies and improvement in available hardware resources on personal computing devices.

As part of our future work, we intend to improve upon PDCP architectural design by adding additional components including a scheduler and a shared parallel file system. We will also branch out to convert *docker-compose.yml* into a Kubernetes service deployment YAML. This is so that PDCP's cross-platform capability can be extended to existing cloud infrastructure and not limited to just personal computing devices. Furthermore, building on the concept of PDCP, we will explore the idea of a general interactive-book framework, where the back-end components (similar to PDCP's compute containers) can be swapped out and replaced with relevant containers, such as those supporting database management, operating systems, big data engineering, and web development courses, to name a few.

The GitHub containing all the source codes for PDCP can be found at REDACTED.

References

[1] J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, et al., Xsede: accelerating scientific discovery, *Computing in science & engineering*, *16*(5):(2014), 62–74.

[2] R. Ricci, E. Eide, C. Team, Introducing cloudlab: Scientific infrastructure for advancing cloud architectures and applications, *; login:: the magazine of USENIX & SAGE*, *39*(6):(2014), 36–38.

[3] L. B. Ngo, J. Kilgannon, Virtual cluster for hpc education, *Journal of Computing Sciences in Colleges*, *36*(3):(2020), 20–30.

[4] S. M. Diesburg, P. A. Gray, D. Joiner, High performance computing environments without the fuss: the bootable cluster cd, in *19th IEEE International Parallel and Distributed Processing Symposium*, pages 8–pp (IEEE, 2005).

[5] I. Babic, A. Weeden, M. Ludin, S. Thompson, C. Peck, K. Muterspaw, A. F. Gibbon, J. Houchins, T. Murphy, Littlefe and bccd as a successful on-ramp to hpc, in *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*, page 73 (ACM, 2014).

[6] E. Shoop, R. Brown, E. Biggers, M. Kane, D. Lin, M. Warner, Virtual clusters for parallel and distributed education, in *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 517–522 (ACM, 2012).

[7] P. Marshall, M. Oberg, N. Rini, T. Voran, M. Woitaszek, Virtual clusters for hands-on linux cluster construction education, in *Proc. of the 11th LCI International Conference on High-Performance Clustered Computing* (2010).

[8] J. C. Adams, T. H. Brom, Microwulf: a beowulf cluster for every desk, *ACM SIGCSE Bulletin*, *40*(1):(2008), 121–125.

[9] D. Toth, A portable cluster for each student, in *2014 IEEE International Parallel & Distributed Processing Symposium Workshops*, pages 1130–1134 (IEEE, 2014).

[10] L. B. Ngo, Experience in teaching cloud computing with a project-based approach, *Journal of Computing Sciences in Colleges*, *38*(3):(2022), 107–119.

[11] L. B. Ngo, H. Bui, Sustainable and scalable setup for teaching big data computing, *Journal of Computational Science*, *14*(1).

[12] R. Keller Tesser, E. Borin, Containers in hpc: a survey, *The Journal of Supercomputing*, *79*(5):(2023), 5759–5827.

[13] N. Zhou, H. Zhou, D. Hoppe, Containerization for high performance computing systems: Survey and prospects, *IEEE Transactions on Software Engineering*, *49*(4):(2022), 2722–2740.

[14] S. Abraham, A. K. Paul, R. I. S. Khan, A. R. Butt, On the use of containers in high performance computing environments, in *2020 IEEE 13th International Conference on Cloud Computing (CLOUD)*, pages 284–293 (IEEE, 2020).

[15] C. Kozanitis, A. Bilas, Virtual clusters: Isolated, containerized hpc environments in kubernetes, in *High Performance Computing. ISC High Performance 2022 International Workshops: Hamburg, Germany, May 29–June 2, 2022, Revised Selected Papers*, volume 13387, page 347 (Springer Nature, 2023).

[16] University of Buffalo and Ohio Supercomputing Center, HPC Toolset Tutorial, https://github.com/ubccr/hpc-toolset-tutorial, 2025.

[17] J. T. Palmer, S. M. Gallo, T. R. Furlani, M. D. Jones, R. L. DeLeon, J. P. White, N. Simakov, A. K. Patra, J. Sperhac, T. Yearke, et al., Open xdmod: A tool for the comprehensive management of high-performance computing resources, *Computing in Science & Engineering*, *17*(4):(2015), 52–62.

[18] A. Bruno, D. Sajdak, Coldfront: Resource allocation management system, in *Practice and Experience in Advanced Research Computing 2021: Evolution Across All Dimensions*, pages 1–5 (2021).

[19] E. Chen, M. Asta, Using jupyter tools to design an interactive textbook to guide undergraduate research in materials informatics, 2022.

[20] N. Naik, Building a virtual system of systems using docker swarm in multiple clouds, in *2016 IEEE International Symposium on Systems Engineering (ISSE)*, pages 1–3 (IEEE, 2016).

Efficiency Transition in Matrix Multiplication: Hybrid Compilation Paths Outperform Traditional Approaches at Scale

ABSTRACT

There are potential benefits to performance by bypassing the traditional compilation steps of translating high-level code into machine code. By analyzing the theory of skipping the assembly phase, we aim to discover if this approach can lead to faster execution. The analysis considers compiler design, machine code generation techniques, and the limitations of current compilers. The findings suggest that, while direct machine code generation (compiling C→assembly→machine code) is feasible, its practicality depends on problem scale and domain-specific requirements. For large matrices (1500×1500), assembly-derived code achieved a 4.94% speed improvement over traditional compilation, demonstrating scalability advantages.

KEY WORDS

Matrix Multiplication, Performance Variability, Execution Performance, Computational Efficiency.

1. Introduction

Matrix multiplication is a cornerstone operation in computational mathematics. It has vast applications spanning fields from computer graphics and machine learning to scientific simulations [1]. The performance of matrix multiplication algorithms impacts the efficiency of countless systems making its optimization a pertinent focus in high-performance computing research.

Traditional implementations of matrix multiplication algorithms are in the time complexity of $O(n^3)$ which become increasingly expensive as the matrix dimensions grow. While there have been algorithmic improvements, such as Strassen's algorithm ($O(n^{2.807})$) [2] and Coppersmith-Winograd algorithm ($O(n^{2.376})$) [3], they often have excessive overhead for practical matrix sizes. Subsequently, much research has been focused on optimizing the standard algorithm through various techniques such as loop unrolling, tiling, vectorization, and parallelization.

While algorithmic advances are critical, the compilation process itself may introduce performance bottlenecks that remain underexplored. Previous research has studied optimization techniques for matrix multiplication, focusing on algorithmic improvements (Strassen, Coppersmith-Winograd), compiler optimizations, and hardware-specific tuning. However, little attention has been given to understanding the fundamental performance differences between code that is compiled directly to machine code versus code that is directly translated to assembly and then compiled, especially when optimizations are disabled to isolate compilation-pathway effects.

The question of whether a compiler's conversion to assembly can introduce inefficiencies is relevant as we consider the future of compiler design [4]. Some researchers have suggested that direct machine code generation from high-level languages might offer performance benefits by removing these intermediate steps [5]. On the contrary, others have suggested
that assembly is needed for code improvements that would be difficult to achieve.

This study isolates compilation effects by comparing unoptimized C code to it unoptimized assembly counterpart. By comparing the execution times across varying matrices dimensions (500×500, 1000×1000, 1500×1500), we aim to isolate the path of compilation itself, independent of optimization techniques.

This research is intended to contribute to compiler design and computational performance by analyzing it in several ways. First, it provides data on the performance differences between compiled C code and its assembly counterpart in a controlled environment. Second, it examines how these differences scale with problem size, revealing an unexpected trend where the performance gap tends to widen with greater matrices. Finally, it offers a perspective into the potential benefits or drawbacks resulting from bypassing the traditional compilation steps for computationally heavy tasks.

2. Body of Paper

Methodology

To investigate our hypothesis that traditional compilation pathways might introduce inefficiencies during the conversion from a high-level code to assembly, we intentionally isolated this variable. By comparing the performance of C code compiled directly to machine code against the same code first compiled to assembly and then to machine code-both with zero optimizations-we looked to accurately measure how the compilation pathway affects the efficiency of the execution.

Experimental Setup:
All experiments were conducted using the Windows Subsystem for Linux (WSL) running Ubuntu. Performance measures were obtained using the GNU profiler (gprof), a

standard tool for analyzing program execution times. For reliability, each test was performed ten times, with the results averaged to account for variability.

The test environment consisted of the following:

- Operating System: WSL Ubuntu
- Compiler: GCC (GNU Compiler Collection)
- Profiling Tool: GNU gprof
- Optimization Level: Disabled (-O0)

Implementation Details:

For this study, the matrix multiplication algorithm was the standard and written in C. To create the assembly version, the C code was compiled with the -S flag to generate assembly code. This was then assembled without modifications to maintain integrity by preventing the compiler from modifying the code generated from the initial script. This enabled us to isolate the impact of the compilation pathway rather than the differences in the algorithm implementation.

The compiler optimizations were deliberately disabled using the -O0 flag for both versions to allow focus on the differences between the compilation pathways rather than the compilers optimization capabilities.

Testing Procedures:

The procedures consisted of the following steps:

- Compile the C implementation directly to executable code
- Compile the C implementation to assembly, then compile the assembly to executable code
- Run each implementation ten times for the different matrix sizes
- Record execution times using gprof
- Calculate average execution times for each matrix size
- Compute the percentage difference in performance

Additionally, using ELF file generation was explored, but it was found that these implementations were consistently slower than both the C and assembly version. This suggest that the standard compilation pathway provided some benefits.

Results

The performance measures across the different matrix sizes revealed an interesting pattern regarding the efficiency of the assembly derived implementation compared to the direct C implementation.

The performance of the 500×500 matrix is as follows. The C implementation averaged 0.2358 seconds across ten runs with minimal variations standard deviation was approximately 0.003 seconds. The assembly derived implementation averaged 0.2402 seconds, with a deviation of approx. 0.0046 seconds, approximately 1.86% slower than the C version.

As the matrix size increased to 1000×1000, the performance shifted. The C implementation averaged 2.1278 seconds (Standard Deviation: 0.102 seconds). Whereas the assembly derived version 2.1238 seconds (Standard Deviation: 0.063 seconds). This represents a small improvement of approximately 0.19% for the assembly version.

With the 1500×1500 matrices, the trend became more noticeable. The C implementation averaged 10.0321 seconds (Standard Deviation: 0.908 seconds), while the assembly derived averaged 9.5366 seconds (Standard Deviation: 0.433 seconds). This shows a 4.94% improvement in favor of the assembly version.

Analysis

The observed pattern shows several important insights into the compilation process and its impact on computational efficiency. For smaller matrices (500×500), the assembly-derived implementation was marginally slower (1.86%) and slightly more variable ($\sigma = 0.0046$ vs. 0.003 seconds). However, as matrices grew to 1500×1500, the assembly version became both faster (4.94% improvement) and more consistent ($\sigma = 0.433$ vs. 0.908 seconds for C). This suggests that compilation pathways influence not only raw speed but also execution stability, particularly at scale.

There is a point where the assembly derived version transitions over from being slower to being faster. The crossover from c-dominated to assembly-dominated performance occurs around 1000×1000 matrices, where assembly begins to outperform C in both speed and consistency. The reduced variability in assembly implementations (e.g., $\sigma = 0.063$ seconds vs. 0.102 seconds for C at 1000×1000) aligns with our hypothesis that direct assembly pathways help to mitigate any inefficiencies in memory access or instructional scheduling. This trend strengthens for larger matrices. At 1500×1500, assembly achieves a 52% reduction in variability ($\sigma = 0.433$ vs. 0.908 seconds for c) alongside a 4.94% speed improvement. These results show that assembly-derived code scales more gracefully, particularly as memory hierarchy utilization (L1/L2/L3 caches, main memory) becomes a dominant factor in performance.

The observed performance patterns align with our theoretical premise discussed in the introduction. As matrix sizes increase, the compilation pathway becomes significant, potentially due to several underlying mechanisms. The assembly-derived execution may retain

structural characteristics that allow for more efficient memory access for large data structures. This becomes noticeable as the matrix size grows and memory hierarchy usage (L1/L2/L3 caches, main memory) becomes a dominate factor in performance.

Furthermore, modern CPUs use complex branch prediction and execution mechanisms that could respond to small variations in the instruction ordering. The direct c-to-machine code path and the c-to-assembly-to-machine code path likely result in slightly differing instruction sequences, which would explain the diverging performance characteristics as computational demands increase. These differences, while negligible for small workloads, could appear to compound at larger scales.

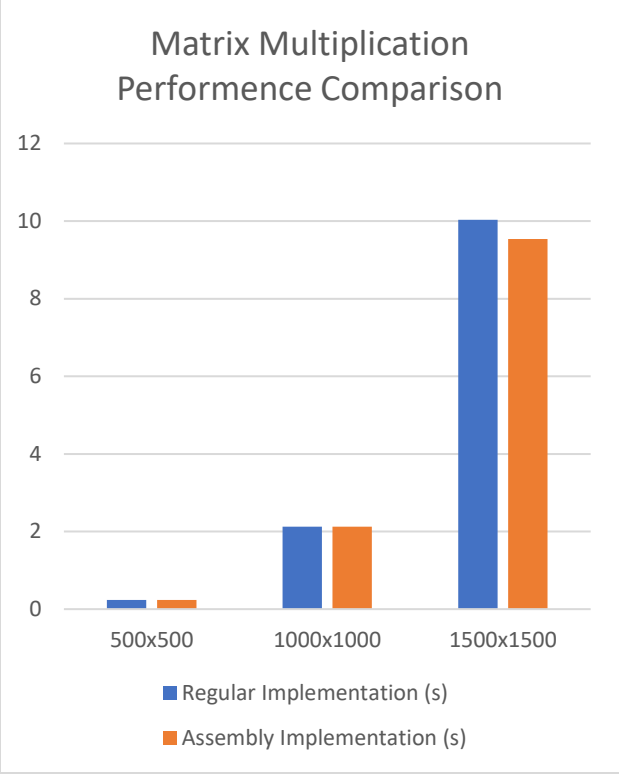
Several factors could contribute to the observed performance differences. First, as matrix sizes increase, memory access patterns play a larger role in performance. The assembly implementation may maintain more efficient memory access at larger scales. Another factor could be explained by the modern CPU. They use complex predictions that could be better used by the CPU that reduce pipeline stalls and branch mispredictions.

Caching behaviors also becomes quite critical with large matrices. Slight discrepancies in instruction scheduling between two compilation pipelines could result in different utilization patterns. These would be magnified with larger matrices.

Using the trends observed, we can estimate that the performance advantages would continue to increase for matrix sizes beyond 1500×1500. A rough estimation based on the previous data points show that for a 2000×2000 matrix, we would see an improvement of 8-10%. For a matrix that is 3000×3000, we would expect to see and improvement of 25-20%; assuming it follows the previous conventions.

These findings have significant implications for compiler designs and optimization strategies. The performance advantages observed from assembly derived code for larger computational tasks show that there may be value in revisiting traditional compilation pipelines. Future compiler development may benefit from including a hybrid approach that uses the advantages of assembly representation for certain computational patterns while maintaining the convenience of high-level language compilation for general purpose code. Additionally, these results highlight the importance of tailoring compilation to specific problem domains and scales rather than applying universal compilation approaches across all computational contexts.

2.1 Graphs, Tables, and Photographs



This graph compares the performance of regular and assembly implementations across three matrix sizes. For small matrices (500×500), the regular implementation shows a slight advantage. With medium-sized matrices (1000×1000), both implementations perform nearly identically. The assembly implementation only shows a substantial improvement with large matrices (1500×1500), achieving a 4.94% performance gain.

Matrix Size	C Time (s)	Assembly Time (s)	% Difference	Variability σ
500×500	0.2358	0.2402	+1.86	0.003 vs. 0.0046
1000×1000	2.1278	2.1238	-0.19	0.102 vs. 0.063
1500×1500	10.0321	9.5366	-4.94	0.908 vs. 0.433

3. Conclusion

This study investigated the performance of bypassing traditional compilation steps by comparing unoptimized C code against its assembly-derived counterpart for matrix multiplication. The findings reveal that while assembly implementations initially lag for smaller matrices, they exhibit significant advantages at scale. For larger matrices, the assembly version achieved a 4.94% speed improvement and 52% lower variability ($\sigma = 0.433$ vs. 0.908 seconds for C), demonstrating superior efficiency as memory hierarchy

utilization (caches, main memory) dominates performance. A critical crossover occurs at 1000x1000, where assembly transitions to outperforming C in both speed and consistency ($\sigma = 0.063$ vs. 0.102 seconds), suggesting that direct compilation pathways may mitigate inefficiencies in instruction scheduling or memory access. These results show how structural characteristics of assembly code, such as predictable memory patterns and reduced pipeline stalls, align with modern CPU architectures, compounding benefits for large-scale computations.

However, the study has limitations. Experiments were conducted on WSL, potentially introducing overhead compared to native environments, and relied solely on GCC with optimizations disabled, limiting real-world applicability where optimizations are typically used. Limitations for larger matrices (e.g., 8–10% improvement at 2000×2000) remain speculative and require empirical validation.

These findings hold promise for applications demanding high-performance computing, such as machine learning or scientific simulations, where assembly-derived code could enhance both speed and predictability. Real-time systems, reliant on consistent execution times, may also benefit from its reduced variability. From a compiler design perspective, hybrid approaches could selectively delegate performance-critical routines to assembly-generation modules, while retaining high-level language convenience for general-purpose code.

Further work could explore the impact of enabling compiler optimizations across both pathways, validating results on native hardware and diverse compilers, and investigate automated tools for targeted assembly generation. While traditional compilation pathways remain practical for most applications, this work highlights the value of rethinking strategies for large-scale tasks by tailoring compilation approaches to problem scale and domain. Future systems could unlock significant gains in both performance and consistency.

References:

[1] K. Goto and R. A. van de Geijn, Anatomy of high-performance matrix multiplication, *ACM Transactions on Mathematical Software* 34 (2008), 1–25.
Link: https://doi.org/10.1145/1356052.1356053

[2] V. Strassen, Gaussian elimination is not optimal, *Numerische Mathematik* 13 (1969), 354–356.
Link: https://doi.org/10.1007/BF02165411

[3] D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progressions, *Journal of Symbolic Computation* 9 (1990), 251–280.
Link: https://doi.org/10.1016/S0747-7171(08)80013-2

[4] C. Lattner and V. Adve, LLVM: A compilation framework for lifelong program analysis & transformation, Proc. International Symposium on Code Generation and Optimization (CGO), San Jose, CA, 2004, 75–86.
Link: https://llvm.org/pubs/2004-01-30-CGO-LLVM.html

[5] B. Kuzma, I. Korostelev, J. P. L. de Carvalho, J. E. Moreira, C. Barton, G. Araujo, and J. N. Amaral, Fast matrix multiplication via compiler-only layered data reorganization and intrinsic lowering, arXiv:2305.18236 (2023).
Link: https://arxiv.org/abs/2305.18236

LINTUNET: A HYBRID TRANSFORMER-CNN ARCHITECTURE FOR BRAIN TUMOR SEGMENTATION

Lawrence Menegus, Dongsheng Che
East Stoudsburg University
lmenegus@live.esu.edu, dche@esu.edu

ABSTRACT

Brain tumor segmentation is vital in medical imaging for accurately identifying tumor regions in MRI scans. While U-Net and other CNN-based models have been widely used, they struggle with capturing long-range dependencies due to their localized receptive fields. To overcome this, we propose LinTUNet, a Hybrid Linformer Transformer U-Net, which integrates sparse self-attention into the U-Net architecture using a Sparse Linformer for efficient attention computation. By incorporating Transformer-based attention in the bottleneck layer, our model enhances feature extraction while maintaining computational efficiency. Performance evaluation using key metrics shows that LinTUNet surpasses U-Net in segmentation accuracy, achieving higher IoU and F1-scores. This paper presents our approach to developing a Hybrid Transformer-CNN model, demonstrating its advantages in medical image segmentation and deep-learning-driven healthcare applications.

KEY WORDS

Brain Tumor Segmentation; U-Net; Transformer; Sparse Linformer; Attention Layer; Convolutional Neural Network

1. Introduction

Deep learning-based segmentation models have become a critical component of medical imaging, particularly for identifying and segmenting brain tumors in MRI scans. Image segmentation enables precise analysis of specific structures, which is essential for accurate diagnosis and treatment planning [1]. Traditional Convolutional Neural Network (CNN)-based architectures, such as U-Net, have shown significant success in biomedical image segmentation [2]. However, CNN-based models face several challenges that limit their performance in segmenting complex tumor structures.

One major limitation of CNNs is their restricted receptive field, which prevents them from capturing long-range dependencies and global contextual relationships within medical images [3]. CNNs utilize small convolutional kernels, making it difficult to segment tumors with irregular boundaries accurately [4]. As a result, segmentation errors often occur, particularly when tumor shapes and sizes vary significantly across patients. Additionally, CNNs require extensive computational resources and long training times, making them less practical for real-time clinical applications [5]. Another drawback is their poor global awareness, as CNNs operate on local patches rather than considering the entire image context, leading to inconsistent boundary predictions [6]. CNN-based models depend on large annotated datasets, which are costly and time-consuming to obtain in the medical domain [6]. These challenges highlight the need for a segmentation model that can efficiently capture global spatial relationships while maintaining computational feasibility.

To address these issues and limitations, we propose LinTUnet, a hybrid Transformer-CNN model that integrates convolutional layers for local feature extraction with a Transformer-based Sparse Linformer self-attention mechanism to capture long-range dependencies. By combining the strengths of CNNs and Transformers, LinTUnet enhances segmentation accuracy while optimizing computational efficiency, making it particularly suitable for brain tumor segmentation in MRI scans.

LinTUNet leverages both Linformer and Sparse Transformer mechanisms to overcome the limitations of traditional CNN-based segmentation models, particularly in handling complex tumor structures in MRI scans. Linformer addresses the inefficiency of standard Transformers by reducing the quadratic complexity $O(n^2)$ of self-attention to linear complexity *O(n)* [7]. Instead of computing full self-attention matrices, Linformer compresses key and value projections into a lower-dimensional representation, significantly improving computational efficiency while preserving essential long-range dependencies. This allows LinTUNet to capture global spatial relationships without the excessive memory and processing requirements of full Transformers, making it feasible for high-resolution medical imaging [7].

Meanwhile, the Sparse Transformer enhances

segmentation accuracy by focusing computational resources on the most relevant spatial features rather than processing the entire image uniformly. Unlike traditional self-attention, which treats all pixels equally, the Sparse Transformer selectively attends to a subset of tokens, prioritizing regions with important tumor-related features [8]. This sparsity reduces redundancy and enhances the model's ability to detect tumors with varying shapes and textures while maintaining computational efficiency.

Together, Linformer and Sparse Transformer complement each other in LinTUNet. Linformer ensures that the model can process long-range dependencies efficiently, while the Sparse Transformer improves segmentation precision by directing attention to critical areas within the MRI scan [8]. By integrating both mechanisms, LinTUNet achieves superior segmentation accuracy while optimizing computational feasibility, making it a powerful solution for brain tumor segmentation in medical imaging.

2. Overview of U-Net (CNN) Architecture

U-Net is a convolutional neural network (CNN) designed for image segmentation, particularly excelling in pixel-level classification tasks such as tumor identification in MRI scans. It follows an encoder-decoder structure with skip connections, allowing it to segment medical images efficiently while preserving fine-grained spatial details.

The encoder compresses spatial information through downsampling operations, while the decoder reconstructs the segmented regions using transposed convolutions and concatenated feature maps from the encoder. This architecture effectively retains both low-level and high-level features, making it ideal for medical imaging tasks [2].

2.1 Encoder

The encoder consists of multiple downsampling blocks, each containing:

- Two convolutional layers with ReLU activation
- Batch normalization for stabilizing training
- Max pooling for reducing spatial dimensions

This hierarchical feature extraction process helps the network learn both fine-grained and abstract features, which are essential for precise image segmentation[2].

2.2 Bottleneck

The bottleneck layer is positioned between the encoder and decoder, capturing the most abstract representations of the input image. It acts as a bridge between feature extraction and segmentation refinement, ensuring essential patterns are preserved before reconstruction begins [4].

2.3 Decoder

The decoder restores the image resolution while maintaining segmentation accuracy. It achieves this by:

- Upsampling through transposed convolutions
- Concatenating feature maps from the encoder layers
- Applying double convolutions to refine segmentation accuracy

This ensures the precise reconstruction of the predicted mask.

2.4 Skip Connections

Skip connections play a crucial role in U-Net by passing detailed spatial information from the encoder to the decoder. This prevents information loss and improves segmentation precision, particularly for complex structures like tumors [2].

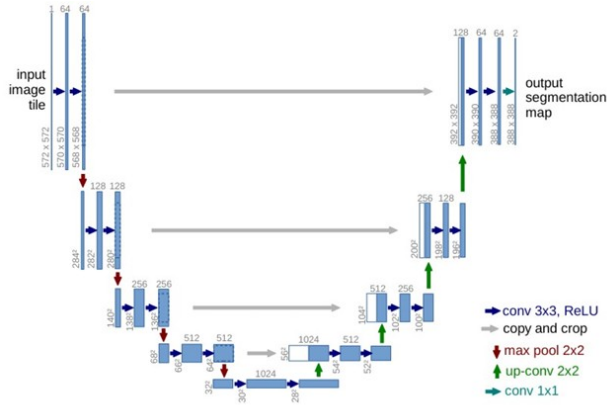


Figure 1: The U-Net Architecture proposed by Ronneberger et al. In 2015 [2].

3. Proposed Model: LinTUNet

To address the limitations of U-Net and CNN-based models, we introduce LinTUNet, a hybrid Transformer-CNN model that enhances segmentation accuracy by incorporating Transformer-based self-attention into the U-Net bottleneck.

3.1 Architecture of LinTUNet

LinTUNet builds upon the CNN based U-Net's encoder-decoder framework but enhances it with Linformer-based self-attention at the bottleneck stage to improve feature representations. Like a traditional Unet CNN this hybrid consists of a Decoder, Bottleneck or in this case a linformer attention layer bottleneck and a decoder.

The encoder (downsampling path) and decoder (upsampling path) are the same as the traditional CNN. The encoder consists of multiple convolutional layers for local feature extraction, followed by Batch Normalization, Dropout, and ReLU activations to improve generalization and prevent overfitting. These layers progressively reduce the spatial dimensions while capturing hierarchical features crucial for accurate segmentation. The decoder utilizes transposed convolutions to restore spatial resolution, while skip connections from the encoder reintegrate fine-grained details lost during downsampling. Finally, a 1x1 convolution is applied to produce the segmented tumor mask, ensuring precise localization of tumor regions.

3.2 Sparse Linformer Attention Layer

At the bottleneck stage, Sparse Linformer transforms the encoded feature map into a sequence representation to apply self-attention, effectively capturing long-range dependencies and improving segmentation consistency. Figure 2 illustrates an example of our proposed Sparse Linformer Attention Layer in the LinTUNet bottleneck, highlighting its role in feature transformation and efficient attention computation.

A Linformer reduces the computational complexity of the standard Transformer attention mechanism from $O(N^2)$ to $O(N)$ by projecting the sequence into a lower-dimensional space [7]. This is achieved using a low-rank approximation of the attention matrix, making it computationally efficient for tasks like brain tumor segmentation.

In standard self-attention, the attention matrix is computed as:

$$A = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \quad (1)$$

where

$$Q = XW_Q, K = XW_K, V = XW_V \quad (2)$$

represent the query, key, and value matrices. The complexity of this computation is $O(N^2)$, making it impractical for large medical images [7].

The Linformer approximates this by introducing low-rank projections for the key and value matrices:

$$K' = EK, V' = FV \quad (3)$$

where

$$E, F \in \mathbb{R}^{rxN} \quad (4)$$

are projection matrices that reduce the sequence length from N to r ($r \ll N$) [7]. The modified attention computation becomes:

$$A = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \quad (5)$$

leading to a output of:

$$Linformer - Attention(Q, K, V) = A'V' \quad (6)$$

This transformation reduces computational complexity to $O(N)$ while maintaining segmentation accuracy. For our proposed Model we used a Sparse Linformer. A sparse Linformer further optimizes this approach by applying sparsity constraints on the projection matrices E and F, ensuring that only a subset of elements contribute to the computation [8,9]:

$$K' = S(E)K, V' = S(F')V \quad (7)$$

where S(E) and S(F) select the most significant rows to preserve spatial information while reducing memory usage.

The concept of sparse attention mechanisms was introduced in the paper "Generating Long Sequences with Sparse Transformers" by Child et al.[9], which demonstrated that sparse factorizations of the attention matrix can reduce computational complexity while effectively modeling long-range dependencies. LinTUNet uses this mixture of a Sparse and Linformer attention layer in a sequence during the bottleneck stage which then is reshaped back into a 2D feature map before entering the decoder.

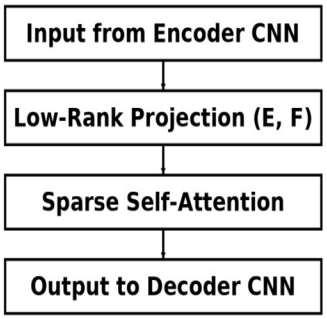


Figure 2. Example of our proposed Sparse Linformer Attention Layer of the LinTUNet Bottleneck.

The Linformer reduces the size of the attention layer by compressing pixel-to-pixel interactions and projecting the information into a lower-dimensional space. It leverages low-rank approximations, ensuring that only the most relevant information is retained from the attention calculations.

The Sparse Linformer further enhances this approach by selecting only the most important rows from the compressed data, significantly reducing memory usage while maintaining the model's ability to capture essential spatial relationships.

4. Training and Model Evaluation

4.1 Data Collection

The dataset used in this study comprises 2,146 MRI brain scan images specifically curated to detect brain tumors, sourced from a Kaggle dataset for semantic segmentation tasks [10]. To optimize the training process, the dataset was divided into three subsets: 1,501 images for training, 429 images for validation, and 215 images for testing. This division ensured the models were trained on a diverse set of labeled data while being fine-tuned and evaluated on distinct subsets, thereby enhancing their ability to generalize to unseen data. Each MRI image is accompanied by pixel-wise annotations, providing segmentation masks that delineate the tumor regions. This segmentation approach is particularly critical for models like U-Net and LinTUNet, as it enables them to not only detect the presence of tumors but also accurately identify tumor boundaries. The dataset includes a variety of MRI images showcasing tumors with different shapes, sizes, and locations, allowing the models to learn comprehensive feature representations across diverse tumor characteristics.

4.2 Feature Selection and Augmentation

This study utilizes PyTorch for medical image segmentation, specifically targeting brain MRI scans. A custom preprocessing pipeline was developed to pair MRI images with their corresponding segmentation masks, formatted in COCO-style annotations [11]. The pipeline incorporates data augmentation techniques such as resizing, grayscale conversion, tensor conversion, and normalization to enhance the training process and improve model generalization. The dataset is structured into distinct directories for training, validation, and testing, with mechanisms to ensure accurate pairing of images and masks. Additionally, a custom dataset class, built on PyTorch's Dataset class, facilitates efficient batch processing and GPU-accelerated transformations via CUDA, significantly reducing training times [12].

The implementation includes functions for managing COCO-style annotations and visualizing image samples overlaid with segmentation masks, enabling quality assurance. Error handling mechanisms were integrated to address mismatches between images and masks, ensuring only valid pairs are used during training. These features combined with the robust data pipeline, contribute to the model's accuracy, efficiency, and scalability in the medical imaging domain.

4.3 Mask Generation and Dataset Preparation

In this study, segmentation masks for the MRI brain scans were generated using the COCO annotation format, which provides pixel-level annotations of tumor regions. The dataset's annotation file, stored in JSON format, was processed to extract segmentation data for each MRI image. A custom create_mask function was developed to iterate through the segmentation points specified in the annotations. These points, stored as a list of coordinates outlining the tumor boundary, were converted into polygonal shapes using the skimage.draw.polygon function. This method filled the tumor regions with white pixels (255), effectively creating binary masks that highlighted the tumor areas. The generated masks were saved as TIFF files, with each mask corresponding to a specific MRI image. Figure 3 illustrates a Brain Scan MRI image with the COCO-Style Annotation, demonstrating the pixel-level segmentation used in this study.

The images and their corresponding segmentation masks were organized into separate directories for training, validation, and testing. A *mask_folders_if_not_exist* function was using dynamic programming implemented to ensure that masks were only generated if the output directories did not already exist, thus avoiding redundancy. To maintain data integrity, a *compare_folders* function was developed to identify and remove mismatched images and masks. Finally, a *train_test_split* function was employed to partition the data into training, validation, and test sets, ensuring consistency across the models. This structured approach ensured that the data was well-prepared for model training and evaluation, enabling robust performance for both U-Net and LinTUNet.

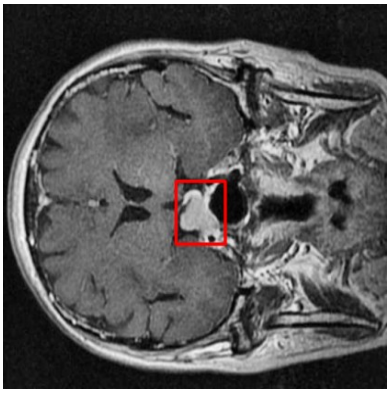


Figure 3. Brain Scan MRI image with the COCO-Style Annotation

4.4 Training Strategy

The LinTUNet model is trained using Binary Cross-Entropy with Logits Loss (BCEWithLogitsLoss), a widely used loss function for binary segmentation tasks like tumor detection in brain MRI scans. This loss function combines the sigmoid activation function with binary cross-entropy loss, ensuring stable numerical computation and penalizing incorrect predictions. The BCE loss is calculated as:

$$L=-\frac{1}{N}\sum_{i=1}^{N}\left[y_i\log\left(\sigma\left(x_i\right)\right)+\left(1-y_i\right)\log\left(1-\sigma\left(x_i\right)\right)\right] \quad (8)$$

where

y_i is the ground truth label, x_i is the predicted value (logits), and

$$\sigma\left(x_i\right)=\frac{1}{\left(1+e^{-x_i}\right)} \quad (9)$$

is the sigmoid activation function. This loss function is particularly effective in handling imbalanced datasets, which is crucial for medical imaging applications [13].

For optimization, the Adam optimizer [14] is chosen due to its adaptive moment estimation, which dynamically adjusts learning rates based on past gradient updates. Adam's parameter updates follow:

$$m_t=B_1 m_{t-1}+\left(1-B_1\right)g_t \quad (10)$$

$$v_t=\beta_2 v_{t-1}+\left(1-\beta_2\right)g_t^2 \quad (11)$$

$$m_t=m_t\frac{m_t}{1-\beta_1^t},\ v_t=\frac{v_t}{1-\beta_t^t} \quad (12)$$

$$\theta_t=\theta_{t-1}-\alpha\frac{m_t}{\sqrt{v_t}+\epsilon} \quad (13)$$

where m_t, v_t are the first and second moment estimates of the gradient, β_1, β_2 control the decay rates, and α is the learning rate.

To further stabilize training for the models, a StepLR learning rate scheduler is applied, which reduces the learning rate at predefined intervals:

$$\alpha_t=\alpha_0\gamma^{\lfloor t/step\rfloor} \quad (14)$$

where α_0 is the initial learning rate, γ is the decay factor, and t is the current epoch number. This ensures that the learning rate is high during initial epochs for rapid learning but decays over time for fine-tuning.

To accelerate training and reduce memory consumption, Automatic Mixed Precision (AMP) is employed. AMP dynamically switches between single-precision (FP32) and half-precision (FP16) computations, allowing for larger batch sizes and faster training while maintaining numerical stability [15]. A critical component of AMP is gradient scaling, which prevents underflow issues in FP16 computations by scaling the gradients before backpropagation:

$$g=Sg \quad (15)$$

where S is a scaling factor applied to the gradients g, ensuring that small values do not get truncated to zero. This approach significantly speeds up deep learning models trained on large medical datasets without affecting accuracy [15].

The model training process follows a structured pipeline that includes data augmentation, GPU acceleration, and progressive learning rate adjustments. Each epoch begins with a forward pass, where in put MRI images are processed through the encoder, Sparse Linformer attention layer, and decoder. The model's output is compared against ground truth segmentation masks using BCEWithLogitsLoss, and gradients are computed for backpropagation. To prevent vanishing or exploding gradients, gradient clipping is applied by restricting the norm of the gradients:

$$g=\frac{g}{max\left(1,\frac{/\!/\,g\,/\!/}{c}\right)} \quad (16)$$

where g is the gradient vector and c is the clipping threshold. The validation phase follows each training epoch, assessing model performance on unseen data. The learning rate scheduler is updated at predefined intervals to stabilize training and enhance generalization[14].

To calculate the overall accuracy we used the pixel-wise accuracy calculation for model training and evaluation.

Which used the raw predictions from the model (logits) are passed through a sigmoid function to convert them into binary values (0 or 1). The same process is applied to the ground truth masks to ensure consistency. The predicted and actual masks are then compared element-wise, and the number of correctly classified pixels is summed. Finally, this sum is divided by the total number of pixels in the mask to obtain the accuracy.

$$Accuracy = \frac{\left(\sum_i^n 1\left(y_{(pred_i)} = y_{(true_i)}\right)\right)}{N} \quad (17)$$

In pixel-wise accuracy calculation $y_{(pred_i)}$, represents the predicted binary mask, while $y_{(true_i)}$ is the actual ground truth mask. The total number of pixels in the mask is denoted by N. Accuracy is computed as the ratio of correctly classified pixels to the total pixels, using the indicator function $1(\cdot)$, which returns 1 if the prediction matches the ground truth and 0 otherwise. This ensures an accurate measure of segmentation performance across the dataset [17].

4.5 Segmentation Performance Metrics

To evaluate segmentation performance, Intersection over Union (IoU), Dice Score (F1 Score), Precision, and Recall are computed. IoU, also known as the Jaccard Index, measures the degree of overlap between predicted and ground truth segmentation masks and is defined as:

$$IoU = \frac{A \cup B}{A \cap B} \quad (18)$$

Higher IoU values indicate better segmentation performance [18]. Similarly, the Dice Score (F1 Score) quantifies segmentation accuracy by measuring the harmonic mean of precision and recall:

$$Dice = 2\frac{A \cap B}{A + B} \quad (19)$$

This metric is particularly useful in medical imaging, as it balances false positives and false negatives.

Precision and recall further assess the model's reliability. Precision evaluates the proportion of correctly predicted tumor pixels relative to all predicted tumor pixels:

$$Precision = \frac{TP}{TP + FP} \quad (20)$$

where Tp = True Postitives, FP = False Positives, FN = False Negatives. These metrics ensure that the model does not produce excessive false positives or false negatives [19].

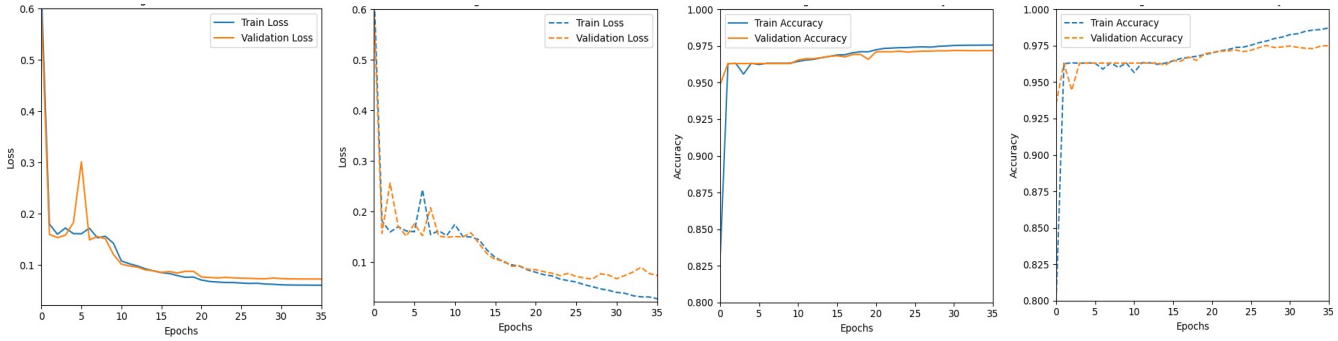
Throughout training, key performance indicators such as training and validation loss, accuracy, IoU, precision, and F1-score are recorded per epoch to monitor the model's learning progress. This ensures early detection of overfitting or underperformance, allowing necessary adjustments to be made. Once trained, the final model is evaluated on an independent test set to validate its generalization ability on unseen MRI scans. By integrating Sparse Linformer self-attention in the bottleneck stage, BCEWithLogitsLoss, Adam optimization with learning rate scheduling, and mixed precision training, LinTUNet achieves an efficient and accurate segmentation while maintaining computational efficiency. This combination of Transformer-based self-attention and convolutional feature extraction enables the model to effectively capture both local and global contextual information, making it highly suitable for image segmentation tasks.

5. Results and Discussion

5.1 Training and Validation Loss and Accuracy

This project was conducted on Google Colab, utilizing Google Cloud's computational resources. The results presented below are based on one of several trial runs of the Jupyter notebook. Due to variability in cloud resource allocation, minor differences may occur between runs. However, LinTUNet consistently outperformed U-Net across all trials, with the extent of improvement varying. Despite these minor fluctuations, the overall trends and conclusions remain impressive and robust. The following graphs and visuals represent one such trial, providing a detailed analysis of U-Net (CNN) and LinTUNet (Transformer-CNN), highlighting their respective loss functions, accuracy trends, and segmentation performance.

Demonstrated in Figure 4, LinTUNet slightly outperforms U-Net in both accuracy and loss reduction during the entirety of both model's training and evaluation. In visuals (a) and (b), the loss curves of U-Net and LinTUNet, respectively, indicate that LinTUNet achieves a slightly lower loss. Moreover, LinTUNet attains an impressive 98.95% accuracy, slightly surpassing U-Net's 97.57%, as illustrated in visuals (c) and (d). This suggests that LinTUNet not only enhances pixel-wise classification accuracy but also ensures more consistent and precise segmentation across the dataset. The combination of lower loss and higher accuracy reinforces LinTUNet's ability to effectively capture long-range dependencies and spatial relationships, leading to superior segmentation performance compared to traditional CNN-based architectures.



(a) U-Net Loss (b) LinTUNet Loss (c) U-Net Accuracy (d) LinTUnet Accuracy

Figure 4. U-Net and LinTUNet loss and accuracy curves, illustrating model performance. (a) U-Net's loss curve. (b) LinTUNet's loss curve. (c) U-Net's accuracy curve. (d) LinTUNet's accuracy curve.

(a) Input Image (b) U-Net's Prediction (c) LinTUnet Prediction

Figure 5. Model predictions for the Image Segmenation. (a) The input image used for both U-Net and LinTUNet models. (b) U-Net's predicted segmentation mask (c) LinTUNet's predicted segmentation mask.

5.2 Segmentation Performance

Table 1 presents a comprehensive comparison of UNet (CNN) and LinTUNet across key segmentation performance metrics this includes F1 Score, Intersection of Union and overall precision.

Table 1. Results of the all metrics of U-Net and LinTUNet

Metric	U-Net (CNN)	LinTUNet (ours)
F1 Score	0.6013	**0.8675**
IoU	0.4321	**0.7668**
Precision	0.7871	0.7871

As shown in Table 1, LinTUNet significantly outperforms U-Net in image segmentation across multiple key metrics. It achieves an impressive F1 Score of 0.8675, compared to U-Net's 0.6013, demonstrating a much better balance between precision and recall. This indicates that LinTUNet generates fewer false positives and false negatives, resulting in more accurate and reliable segmentation

Another crucial metric, Intersection over Union (IoU), further highlights LinTUNet's superiority in spatial localization. With an IoU of 0.7668—considerably higher than U-Net's 0.4321—LinTUNet more accurately aligns predicted tumor regions with the input images. This improvement is visually evident in Figure 5, where LinTUNet's segmentation (c) closely matches the input image of the tumor related area (a), whereas U-Net's prediction (b) deviates significantly, with the segmented tumor area appearing overly broad and imprecise.

Interestingly, both models achieve an identical precision of 0.7871, indicating that they identify tumor pixels with similar correctness. However, LinTUNet's higher F1 Score and IoU suggest superior overall segmentation by reducing false negatives, making it more effective at detecting the tumor regions.

This analysis covers the entire prediction performance of both models, confirms that LinTUNet consistently delivers more precise and accurate tumor segmentation

than U-Net, making it a more reliable approach for medical image analysis.

5.3 Execution Time

Beyond accuracy and other performance metrics, LinTUNet offers a significant advantage in inference speed, making it much more efficient for real-time applications. It processes an image in just 0.0002 seconds (0.2 milliseconds), whereas U-Net takes 0.0014 seconds (1.4 milliseconds). This means LinTUNet generates predictions 7 times faster than U-Net, a crucial improvement for high-throughput medical imaging and real-time diagnostic applications. A key contributor to this efficiency could be from the automatic mixed precision (AMP) autocasting, which optimizes computations by dynamically switching between single-precision (FP32) and half-precision (FP16) during training, helping to increase processing power and computational efficiency.

6. Conclusion

Our proposed hybrid Transformer-CNN model, LinTUNet, consistently outperforms traditional CNN-based methods like U-Net in image segmentation. By integrating Sparse Linformer self-attention into the U-Net architecture, LinTUNet captures long-range dependencies while maintaining computational efficiency. Its ability to deliver more accurate image segmentations while significantly reducing processing time makes it highly suitable for real-time medical applications. Having demonstrated the advantages of this approach, we anticipate future studies applying LinTUNet to larger and more diverse medical imaging datasets, including multi-modal MRI scans segmentation tasks, to further validate its effectiveness. By harnessing the power of Transformers for medical imaging, LinTUNet has the potential to improve diagnostic accuracy, enhance early detection, and ultimately contribute to better patient outcomes, potentially saving lives.

References:

[1] G. Litjens et al., "A survey on deep learning in medical image analysis," Medical Image Analysis, vol. 42, pp. 60-88, 2017. [Online]. Available: https://doi.org/10.1016/j.media.2017.07.005

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," arXiv preprint, arXiv:1505.04597, 2015. [Online]. Available: https://arxiv.org/abs/1505.04597

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint, arXiv:1409.1556, 2014. [Online]. Available: https://arxiv.org/abs/1409.1556

[4] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-Net: Learning dense volumetric segmentation from sparse annotation," Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2016, pp. 424-432. [Online]. Available: https://doi.org/10.1007/978-3-319-46723-8_49

[5] F. Isensee et al., "nnU-Net: A self-adapting framework for U-Net-based medical image segmentation," Nature Methods, vol. 18, pp. 203-211, 2021. [Online]. Available: https://doi.org/10.1038/s41592-020-01008-z

[6] B. H. Menze et al., "The multimodal brain tumor image segmentation benchmark (BraTS)," IEEE Transactions on Medical Imaging, vol. 34, pp. 1993-2024, 2015. [Online]. Available: https://doi.org/10.1109/TMI.2014.2377694

[7] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," arXiv preprint, arXiv:2010.11929, 2020. [Online]. Available: https://arxiv.org/abs/2010.11929

[8] A. Wang et al., "Linformer: Self-attention with linear complexity," arXiv preprint, arXiv:2006.04768, 2020. [Online]. Available: https://arxiv.org/abs/2006.04768

[9] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," arXiv preprint, arXiv:1904.10509, 2019. [Online]. Available: https://arxiv.org/abs/1904.10509

[10] P. Darabi, "Brain tumor image dataset (semantic segmentation)," Kaggle, 2021. [Online]. Available: https://www.kaggle.com/datasets/pkdarabi/brain-tumor-image-segmentation

[11] M. Deng, K. He, and R. Girshick, "COCONut: Modernizing COCO segmentation," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2024/papers/Deng_COCONut_Modernizing_COCO_Segmentation_CVPR_2024_paper.pdf

[12]Deep learning software, NVIDIA Developer, 2023. [Online]. Available: https://developer.nvidia.com/deep-learning-software

[13] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, Cambridge, MA: MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint, arXiv:1412.6980, 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[15] P. Micikevicius et al., "Mixed precision training," International Conference on Learning Representations (ICLR), 2018. [Online]. Available: https://openreview.net/forum?id=r1gs9JgRZ

[16] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3431-3440. [Online]. Available: https://doi.org/10.1109/CVPR.2015.7298965

[17] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. Jorge Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, 2017, pp. 240-248. [Online]. Available: https://doi.org/10.1007/978-3-319-67558-9_28

[18] J. P. Cohen, G. Bertin, and H. Frappier, “Mitigating Bias in Radiology Machine Learning: 3. Performance Metrics,” Journal of Medical Imaging, vol. 9, no. 4, 2022. [Online].Available:https://pmc.ncbi.nlm.nih.gov/articles/PMC9530766

[19] A. J. Carrillo-Perez, P. Jimenez-Carretero, and R. Gonzalez-Diaz, “Towards a guideline for evaluation metrics in medical image segmentation,” BMC Research Notes, vol. 15, no. 1, pp. 1–10, 2022. [Online]. Available: https://bmcresnotes.biomedcentral.com/articles/10.1186/s13104-022-06096-y

Birds of a Feather

Shared Course Planning in PASSHE
Hosted by:

Charles Girard Shippensburgh University cdgira@ship.edu

PASSHE has started a push for course sharing across the state system. Their new share portal, Academic Sharing | PA State System of Higher Education, is set up to help students find courses they can take at other institutions more easily. Rather than just choosing courses at random and offering them, I am hoping to discuss how the various computer science programs across the PASSHE system might coordinate to use this resource to strengthen our programs. For example, knowing that one school will have a C programming course offered in the fall so that we can dedicate resources to running an elective or an additional general education course. The first would just be identifying common courses and then how we could balance them so enrollments stay strong while hopefully giving students more flexibility.

Mobile App Development: Teaching Strategies, Experiences, and Best Practices
Hosted By

Dave Tucker,	Pennsylvania Western University (Edinboro),	dtucker@pennwest.edu,
Alawya Alawami	Pennsylvania Western University (Clarion)	aalawami@pennwest.edu

Mobile application development has become an integral part of computer science and information systems curricula, yet instructors face numerous challenges in teaching this subject. These challenges include selecting appropriate development environments, choosing programming languages, and deciding which mobile operating system to target. Additionally, designing assignments that balance complexity with practical integration on actual devices can be difficult. Additional discussion will center around how to accommodate students who do not have access to Android devices, given iOS is difficult to deploy on.

This Birds of a Feather (BoF) session aims to foster a PASSHE community of mobile app instructors, facilitating the exchange of teaching strategies, experiences, and best practices. We will explore various development environments, including Android-only, iOS-only, and cross-platform solutions, as well as discuss methods for deploying applications on iPhones. Furthermore, this session will serve as a platform to share both successful and challenging experiences, with the goal of creating a repository of student project ideas.

Abstracts

2025: PACISE

Title: Design and development of 3D UI layout for virtual and augmented reality using a model-based design tool

Abstract:

Virtual and augmented reality (VR, AR) has an increasing impact on the market in many fields, from education and medicine to engineering. The design of inclusive and immersive user interfaces for virtual and augmented reality systems remains challenging for the human-computer interaction (HCI) community. However, building an optimal 3D user interface (UI) design is not an easy process due to the noisiness of user behavior and the variability of user preferences. Additionally, designing 3D user interfaces (UIs) with limited research to understand the usability of VR/AR systems is detrimental to the goal of providing rich, inclusive experiences for all users.

In this paper, we first explore the foundations of ability-based design and design engineering used in engineering to design products and systems. Thus, by using techniques from design engineering, we systematically investigate the design parameters that dominate user performance and comfort when interacting with UI layouts. So, this research aims to explore the effects of physical ergonomics, cognition, and visual perception on the usability of VR and AR systems. Then, we categorize the identified parameters into the relevant controllable and uncontrollable parameters that dominate user performance and comfort when interacting with 3D user interfaces. Due to the difficulty in extracting data in the field from actual users or generating realistic data from proxy users, we use a model-based approach that involves two steps: (1) identification and examination of pertinent models of human performance and (2) determination of the optimal settings of controllable parameters using these models. A model-based approach offers the potential for cost and time-effective evaluation of user performance without the need for intrusive measures.

Second, we create a novel model-based design toolkit that facilitates the design, creation, and exploration of inclusively immersive 3D UI layouts. After the identification of optimal parameter settings, we will integrate these settings into our UI design toolkit and enable the toolkit to generate the most optimal UI given specifications set by the designer. This toolkit can be used to construct UI layouts that accommodate the user's unique perceptual, cognitive, and physical capabilities. Additionally, we use the design parameter analyses to convert the parameters into predictive models, which are then used to construct a single objective function to optimize. Additionally, we apply a user-involved approach to the toolkit through preference learning to integrate the designer’s feedback into the optimization process and suggest alternative configurations pertaining to user capability at design time.

Finally, we design our UI design toolkit for the Hololens 2, an optical see-through HMD device. The Hololens 2 provides an AR experience by projecting holograms onto the user’s real-world environment. The Hololens 2 also utilizes spatial mapping, which provides a detailed representation of real-world surfaces around the device's environment. Using Hololens 2, we

define the interaction space of this AR system as a 3D Cartesian grid consisting of locations where a person can reach and manipulate objects with a fixed torso position. The interaction space is divided into elements called voxels, which are used to determine the optimal placement of UI elements in terms of physical ergonomics, text readability, and color harmony, as described in the following subsections. We also show how to generate an optimized UI layout using preference learning. Our research has shown that utilizing a competency-based design perspective can be beneficial in focusing on competencies throughout the design process, resulting in systems that maximize human potential. We expect that the identification of critical design parameters and the preliminary framework provided by the design toolkit can provide a foundation for further work to improve the quality of immersive experiences provided to users of various abilities.

References:

1. M. Bachynskyi, G. Palmas, A. Oulasvirta, and T. Weinkauf. Informing the design of novel input methods with muscle coactivation clustering. *ACM Trans. Comput.-Hum. Interact.*, 21(6), Jan. 2015. doi: 10.1145/ 2687921
2. A. Blandford. *Semi-structured qualitative studies*. 01 2013.
3. D. Cohen-Or, O. Sorkine, R. Gal, T. Leyvand, and Y.-Q. Xu. Color harmonization. *ACM Trans. Graph.*, 25(3):624–630, July 2006. doi: 10.1145/1141911.1141933
4. S. Dias, J. Diniz, E. Konstantinidis, T. Savvidis, V. Zilidou, P. Bamidis, A. Grammatikopoulou, K. Dimitropoulos, N. Grammalidis, H. Jaeger, M. Stadtschnitzer, H. Silva, G. Telo, I. Ioakeimidis, G. Ntakakis, F. Karayiannis, E. Huchet, V. Hoermann, K. Filis, E. Theodoropoulou, G. Lyberopoulos, K. Kyritsis, A. Papadopoulos, A. Depoulos, D. Trivedi, R. Chaudhuri, L. Klingelhoefer, H. Reichmann, S. Bostantzopoulou, Z. Katsarou, D. Iakovakis, S. Hadjidimitriou, V. Charisis, G. Apostolidis, and L. Hadjileontiadis. Assistive hci-serious games co-design insights: The case study of i-prognosis personalized game suite for parkinson's disease. *Frontiers in Psychology*, 11, 2020.
5. J. J. Dudley, J. T. Jacques, and P. O. Kristensson. *Crowdsourcing Design Guidance for Contextual Adaptation of Text Content in Augmented Reality*. Association for Computing Machinery, New York, NY, USA, 2021.
6. J. a. M. Evangelista Belo, A. M. Feit, T. Feuchtner, and K. Grønbæk. *XRgonomics: Facilitating the Creation of Ergonomic 3D Interfaces*. Association for Computing Machinery, New York, NY, USA, 2021.
7. K. Z. Gajos, D. S. Weld, and J. O. Wobbrock. Automatically generating personalized user interfaces with supple. *Artificial Intelligence*, 174(12):910–950, 2010. doi: 10.1016/j.artint.2010.05.005
8. A. Goicoechea, D. Hansen, and L. Duckstein. *Multiobjective decision analysis with engineering and business application*. John Wiley and Sons Inc., New York, NY, USA, 1982.
9. D. Hasler and S. Suesstrunk. Measuring colourfulness in natural images. *Proceedings of SPIE - The International Society for Optical Engineering*, 5007:87–95, 06 2003. doi: 10.1117/12.477378
10. J. D. Hincapié-Ramos, X. Guo, P. Moghadasian, and P. Irani. Consumed endurance: A metric to quantify arm fatigue of mid-air interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, p. 1063–1072. Association for Computing Machinery, New York, NY, USA, 2014. doi: 10.1145/2556288. 2557130

11. R. Marler and J. Arora. The weighted sum method for multi-objective optimization: new insights. *Structural and Multidisciplinary Optimization*, 41:853–862, 2010.
12. Y. Matsuda. *Color design*. Asakura Shoten, 1995.
13. L. McAtamney and E. Nigel Corlett. Rula: a survey method for the investigation of work-related upper limb disorders. *Applied Ergonomics*, 24(2):91–99, 1993. doi: 10.1016/0003-6870(93)90080-S
14. M. Mott, J. Tang, S. Kane, E. Cutrell, and M. R. Morris. "I just went into it assuming that I wouldn't be able to have the full experience": Understanding the accessibility of virtual reality for people with limited mobility. In *ASSETS 2020*. ACM, October 2020.
15. G. Stockman and L. G. Shapiro. *Computer Vision*. Prentice Hall PTR, USA, 1st ed., 2001.
16. K. Todi, D. Weir, and A. Oulasvirta. Sketchplore: Sketch and explore with a layout optimiser. pp. 543–555, 06 2016. doi: 10.1145/2901790. 2901817
17. D. Tolani and N. Badler. Real-time inverse kinematics of the human arm. *Presence (Cambridge, Mass.)*, 5:393–401, 02 1996. doi: 10. 1162/pres.1996.5.4.393
18. E. Triantafyllidis and Z. Li. The challenges in modeling human performance in 3d space with fitts' law. *CoRR*, abs/2101.00260, 2021.
19. J. O. Wobbrock, S. K. Kane, K. Z. Gajos, S. Harada, and J. Froehlich. Ability-based design: Concept, principles and examples. *ACM Trans. Access. Comput.*, 3(3), Apr. 2011. doi: 10.1145/1952383.1952384
20. L. Zadeh. Optimality and non-scalar-valued performance criteria. *IEEE Transactions on Automatic Control*, 8:59–60, 1963.

Automating Faculty Services in Universities for Enhanced Efficiency

Naresh Adhikari
Department of Computing and Security, Slippery Rock University
naresh.adhikari@sru.edu

Abstract:

Faculty members in universities undertake a diverse range of responsibilities beyond teaching and research, including faculty recruitment, club advising, peer observation, peer evaluation for tenure, promotion, sabbatical, conference organization, and professional development workshops, among other tasks. These tasks often require significant administrative effort, leading to inefficiencies and workload imbalances. Automation presents a viable solution to streamline these services, reducing faculty burnout and enhancing institutional effectiveness.