JoCsE

Journal Of Computational Science Education

Promoting the Use of
Computational Science
Through Education

Contents

Introduction to Volume 15, Issue 2

David Joiner, Editor
Kean University
fdjoiner@kean.edu

This issue of our journal features six diverse contributions that represent the current advances in computational training, high-performance computing education, data analytics in community colleges, and practical AI applications. We present three papers from the Strategies for Enhancing HPC Education and Training workshop at PEARC24. as well as three additional submitted articles.

This issue's featured articles are as follows:

(1) Bautista and Sukhija on guidelines to support data analytics programs in community colleges, addressing the challenges of instructor availability and rigorous student commitments.

(2) Nite et al.'s analysis of the expanded metrics for assessing a CyberTraining project at Texas A&M University.

(3) Stevens et al.'s interdisciplinary introduction to high-performance computing at Wake Forest University, bridging multiple academic disciplines.

(4) Krishnasamy et al.'s exploration of HPC training courses in Luxembourg, focusing on CUDA, OpenMP, and OpenACC.

(5) Matlin's development of self-paced computational science modules in Python, aimed at supporting undergraduate students.

(6) Reagan et al.'s student paper on using generative language models, like GPT-3.5 and GPT-4, for GUI development.

We encourage you to submit your work to the Journal of Computational Science Education. Computational science is an increasingly important interdisciplinary field, offering insights into complex systems, accelerating discovery, and helping to solve diverse problems. We welcome high-quality papers describing instructional materials, successful projects, or research on instructional efficacy. Whether you are faculty or a student, your contributions are valuable to advancing computational science education. Additionally, if you have expertise in computational science, consider volunteering as a reviewer to support our peer review process. Together, we can share successes and inspire others to develop and adopt computational science in education.

Sincerely, Dave Joiner

Creating Guidelines to Supplement the Data Analytics Program in Community College toward Preparation of STEM and HPC Careers

Elizabeth Bautista
NERSC, Lawrence Berkeley National Laboratory
ejbautista@lbl.gov

Nitin Sukhija
Slippery Rock University of Pennsylvania
nitin.sukhija@sru.edu

ABSTRACT

Data science continues to create opportunities in the technology and HPC industry resulting from growing data sets, the need for more insights, the necessity of automation, the evolving roles and changes in job descriptions as those positions are needed and the shortage in the workforce with this talent. However, despite the growing demand, not enough students are learning the basic skills or being able to be given opportunities for hands-on work. In the Northern California Community College system, many of the students return to school after having graduated with a bachelor's degree or find the need to gain new skills to enhance their resume or to change careers altogether. Unfortunately, in the community colleges, there are not enough classes or instructors who are trained in data science to teach the class. In the four-year university, the program is usually waitlisted for transfer students from the community college. This paper is a continuation of the work after the National Energy Research Scientific Computing Center (NERSC) partnered with Laney College to start a Data Analytics program. After two years, they are challenged with not enough instructors to the number of students that are interested in the program. Further, approximately 40% of students are struggling to continue the rigorous material they need to learn. These students may have to work to support families and are unable to put in the 20-40 hours of work to earn a living as well as the 20-40 hours of study and homework that the program requires. Therefore, Laney partnered with Codefinity, an online education program that has a track for Python Data Analysis and Visualization. In the prior year, students who learned how to visualize data with Grafana [8] were able to master it in two months and create new visualizations for NERSC Operations staff [3]. Codefinity [6] provides a similar program where in six courses they learn about the data and learn a visualization program at their own pace of either a three-month plan or an annual plan. This paper describes the early results of using these guidelines to aid students who are not able to adhere to a rigorous study program in a community college as well as for students who want this skill in addition to the concentration they are currently taking.

1 INTRODUCTION

As new data sets are created and data volume grows, organizations are looking for more insights than ever before because they want to leverage the collected data. With the improvement of technologies like machine learning and artificial intelligence being much more accessible, organizations not only want their routine work to be automated but the job itself enhanced so that the human is doing something smarter and more effectively.

At NERSC, the Operations Technology Group (OTG) staff are the 24x7 onsite site reliability engineers who are the first responders to anything that occurs in the data center. This particular job description requires knowledge of system administration of HPC systems, local and wide area networking, a three-tier onsite storage and data center facility management at minimum for staff to be successful. NERSC is one of the largest facilities in the world devoted to providing computational resources and expertise for basic scientific research. NERSC currently supports close to 10,000 users globally across almost 1,000 scientific projects.

As NERSC moves toward an exascale system, one of the OTG's strategic plan is to use the data collected by the Operations Monitoring and Notification Infrastructure (OMNI) system to create a new paradigm called Artificial Intelligence Operations (AIOps) [1]. This means, the analysis of the data should be able to predict usual problems before they occur and mitigate them without human intervention.

But in today's shortage of talent, how will they be able to hire staff who can do this work unless they participate in the educational system to help create a pipeline of staff? The ongoing solutions reported in prior papers have been effective but this industry needs more and with the constraints of not enough faculty to many more students interested in learning these skills, NERSC partnered again with Laney College who partnered with Codefinity to create guidelines for not only their students but all students in general to help supplement the current shortage in faculty for these classes [5, 11].

This paper documents the process for creating a pipeline where data analytics students can supplement their education, non-data analytics students can learn these skills, and students in the program who cannot adhere to the rigorous program have an alternative. Section 2 will explain how the partnership with Codefinity was established. Section 3 shows how the program was designed by faculty and NERSC staff with Codefinity. Section 4 provides the logistics of funding for this program. Section 5 will provide case

studies of positive outcomes from the trial year of the program. Section 6 will provide lessons learned, future work and final thoughts to be able to continue the program.

2 BACKGROUND

This paper is the continuation of work done with Laney College over the last two to three years [2–4]. We started with creating pathways for disadvantaged students toward getting them educated and trained for STEM and HPC careers. Following that, we strengthened the community college's computer science program by creating a data analytics program that can be offered to students to give them an advantage in getting a job in this field and also to provide them leverage for when they transferred to a four year university [9].

As the need and popularity of the program grew, funding for the program was more challenging to obtain. Therefore, a decision was made to help supplement the program through other means to assist students who are not able to continue with the rigorous study required to complete the educational portion and obtain the hands-on experience they needed to get a job. We decided to leverage online training programs and investigated several. One of the potential programs we investigated was Merit America [10]. This program was online, provided various tech programs including coding and data analytics. They also provided a low cost onramp into the program with a stipulation that pay back isn't required until the student is earning at least 40K a year in a job. Cost is a limiting factor in this instance. Although there is practically zero start up cost, the student would go into debt again. This did not work for returning students who already have a four year degree, college debt and yet do not have a job which is why they are back in a community college.

Another program we investigated is Coursera [7]. This program had many choices and the courses seemed like a duplicate of what students already took in the community college. They were also too fundamental for the more advanced students. And though they provided a programming segment, it does not specifically have courses in data analytics. Course was not as prohibitive even though you pay as you go along but sometimes, the decision is not only about the cost.

There were many bootcamps online that were also provided that promised to gain a student a skill in a short amount of time. Again, these courses are very expensive and would be prohibitive for students already in debt. Further, while they present much marketing that they will help students get a job, interviews with prior students show that although they have a job, it took them more than one year after taking the program and it is not the career they necessarily envisioned for themselves.

We decided to use Codefinity is because of the following reasons:

- The cost was affordable for students and did not get them into additional debt.
- The programs were what we were looking for including data science, data analytics, machine learning and artificial intelligence, which could supplement the Laney curriculum.
- There is a hands-on component where students complete projects in the software's workspace or students and faculty

can design a project that groups of students can complete. We hope it's the next best thing to an internship.
- The timeline to complete a course is up to the students up to one year before the course needs to be redone and paid for again. Further, the quicker the students complete a course, the faster they could complete the program.

Negotiation to partner with Codefinity included the potential for Laney to subsidize the cost for students, allowing the cost to be even less. The program costs 25% less as long as students complete the program. The 25% is credited to students when they successfully complete the program and applied toward the next course. The courses cost 50% less as they complete 70% of the program therefore, there is an incentive for students to complete courses as well as the program.

3 PROGRAM DESIGN

Codefinity, the Laney faculty and NERSC staff collaborated to "pick and choose" various classes in their programs to create a new online course of study that would allow students to do a course online if they cannot participate in school and extend the course of study to more than a semester should a student need it. Further, there were stand alone courses not part of the data analytics track that could supplement Laney's classes in programming such as a python course for beginners and Pandas for more advanced students. For students in the data analytics track, a course called Business Analysis Fundamentals is being used to help students get hands-on practice in looking at business data and creating analytics graphs for them to answer those insights needed by companies. For more advanced programmers in the analytics track is a class that uses python with visualization software and linear regression with python. The addition of these more advanced courses will help students planning on applying to the four year university data analytics program. The advantage of doing the courses online is that they complete them in their own time. As long as they can set aside time each day or each week to complete a series of modules, life does not have to compete with their education and training.

For example, in the beginning python class, the course has 192 chapters which should take a total of 16 hours for the average students who will need to do homework to complete 168 tasks. While the 16 hours may seem daunting, it is a total 16 hours that can be done across a semester or one month, as long as they complete all the tasks. All that's needed is for the student to commit to the time, a device with an approved browser and the motivation to learn.

4 COST AND FUNDING

Codefinity agreed to the following plans for our students:

- An annual plan that can be billed monthly or annually. This would be the best plan for a single course. The billing cycle renews if the student does not complete the course within in this period.
- A three month plan for a single course that can be billed every three months or monthly. The billing cycle renews if the student does not complete the course within this period.
- A monthly plan. This is the most expensive of the plans and this plan does not qualify for the school discount. As with

the above two plans, if the student does not complete the course in this time frame, the billing cycle renews.

- An annual professional plan where a student can start up to five courses and can take up to one year to complete them. While this costs more, it is less expensive than to pay for a course at a time.

As previously mentioned, Laney will subsidize 25% of the annual cost or the three month plan cost, and is credited to the student as they complete each course. When they are about 70% completed, they can get up to a 50% discount. A cost comparison to Laney's current tuition cost which is currently free as long as the student lives within the area served by the school, while more expensive, it is less expensive than a semester at the current state university and is subsidized by the school. Currently, the school has funding for three years. We currently offered this program in the spring of 2024. We will continue to offer this program throughout the year as long as we can secure funding. Financial support for community colleges is always dire. It is our hope that we have expanded the Computing Science Department with this program will continue to get us yearly funding from the state. We also have a few grants that we hope to complete before the fall to give us additional funding.

5 EARLY RESULTS

In the spring of 2024, 30 pilot students entered the Codefinity program. Fifty percent of the students are part of our two previous programs and have opted to take the one year professional plan. The other twenty-five percent are newly enrolled fall students in the computer science program and the last twenty-five percent are new students not in a computer science program. The second batch of fifty percent of students are taking either an annual plan or a monthly plan per course.

As of the end of May 2024, 10 students competed five courses and when they applied to a summer internship, six students were accepted to a Port of Oakland Authority internship, two students at the San Francisco Airport Commission and the other two are in the summer program at LBNL. The other students have completed a minimum of one semester of work in their programs. One student was accepted to New York University's Data Science program as a junior and a second student was able to transfer to California State East Bay's Computer Science Program. A third student was able to be accepted to the University of California, Davis's Computer Science Program. Two students earned a summer internship at Stanford Linear Accelerator Laboratory.

Although not every student can say that they earned an internship or were able to complete a course, none of the 30 students in the pilot program left the program because they could not complete the courses. In fact, those students who earned a transfer to the four year universities plan to continue with the online program.

6 LESSONS LEARNED AND FUTURE WORK

An important lesson for us is that students need flexibility and an educational institution needs to provide this to ensure success for their students. The tutoring and mentoring program continues to help the students. We continue to create partnerships with Bay Area organizations to provide internships for our students. Our big issue is continued funding especially now that normal classes are back and more students are enrolling due to the results of lack of employment from COVID. A Bureau of Labor Statistics survey in 2023 noted that in the US 2.5 million people lost work because the employer closed or lost their business. Many of them went into another industry. The impact of the pandemic today continues as companies who may have survived the closure now find that they need to shut down. In Oakland, where Laney College is located, 25-30 businesses close every month in multiple industries. This is in addition to the layoffs that continue to happen.

As we lose these companies, people will look for alternatives and will want to gain a new skill. The tech industry is a good possibility for them because our salaries tend to be higher even when the jobs are entry level. However, educational organizations need to provide the training and hands on experience to update skills.

At NERSC, we have experienced that students can quickly understand how to analyze and visualize data. As we continue to gather data, we need these skill sets so that Site Reliability Engineers can support their data center. The future work involves being able to secure funding so that the program can grow as enrollment grows and to continue recruiting more employers to participate in our internship program.

REFERENCES

[1] Elizabeth Bautista, Melissa Romanus, Thomas Davis, Cary Whitney, and Theodore Kubaska. 2019. Collecting, monitoring, and analyzing facility and systems data at the national energy research scientific computing center. In *Workshop Proceedings of the 48th International Conference on Parallel Processing.* ACM, Kyoto Japan, 1–9. https://doi.org/10.1145/3339186.3339213

[2] Elizabeth Bautista and Nitin Sukhija. 2021. Employing directed internship and apprenticeship for fostering HPC training and education. *JOCSE* 12, 2 (2021). https://doi.org/10.22369/issn.2153-4136/12/2/8

[3] Elizabeth Bautista and Nitin Sukhija. 2023. Creating pathways in disadvantaged communities towards STEM and HPC. *JOCSE* 14, 2 (2023), 2–5. https://doi.org/10.22369/issn.2153-4136/14/2/1

[4] Elizabeth Bautista and Nitin Sukhija. 2024. Data analytics program in community colleges in preparation for STEM and HPC careers. *JOCSE* 15, 1 (2024), 59–63. https://doi.org/10.22369/issn.2153-4136/15/1/12

[5] Berkeley Lab. 2022. *Berkeley lab demographics.* https://diversity.lbl.gov/berkeley-lab-workforce-demographics-fy2022/

[6] Codefinity. 2024. *Boost your career...* https://codefinity.com

[7] Coursera. 2024. *Your path to career growth.* https://www.coursera.org

[8] Grafana Labs. 2024. . https://grafana.com/

[9] Rafael A. Irizarry. 2020. The role of academia in data science education. *Harvard Data Science Review* 2 (2020).

[10] Merit America. 2024. *Start your tech career...* https://meritamerica.org/

[11] Public Policy Institute of California. 2014. *California's need for skilled workers.* https://www.ppic.org/publication/californias-need-for-skilled-workers/

Assessing the Impact of a CyberTraining Project: Expanding the Metrics

Sandra B. Nite
Texas A&M University
s-nite@tamu.edu

Joshua Winchell
Texas A&M University
jwinchell@tamu.edu

Marinus 'Maikel' Pennings
Texas A&M University
pennings@tamu.edu

Dhruva K. Chakravorty
Texas A&M University
chakravorty@tamu.edu

Keith Jackson
Texas A&M University
kjackson@tamu.edu

ABSTRACT

As training on cyberinfrastructure resources becomes more common, we show the progression of metrics used to measure the effectiveness and impact of informal computational training courses that are provided by the Texas A&M University High Performance Research Computing facility. These courses were built to support researchers from research groups that have a background in computing practices. As such, the courses were structured as information-sharing sessions with the primary method to measure course success being frequency of participation. While these metrics inform about the interest in these courses, they relied on researchers continuing the learning process in their laboratories. As computing becomes ubiquitous in research programs, researchers who have no peer-learning mechanisms participate in these courses. Researchers are now participating in a continuum of courses that cover introductory to advanced topics and rely on them to build proficiency in research computing technologies.

We report on a pilot program that pivots along the way to support these researchers. We collected additional metrics to learn about the impact of the training materials for individual researchers. These metrics include participation in course activities, time spent logged on compute clusters, and views of course recordings and other asynchronous training materials. Surveys are now structured to identify the needs of individual researchers. Some of these metrics require additional processing time but will assist in understanding how researchers learn in these environments.

KEYWORDS

cybertraining, short courses, high performance computing

1 INTRODUCTION

Texas A&M University (TAMU) High Performance Research Computing (HPRC) was established in 1989 [13] and serves the research computing needs of all Texas A&M universities and state agencies and provides researchers with exceptional High Performance Computing (HPC) resources. Besides hardware, users have access to several commercial, free, and open-source software packages. HPRC staff includes system administrators and research scientists as well as graduate and undergraduate student assistants.

2 THE HPRC TRAINING PROGRAM

In 2017 TAMU HPRC was awarded a CiSE ProS Cybertraining award that was funded by the National Science Foundation (NSF). This funding, followed by other state and federal funding agencies helped expand training opportunities to the research community. As part of this effort, development of the short course program accelerated. A list of topics in computing were identified as key areas of importance for researchers interested in computing. For each topic area, complementary learning objectives and outcomes were identified to guide the design of courses. Delivery was offered remotely and in-person, and hybrid instruction was introduced in 2018. Initially, each course had a lecture component followed by a series of hands-on exercises. Exercises were incorporated throughout the courses to keep attendees engaged throughout the training [1, 3, 8, 10, 11]. After considerable trial and error, it was determined that two and a half hours was the appropriate duration for a short course. Each course was followed by a brief survey asking researchers whether the course content suited their learning needs. The short courses are modular, and a series of short courses can be stacked on top of each other to develop a semester-long series of classes. While this model was refined over the years, the underlying structure largely remained the same. Researchers can learn from a series of topics from introductory to advanced levels.

Each semester TAMU HPRC offers 60+ CI-training focused camps, hackathons, workshops, one-on-one consulting sessions, and train-the-trainers programs for scientific applications. The guiding principle for these training programs is to help researchers use our CI resources for science. In this spirit, the focus remains on teaching researchers how to use science and engineering applications software and/or workflows on CI. Underlying technologies (e.g., containerization, AI/ML frameworks, composability) are taught under the auspices of these applications. We have developed pedagogical approaches to CI training and were arguably the first group to offer micro credentials in research computing. Our informal training program offers 30+ CI-specific courses in 3 pedagogical formats to 4,000+ researchers every semester. These training courses are modular and emphasize hands-on activities. Asynchronous self-paced training for research workflows and CI are offered on the Canvas and Google Classroom Learning Management Systems. These asynchronous classes give researchers the opportunity to learn at

their own pace, while earning micro credentials. In a parallel thrust, the TAMU HPRC YouTube channel offers ADA-compliant training videos in 5-, 50- and 160-minute increments. The channel has 1,100+ subscribers and typically garners 1,100 views every month. Finally, HPRC is home to CI communities like the NSF SWEETER CyberTeam and BRICCs consortia. These efforts have extended to tutorials and workshops at the annual SC, PEARC, and IEEE Frontiers in Education conferences as well. These training programs dovetail into one-on-one deep consulting with researchers via the "Bring Your Own" Science, Code, and Data series. Research focused training programs for skilled CI researchers have been supported by previous NSF CyberTraining awards (OAC-1829799). HPRC scientists are members of the MATCH CSSN (Computational Science Support Network) and CCEP (CSSN Community Engagement Program) awardees. They also lead into our strong K-12 programs that extend from teacher preparation workshops with IEEE and ACM at (SuperComputing22 (SC22) and SuperComputing23), and weeklong summer camps that have introduced over 400 school students to computing. The strength of these programs is best represented by our ability to recruit students from various backgrounds. Local news outlets have featured the camps as well, broadening the reach throughout the community.

The focus for this study is the short courses and primers that are taught each semester. This is the largest piece of our training program, and we are working to improve the metrics for formative assessment. Summative assessment will occur at the end of the first year of the ACES testbed implementation. HPRC provides a series of short courses and primers each fall and spring semester to teach researchers how to enhance their use of research computing clusters. Primers are hour-long events and are often taught by graduate students. Short courses are 2.5-hour-long events taught by HPRC staff and industry partners. These courses build from introductory topics to advanced courses. The content of HPRC's short courses includes the basics needed to use the clusters, coding specifications for certain types of processing, and programming languages for specific applications. They are provided over a period of about 10 weeks, with a morning and afternoon training event on Tuesdays and Fridays. There is a mixture of face-to-face, online through Zoom, and hybrid venues. In particular, the ACES courses are delivered on Zoom or as hybrid courses so that they are available to attendees across the U.S. Most of the training courses incorporate practice exercises for the participants to work through to ensure their understanding of the concepts and procedures being presented. As the series of courses were developed, the most important metric was a count of participants registered for the courses. This metric showed that the courses were being attended and gave information about which courses enjoyed the highest attendance. This has continued to be the primary metric of interest. As the series of short courses expanded, the metrics used are changed to support a better understanding of the effectiveness and impact of the training program. The knowledge gained about the effectiveness of the training will be useful in the effort to increase the impact on the university campus and impact a broader community of researchers across the United States.

3   PURPOSE OF THE STUDY

In this study we show the process we have used to gather better data about our training program. We will examine what is missing from a thorough understanding of the impact of the short course training program at our institution and beyond as we reach out across the United States to provide high performance compute resources for research. Our training program will expand, and we need better metrics to fully understand the impact and how to improve our effectiveness in providing the resources researchers need. Our research questions are: 1) How well are our current short course and primer offerings serving the needs of the research community? 2) What metrics can deepen our understanding of what researchers need and how to fulfill those needs?

4   METHODOLOGY

For the first few years, registration numbers and numbers of courses offered were the primary metrics considered. However, the courses are continually evolving and offerings expanded as compute cluster components become more sophisticated. For example, with the composability available on FASTER and ACES, some short courses needed to address this. Similarly, with the novel accelerators offered on these clusters, the porting of code is not a straightforward process. Thus, the training model for these courses needed to evolve [7]. As we considered the metrics of numbers of short courses and numbers of registrations for each, it became clear that we needed to improve our data collection to include additional information about the impact of our training program. The number of registrations was not sufficient to determine the level of interest. We needed to take attendance to see how many actually followed through with their intent to take the course. Further, in order for researchers to use the clusters as taught in the short courses, attendees needed to do more than watch a presenter. Thus, we have worked to ensure that the short courses are not lecture presentations but instruction with brief exercises for participants to try out during the training. To this end, we added a new metric in the Fall 2023 semester, tracking the number of attendees that logged in to the cluster. Besides gathering additional information to examine this new metric, we realized we needed a different teaching style from lecture and demonstration. We needed to know how many attendees participated by logging on and trying the exercises given in the training. From experiences teaching credit courses, we know that attendees who watch the presenter but do not log on to the computing cluster and complete the exercises are much less likely to use the cluster later. We shifted our focus beyond the metrics of frequency of registrants and attendees. We began checking the cluster during the training to ensure attendees are logged into the clusters. If we see they are not, we offer help at the beginning of the training to get them logged on to the cluster so that they can complete the tasks and exercises throughout the course

5   RESULTS

Although the HPRC training program is multi-pronged, we limited ourselves here to the most homogeneous events that are most often in advertisement and delivery across semesters. Thus, we focused on the short courses and primers offered regularly during school semesters, with the exception of Summer 2020, when a
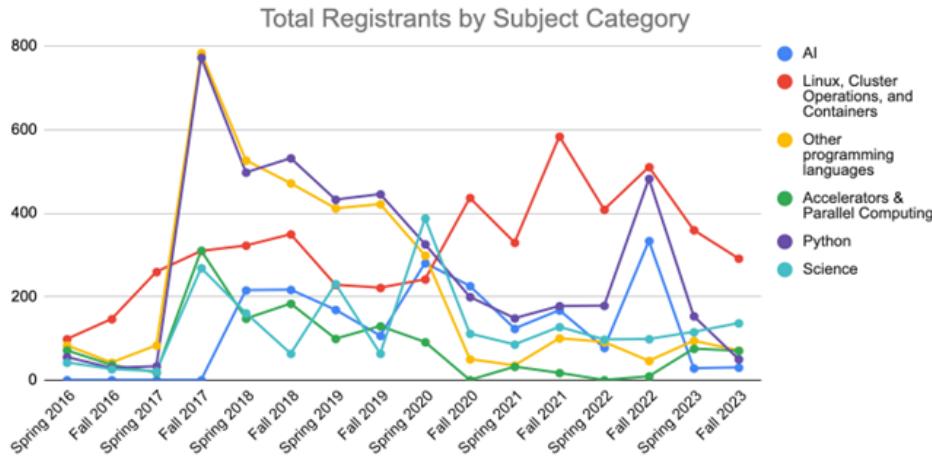
Figure 1: Registration counts for groups of classes from Spring of 2016 to Fall of 2023.

series was offered online during the COVID-19 pandemic. Short courses are generally in Fall and Spring because the Summer is busy with K12 outreach and conferences. There is some difficulty in analyzing even the short course and primer data because our specific course offerings varied by semester, meaning a course-by-course representation would be both too bulky and too difficult to use to draw cross-semester conclusions. Thus, we grouped our classes into six broad categories that we tracked across semesters.

Figure 1 shows the registrations for six categories of short courses taught during school semesters since Spring of 2016. From these we can see the areas in which interest has grown or waned over time. The number of registrations gave us information about the level of interest in the courses, but the number of attendees gave us a greater understanding of the strength of that interest. These metrics for the short courses delivered from 2016-2018 and for 2020 were previously reported [2, 4]. We make two observations from Figure 1. Firstly, there is a jump in registrations in 2017, especially for courses about "Python" and "Other programming languages," as new funding allowed for remote classes and expanded offerings of in-person classes [2]. Secondly, there was a drop in overall registration in the pandemic years, which is now beginning to recover.

In Figure 2, we look at more detailed metrics for the Fall 2023 semester, checking attendance and cluster logins in addition to registrations. These courses were offered at no cost; thus, researchers who registered were not fully committed if something else arose. The metrics that were especially useful are the number who attended and the number who logged into the cluster.

In addition to our live courses, we process videos from recordings of short courses and provide YouTube videos on the TAMU HPRC channel, providing greater access for those who cannot attend or want to review the material. There have been almost 10,000 views to date. Some courses (e.g., use of containers, programming, cybersecurity) are very popular with researchers [5, 12].

Our first research question was, "How well are our current short course and primer offerings serving the needs of the research community?" The metrics shared thus far have several indicators that we are meeting those needs well. We can see this in the numbers

that attended the new offerings. One of the newest clusters, ACES, just reached the testbed state. This cluster has a number of specialty resources for researchers. During the summer of 2023, we held a training conference and invited researchers to attend to be introduced to these resources. The same was true of the newest cluster, Launch, which went online in December 2023. A quick look shows that attendees were extremely happy with the training. We identify needed improvements through surveys.

The second research question was, "What metrics can deepen our understanding of what researchers need and how to fulfill those needs?" We recognized through this study that there are additional metrics that could be useful as we continue to improve our training program. First, we would like to know how many attendees completed the exercises during the courses. We have discussed possibly giving them an assignment to complete and providing a certificate for attending and completing the assignment. However, there is work to do to prepare the assignment as well as preparing and sending certificates after checking the assignment. We are considering customizing the survey for different courses to gain optimal information.

6   DISCUSSION

We believe collecting compute times on the clusters for short course attendees in subsequent semesters could add to the useful metrics. The average compute time of short course attendees will help us see how much they use the HPRC resources in their classes and/or research, if we can obtain these data. A study on e-resources usage revealed that among the five factors that had an impact on usage, "influencers" were the most powerful in affecting intention to use the resources. E-resources need to be organized, be easy to access, and meet the researcher's needs. With a good first experience, users will try again and spread the information to friends and colleagues [14]. A survey question about how likely an attendee is to recommend our training courses might prompt the attendee to encourage fellow researchers to avail themselves of the training sessions. Finally, we plan to construct a survey for PIs to ascertain how much they believe the short courses help their students in their

Figure 2: Frequency of registrations, attendance, and engagement for Fall 2023 Short Courses.

research projects. A similar survey could be sent to all attendees at the end of each semester's series of primers and short courses. We may create a survey to ask if there are other training courses that would further enhance the work of the researchers that use our clusters.

Understanding the impact of a training program on the intended audience is not simple or straightforward. Counts of registration and attendance are the bare minimum metrics to inform the training provider. Actual engagement in the training goes a step forward. Before that metric can be used, the design of the training may need to be changed as well. HPRC's training program began more as a lecture series where attendees were encouraged to follow along on their computers, but little help was available during the session to ensure that level of engagement. The courses have evolved in a way that provides helpers in the session to assist attendees who have difficulty accessing the computing clusters. In addition, exercises for the attendees are embedded. The presenter pauses for response from the attendees, answers questions, and then shows the process and answer for the exercises. We are noticing that the engagement assessed through percentage of those logged onto the cluster is approaching 100% of the attendees.

6.1 Expanding Beyond the Institution

The metrics discussed have been primarily considered in view of the impact of the short courses at our institution, TAMU. However, our new ACES cluster is available to researchers across the United States and just moved into the testbed phase. We plan to use many of the metrics discussed previously on the short courses for which attendees use the ACES cluster. Accounts are controlled through ACCESS, (the successor of XSEDE), but we will be able to collect information about logins and compute times on this cluster. These metrics and others we may design for the future will provide information about how our short courses impact the greater research community that uses HPC resources. It is currently and will always be necessary to develop new training for the latest hardware and software used in HPC [6]. We need a variety of metrics to help us understand better where to focus our efforts moving forward.

7 CONCLUSIONS

In discussing the metrics involved in the cyberinfrastructure training program, we approach the topic from the standpoint that the results we obtained gave us ideas about how various factors may be related; in the relatively open ecosystem of a university–plus external outreach–there is a limit to how specific our conclusions may be regardless of how much data we are able to collect, but we can at least identify trends to explore. We can suggest theories and sometimes gather data to test that theory, but we often must simply operate on our conjectures [9].

> The key concept is not that the metrics themselves had a direct causal effect on eventual outcomes, but rather that the metrics were chosen so that actions and decisions which moved the metrics in the desired direction also moved the organization in the direction of the desired outcomes and goals (p. viii) [14].

This quote defines the purpose of metrics in cyberinfrastructure training programs. In general, the goals are to serve the research community by providing various training resources to promote their research activities on HPC resources for the good of society.

Thus, adaptation, innovation, and assessment are in constant movement as technology quickly develops continually. We do our best to use the metrics to understand and conjecture about what will be needed next and work to develop the resources needed to support that work.

ACKNOWLEDGEMENTS

REFERENCES

[1] D. Christopher Brooks, Jearl Walker, and Paul Baepler. 2016. *A Guide to Teaching in the Active Learning Classroom: History, Research, and Practice.* Stylus Publishing.

[2] Dhruva K. Chakravorty, Marinus 'Maikel' Pennings, Honggao Liu, Zengyu 'Sheldon' Wei, Dylan M. Rodriguez, Levi T. Jordan, Donald 'Rick' McMullen, Noushin Ghaffari, and Shaina D. Le. 2019. Effectively Extending Computational Training Using Informal Means at Larger Institutions. *Journal of Computational Science Education* 10, 1 (2019), 40–47. https://doi.org/10.22369/issn.2153-4136/10/1/7

[3] Dhruva K. Chakravorty, Marinus 'Maikel' Pennings, Honggao Liu, Zengyu 'Sheldon' Wei, Dylan M. Rodriguez, Levi T. Jordan, Donald 'Rick' McMullen, Noushin Ghaffari, Shaina D. Le, Derek Rodriquez, Crystal Buchanan, and Nathan Gober. 2019. Evaluating Active Learning Approaches for Teaching Intermediate Programming at an Early Undergraduate Level. *Journal of Computational Science Education* 10, 1 (2019), 61–66. https://doi.org/10.22369/issn.2153-4136/10/1/10

[4] Dhruva K. Chakravorty, Lisa M. Perez, Honggao Liu, Braden Yosko, Keith Jackson, Dylan Rodriguez, Stuti H. Trivedi, Levi Jordan, and Shaina Le. 2021. Exploring Remote Learning Methods for User Training in Research Computing. *Journal of Computational Science Education* 12, 2 (2021), 11–17. https://doi.org/10.22369/issn.2153-4136/12/2/2

[5] Dhruva K. Chakravorty and Minh Tri Pham. 2022. Evaluating the Effectiveness of an Online Learning Platform in Transitioning Users from a High Performance Computing to a Commercial Cloud Computing Environment. *Journal of Computational Science Education* 11, 1 (2022), 93–99. https://doi.org/10.22369/issn.2153-4136/11/1/15

[6] Eric Coulter, Jodie Sprouse, Resa Reynolds, and Richard Knepper. 2019. Extending XSEDE Innovations to Campus Cyberinfrastructure - The XSEDE National Integration Toolkit. *Journal of Computational Science Education* 10, 1 (2019), 16–20. https://doi.org/10.22369/issn.2153-4136/10/1/3

[7] Zhenhua He, Sandra Nite, Joshua Winchell, Abhinand Nasari, Hieu Le, Jiao Tao, Dhruva Chakravorty, Lisa M. Pereze, and Honggao Liu. 2023. Development of a training framework for novel accelerators. In *Proceedings of the 2023 IEEE Frontiers in Education (FIE) Conference: Engineering Education in a Diverse, Global World.* College Station TX. https://doi.org/10.1109/FIE58773.2023.10343498

[8] Richard Lawrence, Tri M. Pham, Phi T. Au, Xin Yang, Kyle Hsu, Stuti H. Trivedi, Lisa M. Perez, and Dhruva K. Chakravorty. 2020. Expanding Interactive Computing to Facilitate Informal Instruction in Research Computing. *Journal of Computational Science Education* 13, 1 (2020), 44–49. https://doi.org/10.22369/issn.2153-4136/13/1/8

[9] Miriam Madsen. 2021. The Configurative agency of metrics in education: A research agenda involving a different engagement with data. *Journal of Education Policy* 36, 1 (2021), 64–83. https://doi.org/10.1080/02680939.2019.1682679

[10] Michael Prince. 2004. Does active learning work? A review of the research. *Journal of Engineering Education* 93, 3 (2004), 64–83. https://doi.org/10.1080/02680939.2019.1682679

[11] Kem Saichaie, D. Christopher Brooks, Phil Long, Bob Smith, Richard Holeton, Carole Meyers, Adam Finkelstein, and Shirley Dugdale. 2017. 7 Things You Should Know About Research on Active Learning Classrooms. Educause.

[12] Jinsil Hwaryoung Seo, Michael Bruner, Austin Payne, Nathan Gober, Donald 'Rick' McMullen, and Dhruva K. Chakravorty. 2019. Using Virtual Reality to Enforce Principles of Cybersecurity. *Journal of Computational Science Education* 10, 1 (2019), 81–87. https://doi.org/10.22369/issn.2153-4136/10/1/13

[13] Texas A&M High Performance Research Computing. 2024. https://hprc.tamu.edu

[14] Vandana, Ruchi Jain Garg, Vinod Kumar, and Alok Kumar Singh. 2023. Investigating the Impact of Usage Factors on Satisfaction and Intention to use e-Resource. *Performance Measurement and Metrics* 24, 1 (2023), 1–11. https://doi.org/10.1108/PMM-08-2021-0042

An Interdisciplinary Introduction to High Performance Computing for Undergraduate Programs

Cody Stevens
Wake Forest University
stevca9@wfu.edu

Sean M. Anderson
Wake Forest University
anderss@wfu.edu

Adam Carlson
Wake Forest University
carlsoas@wfu.edu

ABSTRACT

The new strategic framework of Wake Forest University seeks to build and strengthen signature areas of excellence in research, scholarship, and creative work that cross academic and institutional boundaries. To support this initiative, the High Performance Computing (HPC) Team has developed an *Introduction to High Performance Computing* undergraduate course that is accessible to students of all levels and of all academic domains. The objective of this course is to build a curriculum that presents HPC as an essential tool for research and scholarship, enables student-faculty collaboration across all disciplines, and promotes student participation in academic research during their undergraduate studies.

KEYWORDS

High Performance Computing, HPC, Cluster Computing, Pedagogy, Education, Interdisciplinary

1 INTRODUCTION

Wake Forest University (WFU) is an R2 liberal arts institution located in Winston-Salem, NC with an undergraduate population of 5,500 and a graduate population of 1,600 across the Reynolda Campus and Business School programs. WFU follows the teacher-scholar model with a student-faculty ratio of 10:1. There is strong support for undergraduate research and experiential learning programs throughout the University. Undergraduate research is so paramount to the University mission, that WFU has a dedicated center, the Undergraduate Research and Creative Activities (URECA) Center, just for this purpose, that provides internal grants to undergraduate students for research fellowships during the summer with a faculty advisor.

Research at the university is supported by the WFU High Performance Computing (HPC) Facility[13]. The facility's main asset, the Distributed Environment for Academic Computing (DEAC) Cluster, provides 4,000 CPU cores, 20 TB of RAM, and 280 TB of storage to researchers across the University. The DEAC Cluster runs on the Red Hat Enterprise Linux (RHEL) 7 operating system. The 280 TB of storage is provided by a NetApp A300 storage array, and is served over NFS to all compute and login nodes. The DEAC Cluster currently supports over 500 faculty, students, and staff from across 15 different academic departments. Training and support for

this resource is provided by us (the HPC Team) and this resource is heavily used throughout the *Introduction to High Performance Computing* course.

2 BACKGROUND

Collaboration between us and the Computer Science (CS) Department first began during the Fall semester of 2017. During this time the CS Department was interested in recruiting a team of students to compete in the Student Cluster Competition (SCC) held at the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC). This competition tasks a team of six undergraduate students to compete in a 48-hour competition where each team designs and builds a small HPC cluster. The team must install and run up to five software applications including benchmarks, visualization programs, and a wide range of scientific packages. We attended weekly classes led by a CS faculty member, provided mentorship to students, and presented on HPC topics relevant to the SCC; for instance, on the Slurm scheduler and resource manager, on cluster hardware and networking, and on software compilation. Through this collaboration, four cohorts of students were accepted to compete at the SCC both in-person (2018, 2019) and remotely (2020, 2021).

While this initial class was successful in preparing students to compete in the annual SCC, it did have some shortcomings in both content and student outcomes:

First, this course was offered at the 300-level within the CS curriculum, with a 200-level prerequisite. This prerequisite meant that any student who wished to enroll in the course and participate in the competition would need to have taken at least two courses in computer programming and one course in data structures and algorithms.

Second, as this course was a 300-level elective course, enrollment primarily consisted of students in their Junior or Senior year. While these students did learn many principles of HPC, there was little opportunity for them to use these skills within research programs before graduating.

Third, with the course's primary focus being to train a team to compete in the SCC, there was an implicit enrollment capacity of six students, which is the maximum team size to compete. Assessment for this course was directly tied to performance in the SCC, and therefore there was a disconnect for any student who potentially enrolled in the course and was not on the SCC team roster in both course content and assessment.

These shortcomings drastically limited the broader impact of this initial undergraduate HPC class. We found that the prerequisite courses did not yield the necessary skills to work in an HPC environment, such as familiarity with the Linux command line and

filesystem. Class time was mostly dedicated to learning these fundamental skills, the basics of job scheduling, and exploring computer hardware. This meant that students were not adequately prepared for the SCC as they did not have enough time to sufficiently learn and comprehend more advanced topics like installing and configuring the scheduler, compiling software from source, and familiarizing themselves with the application workflows. It was from this collaboration that the current *Introduction to High Performance Computing* course was developed, and it is designed to address these shortcomings.

3 COURSE DEVELOPMENT

HPC can often be taught from two different pedagogical approaches. The first is on how to *enable* HPC and focuses on the perspective of computer scientists, computer engineers, and cyberinfrastructure professionals who architect and design the hardware, software, and computing environment to support HPC workloads. The second is on how to *use* HPC as a research tool and focuses on the perspective of researchers from many different scientific domains, such as Physics, Biology, Chemistry, Mathematics, and Computer Science, who use HPC to solve complex real-world problems. When developing this course, our first priority was to make it as accessible to as many students as possible across the University. Other institutions that offer HPC courses do so through their Computer Science or Computing Engineering departments[6, 15, 16], and primarily focus on the first approach described above. In order to expand the use of HPC resources across all departments on campus, creating an interdisciplinary course focusing on the second approach was critical to meeting this objective.

3.1 Course Structure

The *Introduction to High Performance Computing* course is currently offered as a Special Topics course within the CS Department at WFU. It satisfies degree requirements for students pursuing a minor or a Bachelor of Arts in Computer Science, but does not satisfy any degree requirements for the Bachelor of Science program. The course meets twice a week for 50 minutes and students earn two credit hours upon completion. The course is unique within the CS Department as it is taught by all three of us (members of the HPC Team). Each instructor has a different background and experience in HPC, and provides their own perspective to the topics covered in the course. The course is currently offered during the Spring semester where it serves as an on-ramp to any student hoping to pursue summer research; this also coincides with two proposal deadlines – for the URECA Center's Summer Research Fellowship and for the SCC in May. We advocate for students to pursue these two opportunities if they are interested in the content of the course, and work directly with students to match them with a faculty research advisor for a URECA fellowship or a hardware vendor for the SCC proposal. The course then provides the necessary training for students to be successful in either endeavor.

Assessment for the course comes from biweekly projects and two larger midterm and final projects. Each project is associated with a curriculum module within the course, and students must complete the tasks by interacting with the DEAC Cluster and answering supplemental questions. We purposefully avoid timed assessments during class, such as quizzes or exams, as tasks that run in an HPC environment are asynchronous, and it can be unreasonable to expect those tasks to finish within a specific period of time. HPC systems have a constant flux of workloads, and there may be periods where job scheduling can take hours or days. Rather than reserving cluster resources for each class and impacting other research workloads, we believe that treating our students just like researchers gives them a more realistic experience of how to use and interact with a shared resource that is used by hundreds of people daily.

3.2 Course Curriculum

The *Introduction to High Performance Computing* course introduces topics using several modules. Each module is presented to the class through a lecture with slides, a hands-on activity performed in class, and a project that students begin during class but have up to a week to complete outside of class. The WFU HPC facility is heavily used throughout the course and students connect and interact with it using Visual Studio Code. We chose this tool because it is platform agnostic and provides students with the same experience no matter what device they bring to the classroom.

We removed all prerequisites for enrollment in order to make the course as accessible as possible to students from all academic domains. Students are not expected to have any prior programming experience before taking the course. We selected Bash and Python as the main applications (and scripting languages) used in the course due to their ubiquity on modern HPC systems and relevancy in data science. We provide all project code for the students, who interact with them on the DEAC Cluster and learn how to monitor their behavior and performance on a real HPC system. The topics covered in the the Spring 2024 semester are listed in the following sections.

3.2.1 Module 1: The Linux Computing Environment. For many students the concept of logging into a remote server is foreign to them. While they can see output of programs and Linux commands through Visual Studio Code, it can be difficult for new users to grasp that these programs are not running on one's personal laptop or workstation. In terms of distributed computing where jobs are submitted through a batch scheduler, such as Slurm, this can add another layer of abstraction that makes the topic more confusing for students, which is why in this module we have students run all exercises on one of the DEAC Cluster's login nodes.

In this module we primarily focus on basic Linux commands such as cd, ls, and mkdir. We also discuss the Linux filesystem and distinguish between absolute and relative paths. We leverage tools such as bashcrawl[10] and the "Password in a Haystack" challenge provided by the Hands on with Frontier GitHub repository[2]. The project for this module requires the students to also utilize helpful Linux commands such as wc, grep, and echo.

Previous iterations of this module had a deeper focus on Linux file permissions, and covered more Linux commands, such as how to determine a server's hostname and IP address, but this overall seemed irrelevant to the audience at this stage of their introduction to Linux, and this content was removed to streamline the course material.

Figure 1: Cody Stevens describes the architecture of a compute node while students disassemble Cisco B200 servers. Photo credit: Sean M. Anderson.

Figure 2: Adam Carlson highlighting the raised flooring and cabling of the WFU Data Center during a tour with students. Photo credit: Sean M. Anderson.

3.2.2 Module 2: Modulefiles and Environment Variables. The WFU HPC Facility heavily relies on environment modulefiles to manage different versions and flavors of scientific software available to researchers, and is common amongst many HPC Centers across the world. This module reinforces paths in Linux from the previous module, and we have students create temporary and persistent enviornment variables that they will use throughout the rest of the course.

The project for this module creates a bespoke environment for each student, and statically compiles several simple C programs that are named after common dictionary words. Students are tasked with identifying the environment module with their username, and analyze the changes to their environment that the module file makes to find a secret hidden program. In this project students utilize module commands such as show and load, and Linux commands such as find and which to determine their secret program.

3.2.3 Module 3: Cluster Components and Hardware. This module focuses on the hardware that composes an HPC Cluster, and what resources are available to user jobs. We discuss the components of a cluster compute node, such as CPU Cores and Memory (RAM). Students disassemble decommissioned Cisco B200 blade servers that were previously compute nodes from the DEAC Cluster to physically see how these components are integrated together in a different form factor from their laptop or a desktop workstation.

One of the highlights of the course is in this module where the students take a tour of the WFU Data Center, and see common elements of a data center, such as a raised flooring. The tour covers topics such as power, networking, and cooling within the data center and students get to see the physical hardware that composes the DEAC Cluster. This entire module helps address the layer of abstraction mentioned before in Module 1, and gives the students a physical sense of what resources are available to them when they are about to submit a job.

3.2.4 Module 4: The Slurm Resource Manager and Scheduler. At this point in the course, students should be familiar with the Linux

environment, and should understand the basics of how modulefiles work and the resources available to them on the DEAC Cluster. This module ties all of these components together and covers the basics of submitting a batch job through Slurm using a batch script, and how to monitor a job and the state of a Slurm cluster using the commands squeue and sinfo.

For this project we utilize the software application Blender to render one of the sample files that we have tweaked slightly to provide some WFU flair. While this application may not be common on a given scientific computing cluster, overall we think this is a great application for the students because the rendering process for a single scene can take up to 16 minutes with a single core, and consumes roughly 8Gb of memory or RAM and produces a nice picture and visual feedback for the students. We utilize these resource requirements to show the students common errors that Slurm will provide when too little time or memory is requested. Students also see a significant speedup when requesting additional CPU cores, and students can see the benefit of allocating more CPU cores to a given Slurm job.

We have also used Blender in class to demonstrate the ability of Slurm Job Arrays by rendering one of the sample movies provided by Blender, where each job with the job array renders a single scene before they are all stitched together to create the final movie. We did not use this example in the most recent iteration of the course due to time constraints, but feel it is worth mentioning as we plan to use it again in future offerings of the course.

We believe that the conclusion of this module marks the completion of what a typical HPC training course for a new user would cover. At this point in the course a student should be able to:

- Login to a login node, and interact with the login node using basic Linux commands to organize and store their data.
- Interact with environment modulefiles to see which software applications are available and load appropriate modulefiles for their research.

• Understand the basic resources a cluster provides for their job, and how to create a Slurm batch script to submit jobs to the cluster.

The largest hurdle we have observed for new users to the DEAC Cluster is learning the basic Linux environment. While this content could be covered in a one to two day workshop, we believe that the weekly projects that all require Linux commands to be executed on the login nodes gives the students more structured experience that they would not otherwise get from a one-on-one training session, or a small workshop. We believe that content up to this point in the course could be offered as a half semester class for any prospective student interested in research, and at this point in the course a student should be able to adequately pursue independent research with a faculty advisor.

It is at this time that we also introduce students to a general HPC workflow using Python and the Palmer Penguins dataset[4]. From this moment on in the course Python will be used more heavily, and this interactive activity gives a good example for how data can be preprocessed before analysis, and provides some context for what future Python programs used in the course may be doing.

3.2.5 Midterm: Asymptotic Complexity. As previously mentioned, the first half of the course could be described as a general training for a prospective researcher. The second half of the course serves as an introduction to many concepts we see in HPC, but may not be relevant to every user on a given research cluster. From this point on all work is submitted through batch submission using Slurm.

In this module we discuss asymptotic complexity using real world examples to describe linear, sub-linear, polynomial, and exponential scaling for computational complexity. We also discuss Amdahl's Law and discuss the consequences of requesting too many resources for a given problem. For the midterm we use a modified version of the color quantization using K-means clustering example with scikit-learn[8, 11]. We provide the students with five images

Figure 3: 3D Render of the WFU-customized Cozy Bakery[9] scene using Blender.
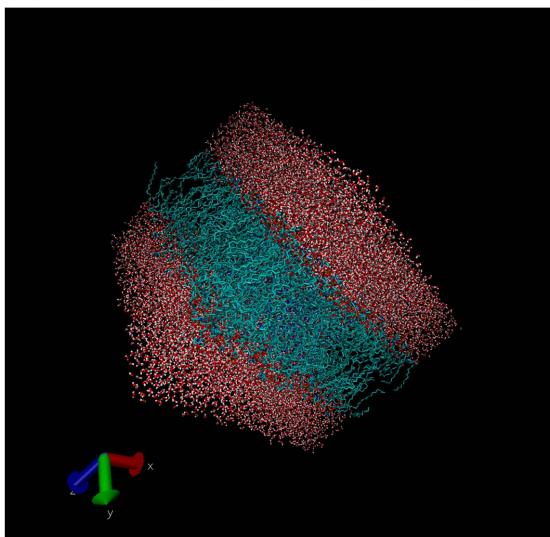
of the same subject with varying pixel dimensions to show the speedup as the size of the image increases. Like the previous application, Blender, utilizing images provides a less abstract approach to the topic and with this example program we observe that the sampling of the image performs at the same or worse speed using multiple processors than the assignment of colors using k-means or random clustering, which decreases in computation time as more processors are added to calculations. What the students ultimately observe is that there are distinct parts of the program, and they each do not scale proportionally to one another as more resources are given to the problem.

3.2.6 Module 5: Networking and Storage. Networking and storage are essential components to any HPC environment. We instruct the students that on the DEAC Cluster, there exist three directories the students can actively write and save data to. There is their home directory located under /home which serves as their starting point on the cluster, but that it is best practice to store input and write output of their research endeavors to their RESEARCHPATH which is a directory that exists on a larger pool of storage that is managed by a quota. On the DEAC Cluster, both of these paths exist on a shared filesystem and are shared via NFS to the login and compute nodes.

In this module we also discuss the concept of latency for data transfers between a shared storage array and the compute and login nodes. We discuss local storage, which is commonly referred to as scratch storage, that is temporary in nature and that in theory provides better performance over reading and writing data over an NFS filesystem. In practice, we see a minimal performance gain using the SSD local scratch directories on the DEAC Cluster versus the NFS filesystem provided by the NetApp A300 storage array, but we still believe this module is important as latency is something that will always persist in the field of HPC.

3.2.7 Module 6: Scientific Software. Compiling scientific software is a critical component of enabling HPC workloads, and in this module students learn about how software is installed and managed in an HPC environment. A simple example of compiling Nyancat CLI[7] using GNU make is done in class, and we discuss different versions of software as well as the performance of software when compiled using different compilers, such as GNU and Intel. We discuss what files are created when software is compiled and installed, and environment variables, such as PATH and LD_LIBRARY_PATH, that are relevant to using software that is compiled from source.

This module directly ties back to Module 2, and gives a more practical example of why we utilize modulefiles in an HPC environment. For this project the students are tasked with installing Python 3.9 into a scratch directory. Due to the DEAC Cluster having a base compiler of GCC 4.8.5, the students would encounter an error if they did not use a more recent compiler, such as GCC 10.2.0, which they are instructed to use. Students are tasked with installing their Python software into a scratch directory which directly references the content covered in Module 5, and students do this both interactively on a login node as well as on a compute node with a job submitted through Slurm.

Figure 4: Visualization using VMD of the protein structure output of GROMACS by Cassandra Hung, Spring 2024.

3.2.8   Module 7: GPU Computing. With the rise of Generative Artificial Intelligence (AI) and the prevalence of NVIDIA Data Center GPUs in HPC facilities today, this is a module that is relevant to the current landscape of HPC. This module is split over two different lectures. The first lecture discusses GPU hardware, and the history of the GPU and how it has advanced today up to NVIDIA's latest announcement of the Grace CPU and Blackwell GPU architectures and the GB200. This lecture explains the historical nature of GPU hardware, and how it differs from the CPU and CPU memory. The second lecture focuses on AI, and the software that enables it, such as TensorFlow and PyTorch. We discuss the rise of AI since the early 2010's up to the Generative AI and Large Language Models we see today.

We use this module to introduce the concept of interactive Slurm jobs, as GPU compute nodes have hardware and software that is not available on the DEAC Cluster login nodes, such as the NVIDIA driver. The students for this project deploy Meta's Llama2 13b-chat model[14] using llama.cpp[3]. When compiled this software produces a chat bot the students can interact with and compare the performance to other Generative AI tools, such as ChatGPT. This code is executed on CPU compute nodes with varying numbers of CPU cores, and on the GPU nodes using NVIDIA V100 GPUs.

3.2.9   Module 8: Parallel Frameworks. The last module covered within the class is on parallel frameworks such as message passing and threading. We primarily focus on MPI and OpenMP for each of these frameworks, and we discuss the memory management and use cases for each. Students complete a hands-on activity in teams that tasks them with sorting cards given different requirements. Students are responsible for assigning themselves to different roles in the sorting process as they work together, just as MPI processes and OpenMP threads would interact within the same context, and we observe the real time it takes us as humans to sort the cards using this division of labor.

Table 1: Student enrollment per semester.

Semester	Total Enrollment	Male	Female
Fall 2022	7	6	1
Spring 2022	12	10	2
Spring 2023	16	8	8
Spring 2024	18	9	9

The final project brings both of these concepts together, and tasks the users with installing the molecular dynamics software GROMACS[1] from source and comparing the performance via throughput of a simulation using different combinations of MPI processes and OpenMP threads. Students compare over 50 different combinations using up to 48 CPU cores for a single job, and then produce a visualization of the resulting protein structure using the VMD software package[5].

4   OUTCOMES

Total enrollment in the *Introduction to High Performance Computing* course has grown each semester, with our most recent semester having a total of 18 students. There has been an equal number of men and women enrolled in the course for the previous two semesters, but more than half of all students have been Computer Science majors. This is to be expected, as the course if offered through the CS Department, but this bias needs to be reduced in order to foster more interdisciplinary collaboration. Removing prerequisite classes has made the course more accessible, but there are still many opportunities for more inter-departmental collaborations.

From the Spring 2023 cohort, we had 4 students participate in the Student Cluster Competition (SCC) through the virtual IndySCC competition. Our WFU team, the Daemon Deacons, finished 4th overall in the competition and 1st amongst US teams. One of the software applications for the IndySCC was GROMACS which students already had prior experience with from our course. The final result corroborated that this team was better prepared for the SCC challenges when compared to teams from previous years.

This past semester we saw former students using the WFU HPC Facility for courses in finance, natural language processing, and science-guided machine learning. The WFU HPC Facility supported entire classes on the latter two topics. One student from the finance course had taken our class in the Spring 2023 cohort, and decided

Table 2: Student enrollment by academic major.

Major	Students
Computer Science	27
Mathematics & Statistics	11
Finance	4
Engineering	3
Economics	3
Biology	2
Undeclared	2
Political Science	1

to use the DEAC Cluster to perform a look-back analysis to build the best stock portfolio for her final project. This work was done independently of any other student or faculty involvement and was thanks to their experience in the *Introduction to High Performance Computing* course. From our most recent Spring 2024 cohort, two students have received URECA Research Fellowships for Summer 2024 and will be pursuing research with faculty in the CS Department; one student will be pursuing research opportunities in the Economics Department later this year. We will be working with these students to sponsor travel and accommodations to attend the SC24 conference later this year.

It is this engagement that we hope to foster through our course. By exposing these topics to a diverse student body we can ensure that the HPC resources are well-utilized and continue to contribute to student success throughout their time at WFU.

5 CONCLUSIONS

There is an implicit bias when offering HPC topics from only one academic department – this contradicts the interdisciplinary nature of HPC work. The demand for these resources in higher education will continue to expand as more and more academic disciplines attempt to educate students on the most relevant problems in today's technological and data-driven world. Advances in artificial intelligence and machine learning will necessitate powerful compute resources that can only be provided by HPC facilities. These facilities can also provide students with a unified computing environment and are capable of handling the additional computational requirements of courses that adequately prepare students for employment and post-graduate opportunities. By developing curricula that are accessible to students across all academic disciplines and different skill levels, the concepts of HPC can be introduced early in the undergraduate career path, and can have greater success later as HPC becomes more integrated in other academic programs. This provides greater utilization of HPC facilities, and eases the burden of individual training on faculty and other cyberinfrastructure professionals.

6 FUTURE WORK

The *Introduction to High Performance Computing* course in its current iteration has been offered for four semesters as a Special Topics course within the Department of Computer Science. Our next step is to establish a permanent course reference number within the CS curriculum and expand the number of academic credit hours to 3. With our permanent course listing and our refined curriculum, we intend to publish our projects on GitHub[12] to allow anyone to download each module and project, with bootstrap scripts for popular Linux flavors. Lastly, to break away from any departmental bias and to cater to the true interdisciplinary nature of HPC, we will be working towards creating our own academic program and will collaborate with departments across campus to create accessible projects and coursework for faculty to incorporate into their classes and majors.

ACKNOWLEDGMENTS

Computations were performed using the Wake Forest University (WFU) High Performance Computing Facility, a centrally managed computational resource available to WFU researchers including faculty, staff, students, and collaborators.

The authors would like to thank Dr. William Turkett , Chair of the Department of Computer Science, for fostering this ongoing collaboration, and Dr. Sam Cho for developing the initial High Performance Computing course which laid the foundation for the *Introduction to High Performance Computing* course as it is today.

Special thanks to the Information Systems Leadership Team for supporting the authors in this initiative to bolster the strategic framework of the University and foster a community of learning.

REFERENCES

[1] Mark Abraham, Andrey Alekseenko, Cathrine Bergh, Christian Blau, Eliane Briand, Mahesh Doijade, Stefan Fleischmann, Vytautas Gapsys, Gaurav Garg, Sergey Gorelov, Gilles Gouaillardet, Alan Gray, M Eric Irrgang, Farzaneh Jalalypour, Joe Jordan, Christoph Junghans, Prashanth Kanduri, Sebastian Keller, Carsten Kutzner, Justin A Lemkul, Magnus Lundborg, Pascal Merz, Vedran Miletić, Dmitry Morozov, Szilárd Páll, Roland Schulz, Michael Shirts, Alexey Shvetsov, Bálint Soproni, David van der Spoel, Philip Turner, Carsten Uphoff, Alessandra Villa, Sebastian Wingbermühle, Artem Zhmurov, Paul Bauer, Berk Hess, and Erik Lindahl. 2024. GROMACS 2023.4 Source code.
[2] Oak Ridge Leadership Computing Facility. 2021. Hands on with Frontier. Retrieved February 2024 from https://github.com/olcf/hands-on-with-frontier
[3] Radoslav Gerganov. 2023. LLaMA c++. Retrieved April 2024 from https://github.com/ggerganov/llama.cpp
[4] Kristen B Gorman, Tony D Williams, and William R Fraser. 2014. Ecological sexual dimorphism and environmental variability within a community of antarctic penguins (genus Pygoscelis). *PLoS One* 9, 3 (March 2014), e90081.
[5] William Humphrey, Andrew Dalke, and Klaus Schulten. 1996. VMD – Visual Molecular Dynamics. *Journal of Molecular Graphics* 14 (1996), 33–38.
[6] Steve Jones. [n. d.]. ME 344: Introduction to High Performance Computing. https://bulletin.stanford.edu/courses/2191441
[7] K Lange. 2013. Nyancat CLI. Retrieved April 2024 from https://github.com/klange/nyancat
[8] Robert Layton, Olivier Grisel, and Mathieu Blondel. [n. d.]. Color Quantization using K-Means. Retrieved March 2024 from https://scikit-learn.org/stable/auto_examples/cluster/plot_color_quantization.html
[9] Nicole Morena. 2023. Cozy Bakery - Blender 3.5 Splash Screen. https://www.artstation.com/nickyblender
[10] notklaatu. 2019. bashcrawl. Retrieved February 2024 from https://gitlab.com/slackermedia/bashcrawl
[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[12] Cody Stevens, Sean M. Anderson, and Adam Carlson. 2024. Introduction to High Performance Computing. https://github.com/WFU-HPC/Introduction-to-High-Performance-Computing
[13] Information Systems and Wake Forest University. 2021. WFU High Performance Computing Facility. https://doi.org/10.57682/G13Z-2362
[14] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:cs.CL/2307.09288
[15] Iowa State University. [n. d.]. COMS 424: Introduction to High Performance Computing. https://catalog.iastate.edu/azcourses/com_s/
[16] Rich Vuduc, Andrew Becker, Catherine Gamboa, and Amanda Deisler. 2023. CSE 6220: Intro to High-Performance Computing. https://omscs.gatech.edu/cse-6220-intro-high-performance-computing

HPC Courses Training Organization and Experiences in Supercomputing Luxembourg EuroCC: National Competence Centre (NCC)

Ezhilmathi Krishnasamy
University of Luxembourg
ezhilmathi.krishnasamy@uni.lu

Pascal Bouvry
University of Luxembourg
pascal.bouvry@uni.lu

ABSTRACT

High-performance computing (HPC) is a crucial field in science and engineering. Although HPC [23], is often viewed as a pure field of computer science or a subset of it, it actually serves as a tool that enables us to achieve exceptional results in science and engineering [22]. Since early on, computers have been primarily utilized for extensive arithmetic computations. However, recent advancements in electronics have also made edge computing integral to high-performance computing. Additionally, we have witnessed remarkable growth in computer architecture, leading to the development of powerful HPC machines, with supercomputers now reaching exaflop powers. Nevertheless, there are still challenges in utilizing these powerful machines due to the lack of knowledge in integrating physics and mathematics into HPC. Furthermore, complications with the software stack and common parallel programming models that target exascale computing (heterogeneous computing) persist. In this context, we present our effective course design for HPC training, focusing on CUDA, OpenACC, and OpenMP courses, which aim to equip STEM graduates with HPC knowledge. We also discuss how our training stands out in comparison to other NCC training frameworks in the EuroCC context and promotes lifelong learning.

KEYWORDS

CUDA, OpenMP, OpenACC, EuroCC, HPC Training

1 INTRODUCTION

The concept of a computer has existed since the 19th century. In the early nineteenth century, Charles Badge had the idea of a differential machine [20], one of the early visions of present-day computers. Over the years, it has been developed, and now we are in the era of exascale computing power. However, as we develop advanced computer architecture, software and parallel programming models, we also see an increasing gap in the efficient use of these powerful machines. We recognize that HPC centres have provided training for STEM disciplines for decades, but the limited number of trainers has hampered the ability to scale HPC training, and much of the training has yet to reach a broad audience. The opportunity to introduce parallel programming and supercomputer architecture to a broader

group of students studying mathematics, engineering, and physics is crucial because their applications are very demanding in terms of using the power of supercomputers, whether in computational fluid dynamics, material science, or cosmology. In this work, we present a different approach that uses online courses in order to scale HPC training. At the same time, we are fortunate to live with relatively advanced technology around us; for example, our smartphones have more than a single CPU core and our laptops. However, the question is whether those with an educational background in STEM are aware of this or whether they can use these cores efficiently if they want to do any computations. Even though academic programs exist for computational science or scientific computing, they are still not well-received, and we do not have enough people graduating from those programs. Recently, in the EU, an initiative focused on the master's programme in high-performance computing[1], which focuses on from computer science to scientific computing. This initiative aims to produce more graduates who can efficiently use the available HPC power we have now and in the future to enable better science and engineering.

2 RELATED WORK

There are numerous platforms and organizations that offer HPC training courses through MOOCs [3, 4, 7, 8, 11], online courses from reputable universities, PRACE Training Events [19], and HPC centers such as BSC, Julich, and CINECA. Additionally, various MOOC platforms provide different learning paths for specific courses. Massachusetts Institute of Technology also offers online courses for a wide range of topics [9]. However, there are some constraints that could hinder learners from achieving their learning goals:

- MOOC courses usually require payment and may necessitate subscribing to multiple courses.
- If a participant misses an online session, they may not be able to catch up with the course content, even if they can access the slides afterwards due to limitations in the slide information.
- Some sessions may only include 2-3 hours of presentation recordings without practical sessions. For instance, in parallel programming courses, participants may not have the opportunity to apply what they have learned from the lecture videos.

3 IMPORTANCE OF HPC IN STEM

Computers are essential for performing calculations in STEM fields, which require extensive computation and visualization. However,

[1]https://eumaster4hpc.uni.lu/

there is a noticeable gap in high-performance computing (HPC) skills among graduates and individuals with backgrounds in physics, engineering, and mathematics. For instance, someone studying mechanical engineering may be proficient in using specific software for parallel computing but may lack knowledge about the parallelization capability of the software or the HPC machine being used. Each educational program typically emphasizes its own field, which is appropriate. However, it is also important to introduce programming and computer architecture to graduates and postgraduates alongside their core STEM courses. This approach not only fosters interdisciplinary understanding but also encourages parallel simulation and computation, leading to enhanced creativity among engineers, physicists, and mathematicians and potentially accelerating technological advancements.

Efforts have been made to integrate computer and programming knowledge into STEM fields, resulting in the emergence of a new educational profile called "computational science." For example, a specialization like computational science: mechanics focuses on solids, fluids, and heat transfer, emphasizing methodologies and their computer implementation, including parallel architecture and parallel programming. We must realize how many universities worldwide offer these kinds of educational profiles. Therefore, it is essential not only for people following computational science to be exposed to computers and parallel programming but also to the entire STEM background. As we witness, we come across supercomputers and parallel programming applications in every field, whether artificial intelligence, material science, or mechanical engineering.

4 HPC KNOWLEDGE IN STEM

In the past decades, students studying computational science have been introduced to parallel architecture and parallel programming, which has enabled them to run simulations on supercomputers. Today, we are exposed to exascale supercomputers, laptops with powerful cores, and even our mobile phones. This growth is expected to continue, making a good understanding of parallel architecture and parallel programming beneficial for speeding up computations or visualizations in any STEM field.

However, we can not suddenly impose the idea that people from all STEM backgrounds must study parallel architecture or parallel programming. Therefore, interested individuals with a STEM background should be able to easily follow individual courses in computer architecture and parallel programming. Our training program is designed to help anyone quickly learn parallel programming to target various parallel architectures, such as GPU or multicore CPU.

5 HPC TRAINING INITIATIVE IN EUROPE

Europe has taken a significant initiative to promote HPC education and training. Previously, PRACE [10] made substantial efforts in promoting HPC education and training, along with other activities, such as building and maintaining new HPC machines in a centralized approach. After PRACE concluded, the EU initiated a new project called EuroHPC JU[2]. Its functionality is similar to PRACE, but it has a unique approach to research, installation of

---

new HPC machines, education, and training. One of their primary training initiatives is EuroCC [6], in which many partner EU countries are participating and providing HPC training in their own country using either their own supercomputers or one of the JU supercomputers. This initiative is typically referred to as the National Competence Centre (NCC). Since Luxembourg is part of the EU, it also takes part in the EuroCC initiative, which promotes activities in high-performance computing (HPC) in Luxembourg, along with providing education and training in HPC. The EuroCC project is currently in its second phase, which began on September 1st, 2023, and will continue until August 2026 [5].

5.1 EuroCC

EuroCC was established around September 2020 [6], and since then, each EU member state has been promoting HPC training activities within their respective countries, particularly targeting individuals with a background in STEM education. As the initiative is relatively new, most of the National Competence Centers (NCCs) are working independently to develop courses that promote HPC education in their own countries. As part of this initiative in Luxembourg, we have conducted education and training in HPC, focusing on parallel programming courses in CUDA, OpenACC, and OpenMP.

5.2 HPC Training in NCC Luxembourg

Geographically, Luxembourg is a very small country compared to its neighbouring countries. NCC Luxembourg has three partners: Luxinnovation[3], Luxembourg University[4] and LuxProvide. Although it has many industries, including engineering and technology, especially the financial and law sectors, many professionals and graduates still lack HPC knowledge in the sense of using HPC machines (for example, intra-node and accelerators) efficiently and parallelizing their applications. Therefore, we provide HPC training in Luxembourg to promote scientific computing within Luxembourg and beyond. Our training is in English and uses the national supercomputer MeluXina (it is also part of the JU supercomputer).

Figure 1: The NCC Luxembourg training website page.

---

6   TRAINING STRUCTURE

Since our NCC is very young and before we start training, we wanted to make sure we provide excellent and optimistic training within HPC. For this, we have adopted two well-known and well-practised instructional design (ID) approaches: the ADDIE model [13], see Figure 2 (left) and Merrill's First Principle Instructions [17], see Figure 2 (right).

The ADDIE model consists of five key components:

- Analysis
- Design
- Development
- Implementation
- Evaluation

Analysis. Initially, we distributed questionnaires to public and private institutes in Luxembourg. The questionnaires aimed to gather information on the use of parallel simulation, GPU usage, knowledge of efficient multicore CPU utilization, and more. For example, we asked about the fields in which parallel simulation is being used, the use of GPUs or willingness to use GPUs, and knowledge of efficient multicore CPU utilization. This information will provide an overview and help reduce the performance gap of HPC skills in Luxembourg. The responses provided valuable insights, allowing us to understand the current landscape and identify areas for improvement. To identify knowledge and skills gaps, we collected essential information to design our HPC training. This included determining the target audience, skill levels, computing resources, and participant's interest in various HPC training topics. Based on our analysis, we decided to organize the training into the following courses:

- CUDA
- OpenACC
- OpenMP
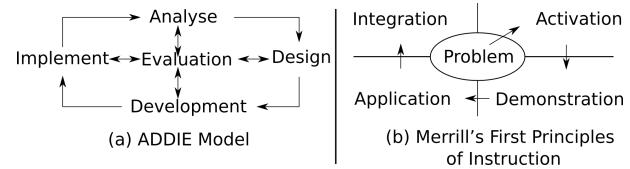
Design. In the design phase of the ADDIE model, we focused on different topics and their learning outcomes. We primarily concentrated on the course structure and teaching aids, such as choosing the website layout for hosting course material to include both lecture material and hands-on sessions. We also selected appropriate topics (learning outcomes) for lectures and hands-on sessions. For hosting the teaching material, we decided on the GitHub repository and created the hands-on session; we chose MkDocs to create the website[5], please see Figure 1. During the design phase, we decided to structure our training as follows:

- Preparation Session (approximately 1 hour)
- Break 1 (lunch break, up to 60 min.)
- Lecture Part 1 (strictly 45 min.)
- Break 2 (up to 15 min.)
- Lecture Part 2 (strictly 45 min.)
- Break 3 (up to 15 min.)
- Hands-on Session 1 (strictly 45 min.)
- Break 4 (up to 15 min.)
- Hands-on Session 2 (strictly 45 min.)

We limit each lecture and hands-on session (except the preparation session) to a maximum of 45 minutes. This is primarily based

---
[5]https://ncclux.github.io/NCC-Trainings/

on attention span and learning, cognitive load theory, spacing effect, student engagement, information retention, and learning outcomes [12, 14, 16, 18, 21]. In between lectures and hands-on sessions, we have a 15-minute break, which is also loosely based on Pomodoro Technique [15], where a short period of break increases the learning capacity. Our courses are meant to last only half a day, which is 4-5 hours. We believe that this would be very efficient and compact. In addition, HPC courses are mostly self-thought-out after participants take the classes. The more someone practices, the more experience they will get. Therefore, keeping the course for an entire day or more than a day would not be very efficient in terms of learning outcomes.

Development. In the Development section, we conduct our training online to ensure that everyone can participate without any constraints, such as needing to be on-site (which would require travel time) or take leave from work. This is especially important as it allows people with transportation issues to participate as well. Additionally, we wanted our training activities to reach beyond Luxembourg, so we prefer to keep our training online. During the development stage, we prepared the lecture and hands-on session using GitHub with MkDocs, while ensuring that our training material fulfills the learning outcomes. We also utilized Material for MkDocs, which provides a nice layout for text, figures, and coding.

Figure 2: A schematic overview of the ADDIE Model (left) and Merrill's First Principles of Instruction (right).

Implementation. This phase involves delivering the course to the participants. We typically announce the course 5-6 weeks before it starts to allow participants to register. We use various communication channels such as Twitter, LinkedIn, and our course webpage to promote the training. After each course, we gather feedback from the participants to make improvements.
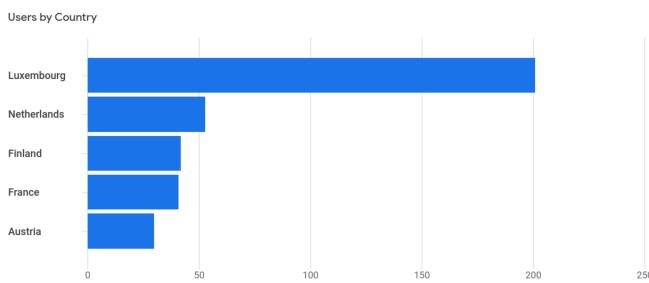
Evaluation. We analyze the feedback collected during and after the training to determine if any suggestions need to be adopted for future iterations. We maintain this approach on a regular basis to ensure efficient learning outcomes for learners. Even if we meet the learning outcomes after the training, we still look for further improvement in the course development. Typically, after the course, we send out a questionnaire to participants to improve our course content. For example: 1) Was the trial-run session informative, or do you need more instructions on the training page? 2) Did the lecture portion was informative and aligned with the course learning outcomes? 3) Were the hands-on sessions compatible with the lecture content and aligned with the learning outcomes?

We received the following feedback: participants would like more information on the trial-run session, and in addition, they would like to participate in advanced courses for inter-node computation.

This feedback will be included in the following training sessions, and new training on inter-node environment is being prepared.

Participants liked our approach, such as 45-minute lectures (45 X 2) with breaks and 45-minute hands-on sessions (45 X 2). This is because in some courses, the lecture part goes beyond 45 minutes, and the hands-on session also goes on continuously without a break. Additionally, they are really happy with the hands-on sessions and how they are organized, which are aligned with the lecture part and learning outcomes. Overall, participants are satisfied with each of our courses, which deal with logging in to supercomputers, learning parallel programming, and performing analysis.


Figure 3: The number of participants who actively used the learning website during and after the course.

We also monitor our training page through Google Analytics for further analysis of hands-on sessions; for example, some participants spend less or more time on one topic. This eventually leads to more thought about that topic for further improvement; for instance, Figure 3 shows the user's statistics (by country) we obtained from Google Analytics.

As we have discussed earlier, the ADDIE model has been adopted to develop our courses; however, we paid even more attention to the hands-on session and followed Merrill's First Principles of Instruction. The following shows those principles and how they have been adopted to organise our hands-on session:

- Problem-centered: We start with a simple `hello world` program and then progress to more BLAS examples.
- Activation: Each part of the hands-on session addresses key questions derived from the learning outcomes stated in the course description, ensuring active participation throughout the session.
- Demonstration: Initially, we present an example of a specific problem, such as vector addition. We display the C/C++ code of the sequential version and provide the framework for converting it to the parallel version. Participants are required to focus on crucial areas to subsequently convert the code to the parallel version.
- Application: We align practice activities with learning outcomes and gradually reduce guidance to foster learner's independence. For instance, we introduce entirely new questions based on previous problems that are still closely related. Please refer to list 7.4 for an example, where we pose questions that are linked to previous ones, but participants need to work independently.

- Integration: We encourage learners to apply their learning to their own scientific problems at work or in research. Towards the end of the training, we prompt participants to apply the learning outcomes to their own problems. For example, the CUDA programming model can be employed in fields such as CFD, solid mechanics, material science, and biomedical applications.

7 DETAILED TRAINING OUTLINE

In this section, we provide practical details about our training, focusing on learning outcomes to give readers a better idea of how the training was conducted.

7.1 Preparation

Our courses are tailored for individuals with a STEM educational background, so we don't anticipate extensive experience with the HPC machine. Therefore, we commence our training with a preparation session. During this session, participants can log into the MeluXina[6] HPC machine to familiarize themselves with it. This is crucial as they will need to use the machine later for the practical sessions. The purpose of this session is to minimize any time loss during the practical sessions and to assist participants in loading the necessary modules for compiling and executing the test problem. This way, they will already have an understanding of how to log in to the machine, load the modules, compile, and execute the tasks. We schedule this session for about an hour, usually before lunch, typically between 11:00-12:00.
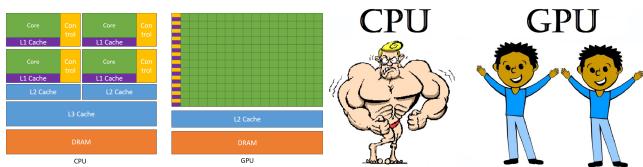
7.2 Lecture part 1

In this first part of the lecture we will outline the course structure and discuss the significance of HPC (High Performance Computing) along with its various applications. We will cover examples of its use in computational fluid dynamics and material science. Since our participants come from diverse educational backgrounds, we aim to demonstrate the relevance of HPC in their respective fields rather than focusing on a single domain.

We will begin by explaining the basic architecture of computers. Our goal is to establish a solid understanding of parallel architecture before diving into parallel programming syntax, such as CUDA or OpenMP. By understanding the distinctions between multicore CPUs vs. single-core CPUs and CPUs vs. GPUs, participants will be better prepared to grasp parallel programming concepts. This practical approach is similar to understanding a car's design before learning to drive. It's beneficial to comprehend the car's design, such as the number of available gears, speed, camera settings, etc., for better manoeuvring the vehicle.

One example that illustrates the distinct difference between GPUs and CPUs is their core count. GPUs have more cores than CPUs, making them faster. For instance, the Intel® Core™ i7-10700K Processor has a base frequency of `3.80` GHz, while the Nvidia Ampere GPU has a frequency of `0.765` GHz. Although the CPU's frequency is higher, the GPU can achieve high throughput due to its numerous cores and ability to handle parallel threads more efficiently than the CPU. Additionally, we provide various figures

---

[6]https://www.luxprovide.lu/meluxina/

Figure 4: A simple illustration of the difference between the CPU and GPU (left). The CPU has a higher frequency than the GPU; however, the GPU operates in groups using SIMT (right).

to support these concepts; Figure 4 is an example that visually demonstrates the disparity between CPU and GPU.

In the subsequent sections, we will delve into essential code syntax and APIs of parallel programming models, explaining their use cases and suggesting effective utilization strategies.

7.3 Lecture Part 2

The second part of the lecture analyses more detailed aspects of the topic. For example, when discussing CUDA programming, we start by explaining how to print a simple `hello world` program from the GPU. We then cover fundamental concepts in parallel programming applicable to both CPUs and GPUs. Additionally, we explain memory management, and how parallel threads can be created and used efficiently. Finally, we conclude lecture part 2 by presenting examples of BLAS operations and emphasizing the use of profiling tools.

7.4 Hands-on Session 1

For the hands-on session, we have developed a website that participants can access, for example, in OpenACC [2]. We cover a minimum of 5 key topics in the hands-on session; the first session will cover topics 1 to 2. Each section will examine a specific topic concept in detail with an explanation and will also include questions. Instead of simply asking participants to solve problems, we pose questions related to techniques that could be used to solve a problem. For instance, in the CUDA programming course, in the vector addition section[7], we ask the following questions for the hands-on session topic 1:

- What happens if you remove the `_syncthreads()` from the `__global void vector_add(float *a, float *b, float *out, int n)` function?
- Can you remove the if condition $if(i \leq n)$ from the `global void vector_add(float *a, float *b, float *out, int n)` function? If so, how can you do that?
- Here, we do not use the `cudaDeviceSynchronize()` in the main application. Can you figure out why we do not need to use it?

7.5 Hands-on Session 2

In the second part of the hands-on session, we will cover the remaining training topics. The hands-on exercises are well-designed, with questions for participants to work on. Each topic will cover

---

[7]https://ncclux.github.io/NCC-Trainings/cuda/exercise-2/

specific concepts, from fundamental to advanced, and we have provided code skeletons for each concept to ensure that participants can complete the exercises within the allotted time. Even if someone cannot finish during the training session, they can still use the material for self-paced learning, as we plan to keep the training material available for an extended period.

8 TOPICS AND LEARNING OUTCOMES

The "Topics and Learning Outcomes" section outlines the three courses and their respective learning outcomes, as well as our approach to training based on these outcomes.

8.1 Training in CUDA

The CUDA training is designed for participants at beginner to intermediate levels [1], and the learning outcomes are as follows:

- Understanding the GPU architecture (and also the difference between GPU and CPU)
  - Streaming architecture
  - Thread blocks
- Implement the CUDA programming model
  - Programming structure
  - Device calls (thread blocks organization)
  - Host calls (kernel calls)
- Efficient handling of memory management
  - Host to Device
  - Unified memory
- Apply the CUDA programming knowledge to accelerate examples from science and engineering
  - Iterative solvers from science and engineering
  - Matrix multiplication, matrix-vector multiplication, BLAS routines, etc.

Lectures. In the "Lectures" section, the focus is on comparing the basic architecture of the CPU and GPU. The concept of streaming multiprocessors (SMs), thread blocks, and single instruction multiple threads (SIMT) programming methodology on GPUs is discussed. The organization of SMs in the GPU and its memory organization between global, L2, shared, and L1 are detailed. The organization of thread blocks in the CUDA programming model is emphasized, along with the creation and conversion of 1D, 2D, and 3D thread blocks as needed. The latest GPU architecture advancements and compute capability are also presented. A simple CUDA programming model is demonstrated, including the construction of device functions and device calls from the host (kernel) using necessary API and CUDA qualifiers. Finally, essential BLAS operations like vector addition and multiplication using the CUDA programming model are showcased. The section concludes with a brief introduction to profiling the code and identifying bottlenecks, such as GPU occupancy and data transfer time between CPU and GPU.

Hands-on Session. During the hands-on session, participants can go through the example and work on some questions based on that example. In this way, they would have a better understanding, which they could connect with lectures and hands-on sessions.

We structured our hands-on session as follows:

- Hello World: It shows how to use simple CUDA APIs to work on `Hello World`, for example, kernel calls and how to include the thread blocks in the kernel block.
- Vector Addition: It is interesting, in terms of creating more thread blocks, for example, 1D simultaneously, how computation can be offloaded to GPU, such as data transfer to GPU and transferring back solution to CPU.
- Matrix Multiplication: Till now, participants have seen how to make just one loop into a 1D block that matches SMIT, whereas in matrix multiplication, out of three loops, inner loops should be sequentialised. This is an excellent example of parallelizing multiple loops.
- Shared Memory: Most of the time, we might end up doing computation, which uses data that is being used often. Hence, knowing how to keep data in the shared memory decreases the latency; we support this example with matrix multiplication.
- Unified Memory: It simplifies the CUDA programming structure. Most programmers make data-handling mistakes between CPU and GPU. Therefore, `unified memory` option would be a good option if someone does not want to pay more attention to data movement between CPU and GPU.
- Profiling and Performance: It is essential to understand how the given code is being executed efficiently in the GPU; we show some examples of Nsight Systems and Nsight Compute.

8.2 Training in OpenACC

Below are the learning objectives for the OpenACC programming model.

- Understanding the GPU architecture (including the differences between GPU and CPU)
  - Streaming architecture
  - Thread blocks
- Implementing the OpenACC programming model
  - Compute constructs
  - Loop constructs
  - Data clauses
- Efficient memory management
  - Host to Device
  - Unified memory
- Applying OpenACC programming knowledge to accelerate science and engineering examples
  - Iterative solvers from science and engineering
  - Matrix multiplication, matrix-vector multiplication, BLAS routines, etc.

In our OpenACC course, we primarily focus on GPU execution. This is because we aim to provide an alternative option for GPU programming, instead of the CUDA programming model. During the lectures, we investigate the fundamental differences between GPU and CPU architectures, including details such as streaming multiprocessors (SMs), GPU processing cluster (GPC), and memory hierarchy in the GPUs. We also discuss thread blocks in OpenACC, which, despite its similarity to directive programming, provides a low-level API for defining thread blocks that can be set manually.

The first part of the lecture covers important APIs from OpenACC, such as `parallel` and `kernel`, which are used for parallelizing loops. Data movement in the programming model is a crucial aspect of GPU programming, and we cover high-level APIs available in OpenACC for data movement between CPU and GPU, such as `create`, `copyin`, `copyout`, and `copy`. Additionally, we demonstrate working examples of BLAS routines, which involve nested loops and data movement of dynamic arrays between CPU and GPU. Finally, we explore the use of `unified memory` programming options and profiling tools to efficiently parallelize sequential code on a GPU.

Hands-on Session. During the hands-on session, we covered the following topics:

- Compute Constructs and Parallelize Loops: We provided examples of using `parallel` and `kernels`, such as the `hello world` example with a loop.
- Data Locality: We discussed OpenACC's low-level and high-level APIs for data movement between GPU and CPU, and we considered the use of BLAS routines to support this concept.
- Optimize Loops: By default, the OpenACC compiler selects an appropriate number of thread blocks to parallelize the loops. However, sometimes it is beneficial to set multiple thread blocks for better performance. This section covers these approaches.
- Unified Memory: Similar to the concept of CUDA, in OpenACC, enabling `unified memory` is as simple as adding a compiler flag. This simplifies the necessary data transfer API in OpenACC.
- Profiling and Performance: This provides more information during compilation, such as kernel execution thread blocks and the frequency of other API calls throughout the application execution. Additionally, the NVHPC compiler offers numerous options for profiling OpenACC code.

8.3 Training in OpenMP

The OpenMP training introduces parallel architecture and gradually introduces parallel programming; the following are the learning outcomes from OpenMP training.

- Understanding the shared memory architecture
  - Unified memory access (UMA) and Non-unified memory access (NUMA)
  - Hybrid distributed shared memory architecture
- Implementing the OpenMP programming model
  - Parallel region
  - Environment routines
  - Data sharing
- Efficient handling of OpenMP constructs
  - Work sharing
  - Synchronization constructs
  - Single instruction multiple data (SIMD) directive
- Applying OpenMP programming knowledge to parallelize examples from science and engineering:
  - Iterative solvers from science and engineering
  - Matrix multiplication, matrix-vector multiplication, BLAS routines, etc.

Lectures. OpenMP begins with an overview of computer architecture, covering shared and distributed memory architecture. It briefly explains single instruction single data (SISD), single instruction multiple data (SIMD), multiple instruction single data (MISD), and multiple instruction multiple data (MIMD) architectures. This knowledge is essential for understanding how programming can utilize the OpenMP programming model.

Next, the lecture explores the important role of the OpenMP environment, which is necessary for the successful implementation of the OpenMP programming model. The discussion then covers topics such as data sharing and work sharing, including the loop construct, sections construct, single construct, and master construct, which are crucial for SIMD and data lock or to avoid data race conditions.

Finally, the lecture concludes by demonstrating the use of various profiling tools for OpenMP performance analysis.

Hands-on Session. The hands-on session for OpenMP is organized as follows:

- Parallel Region: The `parallel` construct is crucial to understand when learning the OpenMP programming model. In this section, we demonstrate how to use the `parallel` construct and how to invoke it in the application. We also introduce other OpenMP API routines needed for the `parallel` construct.
- Data Sharing Attribute: To avoid data race or deadlock in OpenMP, we explain the methodology and provide examples using `private`, `shared`, `firstprivate`, and `lastprivate`.
- Work Sharing Constructs (loop): The previous `parallel` construct created a parallel region, but often we need parallelism in the loop. For this, we introduce `#pragma omp for`, which enables SIMD programming capability.
- Work Sharing Constructs (loop-scheduling): This section helps programmers efficiently share loops based on the problem and available resources, using `static`, `dynamic`, `guided`, and `auto`.
- Worksharing Constructs (others): Here, we introduce how to parallelize nested loops, which is commonly encountered, especially during BLAS operations.
- Profiling and Performance: We introduce profiling tools such as Intel APS, ARM Forge, Intel Advisor, Intel Inspector, and Intel VTune for a better understanding of profiling the code, memory leaks, latency, finding potential hot spots, etc. Figure 5 shows an example of the hands-on session topics in OpenMP.

9   CONCLUSIONS AND FUTURE DIRECTIONS

The training within NCC Luxembourg took place in 2023. Each course had around 20-30 participants. We have observed that the approach we adopted, especially the course delivery structures, is working very well for HPC courses. We are expanding our offerings to include more domains and additional parallel programming courses such as OpenMP Offloading, SYCL, OpenCL, etc. Additionally, we plan to offer courses in HPC software for science and engineering applications, such as ANSYS, OpenFOAM, GROMACS, and ABINIT. Furthermore, we intend to organize boot camps and

Figure 5: An example from our hands-on session on OpenMP.

hackathons in collaboration with local companies and hardware vendors like Nvidia and AMD.

ACKNOWLEDGMENTS

REFERENCES

[1] 2023-2024. Introduction to GPU Programming Using CUDA. https://ncclux.github.io/NCC-Trainings/cuda/
[2] 2023-2024. Introduction to OpenACC for Heterogeneous Computing. https://ncclux.github.io/NCC-Trainings/openacc/
[3] 2024. Coursera. https://www.coursera.org/
[4] 2024. edX. https://www.edx.org/
[5] 2024. EuroHPC JU - EuroCC1. https://www.eurocc-access.eu/
[6] 2024. EuroHPC JU - EuroCC2. https://eurohpc-ju.europa.eu/
[7] 2024. FutureLearn. https://www.futurelearn.com/
[8] 2024. Khanacademy. https://www.khanacademy.org/
[9] 2024. MIT OpenCourseWare. https://ocw.mit.edu/
[10] 2024. Partnership For Advanced Computing In Europe. https://prace-ri.eu/
[11] 2024. Udacity. https://www.udacity.com/
[12] Donald Bligh. 1985. What's the use of lectures? *Journal of Geography in Higher Education* 9, 1 (1985), 105–106.
[13] Robert Maribe Branch. 2009. *Instructional design: The ADDIE approach.* Vol. 722. Springer.
[14] Nicholas J Cepeda, Harold Pashler, Edward Vul, John T Wixted, and Doug Rohrer. 2006. Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological bulletin* 132, 3 (2006), 354.
[15] Francesco Cirillo. 2018. *The Pomodoro technique: The acclaimed time-management system that has transformed how we work.* Currency.
[16] Scott Freeman, Sarah L Eddy, Miles McDonough, Michelle K Smith, Nnadozie Okoroafor, Hannah Jordt, and Mary Pat Wenderoth. 2014. Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the national academy of sciences* 111, 23 (2014), 8410–8415.
[17] M David Merrill. 2007. First principles of instruction: A synthesis. *Trends and issues in instructional design and technology* 2 (2007), 62–71.
[18] Fred Paas, Alexander Renkl, and John Sweller. 2003. Cognitive load theory and instructional design: Recent developments. *Educational psychologist* 38, 1 (2003), 1–4.
[19] PRACE 2024. *PRACE Training Events.* PRACE. https://events.prace-ri.eu/category/1/.

[20] Doron Swade and Charles Babbage. 2001. *Difference engine: Charles Babbage and the quest to build the First Computer.* Viking Penguin.

[21] John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive science* 12, 2 (1988), 257–285.

[22] U.S. Department of Energy 2024. *Exascale Computing Project.* U.S. Department of Energy. https://www.exascaleproject.org/.

[23] Laurence T Yang and Minyi Guo. 2005. *High-performance computing: paradigm and infrastructure.* John Wiley & Sons.

Scientific Computation in Jupyter Notebooks using Python

Mark Matlin
Bryn Mawr College
mmatlin@brynmawr.edu

ABSTRACT

Computation is a significant part of the work done by many practicing scientists, yet it is not universally taught from a scientific perspective in undergraduate science departments. In response to the need to provide training in scientific computation to our students, we developed a suite of self-paced "modules" in the form of Jupyter notebooks using Python. These modules introduce the basics of Python programming and present a wide variety of scientific applications of computing, ranging from numerical integration and differentiation to Fourier analysis, Monte Carlo methods, parallel processing, and machine learning. [1] The modules contain multiple features to promote learning, including "Breakpoint Questions," recaps of key information, self-reflection prompts, and exercises.

KEYWORDS

Scientific computation, Python, Jupyter notebooks

1 INTRODUCTION

The modules were developed under a grant from the TIDES (Teaching to Increase Diversity and Equity in STEM) project of the Association of American Colleges & Universities. [2] It is intended that the modules will provide a largely independent learning experience in algorithmic thinking and scientific programming for science students, but they also could serve as the core content of or a supplement to a formal course.

Our strategy in the first year of the grant was to design a Computational Methods pilot course, primarily for physics majors but also other physical science students, that would serve as a temporary vehicle to develop, pilot, and assess modular learning units in a blended learning format, and to gain experience with teaching practices that support diverse classrooms. That course, PHYS 350, was offered at Bryn Mawr in the spring of 2015. It was run mainly by the author as a computational lab: students were asked to do readings in the modules ahead of class time, and they worked on the embedded exercises during class. Lecturing was kept to a bare minimum. Help was available from the author, a colleague, and a TA, but the students usually looked to their classmates for help, and that was encouraged.

A secondary goal of the set of modules beyond teaching computation (and the focus of the TIDES project) is to attract to computing and computational science students from groups typically underrepresented in those fields. One approach to that goal is to give students an appreciation of the wide range of individuals who have contributed to the development of computational science, or who have used it in interesting or important ways. This approach is implemented in nearly every module by the inclusion of a brief scientist profile, most of which were developed by the students in PHYS 350. A second approach, which was employed at regular intervals in that course, was to ask students a "reflection question" meant to encourage them to think about how science and computation was relevant to them and could serve their personal goals. Multiple reflection questions are now presented in the modules.

Since the first offering of PHYS 350, the modules have been used more than a half dozen times in labs offered at both the first- and second-year levels, by the author and separately by two colleagues who were not previously familiar with the modules. As a result of these experiences, involving roughly 100 students, we now offer the lab in the second semester of the sophomore year, when we believe our students can get the most from the course while also acquiring computational skills in time for them to be used in upper-level courses.

2 THE MODULES

At the core of our methodology for developing the computational materials was an open-source technology called a "computational notebook." These integrate text, formatted equations in LaTeX, computations, visualizations, and other media such as sound and video, and render it in a web browser window. Computational notebooks easily allow students to author, reproduce, and adapt code, and insert text and hyperlinks. We believed the notebook format of computation would allow us to implement our blended approach to create modules that could be easily distributed to other institutions. Most importantly, the notebook platform allowed us to integrate culturally responsive material and activities directly into the computational instruction modules. In addition, computational notebooks could be used by the students to express their own narratives and personal connections in the form of "reflections" directly alongside their coding work.

Specifically, for our project we used the Jupyter open-source notebook framework (https://jupyter.org/), which supports over 40 computing languages. A significant advantage to utilizing an open-source platform is that students will have access to it free of charge, even after they have left Bryn Mawr College. (An open-source distribution of Python, which includes the Jupyter platform as well as many useful Python packages, is Anaconda, https://www.anaconda.com/.)

---

[1] Much of the scientific computation content was based on the outstanding textbook Computational Physics, Mark Newman, CreateSpace Independent Publishing Platform, 2012.

[2] Former colleagues Elizabeth McCormack (now at Bowdoin College) and Doug Blank (now at Comet ML) were members of the Bryn Mawr TIDES team.

Our 14 initial modules (one with multiple parts) all use the Python programming language. (A zeroth module introducing computation and providing lists of resources is offered as a PDF file; two additional modules on advanced topics have been added more recently.) Python is especially well-suited for computing education and is used by researchers in a variety of fields, including biology, physics, chemistry, linguistics, and computer science. Using authentic tools on authentic problems has been shown to be effective in education [1].

Module 0 introduces computing in general and, over three parts, Module 1 introduces the basics of programming and Python, as well as some software library packages that extend the power of Python for scientific computing. Module 2 discusses the concepts of numerical error and computing time. Module 3 presents the first application of computing to problems relevant in physics: the Euler-Cromer method is used to solve one- and two-dimensional motion problems, without and with drag. (The exercises for this module include ones modeling projectile motion in the distance-dependent gravitational field of Earth, the flight of a rocket accounting for mass loss, and a solar-sail- powered spacecraft.) Module 4 introduces the forward, backward, and central derivatives, as well as interpolation. Our typical second-year student can get through the preceding modules and at least part of Module 4 in a one-semester course.

Subsequent modules cover important topics in scientific computation that a student might find useful in their upper-level courses or in research: numerical integration, linear equations, eigenvalue equations, data analysis /visualization, Fourier techniques, nonlinear equations, ordinary differential equations, partial differential equations, Monte Carlo methods, symbolic computation, object-oriented programming, parallel computing, and machine learning. A detailed table of contents of all the modules is provided in an appendix to this article.

An Instructor's Guide to the modules, including a flowchart of module dependencies, is provided along with the modules (see the Conclusion section). It also includes an overview of all the modules with some instructions for their use, the prompt we present to students for developing scientist profiles, and several reflection prompts.

As noted, the computational modules were designed with several features intended to help meet the goals of the TIDES project to enhance diverse and inclusive learning. Some of these features were formulated with the principles of Universal Design for Learning in mind. (The UDL philosophy is to provide learning materials and modalities that will assist all students, including those from underrepresented groups in STEM.) The features we incorporated in the modules are:

(1) *Context information*: The start of each module lists the prior modules with which students must be familiar in order to undertake the current module; a time estimate for reading and working through the module, including the exercises; and the learning goals for the module. Some modules also include a brief list of some interesting applications of the mathematical tools presented in the module. (Module 0 omits the last two elements, as well as features 2-5 below.)

(2) *Scientist profile*: Each module contains a personal profile of a scientist connected with computing that describes their

work and some interesting facts about their life; most of these profiles were generated by students. In constructing the profiles, the students were asked to focus on more-or-less contemporary individuals whose lives or work they found inspiring in some way. We plan to continue to ask students working with the modules to generate such profiles. A couple of comments received from the students in the PHYS 350 course regarding one of the profiles were "To me, the most inspiring thing about this story was that Dean grew up in a very difficult time for Black Americans, yet still succeeded academically and professionally to a huge degree. It reminds me that no matter where you come from or what disadvantages you might start out with, if you are dedicated, work hard, and love what you do, nothing can stop you from achieving your goals. I think I can apply this to my academics and career choice to help me decide what I spend my life doing based on my true interests." And "His [Dean's] story was very inspirational considering he was an African American growing up in the 60's and 70's and was able to achieve so much despite any adversity he faced. He was able to work on many large projects, not just one, which is something I aspire to do."

(3) *Breakpoints*: These are quick questions provided to prompt students to stop and think about what they have just read—along with answers to those questions—so students can confirm their understanding immediately. (Student feedback from the pilot course motivated inclusion of the breakpoint answers in the modules themselves rather than in separate solution sets.) A couple of example breakpoint questions from the module on numerical differentiation are "Prove this claim that b is the derivative of the quadratic curve $y = ax^2 + bx + c$ at $x = 0$" and "Write out the expression for the quartic (degree 4) approximation to the derivative."

(4) *Exercises*: These ask students to put into practice the concepts they learn in the modules. Those in early modules are relatively straightforward to complete; the exercises increase in difficulty as students progress through the sequence of modules and build their skills and confidence. Many of the exercises involve standard applications or computations that students will encounter in their physics courses. (We hope to add application problems from other scientific disciplines where appropriate in the future.)

(5) *Reflection prompts*: During the Computational Methods pilot course—which was run as an interactive session, sometimes including a short lecture but with the bulk of the time used for students working independently and in small groups—we periodically posed questions designed to improve the students' metacognitive skills. Some of those questions and others have been incorporated into the module notebooks themselves so that students will encounter them no matter the context in which they engage with the modules (i.e., whether in a course or during independent study). Examples of reflection questions posed in the modules are "Which components of this module did you find more difficult to work through, and why do you think they were challenging?" and "When you got stuck, what did you do to get unstuck? Could this or similar actions be helpful if you get stuck in future

work?" The Instructor's Guide includes additional prompts that might prove useful.

(6) *Recaps*: Bullet-point lists near the end of each module summarize the key takeaway ideas.

Other pedagogical elements connected with the modules that can enhance their effectiveness include:

Term project: As part of the pilot course, students completed a term project applying computation to some problem of personal interest to themselves. Most students chose project topics related to other courses they had taken or were taking, but one student chose a topic connected to her personal interest in music. Assignments that invite connections with a student's personal goals and interests can enhance motivation and persistence, two elements of learning found to be key in the retention of students from underrepresented groups in STEM [2, 3].

E-portfolio: To encourage students to see the connections between the modules and to recognize the full scope of their work, we asked each student in the pilot course to compile all of their module-related activities into an electronic portfolio ("e-portfolio") which they can use as evidence of their computational skills when applying to jobs or graduate schools.

3 IMPLEMENTATION

The pilot course produced extensive feedback on the individual modules from the 17 students in the course (nine women from Bryn Mawr College, one woman and seven men from nearby Haverford College), which was used to improve their clarity and sequencing. The feedback resulted in the refinement of breakpoints and self-reflection prompts, as well as the integration of the scientist profiles directly into the modules. The first few of the updated and expanded modules were next taught as part of the laboratory accompanying our second-semester, calculus-based mechanics course for physical science majors. There, we found that attempting to teach the basic material in a few weeks' worth of two-hour lab meetings to primarily first-year students was an overreach. Most of them had no programming experience, and the challenge of learning that skill on top of the physics they were learning in the course, not to mention the other demands of being a first-year student in college, was a bit too much to handle for many of them.

After an extensive departmental discussion, we decided to rearrange our sophomore-level labs so that one of them could be converted to an exclusively computational lab. By placing the basic modules in a second-semester sophomore lab, we intended to leverage the students' greater mathematical maturity while still introducing them to computing early enough in their college careers that they would have many opportunities to reinforce and further extend those abilities in their junior and senior years. The first offering of this computational lab occurred in Spring 2017 and was taught successfully by a new faculty member who had not been involved in the development of the modules. The results were promising: the students were able to get through the first five modules, apparently without significant difficulty, and the faculty member was able to learn and adapt the modules into the course pedagogy. Additionally, the author has now taught this lab five times since the pilot course and has updated the modules each time in response to student difficulties. We will continue to present

the computational learning program in this way each year. More-advanced modules will be assigned in upper-level physics courses, and in our capstone senior seminar we will ask students to complete and present their e-portfolios with a final reflection on their computational learning.

4 CONCLUSION

With the twin goals of providing training in scientific computation to our majors and recruiting and retaining more students from U.S. underrepresented groups in STEM fields, we developed, piloted, and have made publicly available curricular materials on scientific computing.

We intentionally embedded and integrated learning-promoting features such as personal profiles of noteworthy individuals and reflection prompts into the instructional modules in order to send a strong signal to both students and other faculty members that inclusivity and metacognition are valued in our classrooms, and to make it less likely that these features will be overlooked or dropped from courses utilizing the modules due to time constraints. The modules designed in this way may also serve as a model for other faculty members adapting these materials or designing their own.

Our curricular materials can be incorporated into curricula at other institutions. Outcomes will be highly dependent not only on the details of those curricula, but also on the prior knowledge and experiences of students, as well as their expectations.

The computational modules are available as open educational resources under a Creative Commons license, and others are invited to use and adapt them as they wish. The modules and supplementary materials are provided at https://github.com/BrynMawrCollege/TIDES. They're also posted in the Faculty Commons section of the PICUP website at https://www.compadre.org/PICUP/ under the title "Scientific Computation with Python in Jupyter Notebooks." Supplementary materials include notes for a mini lecture on Python, which the author has begun presenting in the first two class meetings to get students started on learning Python. Also included is a Python Quick Reference Guide showing code for common basic course tasks. Exercise solutions are available for instructors upon request. Readers interested in learning more about inclusive teaching practices will find materials from a workshop we offered in the "BMC TIDES Diversity and Equity Workshop" folder on the github site. Those materials, which include handouts for facilitators and participants, as well as feedback forms and additional suggested readings, should be straightforward to adapt to other institutions wanting to run their own workshops.[3] An extended discussion of our project (and the other TIDES projects) can be found in [4].

REFERENCES

[1] Andrea Forte and Mark Guzdial. 2005. Motivation and nonmajors in computer science: Identifying discrete audiences for introductory courses. *IEEE Transactions on Education* 48, 2 (2005), 248–253. https://doi.org/10.1109/TE.2004.842924

[2] Judith M. Harackiewicz and Chris S. Hulleman. 2010. The Importance of interest: The role of achievement goals and task values in promoting the development of interest. *Social and Personality Psychology Compass* 4, 1 (2010), 42–52. https://doi.org/10.1111/j.1751-9004.2009.00207.x

[3] Judith M. Harackiewicz, Yoi Tibbetts, Elizabeth Canning, and Janet S. Hydea. 2014. Harnessing values to promote motivation in education. In *Advances in Motivation and Achievement*, Vol. 18. https://doi.org/10.1108%2FS0749-742320140000018002

---

[3]Jennifer Spohrer contributed to these materials.

[4] Kelly M. Mack, Kate Winter, and Melissa Soto (eds.). 2019. *Culturally Responsive Strategies for Reforming STEM Higher Education: Turning the TIDES on Inequity.* Emerald Publishing.

APPENDIX: MODULE TABLE OF CONTENTS

Module 0: An Introduction to Computing.
- The computational modules
- Why learn computer programming?
- Modern computer capabilities
- Programming concepts
- Starting with Python
- Tips on learning scientific programming
- Your e-portfolio
- Appendices
  - A. Programming resources, Numerical methods resources
  - B. Basic Python topics to know
  - C. How to install Python on your computer or use it online

Module 1: A Brief Introduction to Python & Programming.
- Part I: The Basics
  - Python Overview
    * Debugging
    * Python as a Calculator
    * Strings and Printing
    * User Input
    * Lists
    * Iteration
    * Slicing
    * Booleans
- Part II: Functions, Packages, and Plotting
  - Functions
    * Function Packages
    * User-Defined Functions
    * Function of a Function
  - Numpy and Scipy
    * Making Vectors and Matrices, 1-D and 2-D Arrays
    * Slicing Arrays
    * linspace and arrange
    * Array Operations
    * Optional Arguments
  - Plotting with Matplotlib
- Part III: Algorithm Design
  - List Manipulation
    * Searching a List
    * Sorting a List
  - Recursion
  - References

Module 2: Numerical Errors and Computational Speed.
- Numerical Errors
- Computational Speed and Big-O Notation
- Vectorization
- Profiling

Module 3: Iterative Methods.
- One-dimensional Motion without Drag
- Two-dimensional Projectile Motion without Drag
- Two-dimensional Motion with Drag

Module 4: Differentiation and Interpolation.
- Definition of the Derivative
- From Differences to the Derivative
- Higher-order Approximations to the First Derivative
- Higher-order Derivatives
- Interpolation

Module 5: Integration.
- The Trapezoidal Rule
- Simpson's Method
- Choosing the Number of Steps, N
- Higher-order Methods
- Gaussian Quadrature
- Comparison of Integration Methods
- Integration over Infinite Ranges
- Multiple Integrals

Module 6: Solution of Linear Equations.
- Gaussian Elimination with Back-Substitution
- LU Decomposition
- Other Decompositions

Module 7: Eigenequations.
- Eigenvalues and Eigenvectors
- Applications
  - Principal Axes of Inertia
  - Coupled Harmonic Oscillators
  - Energies of a Quantum System

Module 8: Analyzing Data .
- Linear Least-Squares Fitting
- An Introduction to Pandas for Data Analysis
- Visualizing Data with Bokeh
- Appendix 1: Singular-value Decomposition
- Appendix 2: Principal Components Analysis

Module 9: Fourier Analysis.
- The Fourier Series
- The Discrete Fourier Transform
- Two-Dimensional Fourier Transforms
- The Discrete Cosine Transform
- The Fast Fourier Transform

Module 10: Differential Equations.
- First-Order Equations of One Variable
  - Euler's Method
  - Runge-Kutta Method
- Second-Order Equations of One Variable
- Boundary Value Problems
  - The Shooting Method
  - The Relaxation Method

Module 11: Partial Differential Equations.
- Partial Differential Equations
  - Boundary Value Problems
  - Initial Value Problems
- Other Methods

Module 12: Monte Carlo Methods.
- Random Numbers in Numerical Computation
- Monte Carlo Integration
  - The Mean Value Method
  - Importance Sampling
  - The Transformation Method
- Monte Carlo Simulations
  - The Ising Model
  - Simulated Annealing

Module 13: Symbolic Computing in Python.
- Starting with sympy
- Evaluation
- Derivatives
- Integrals
- Limits
- Power Series Expansions
- Equation Roots
- Simultaneous Equations
- Differential Equations
- Matrix Operations

Module 14: A Brief Introduction to Object-Oriented Programming.
- Why Use Object-Oriented Programming?
- The Idea Behind Object-Oriented Programming

- A Simple Example
- A More Sophisticated Example
  - Two Balls Connected by One Spring
  - Three Balls Connected by Two Springs
    * An Atomic Example
    * A Simplification

Module 15: Parallel Computing.
- Introduction
- The `multiprocessing (multiprocess)` package
- The `concurrent.futures` package
- The `joblib` package
- The `ipyparallel` package
- The `dask` package
- cuda and numba

Module 16: Machine Learning.
- Basics of Neural Networks
  - What is a Neural Network?
  - Neuron Inputs and Outputs
  - The Learning Process
- `PyTorch`
- `TensorFlow`
- `scikit-learn`

A Case Study for using Generative Language Models in GUI Development

Katelyn Reagan
Smith College
kreagan@smith.edu

Maryam Berijanian
Michigan State University
berijani@msu.edu

Dirk Colbry
Michigan State University
colbrydi@msu.edu

ABSTRACT

In the age of advanced open-source artificial intelligence (AI) and a growing demand for software tools, programming skills are as important as ever. For even the most experienced programmers, it can be challenging to determine which software libraries and packages are best suited to fit specific programming needs. To investigate the potential of AI-supported learning, this case study explores the use of OpenAI's ChatGPT, powered by GPT-3.5 and GPT-4, by students to create an image annotation graphical user interface (GUI) in Python. This task was selected because good User Experience (UX) design is a deceptively complex task in that it can be very easy to build a GUI interface but extremely hard to build one that is well designed. The approaches employed in this study included creating a program from scratch that integrates the listed features incrementally; compiling a list of essential features and requesting ChatGPT to modify existing code accordingly; collaborating on specific segments of a user-initiated program; and creating a program anew using GPT-4 for comparative analysis. The findings of this case study indicate that ChatGPT is optimally utilized for responding to precise queries rather than creating code from scratch. Effective use of ChatGPT requires a foundational understanding of programming languages. As a learning tool, ChatGPT can help a novice programmer create competent initial drafts, akin to what one might expect from an introductory programming course, yet they necessitate substantial modifications for deployment of the tool even as a prototype.

KEYWORDS

Generative Language Models, ChatGPT, Graphical User Interface (GUI), User Experience (UX) Design, Software Tools, Artificial Intelligence in Education (AIED)

1 BACKGROUND

ChatGPT is a web-based chat-bot created by OpenAI [9] and launched in November 2022. After reaching over one million users within five days of launching [2], the chat-bot has continued to gain popularity while demonstrating its ability to convey its broad knowledge using natural language. A Generative Pre-trained Transformer (GPT) powers the chat-bot, enabling original text generation that mimics

the language model's training data [3]. The primary iteration of ChatGPT employed in this study was GPT-3.5, accessible freely through an OpenAI account. GPT-4, requiring a premium account for access, was then utilized for comparative analysis with GPT-3.5. This study's emphasis on GPT-3.5 aims to illustrate the use of a technology that is readily available to students. Currently, ChatGPT is not without issues. The program is prone to exuding confidence while formulating statements from flawed logic and providing compelling disinformation [7]. Yet, since this particular chat-bot is still relatively new and continues to get updated and supported by OpenAI [1], the technology has a promising future.

Many software tools and libraries exist to assist in the creation of a graphical user interface (GUI), but finding a specific tool that fits every need of programming task can be time-consuming. Additionally, professional, public facing interfaces must meet the diversity of users' needs and put accessibility at the forefront, ensuring intuitive functionality and smooth navigation of the tools. The design of "production" quality GUIs calls for unwavering attention to detail and likely requires expertise to create functional layouts [8].

According to the chat-bot itself, ChatGPT trained on "a diverse range of internet text up until the knowledge cutoff date in September 2021". Additionally, it can instruct beginner programmers on GUI creation through explanations of concepts, code examples, event handling, problem-solving, and learning material recommendations. The promise of such capacities could help even inexperienced coders expedite the creation of custom GUIs.

2 CASE STUDY

Although deceivingly simple, the creation of a visually intuitive and smooth-functioning GUI requires a moderate degree of interface programming expertise. A series of back-and-forth interactions with ChatGPT demonstrate its ability to guide programmers without user interface programming experience and minimal knowledge beyond a basic definition of a GUI. This challenging project was specifically selected to see how far ChatGPT could be pushed as a learning tool. The particular program was intended to create a segmentation tool to annotate images [5]. Image segmentation is the process wherein the user uses pixel selection tools to identify regions of an image that are of interest (foreground). Image annotation is often the first step in many machine learning and scientific image understanding workflows.

The desired segmentation program must enable the user to easily upload an image, segment the image, and save the segmented image. The user must be able to switch between annotation colors and zoom/pan the image. The resulting segmented labeling of the output must maintain a size identical to that of the original so that coordinate points for the labels correlate directly with the pixels in

the image. Other desirable features include undo/redo buttons, the option to select different colors, a fill bucket or lasso fill tool, and a size scale to change the size of the markups. Ideally, the program would be written in Python and exclude any excess features that would slow the segmentation process down or provide undesirable results.

Several techniques were used to prompt the chat-bot, including:

Method 1 – From Scratch Evolution: asks ChatGPT to create a Python-based image annotation code from scratch, gradually introducing new features into the program.

Method 2 – From Sample: gives ChatGPT code for a previously created GUI that functions but does not have all required features, gradually fixing and developing the code as needed.

Method 3 – From Scratch List Requirements: gives ChatGPT an extensive list of requested features to create a GUI from scratch.

Method 4 – From Scratch Assisted: gives ChatGPT a rough baseline code inspired by previously established code and knowledge gained through the other methods' results.

Method 5 – Using GPT-4: employs GPT-4 to compare the results with those of GPT-3.5.

Each technique was attempted anywhere from one to three times, depending on whether the programmers felt another attempt would be constructive in demonstrating the capability of the chat-bot given different initial conditions.

A complete list of the transcripts used throughout this paper can be found on the following openly available repository: https://github.com/katiereagan/ChatGPT-Conversation.

2.1 Method 1: From Scratch Evolution

The method "From Scratch Evolution" starts by asking ChatGPT to create a Python-based image annotation code from scratch. Using this code as a baseline, new features were gradually introduced into the program.

Attempt 1:
The initial prompt was *"Python-based image annotation program"*. The output (Figure 1) consisted of a basic annotation program using Open Source Computer Vision Library (OpenCV) [10], with the steps outlined above each piece of code.

Underneath the code, ChatGPT provides a brief description that walks the user through each step of the program, noting that OpenCV must be installed in the Python environment:

ChatGPT:
In this code, the image is loaded using `cv2.imread()` *and displayed in a window using* `cv2.imshow()`. *The user can click on the image, and the* `annotate()` *function is called to handle the mouse events. When a point is clicked, it is added to the points list, and the user is prompted to enter an annotation. The point and annotation are stored in separate lists. The annotated points are drawn on the image, and the updated image is displayed using* `cv2.imshow()`. *After the user is done annotating, the annotated points and annotations are*

printed, and the annotated image is saved using `cv2.imwrite()`. *Finally, the image window is closed using* `cv2.destroyAllWindows()`.

Note that this is a basic example, and you may need to customize it based on your specific requirements. Additionally, you'll need to have OpenCV installed in your Python environment to run this code.

Upon further prompting, ChatGPT provided the user with more detailed instructions about installations of OpenCV using the terminal. This initial program did not function properly, generating errors with each run that were outside the programmer's ability to fix. Rather than searching the code for errors, ChatGPT was prompted to create a program that did not use OpenCV. The alternative offered (Figure 2) by the chat-bot depended on Pillow (Python Imaging Library or PIL fork) [6] for image manipulation and Tkinter (Tk interface) [13] for the user interface. The steps outlined in the code and description provided were similar to those used in the OpenCV-based program, varying in the libraries used.

Again, the program generated was a Python app that would not respond, attempting to but never successfully opening a window. This issue could be due to outdated suggestions due to ChatGPT's information intake cutoff occurring in September of 2021, nearly two years prior to this study.

When working with a chat-bot, the general process is to suggest a new feature, debug, and repeat while gradually introducing new features into the program or correcting pre-existing features.

One notable benefit of working with ChatGPT is that, after getting output, the user can ask the system questions about its code to get more insight into the logic. The chat-bot's conversational nature benefits beginners, who may be less familiar with user interface design.

Although ChatGPT has a reputation for producing solutions that are not necessarily accurate, this work found that ChatGPT's explanations could at least serve as a starting point for further explanation.

Ultimately, ChatGPT did not successfully produce a working code and it was decided to start a second attempt guided by the knowledge and experience gained from the first attempt.

Attempt 2:
For the second attempt, the order in which the features were requested changed in hopes of producing a different outcome.

The initial prompt was still *"Python-based image annotation code"*. The code generated once again depended on OpenCV, so a request was made to rewrite the program without using OpenCV. The default alternative seems to be PIL. To better fit the program requirements, the prompt requested the system record the coordinate points of the user annotations. The resulting code (Figure 3) added a portion of code dedicated to this new feature.

The system struggled to open the Python application popup. After several rounds of requests to optimize the provided code to facilitate the generation of the app window, the program remained non-functional. The chat-bot cited techniques such as double buffering, resizing the image, and using batch annotations, but the final product remained unable to load. It became increasingly apparent that ChatGPT struggles to create its code from scratch but can explain pre-existing code. ChatGPT would benefit from more user

```python
import cv2

# Load the image
image = cv2.imread('image.jpg')

# Create a window and bind it to the image
cv2.namedWindow('Image')
cv2.imshow('Image', image)

# Create lists to store annotation coordinates
points = []
annotations = []

# Callback function for mouse events
def annotate(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDOWN:
        # Add the clicked point to the list
        points.append((x, y))
        # Prompt the user to input annotation
        annotation = input('Enter annotation: ')
        annotations.append(annotation)

        # Draw the point on the image
        cv2.circle(image, (x, y), 5, (0, 255, 0), -1)
        # Display the updated image
        cv2.imshow('Image', image)

# Set the callback function for mouse events
cv2.setMouseCallback('Image', annotate)

# Wait for the user to annotate the image
cv2.waitKey(0)

# Print the annotated points and annotations
for point, annotation in zip(points, annotations):
    print(f'Point: {point}, Annotation: {annotation}')

# Save the annotated image
cv2.imwrite('annotated_image.jpg', image)

# Close the image window
cv2.destroyAllWindows()
```

Figure 1: The output ChatGPT generated when prompted by "Python-based image annotation program."

```python
from PIL import Image, ImageDraw
import tkinter as tk

# Initialize the GUI
root = tk.Tk()

# Load the image
image_path = 'image.jpg'
image = Image.open(image_path)

# Create a canvas to display the image
canvas = tk.Canvas(root, width=image.width, height=image.height)
canvas.pack()

# Convert the PIL image to Tkinter-compatible format
image_tk = ImageTk.PhotoImage(image)
canvas.create_image(0, 0, anchor=tk.NW, image=image_tk)

# Create lists to store annotation coordinates
points = []
annotations = []

# Callback function for mouse events
def annotate(event):
    # Add the clicked point to the list
    points.append((event.x, event.y))
    # Prompt the user to input annotation
    annotation = input('Enter annotation: ')
    annotations.append(annotation)
    # Draw the point on the canvas
    canvas.create_oval(event.x - 5, event.y - 5, event.x + 5, event.y + 5, fill='green')
    # Refresh the canvas
    root.update()

# Bind the callback function to the canvas mouse click event
canvas.bind('<Button-1>', annotate)

# Start the GUI event loop
root.mainloop()

# Print the annotated points and annotations
for point, annotation in zip(points, annotations):
    print(f'Point: {point}, Annotation: {annotation}')

# Draw the annotated points on the image
draw = ImageDraw.Draw(image)
for point in points:
    draw.ellipse((point[0] - 5, point[1] - 5, point[0] + 5, point[1] + 5), fill='green')

# Save the annotated image
annotated_image_path = 'annotated_image.jpg'
image.save(annotated_image_path)

print('Image annotation complete.')
```

Figure 2: A revised program that contrasts from the earlier program that depended on OpenCV. The most notable differences occur near the beginning, when importing the PIL and Tkinter libraries.

guidance, such as an outline of the requirements or an initial excerpt. Ultimately, the programmer was again unable to use ChatGPT to produce working code.

Attempt 3:
The third attempt once again changed the order in which the features were requested in hopes of producing a different outcome, starting with the input *"Python-based image annotation code. Can you modify the above code to run without using OpenCV?"*, followed immediately by a request to include an annotation coordinate-tracking feature. The chat-bot's reply (Figure 4) included parts of code that intended to fulfill the desired features.

While the app would open, the user's image would not upload, and the annotations on the canvas were invisible to the user. After

some repetitive requests to fix the errors, it was determined that ChatGPT could not produce a working code.

Method Takeaway: Overall, the Scratch Evolution method was not effective. ChatGPT struggled to output the original code that functioned and to manipulate the initial code while preserving the program's functionality.

2.2 Method 2: From Sample

This method begins by providing ChatGPT with the code for a previously developed GUI [4] that produces a Python app (Figure 5) that functions but does not have all the required features. For example, the program contains no zoom feature, no functioning save feature, and an eraser that undesirably erases the image in

```python
from PIL import Image, ImageDraw
import tkinter as tk

# Initialize the GUI
root = tk.Tk()

# Load the image
image_path = 'image.jpg'
image = Image.open(image_path)

# Create a canvas to display the image
canvas = tk.Canvas(root, width=image.width, height=image.height)
canvas.pack()

# Convert the PIL image to Tkinter-compatible format
image_tk = ImageTk.PhotoImage(image)
canvas.create_image(0, 0, anchor=tk.NW, image=image_tk)

# Create lists to store annotation coordinates
points = []
annotations = []

# Callback function for mouse events
def annotate(event):
    # Add the clicked point to the list
    points.append((event.x, event.y))
    # Prompt the user to input annotation
    annotation = input('Enter annotation: ')
    annotations.append(annotation)

    # Draw the point on the canvas
    canvas.create_oval(event.x - 5, event.y - 5, event.x + 5, event.y + 5, fill='green')
    # Refresh the canvas
    root.update()

# Bind the callback function to the canvas mouse click event
canvas.bind('<Button-1>', annotate)

# Start the GUI event loop
root.mainloop()

# Print the annotated points and annotations
for point, annotation in zip(points, annotations):
    print(f'Point: {point}, Annotation: {annotation}')

# Draw the annotated points on the image
draw = ImageDraw.Draw(image)
for point in points:
    draw.ellipse((point[0] - 5, point[1] - 5, point[0] + 5, point[1] + 5), fill='green')

# Save the annotated image
annotated_image_path = 'annotated_image.jpg'
image.save(annotated_image_path)

print('Image annotation complete.')
```

Figure 3: A revised version of the earlier program that includes the annotation recording feature.

```python
from PIL import Image, ImageDraw
import tkinter as tk

def draw_rectangle(event):
    global top_left_pt, bottom_right_pt, drawing
    x = event.x - canvas_width // 2
    y = canvas_height // 2 - event.y

    if event.type == '4':  # ButtonPress event
        drawing = True
        top_left_pt = (x, y)

    elif event.type == '5':  # ButtonRelease event
        drawing = False
        bottom_right_pt = (x, y)
        draw.rectangle([top_left_pt, bottom_right_pt], outline='green')
        image.show()

def display_coordinates(event=None):
    center_x = canvas_width // 2
    center_y = canvas_height // 2
    top_left_x = top_left_pt[0] + center_x
    top_left_y = center_y - top_left_pt[1]
    bottom_right_x = bottom_right_pt[0] + center_x
    bottom_right_y = center_y - bottom_right_pt[1]
    print(f"Top Left: ({top_left_x}, {top_left_y})")
    print(f"Bottom Right: ({bottom_right_x}, {bottom_right_y})")

image_path = "path/to/image.jpg"
image = Image.open(image_path)
canvas_width, canvas_height = image.size
top_left_pt, bottom_right_pt = (-1, -1), (-1, -1)
drawing = False

root = tk.Tk()
root.title("Image Annotation")
canvas = tk.Canvas(root, width=canvas_width, height=canvas_height)
canvas.pack()

draw = ImageDraw.Draw(image)
photo_image = ImageTk.PhotoImage(image)
canvas.create_image(canvas_width // 2, canvas_height // 2, image=photo_image)

canvas.bind("<ButtonPress-1>", draw_rectangle)
canvas.bind("<ButtonRelease-1>", draw_rectangle)
root.bind("c", display_coordinates)

root.mainloop()
```

Figure 4: ChatGPT's original code, prompted by the user to create a Python-based image annotation code that did not use OpenCV and included an annotation coordinate-tracking feature.

addition to the markups. Depending on the window size, not all of the toolbar may be accessible.

Using this code as a baseline, ChatGPT further developed the code to satisfy the requirements of the segmentation program.
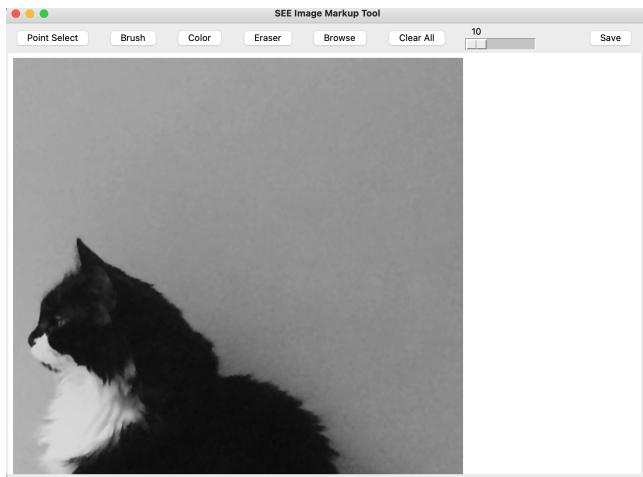
Attempt 1:
The initial input prompt was *"What is the function of this program?"*. ChatGPT replied with a walk through each part of the provided code. Additionally, a concluding overview of the program was as follows:

ChatGPT:
Overall, the program provides a basic image markup tool where users can draw on an image, choose different tools and colors, adjust brush

size, clear the canvas, and save the modified image.

From here, the programmer was then able to gradually request the implementation of new features to fit the requirements of our desired program. While ChatGPT can recognize the sample code initially given to it and can make modifications to implement the requested features, it does a poor job of keeping the code functional. Therefore, it is essential to recurrently ask the chat-bot to check for bugs after each new addition to the program to prevent an accumulation of issues over time. One consistent issue to note when working with ChatGPT is that even the requested corrections may not work. For instance, consider the zoom-in and zoom-out features. The initial requests are as follows:

Figure 5: Screenshot of the Python app generated by the sample code.

User:
Is there a way you can manipulate the earlier code to integrate a zoom functionality into the program? For instance, can you add a "+" button for the user's view of the canvas to zoom in and a "-" button to zoom out? The purpose is to make it easier for the user to see the details of the image only when the view is zoomed in and the overall image only when the view is zoomed out?

ChatGPT:
Certainly! Here's an updated version of the code that integrates zoom functionality with "+" and "-" buttons for zooming in and out of the canvas: . . .

Although the chat-bot's reply seems promising, after some testing, it was noticed that the "+" and "-" buttons are present but do not zoom in and out. ChatGPT was prompted to address this issue:
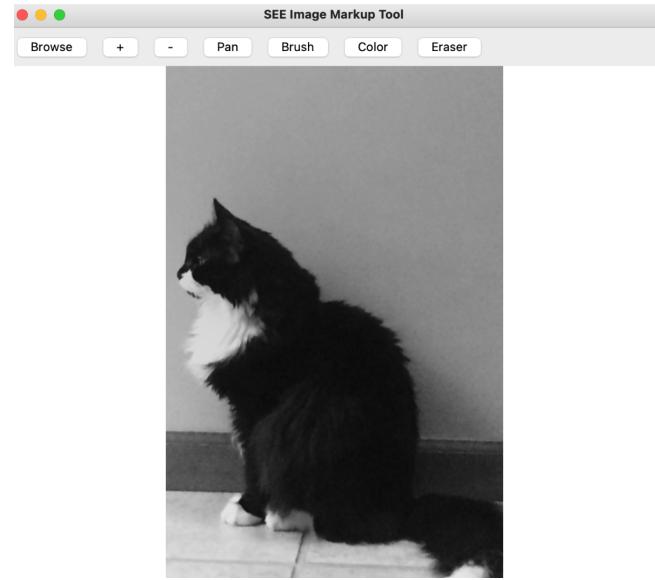
User:
The program zoom buttons do not seem to work. Is there any way you can make some modifications to ensure the logic created enables them to serve their intended purpose?

ChatGPT:
Apologies for the oversight. It seems we missed updating the zoom functionality. Here's the modified code with corrected zoom functionality: . . .

The provided modifications did not address the issue. The conversation continued for several iterations of user input and chat-bot output, resulting in minor tweaks to the program until the buttons functioned. The chat-bot frequently recommended creating custom buttons that had many initial errors. Yet, once the buttons were functional, the features previously established no longer worked properly. The final product (Figure 6) contained buttons that were accessible but did not function. While the zoom buttons work, their activation resulted in the erasing of annotations. Additionally, the

annotations do not align with the mouse and may occasionally become dysfunctional. Only PNG images can be uploaded, and there is no save feature.


Figure 6: Screenshot of the Python app generated by the updated code. Note that the program now has zoom and pan features.

Method Takeaway: Overall, the Sample method was not effective. Due to the conversational nature of working with ChatGPT, it is exceedingly difficult to pinpoint one cause for the ineffectiveness. ChatGPT's learned methods may fail to integrate with the sample code given. The system may find it challenging to process large amounts of code upfront.

2.3 Method 3: From Scratch List Requirements

The method titled "Scratch List Requirements" starts with entering an extensive list of requests and goals into ChatGPT to create an image annotation program in Python. The revised list responds to the issues persisting in the "From Sample" method. Similarly to the previous method, this method used the list of requirements to create an initial code that acts as a baseline. After spotting the errors in the initial program, the programmer focused on fixing individual features one at a time with assistance from ChatGPT to create a functioning program that satisfies all of the listed requirements. After an unsuccessful first attempt, the second attempt depends on a slightly different initial list.
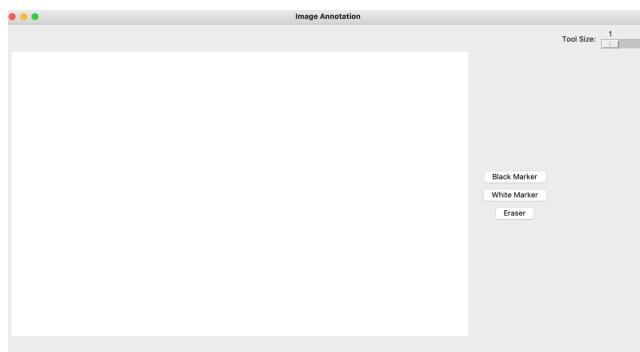
Attempt 1:
The initial user input included a list of program requirements written for the first attempt.

User:
Create a Python-based image annotation code satisfying the following conditions: Do not use CV2, *Instead of using* ANTIALIAS, *use* LANCZOS

or `Resampling.LANCZOS`, *Include a prompt that enables the user to select an image from their local files, Save both the original image and the annotated copy of the image and include a place for the user to choose the location at which the annotated copy will be saved, Enable the user to annotate the image by interacting using a computer mouse, The annotation tools include a black marker, white marker and annotation eraser that have the ability to draw atop the pixels in an image. The user can change the size of the tools throughout their session, with the smallest size being one pixel and the largest size being 20 pixels. Create a coordinate system for the image that look at the pixel location of the annotations, and in which the center of the image is the origin, regardless of its placement on the user's screen, Create a document that records a list of these coordinate points. Prompt the user to choose a file location at which to save this document. Double-check that the code has no errors, by making sure the GUI event loop has* `root.mainloop()` *and that all of the proper packages are imported during initialization.*
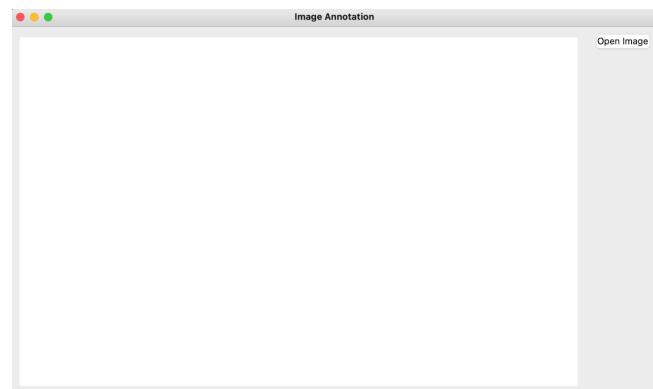
In response, ChatGPT formulated a program that was not functional and produced error messages, as follows. When asked what adjustments would be required to get the code running, ChatGPT supplied a bullet point list and, upon request, offered up code and implemented it into the pre-existing program. Although the chat-bot can spot errors in its code, it continues to output error-ridden code. Eventually, the programmer was able to get the program (Figure 7) up and running. In this version, most of the requested features are present with flaws, except for the upload image function, which is essential to the program. The program can only annotate in dots and has no scroll functionality. Depending on the window size, some of the features may not be accessible.

Figure 7: Screenshot of the initial GUI generated by Method 3, in which a list of requirements was provided.

ChatGPT generates the initial code very quickly. The authors were curious if there would be a way to ask the chat-bot to take more time in its code generation to check for errors throughout. However, GPT-4 takes more time to type the response, and as will be discussed in Method 5, the resulting outputs have higher quality. Although the system can add comments to its code to help the user follow along, one of the limitations of ChatGPT is that there is no way to see the logic behind the output or the sources behind the information.

When focusing on one specific element of the code, ChatGPT frequently makes alterations that result in the dysfunction of the others. After much discussion with the chat-bot, the code (Figure 8) contained fewer functioning features than the initially generated code, albeit with no error messages.


Figure 8: Screenshot of the final GUI generated by Method 3 - Attempt 1. Note that there is no 'save image' feature.

Attempt 2:
For the second attempt, the programmer created a list of more detailed requirements. However, this large amount of input information revealed a limitation of ChatGPT, wherein the chatbot conveyed its limit when taking in long and complicated requests:
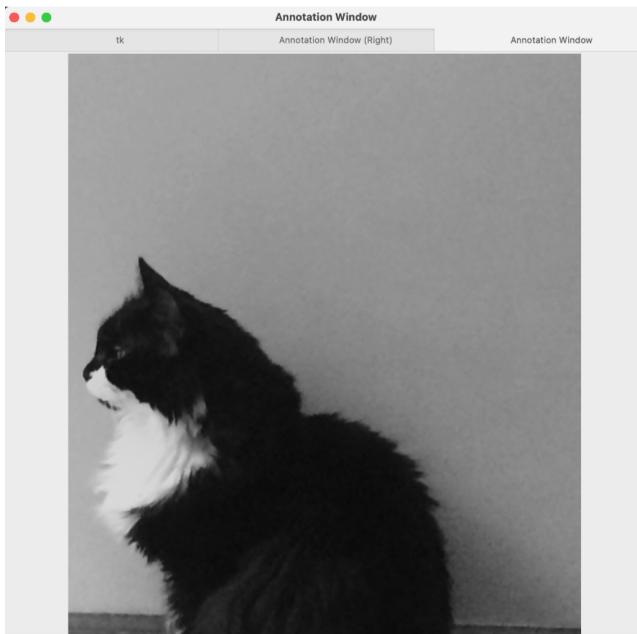
User:
Create a GUI for image annotation that satisfies the following conditions: . . .

ChatGPT:
Creating a complete GUI application with all the described features is beyond the scope of a text-based response. However, I can provide you with an outline of the steps involved in building such an application using Python and some popular GUI libraries. To create the GUI for image annotation with the specified features, you can follow these steps: . . .

The term "text-based response" is most interesting as it was able to form large programs as output during prior attempts and methods.

With some prompting, ChatGPT generated a sample of code for the image annotation program. When asked to review its code, the system identified unfilled parts of the listed requirements. Each ChatGPT-generated code was input as new code to be corrected. At one point, ChatGPT even asked for more specific instructions to best direct its alterations. Eventually, to increase specificity, the chat-bot was requested to create a program (Figure 9) with three tabs; the first one had annotating features such as panning and zooming, the second one was a continuously updated view of the annotated image, and the third tab was a blank window that appeared to serve no purpose.

Figure 9: Screenshot of the GUI generated by Method 3 - Attempt 2. Note that multiple tabs are generated.

With the final layout, the toolbar was not accessible so that the tabs could not serve their designated purposes.

Method Takeaway: Overall, the "From Scratch List Requirements" method was ineffective as it provided too much information upfront for the system to process. Despite the clarity of the input given, ChatGPT failed to provide output that took all elements of the request into account.

2.4 Method 4: From Scratch Assisted

The method titled "From Scratch Assisted" starts with entering a rough baseline code as created through user knowledge gained throughout this study and prior methods. This code generated from scratch by the user does not meet all of the specifications required or run and will be corrected and further developed through recommendations by ChatGPT.

Attempt 1:

The first attempt used Tkinter [13] in a program that contained unresponsive 'Browse', and 'Clear All' buttons; black and white brushes; tool size slider; and zoom and pan features. As such, the GUI was nonfunctional. The general process was to suggest a new feature, debug, and then evaluate the new code based on prior knowledge. Once the code demonstrated some functionality or comprehensible logic by having a few error codes, a new feature would be suggested. ChatGPT successfully interpreted error codes, such as the one included in Figure 10 while also providing explanations when requested.

When given this text-dense error message, ChatGPT responds as follows:

ChatGPT:

The error message `TclError: cannot use geometry manager pack inside . which already has slaves managed by grid` *indicates that you are trying to mix the pack and grid geometry managers in your code, which is not allowed.*

In your code, you are using both grid and pack to manage widgets within the same parent widget. In the init method, you have used a grid to place widgets in the root window. However, later on, you are using pack to manage the annotation_canvas widget.

To fix this error, you need to be consistent and use either grid or pack for all widgets within a given parent widget. In this case, since you are already using grid to manage other widgets, you should use grid for the annotation_canvas widget as well.

The chat-bot also provided an updated code meant to address the issue. One current limitation of ChatGPT is the inability to highlight within blocks of provided code. When requesting that ChatGPT modify pre-written code, it can be challenging to pick out the changes the chat-bot made. Upon request, comments inside the code indicate the changes, but even those can be difficult to spot when considering lengthy blocks of code.

Another feature of ChatGPT is its ability to recall elements from earlier parts of the conversation. For instance, after conversing back and forth about specific flaws in an earlier block of code to suggest changes, the chat-bot implemented the suggestions into the original code. The response was characterized by a comparable level of confidence and positivity:

ChatGPT:

Certainly! Here's the complete code for the image annotation tool based on the discussions we've had…

The final product (Figure 11) had all the required buttons except for a save button; however, the buttons did not successfully activate any features or annotation tools upon selection, leading the GUI to be ultimately ineffective.

Attempt 2:

The second attempt used Plotly Dash [11] in a program that, similar to the one used in Attempt 1, lacked the capability to upload images and had non-functional buttons. The initial code was constructed from scratch drawing from pieces of code gathered throughout previous methods and inspired by examples of programs with similar features.

After using the chat-bot to fix the code and get it running (Figure 12), the layout included a full toolbar that remained accessible regardless of the window size. The markups became erased whenever the tool color or size was changed. The saved image had a white, gridded background and did not retain the same size as the original. In addition, it still lacked the buttons to trigger the requested features. ChatGPT suggested JavaScript to create custom buttons, but they remained dysfunctional. However, the chat-bot seemed reluctant to offer Python alternatives to the JavaScript buttons.

Method Takeaway: Overall, the programmer found this method to be most effective. ChatGPT struggles to produce code from scratch and sometimes uses logic that can be challenging to follow.

```
-----------------------------------------------------------------------
TclError                               Traceback (most recent call last)
Cell In[1], line 197
    193             self.zoom_factor = min(width_ratio, height_ratio)
    195             self.update_canvas()
--> 197 Image_Markup()

Cell In[1], line 56, in Image_Markup.__init__(self)
     53 self.setup()
     55 self.annotation_canvas = tk.Canvas(self.root)
---> 56 self.annotation_canvas.pack(side = tk.LEFT, fill = tk.BOTH, expand = True)
     58 self.annotation_canvas.bind("<ButtonPress-1>", self.start_pan)
     59 self.annotation_canvas.bind("<B1-Motion>", self.pan_image)

File ~/anaconda3/lib/python3.10/tkinter/__init__.py:2425, in Pack.pack_configure(self, cnf, **kw)
   2409 def pack_configure(self, cnf={}, **kw):
   2410     """Pack a widget in the parent widget. Use as options:
   2411     after=widget - pack it after you have packed widget
   2412     anchor=NSEW (or subset) - position widget according to
   (...)
   2423     side=TOP or BOTTOM or LEFT or RIGHT -  where to add this widget.
   2424     """
-> 2425     self.tk.call(
   2426         ('pack', 'configure', self._w)
   2427         + self._options(cnf, kw))

TclError: cannot use geometry manager pack inside . which already has slaves managed by grid
```

Figure 10: A sample of the error code given to, and successfully deciphered by, ChatGPT.
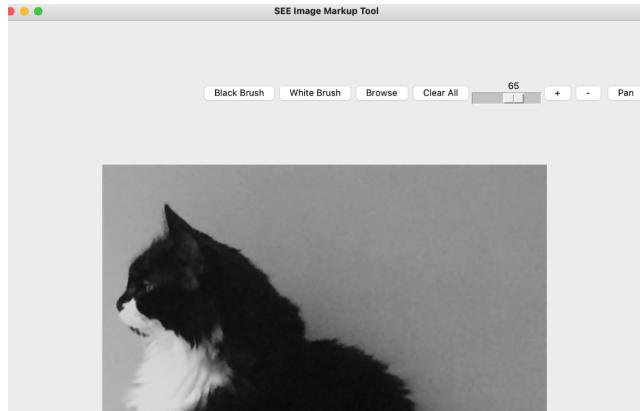

Figure 11: Screenshot of the GUI generated by Method 4 - Attempt 1. Note that the toolbox may not be accessible if the window size is too small.

As such, it is easier for the programmer to start with a clear road map in mind and a rough draft of code.

2.5 Method 5: Using GPT-4

In this methodology, the programmer initially requested ChatGPT to develop a GUI based on a specified set of requirements, though not exhaustively. Subsequently, the generated code was tested, followed by iterative refinements and enhancements, including the addition of new features, aligning this approach with Method 1.

Initially, the GUI encountered an issue where the displayed image was disproportionately small. The GUI, embedded within a Python window, was then transformed into a web-based interface, utilizing Python with Flask [12], HTML, and JavaScript. A critical feature involved the transmission of x- and y-coordinates from user-drawn points to Python. Additionally, the application was designed to overlay an annotation mask on the input image, enabling users to
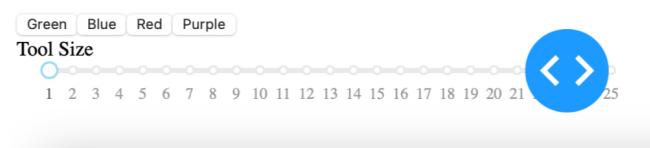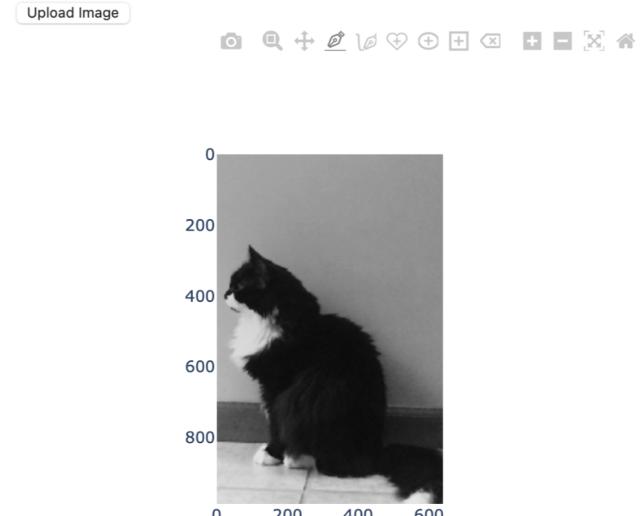

Figure 12: Screenshot of the GUI generated by Method 4 - Attempt 2.

create segmentation annotations without altering the original image. This functionality involved dynamic mask updates, including

real-time display refreshment upon user interaction, and the capacity to save the mask as a separate PNG file. Therefore, the essential features incorporated were: (a) Utilization of two image elements – the original base image and the overlay (mask); (b) Incorporation of a canvas for drawing on the overlay; (c) Functionality to save the overlay as a new image; (d) Dynamic updating of the mask in response to user edits.
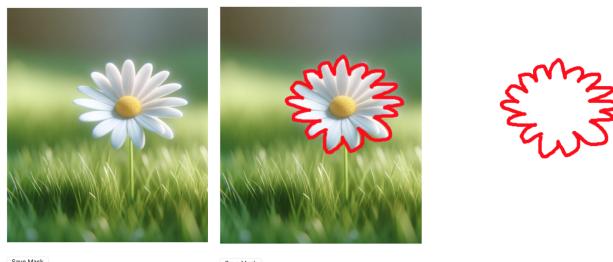
Despite the successful integration of these features, the GUI initially lacked a 'save' button. After multiple iterations, it was discerned that the issue stemmed from the stacking order of HTML elements, with the canvas inadvertently obscuring the button. This realization was attributed to the programmer's prior experience, underscoring the significance of foundational programming knowledge in resolving such issues.

The final refinement involved addressing an issue where the GUI would connect the end point of one drawing segment to the start point of the next upon resumption. ChatGPT resolved this by modifying the mouseup event handler to invoke the beginPath() method. This adjustment ensured that new drawing segments initiated after releasing the mouse button did not connect with the preceding segments, thereby enhancing the application's functionality and user experience.

Figure 13 presents a screenshot of the rudimentary GUI developed with Method 5. The interface of the GUI is minimalistic, primarily displaying the image in its original dimensions alongside a 'Save Mask' button situated beneath the image. Users have the capability to annotate directly on the image; these annotations, appearing in red, are made on a supplementary canvas layer above the image, overlaying it without altering the original. Upon clicking the 'Save Mask' button, only the annotated mask is preserved.

This version of the GUI is intentionally basic, devoid of advanced functionalities such as zooming capabilities, varied pen colors for annotating distinct objects, or the ability to fill enclosed areas in the annotations. Its primary purpose is to facilitate a comparative analysis with the GPT-3.5 generated interfaces and to demonstrate the relative ease and efficiency of GUI generation using GPT-4, which required fewer iterations and refinements in requests.

Potential enhancements could include instructing the chat-bot to fill the annotated areas with a specific color, such as white, while rendering the background in black, thereby further refining the GUI's functionality.

Figure 13: Screenshot of the GUI generated by Method 5. This figure includes the original image (left), the user-annotated image (middle), their respective save buttons below each, and the saved mask after pressing the save button (right).

Method Takeaway: This method demonstrates a level of comparability to Method 1, wherein GPT-3.5 was utilized. The application of GPT-4 in this context revealed a marked improvement in the chat-bot's comprehension capabilities, leading to the fulfillment of requirements with fewer follow-up interactions and iterations. The code produced by GPT-4 was notably effective and met the desired criteria satisfactorily.

However, the experience and prior knowledge of the programmer played a pivotal role in identifying the mentioned issue that ChatGPT was seemingly unable to detect. Consequently, this underscores that sole reliance on ChatGPT, in the absence of foundational programming knowledge, is not advisable.

3 DISCUSSION AND RECOMMENDATIONS

Generally, leveraging prompt engineering with large language models like ChatGPT possibly leads to significantly improved outcomes. By thoughtfully designing prompts, users can more precisely steer the model towards producing the desired responses.

To reduce misunderstandings between the programmer and the chat-bot, specific instructions to ChatGPT are advisable. The programmer should maintain some skepticism regarding the output. While the chat-bot delivers responses that exude confidence, its output remains occasionally incorrect. It is recommended to have a well-defined objective: similar to the broader practice of coding, engaging with ChatGPT yields optimal results when the user approaches with a specific intention in mind. ChatGPT seems to provide the best responses when user requests are specific.

Either start small or break the code down.

ChatGPT typically exhibits limitations when processing large requests, often resulting in the omission of certain parts. It is therefore recommended to initiate interaction with either a small request or an initial draft, subsequently refining it piece by piece. The most effective strategy is contingent upon the nature of the project. If the program has interdependent components, starting with basic features and progressively elaborating them is advisable. If the program is characterized by sequential components, it would be more effective to break down the problem and refine it step-by-step.

Pay attention to the modifications made by ChatGPT and ask questions to understand the logic.

While ChatGPT is capable of analyzing large text inputs, it may not thoroughly address every aspect of the code. Providing suggestions is more beneficial than code corrections, as they can provide starting points for further investigation. Additionally, a compiled list of recommendations may be more comprehensible than suggestions embedded within the code. As stated earlier, ChatGPT cannot be highlighted within a Python code box. While it can insert comments, such annotations can be challenging to distinguish from comments input by the user.

4 CONCLUSION

ChatGPT can be a supportive tool for novice programmers. While the chat-bot is not adept at generating highly functional code from scratch, it excels in identifying errors and interpreting the logic of existing programs. Due to its conversational nature, ChatGPT is

particularly effective at responding to specific questions. In general, the practice of refining prompts through prompt engineering with this advanced language model probably enhances the result quality. By carefully crafting and adjusting prompts, users can direct the model more effectively toward generating outputs that meet their specific requirements.

This study merely scratched the surface of ChatGPT's capacity to aid in user interface programming. A more granular examination of specific methods might enable programmers to discern patterns in the system's responses and check for replaceable conversations.

Additionally, a programmer with experience in graphical user interface development is better positioned to evaluate the potential of chat-bots in enhancing their productivity in GUI design.

REFERENCES

[1] 2024. ChatGPT — Release Notes | OpenAI Help Center. https://help.openai.com/en/articles/6825453-chatgpt-release-notes. Accessed on 3/8/2024.

[2] Katharina Buchholz. 2023. Infographic: Threads Shoots Past One Million User Mark at Lightning Speed. https://www.statista.com/chart/29174/time-to-one-million-users/. Accessed: 2023-11-12.

[3] Enrico Checcucci, Paolo Verri, Daniele Amparore, Giovanni E. Cacciamani, Cristian Fiori, Alberto Breda, and Francesco Porpiglia. 2023. Generative Pre-training Transformer Chat (ChatGPT) in the scientific community: the train has left the station. *Minerva Urology and Nephrology* 75, 2 (April 2023), 131–133. https://doi.org/10.23736/S2724-6051.23.05326-0

[4] Dirk Colbry. 2022. GUI-Basic. https://github.com/see-insight/see-tkinter-gui/blob/dd5c50c5bb0ca92f63cc9b48b85230c2a5d65311/GUI-Basic.py.

[5] Dirk Colbry. 2023. Toward an automatic exploration of algorithm space to speed up image annotation for applications in scientific image understanding. In *IEEE AIPR workshop*. St. Louis, MO.

[6] Jeffrey A. Clark (Alex) and contributors. 2024. Pillow - Python Imaging Library (Fork). https://pypi.org/project/Pillow/. Accessed on 3/8/2024.

[7] Gary Marcus. 2022. What to Expect When You're Expecting … GPT-4. https://garymarcus.substack.com/p/what-to-expect-when-youre-expecting. Accessed: 2023-11-12.

[8] Dan R. Olsen. 2007. Evaluating user interface systems research. https://dl.acm.org/doi/10.1145/1294211.1294256. In *Proceedings of the 20th annual ACM symposium on User interface software and technology (UIST '07)*. Association for Computing Machinery, New York, NY, USA, 251–258. https://doi.org/10.1145/1294211.1294256

[9] OpenAI. 2024. OpenAI. https://openai.com. Accessed on 3/8/2024.

[10] OpenCV. 2024. OpenCV - Open Computer Vision Library. https://opencv.org. Accessed on 3/8/2024.

[11] Plotly Dash. 2024. Dash Python User Guide. https://dash.plotly.com. Accessed on 3/8/2024.

[12] Armin Ronacher. 2024. Flask Web Framework. https://flask.palletsprojects.com/. Accessed on 3/8/2024.

[13] Tkinter. 2024. tkinter — Python interface to Tcl/Tk. https://docs.python.org/3/library/tkinter.html. Accessed on 3/8/2024.

APPENDIX

Here we present the final, fully functional version of the code generated in Method 5 using GPT-4, to facilitate replication and further research by interested parties. This includes the `app.py` file, a Flask [12] application, and the accompanying `index.html`.

In `app.py`, we employ the `render_template('index.html', form=form)` function, which integrates Flask with the HTML frontend, showcasing the interaction between backend logic and frontend presentation. This setup exemplifies the practical application of AI-generated code in a web development context.

app.py:

```python
1  from flask import Flask, render_template, request,
       jsonify, redirect, url_for
2  from flask_wtf import FlaskForm
3  from wtforms import SubmitField
4  from PIL import Image
5  import os
6  import base64
7  from io import BytesIO
8
9  app = Flask(__name__)
10 app.config['SECRET_KEY'] = 'mysecretkey'
11
12 class SaveMaskForm(FlaskForm):
13     save = SubmitField('Save Mask')
14
15 @app.route('/', methods=['GET', 'POST'])
16 def index():
17     form = SaveMaskForm()
18     if form.validate_on_submit():
19         # The functionality of saving the mask
       will be handled in the JS.
20         # This redirect is just to reload the form
        after the POST request.
21         return redirect(url_for('index'))
22     return render_template('index.html', form=form
       )
23
24 @app.route('/save_mask', methods=['POST'])
25 def save_mask():
26     data = request.json
27     mask_data = data['mask_data']
28
29     mask_img = Image.open(BytesIO(base64.b64decode
       (mask_data.split(',')[1])))
30     mask_img_name = "flower_mask.png"
31     mask_img.save(os.path.join('static',
       mask_img_name))
32
33     return jsonify(status='success')
34
35 if __name__ == "__main__":
36     app.run(debug=True)
```

index.html:

```html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-
       width, initial-scale=1.0">
6      <title>Image Annotation</title>
7      <style>
8          #control-panel {
9              margin-top: 20px;
10         }
11     </style>
12 </head>
13 <body>
14
15 <img id="sourceImage" src="/static/flower.png"
       style="position: relative; z-index: 0;" alt="
       Source Image">
16 <canvas id="myCanvas" style="position: absolute;
       top: 0; left: 0; z-index: 1;"></canvas>
17
18 <div id="control-panel">
19     <button type="button" onclick="saveMask()">
       Save Mask</button>
20 </div>
21
22 <script>
23     let canvas = document.getElementById('myCanvas
       ');
24     let ctx = canvas.getContext('2d');
```

```
25    let img = document.getElementById('sourceImage
      ');
26
27    let isDrawing = false;
28
29    img.onload = function() {
30        canvas.width = img.width;
31        canvas.height = img.height;
32    }
33
34    canvas.addEventListener('mousedown', function(
      event) {
35        isDrawing = true;
36        draw(event.pageX - canvas.offsetLeft,
      event.pageY - canvas.offsetTop);
37    });
38
39    canvas.addEventListener('mousemove', function(
      event) {
40        if (isDrawing) {
41            draw(event.pageX - canvas.offsetLeft,
      event.pageY - canvas.offsetTop);
42        }
43    });
44
45    canvas.addEventListener('mouseup', function()
      {
46        isDrawing = false;
47        ctx.beginPath();   // This line clears the
      current drawing path
48    });
49
50    function draw(x, y) {
51        ctx.lineWidth = 5;
52        ctx.lineCap = 'round';
53        ctx.strokeStyle = 'red';
54
55        ctx.lineTo(x, y);
56        ctx.stroke();
57        ctx.beginPath();
58        ctx.moveTo(x, y);
59    }
60
61    function saveMask() {
62        const canvasData = canvas.toDataURL();
63        fetch('/save_mask', {
64            method: 'POST',
65            headers: {
66                'Content-Type': 'application/json'
67            },
68            body: JSON.stringify({
69                'mask_data': canvasData
70            })
71        })
72        .then(response => response.json())
73        .then(data => {
74            if (data.status === 'success') {
75                alert("Mask saved successfully!");
76            } else {
77                alert("Error saving mask!");
78            }
79        });
80    }
81 </script>
82
83 </body>
84 </html>
```