November 2025

Volume 16 Issue 2

JoCsE

Journal Of Computational Science Education

Promoting the Use of Computational Science Through Education

ISSN 2153-4136 (online)

Contents

Volume 16, Issue 2

Introduction to Volume 16 Issue 2 1
David Joiner, Editor

Scaling Instructional Workflows in Data Science Education using JupyterHub and Otter-Grader 2
Sai Annapragada.

Coding through Storytelling: Narrative Reasoning and Software Engineering Education 5
S. Charlie Dey, Jeaime H. Powell, Victor Eijkhout, Joshua Freeze, and Susan Lindsey

Classroom Applications of Question Formulation to Support Problem-Solving in Computer Science 10
James Quinlan and Michael Todd Edwards

Enhancing HPC Education through Virtual Cluster Administration and Benchmarking 16
Charles Ross Lindsey, Jeffrey Valdez, Aaron Jezghani, Will Powell, and Richard Vuduc

Facilitating Academic Research with FPGA Support in a University Data Center 22
Jeevesh Choudhury, Thomas Jennewein, and Gil Speyer

A Novel 3D Recurrent R-CNN for Medical Imaging Feature Detection: A Case Study for Coronary Calcium Detection 29
Vikas Sarvasya, Robert Gotwals, and Liam Butler

Introduction to Volume 16, Issue 2

David Joiner, Editor
Kean University
djoiner@kean.edu

This issue of our journal features six new contributions that showcase current advances in computational science education, data science pedagogy, high-performance computing, and applied machine learning. Together, these papers demonstrate the continued evolution of computational thinking as a cornerstone of modern education and research practice.

This issue's featured articles are as follows:

Sai Annapragada presents Scaling Instructional Workflows in Data Science Education using JupyterHub and Otter-Grader, describing the implementation of a scalable instructional workflow at the University of California, Merced. By integrating JupyterHub, Otter-Grader, and GitHub, the approach streamlines notebook distribution and automated grading, reducing administrative overhead and improving reliability for large data science courses.

S. Charlie Dey, Jeaime H. Powell, Victor Eijkhout, Joshua Freeze, and Susan Lindsey introduce Coding through Storytelling: Narrative Reasoning and Software Engineering Education. This paper explores a pedagogical model that encourages students to "tell the story" of their code—using narrative reasoning to enhance debugging, metacognitive awareness, and clarity of expression in software development.

James Quinlan and Michael Todd Edwards share Classroom Applications of Question Formulation to Support Problem-Solving in Computer Science, describing the use of the Question Formulation Technique (QFT) to help students learn to ask more effective questions. The authors report improvements in engagement and critical thinking, offering practical prompts and reflections for educators seeking to strengthen inquiry in computing instruction.

Charles Ross Lindsey, Jeffrey Valdez, Aaron Jezghani, Will Powell, and Richard Vuduc present Enhancing HPC Education through Virtual Cluster Administration and Benchmarking, detailing the use of a hardware-agnostic "Virtual Cluster" environment to teach system administration and performance benchmarking. Their approach provides students with authentic, hands-on experience in HPC configuration and management while lowering infrastructure barriers.

Jeevesh Choudhury, Thomas Jennewein, and Gil Speyer describe Facilitating Academic Research with FPGA Support in a University Data Center. The paper examines strategies employed at Arizona State University to make FPGA-based research more accessible, highlighting efforts to integrate high-level synthesis tools and lower the expertise threshold for sustainable, energy-efficient high-performance computing.

Vikas Sarvasya, Robert Gotwals, and Liam Butler present A Novel 3D Recurrent R-CNN for Medical Imaging Feature Detection: A Case Study for Coronary Calcium Detection. This student-led research project introduces a deep learning model capable of accurately identifying coronary structures and estimating calcium scores in non-gated chest CT scans, demonstrating the growing role of computational methods in medical image analysis.

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.

Finally, this issue is dedicated to the memory of **Dr. Robert M. Panoff**, whose life and work profoundly shaped our community. I first met Bob in October of 1999, as a nearly-newly-minted Ph.D. applying for a position at Shodor. For me personally, he became both a mentor and a friend, and my years working with him shaped me more than I can say. His unwavering belief that anyone could become a scientist, coupled with his empathy and deep understanding of those around him, were truly legendary. Bob founded the Shodor Education Foundation in 1994 and led it for three decades, building a national community of educators who brought computational science into their classrooms. Through Shodor and the National Computational Science Institute, he trained thousands of teachers and faculty and helped inspire an entire generation of computational thinkers. He was also one of the guiding voices in founding this journal and a member of the editorial board from its creation. For so many of us, the work we continue to do carries forward in his honor and memory.

Sincerely, Dave Joiner

Scaling Instructional Workflows in Data Science Education using JupyterHub and Otter-Grader

Sai Annapragada
University of California, Merced
sannapragada@ucmerced.edu

ABSTRACT

At the University of California, Merced (UCM), an instructional workflow w as a dopted t o s upport t he t eaching o f d ata science at scale. This workflow integrated JupyterHub, Otter-Grader, and GitHub to facilitate browser-based notebook execution, simplify assignment distribution, and automate grading. Initially built around a shared-folder model—where instructors placed course materials in a shared-readwrite directory that automatically appeared as a read-only shared directory for all students—the system transitioned to a GitHub-based setup using nbgitpuller. This shift allowed instructors to distribute assignments and course materials through direct links, removing the need for students to navigate the shared folder manually. By doing this, the need for admin privileges was removed, reducing the risk of accidental deletion of course content from the shared read-write folder. This paper presents our instructional strategy, key challenges addressed, implementation experience, and insights for educational institutions seeking to adopt similar models.

KEYWORDS

Data Science Education, JupyterHub, Otter-Grader, GitHub, Educational Workflows, Scalable Instruction

1 NATURE OF THE TRAINING OR EDUCATION PROGRAM

The pilot implementation, initiated by the Department of Applied Mathematics, introduced the use of JupyterHub in computational courses during the Fall 2023 and Spring 2024 semesters. The pilot implementation was carried out in Data Science (DSC 008), an introductory data science course, with student enrollments averaging between 50-100 students per term. The coursework included weekly assignments, labs, and a term project, with an emphasis on interactive, code-based exploration of datasets. The data science Python package, initially developed by UC Berkeley, was adopted to make programming more accessible to students from diverse backgrounds. As a pedagogical tool, it allows students to carry out basic data science tasks–such as loading, summarizing, and visualizing data without needing to first learn more complex libraries like pandas or matplotlib. With this pilot implementation, several issues related to JupyterHub were identified and resolved, and it helped establish a walkthrough for teaching and sharing course materials using JupyterHub and nbgitpuller.

Jupyter Notebooks allow students to write and execute code directly in their browsers, eliminating the need for individual software installation. Faculty can deliver assignments and instructional content using this unified interface. The environment was designed to promote reproducibility, modular coding practices, and collaborative learning. The need for a scalable and standardized approach became evident as multiple instructors and teaching assistants were involved in managing large classrooms. Consequently, the workflow was expanded to address these operational and pedagogical requirements.

2 STRATEGY

In the pilot implementation, JupyterHub was deployed as a centralized computing environment to host course notebooks and assignments. Faculty members and instructors were granted administrative access, allowing them to upload teaching materials to a shared read-write directory. Students had access to a shared read-only folder from which they could copy content into their own workspace.

While this method was simple to implement, it soon led to several challenges:

- Instructors with administrative privileges could unintentionally overwrite or delete each other's course content.
- Teaching assistants, who were also enrolled in other courses as students, had access to confidential materials such as solution notebooks.

To mitigate these issues and establish clearer boundaries between access roles, we introduced nbgitpulle—a browserbased open-source tool that integrates with JupyterHub and GitHub. In the revised workflow, instructors maintained two separate Git repositories: one private repository containing answer keys and test cases, and a public repository containing only the student-facing materials. Using nbgitpuller, faculty can generate custom links that point to specific repositories and branches, which students could access directly from Cat-Courses (LMS) or email. Upon clicking the link, JupyterHub would launch automatically and pull the necessary files into the student's home environment.

This approach eliminated the need for a shared folder and removed administrative access for instructors and teaching assistants. It also ensured that every student received an identical copy of the materials in an isolated environment, thereby reducing compatibility issues. Larger datasets, where file sizes exceeded GitHub's 25 MB limit, were handled separately by having RISI engineers upload them to the shared folder upon request.

https://doi.org/10.22369/issn.2153-4136/16/2/1

3 ASSESSMENT OR EVALUATION TECHNIQUE

The effectiveness of this workflow was evaluated using a combination of technical observations and feedback from faculty and teaching staff. During both the pilot phase and the actual implementation phase, each user was initially allocated 1GB of RAM with 2 CPU cores. However, for the Data Science course, additional computational power was required to complete assignments efficiently. Upon consultation with faculty, it was decided to offer users the option to choose between 1 GB and 2 GB of RAM, depending on the complexity and computational needs of each assignment. This adjustment helped instructors and students avoid notebook crashes while working on resource-intensive tasks.

Faculty surveys and informal interviews indicated a high level of satisfaction. Once instructors began using JupyterHub and nbgitpuller, the JupyterHub infrastructure team consistently reached out to faculty to check for any issues related to the newly adopted process of distributing and grading notebooks. This included communication through emails, visits to their offices based on availability, and attending classes where JupyterHub was used—particularly in data science-related courses. These direct interactions helped identify challenges faced by instructors and students during instruction. Based on this feedback, the RISI engineer provided on-the-spot technical support to address issues and improve the overall classroom experience.

Moreover, the number of support requests related to software setup and assignment access saw a significant decline. The volume of grading queries was also reduced due to the successful implementation of Otter-Grader. As part of this implementation, course notebooks were converted into an Otter-Grader–compatible format. Otter-Grader is a Python-based package that enables instructors to design assignments and projects directly within notebooks by embedding test cases and solutions and assigning marks to each question. When a student answers a question, Otter-Grader allows them to check their response against the expected output, and if there are any errors, it can generate hints based on the provided solutions. This approach not only helps students verify their work in real time but also allows instructors to automate the grading process through integration with CatCourses(LMS) and Gradescope.

4 EVALUATION OF ITS SUCCESS

The transition to a Git-based, reproducible workflow brought about measurable improvements in operational efficiency and instructional quality. Stability of the JupyterHub environment improved markedly following the RAM upgrade. Administrative challenges were eliminated by removing elevated access privileges for non-technical users. With the integration of Otter-Grader and Gradescope, instructors were able to automate grading workflows, particularly in large classes where manual grading would otherwise have been time-consuming and error-prone. During the pilot phase, Otter-Grader-based Jupyter Notebooks were introduced to help instructors create assignments and projects with embedded test cases, predefined solutions, and marks. These notebooks allowed students to check their work in real time and receive hints based on the provided solutions.

Gradescope, an online grading platform, was then integrated with CatCourses (LMS) using the LTI 1.3 protocol. This integration enabled students to submit their notebooks digitally, and instructors or teaching assistants could grade them automatically using the test cases created within Otter-Grader. The system supported both scanned paper-based submissions and digital notebooks, thereby streamlining the grading process and improving efficiency.

As a result of the combined implementation of JupyterHub infrastructure, Otter-Grader, nbgitpuller, and Gradescope integration, students no longer faced setup-related issues for data science coursework. The infrastructure was reliably available and accessible 24/7, and there was no need for students to install packages manually. This allowed them to focus entirely on the learning objectives. Additionally, faculty and teaching assistants were relieved from setup and grading burdens, allowing them to spend more time supporting students with course content and concepts.

5 LESSONS LEARNED

5.1 Risks of Shared Administrative Access

One of the primary lessons learned was that shared environments with administrative access pose risks to content integrity and privacy. Moving to a version-controlled system using GitHub and nbgitpuller provided a clear separation of responsibilities and improved accountability. However, this transition required faculty to undergo basic training in Git, including branch management and version control. It was also observed that even faculty members with prior experience using Git and related tools were not always following best practices, leading to inconsistencies in content sharing and collaboration.

5.2 Need for Controlled Package Installation

During the rollout, it became evident that instructors required various Python and R packages to support their coursework. To streamline package installation and ensure reliability, a staging JupyterHub environment was introduced. This allowed the infrastructure team to test new packages before deploying them to the production hub, maintaining stability and compatibility.

5.3 Standardizing the Package Request Process

To formalize and simplify the package request process, a ServiceNow request workflow was implemented, and a knowledge base article was developed to guide instructors through the steps. This provided a traceable and standardized process for requesting packages, reducing ad hoc installations and improving coordination between faculty and the support team.

5.4 Adoption of Otter-Grader and Gradescope

Setting up Otter-Grader required an initial investment of time to define test cases and structure the notebooks accordingly. However, once established, the tool provided long-term benefits by enabling automated grading. CatCourses (LMS) integration with Gradescope also required careful configuration but proved effective in delivering a fully automated grading pipeline.

5.5 Development of JupyterHub Documentation

To support faculty and teaching assistants in using these tools effectively, a dedicated documentation website was developed for

JupyterHub. This site includes detailed sections on distributing notebooks using GitHub and nbgitpuller, designing assignments with Otter-Grader, and grading them through Gradescope. The documentation has proven helpful in onboarding new instructors, reducing repeated support queries, and promoting consistent workflows across multiple courses.

6 REPRODUCIBILITY OF INSTRUCTIONAL JUPYTERHUB SETUP AND RESOURCE ALLOCATION PROCESS

All tools utilized in this workflow—JupyterHub, GitHub, Otter-Grader, Gradescope, and nbgitpuller—are open-source or institutionally supported platforms. The setup is reproducible by institutions with access to basic cloud or server infrastructure.

At UC Merced, the JupyterHub solution is deployed on a Kubernetes-based cloud platform. For institutions aiming to replicate this environment, a reasonable starting point for minimum server requirements for JupyterHub on Kubernetes includes 2 CPUs, 4 GB of RAM, and 16 GB of disk space. Storage for each user generally starts at 10 GB and can be adjusted based on specific course needs. However, actual resource requirements will vary depending on the number of users, usage patterns, and the nature of notebooks being executed.

For smaller classes or individual departments, an alternative solution such as The Littlest JupyterHub (TLJH) can be considered. TLJH can be deployed on a user's own virtual machine or physical server. The server must have at least 1 GB of RAM, with 128 MB reserved for TLJH and associated services.

Template repositories and configuration guides are being developed to support reuse and expansion of the setup. The approach requires minimal technical overhead for students, as they access the environment through their browser. Faculty and support staff need moderate technical skills to maintain Git repositories and prepare Otter-based assignments. The modular nature of this workflow allows for easy extension across academic terms and disciplines.

7 RELEVANCE TO THE BROADER RANGE OF TRAINING OR EDUCATION TOPICS

By setting up an infrastructure like the one described in this paper, institutions can adopt tools such as JupyterHub, GitHub, and Otter-Grader for instructional use in a structured and scalable manner. These platforms, widely used in research environments, can be configured to support teaching by providing a unified and accessible environment for coding, analysis, and evaluation.

The browser-based nature of JupyterHub significantly lowers the barrier to entry for students, especially those who are new to programming or do not have access to high-end computing devices. As no local installation is required, students can concentrate on the coursework without technical distractions. Faculty and teaching assistants benefit from consistent workflows that simplify content delivery and assignment management across multiple courses. This instructional model can be extended beyond data science to support computational topics in engineering, biology, economics, and other fields that require hands-on coding and analysis.

8 CONCLUSION

The instructional JupyterHub setup at UC Merced has demonstrated a successful model for scaling data science education through integrated technology and workflows. By transitioning from a shared-folder approach to a Git-based system using nbgitpuller, the solution addressed critical issues around content management, access control, and distribution.

The implementation of Otter-Grader streamlined assessment processes, allowing instructors to automate grading and provide immediate feedback to students. This approach has significantly reduced the administrative burden on faculty while enhancing the student learning experience through consistent access to computational resources and educational materials.

The infrastructure choices made—specifically the Kubernetes-based JupyterHub deployment with appropriate resource allocation—have proven to be reliable and scalable for supporting classes of varying sizes. For institutions with different needs or resources, The Littlest JupyterHub offers a more lightweight alternative. The lessons learned from this implementation highlight the importance of clear access controls, standardized processes for package management, and comprehensive documentation. These insights can inform similar initiatives at other educational institutions seeking to scale their computational teaching environments.

As computational methods continue to expand across disciplines, the approach described in this paper offers a model that can be adapted beyond data science to support diverse educational needs where interactive, code-based learning is beneficial. The browser-based, standardized environment reduces technical barriers for students and allows faculty to focus on teaching rather than troubleshooting.

ACKNOWLEDGEMENTS

Anthropic AI tool Claude Consensus was used in the authoring of this paper as a research tool.

A APPENDIX: ADDITIONAL RESOURCES

The following resources provide more information about the tools and platforms discussed in this paper:

- JupyterHub: https://jupyter.org/hub
- The Littlest JupyterHub: https://tljh.jupyter.org/
- Otter-Grader: https://otter-grader.readthedocs.io/
- Data Science Python Package: https://www.data8.org/datascience/
- Gradescope: https://www.gradescope.com/
- nbgitpuller: https://nbgitpuller.readthedocs.io/en/latest/link.html
- UC Merced JupyterHub: https://jupyterhub.ucmerced.edu/
- UC Merced Grading Documentation: https://ucm-it.github.io/hpc_docs/docs/jupyter/canvas
- Canvas-Gradescope Integration: https://guides.gradescope.com/hc/en-us/articles/23586543164173-Using-Gradescope-LTI-1-3-with-Canvas-as-an-Instruct

Coding through Storytelling: Narrative Reasoning and Software Engineering Education

S. Charlie Dey
Texas Advanced Computing Center
fcharlie@tacc.utexas.edu

Jeaime H. Powell
Omnibond Systems
fjeaime@omnibond.com

Victor Eijkhout
Texas Advanced Computing Center
feijkhout@tacc.utexas.edu

Joshua Freeze
Texas Advanced Computing Center
fjfreeze@tacc.utexas.edu

Susan Lindsey
Texas Advanced Computing Center
fslindsey@tacc.utexas.edu

ABSTRACT

To become a successful software engineer, technical competence alone is not enough. Students must learn to reason about their code, articulate their intentions, and locate errors with clarity and confidence. This paper introduces a pedagogical approach rooted in the metaphor of "telling a story." By encouraging students to narrate their code—identifying protagonists (variables), plotlines (control flow), and conclusions (outputs)—we promote a practice of self-explanation that strengthens metacognitive awareness and debugging skills. Drawing from experiences in the classroom, we show how storytelling helps students pinpoint bugs, communicate intent, and ultimately write more understandable code. We connect these practices with existing research on metacognition, program comprehension, and human-centered computing, and describe how this narrative approach provides a scalable, inclusive, and transferable tool for future computational engineers and scientists.

KEYWORDS

Metacognition, Human-centered programming, teaching coding

1 INTRODUCTION

In undergraduate high-performance computing (HPC) programming courses, a common challenge is students' difficulty in articulating the context of their code during debugging sessions. Typically, students focus narrowly on specific syntactic, semantic, or logical errors, omitting the broader purpose of their program, which hinders effective instructor guidance. Traditional prompts like "walk me through your code" often prove challenging as students simultaneously navigate the problem domain, programmatic issues, and the syntax of a new programming language. To address this, a pedagogical approach grounded in narrative reasoning—a cognitive framework that leverages humans' natural ability to process sequences and causality—has been adopted [2]. This method, implemented through a storytelling exercise, encourages students to shift from a micro-level focus on errors to a macro-level observation of their code's logical narrative, aligning with constructivist learning principles.

1.1 Classroom Implementation

The storytelling approach involves prompting students to "tell the story of your code," often with a playful opener like "Once upon a time…" This shifts their perspective from being immersed in technical details to viewing their code as a narrative with characters (variables), plot (control structures), and resolution (outputs). For example, a student might describe a variable as a character navigating a dataset, making decisions based on conditions, and collecting results. When the narrative falters—such as when a student cannot logically continue the story—it often signals a misunderstanding of a function, algorithm, or logical flow, pinpointing the bug's location. This process also reveals syntactic errors, like missing punctuation or misnamed variables, as students verbalize their logic in a human-readable format.

This paper argues that narrative reasoning—a natural human faculty for explaining sequences, causality, and change—can be harnessed as a powerful tool in software engineering education. Specifically, it supports:

- Self-explanation during coding and debugging
- Improved program comprehension through structured reasoning
- Communication and collaboration via shared narrative framing

Through classroom observations, existing research, and proposed curricular strategies, we introduce a narrative framework that promotes reflective practice among novice programmers.

2 DISCUSSION

Grounded in constructivist learning theory, the storytelling approach encourages learners to actively build mental models by narrating the structure and intent of their code. We present a model that maps programming constructs to narrative elements—such as characters, setting, and plot—to promote comprehension, self-explanation, and reflective practice. Supported by research in metacognition, program comprehension, and learner-centered design, this approach not only improves debugging and problem-solving skills but also fosters engagement and communication, particularly among novice and diverse learners. Together, these foundations establish storytelling as both an effective teaching tool

https://doi.org/10.22369/issn.2153-4136/16/2/2

and a cognitive scaffold for navigating the complexities of programming.

2.1 Educational Framework: Constructivism

This approach is rooted in the constructivist learning framework, which posits that learners actively construct knowledge through experience and reflection [9]. By narrating their code, students externalize their mental models, making assumptions and errors explicit, which facilitates refinement of their understanding. The narrative structure serves as a scaffold, connecting new programming concepts to familiar storytelling patterns, thus enhancing comprehension and retention.

2.2 Model Overview

The educational model for "coding through storytelling" is a narrative pedagogy that encourages students to conceptualize code as a story with narrative elements: characters (variables and objects), plot (control flow), setting (initial conditions), conflict (conditionals or loops), and resolution (outputs or goals). Grounded in constructivism, which posits that learners actively construct knowledge through experience and reflection [9], this model integrates several pedagogical practices to foster metacognition, problem-solving, and communication skills. It aligns with research on program comprehension as storytelling [2] and metacognition in programming [5].

2.3 Benefits of the Storytelling Approach

The narrative-based pedagogy offers several educational benefits, supported by empirical research:

(1) Improved Debugging: Narrating code shifts students' focus to the program's logical flow, revealing inconsistencies or misunderstandings that pinpoint bugs. This aligns with findings on self-explanation, which show that verbalizing thought processes enhances problem-solving [8].
(2) Enhanced Metacognition: The act of storytelling fosters metacognitive awareness, encouraging students to reflect on their understanding and problem-solving strategies, a critical skill in programming education [5].
(3) Increased Engagement: The playful prompt "Once upon a time…" adds levity to debugging, making it more approachable, particularly for diverse learners, as supported by learner-centered computing education principles [4].
(4) Better Communication: Narrating code helps students articulate their intent clearly, a vital skill for collaborative software development, aligning with broader educational goals [4].
(5) Enhanced Program Comprehension: Framing code as a narrative improves understanding by leveraging natural cognitive tendencies to process stories, as evidenced by research on program comprehension [2].

2.4 Academic Foundations

The storytelling approach is supported by several areas of computing education research:

- Narrative Reasoning: A systematic review highlights storytelling's role in software development, noting its benefits in planning, requirements elicitation, and prototyping, providing a theoretical basis for its use in education [2].
- Self-Explanation: Verbalizing thought processes during programming improves learning outcomes, as demonstrated in a randomized experiment with self-explanation assignments [8].
- Metacognition: Metacognitive strategies, such as reflection and self-regulation, are essential for programming novices, and storytelling prompts such reflection [5].
- Program Comprehension: Understanding code as a narrative enhances comprehension by aligning with cognitive processes for processing sequences and causality [1].
- Learner-Centered Design: Storytelling makes programming more accessible and engaging, particularly for diverse student populations, aligning with learner-centered educational approaches [4].

2.5 Metacognition and Self-Explanation

Metacognition—thinking about one's own thinking—is a critical skill in learning to program. Students who engage in self-explanation while coding are more likely to detect misunderstandings and refine their mental models of how code behaves. Loksa et al. [5] describe self-regulation strategies in novice programmers, including planning, monitoring, and debugging as metacognitive acts that enhance learning outcomes.

Narration is one such form of self-explanation. By "telling the story" of what a program does, students externalize their reasoning process—making it easier to spot contradictions, misconceptions, or gaps in understanding.

2.6 Code as Story

Ciancarini et al. [2] explored the idea that program comprehension can be improved by treating code as a narrative structure. They liken variables to characters, control flow to plot, and comments to narration—a framing that helps learners understand not just what the code does, but why. This aligns with how expert programmers often read code: not linearly, but semantically—constructing a story that explains behavior.

Recent neuroscience also supports this framing. Peitek et al. [6] used fMRI to show how code complexity and vocabulary burden working memory during comprehension, aligning with storytelling's role in structuring mental load.

2.7 Debugging, Deconstruction, and Literacy

Debugging as hypothesis testing is a narrative act—imagining what should happen, then identifying where the story breaks. Griffin [3] advocates for "deconstructionist" learning, where reading, tracing, and debugging come before code writing, aligning well with our approach.

Storytelling in teaching also supports literacy, creativity, and critical thinking. Satriani [7] found storytelling enriched vocabulary, engagement, and comprehension in literacy classrooms—benefits that transfer to code comprehension as well.

3 THE NARRATIVE PEDAGOGY

3.1 Characters, Plot, and Purpose

In the classroom, we frame code as a story with:

- Protagonists: Variables, objects, or agents that change or act
- Setting: Initial conditions and inputs
- Plot: Control flow — how the story unfolds step-by-step
- Conflict: Conditionals or loops that create decision points
- Resolution: Output, state changes, or goals achieved

This structure helps students identify the logical and conceptual components of a program more intuitively. By mapping code to familiar storytelling elements, students create a mental schema that aids in both comprehension and retention.

We often begin exercises by giving students a blank narrative framework and asking them to fill it in with elements from a given code snippet or problem description. This reverse-engineering of story from code builds analytical skills, while designing stories before writing code builds synthesis.

3.2 Writing the Story Before the Code

While debugging often reveals where the narrative breaks down, equally powerful is the practice of writing the story first. Before any code is written, students are encouraged to map out the "story arc" of their program:

- Who are the characters? (variables, inputs, actors)
- What is the setting? (initial conditions or assumptions)
- What is the problem/conflict? (what needs to be solved or computed)
- What is the plot? (algorithm or logical steps)
- How does the story end? (output or goal)

This pre-coding narrative acts as a mental simulation of the algorithm and guides students away from immediately jumping into syntax. It shifts the focus from what code to write to what problem to solve—a critical reframe, especially for novice programmers.

This practice also aligns with professional software design principles, such as:

- Test-driven development, where the tests represent the expected outcomes of the story
- Design-first thinking, where logic is mapped out before implementation
- Algorithm sketching, used in pseudocode or flowchart form to visualize intent

By embedding storytelling at the design stage, we encourage foresight, structure, and intentionality in programming. It also supports better communication in team settings, as students can articulate the purpose and logic of their code before implementation.

3.3 Finding the Bug Through the Broken Story

In debugging exercises, students are asked to "read aloud the story" of their code. Where the story breaks—where they pause, contradict themselves, or say "I'm not sure what this part does"—is almost always the bug's location.

This technique externalizes cognition, making their mental model visible. It also mirrors professional practices: code reviews and pair programming sessions often revolve around shared narrative explanations of what code is doing.

We also incorporate peer-debugging activities where students exchange code and provide narrative explanations of what the code is "supposed" to do. This collaborative storytelling not only builds comprehension but fosters peer support and critical dialogue.

4 CLASSROOM IMPLEMENTATION

4.1 "Storytelling Debugging" Sessions

In small-group help sessions, office hours, or lab time, instructors and TAs engage students in live storytelling about their code. These sessions begin with the student walking through their program line by line, describing what each part is supposed to do, in their own words. Interruptions or breakdowns in the narrative usually signal areas of confusion or bugs. The instructor can then prompt deeper reflection with questions like:

- What's the role of this variable here?
- What happens next in the story?
- Does the ending make sense given the plot so far?

This Socratic approach encourages self-correction and metacognitive awareness. These sessions are particularly effective in early-course projects where logic may be simple, but the storytelling gap is often large.

4.2 Reflective Journals and Pair Storytelling

To reinforce the storytelling practice outside structured help sessions, students maintain reflective journals where they document the story of their code weekly. These can include narrative descriptions, flow diagrams, or even short paragraphs written as if explaining their solution to a non-programmer. Prompts might include:

- What was the goal of my code this week?
- Who were the main characters (variables/functions)?
- What surprised me in the process?

In "pair storytelling," students take turns reading and retelling each other's code in small groups. This not only strengthens their understanding of syntax and semantics but also develops critical peer review skills. When a peer can't explain a section clearly, it becomes a clue for the author to refine either the code or their internal logic.

4.3 Rubrics and Evaluation

Assessment in storytelling-based instruction can include narrative clarity as a rubric category. Instructors can evaluate:

- The completeness and coherence of a student's narrative explanation
- The alignment between the intended story and actual code behavior
- The student's ability to identify turning points or conflicts in the logic

These assessments don't replace functional correctness but enrich it, allowing instructors to gain deeper insight into student thinking and development.

4.4 Integration with Curriculum

The narrative framework is not a standalone technique—it can be scaffolded across the curriculum. Early weeks can introduce storytelling as a reflection tool. By mid-semester, storytelling becomes a design strategy before implementation. In later projects, students use it collaboratively for design proposals and team check-ins. Embedding narrative reasoning throughout ensures it's internalized as part of the student's cognitive toolkit.

4.5 Example Activities

Consider a student debugging a parallel computing program designed to process large datasets. When asked to explain their issue, they might initially point to a specific line causing a runtime error. By prompting them to "tell the story of your code," the instructor encourages a narrative: "Once upon a time, a variable named data_chunk set out to process a portion of the dataset. It entered a loop to compute averages, but then it got stuck because the loop never ended." This narrative might reveal that the student misunderstood the termination condition of the loop, highlighting a logical error. As the student continues, they might notice a missing semicolon that disrupted the syntax, which becomes apparent when explaining the code's flow. This example illustrates how storytelling shifts the student's perspective, enabling them to identify both logical and syntactic issues. Examples include:

- The Delivery Bot: Debugging broken logic by following the story of a package.
- Game Storyboarding: Mapping input, flow, and outcome in games like Rock-Paper-Scissors.
- Data Visualization Stories: Connecting data interpretation with narrative.
- Peer Storytelling: Retelling and debugging a partner's code to highlight clarity and intent.

4.5.1 Example 1: The Delivery Bot (Narrative Debugging). Students are given the following function:

```
def setDeliveryTime(package):
   if not package['priority'] and package['weight'] > 10:
        return "afternoon"
    else:
        return "morning"
```

They are asked to tell the story of what happens to each package:

```
package1 = {'priority': True, 'weight': 12}
package2 = {'priority': False, 'weight': 8}
package3 = {'priority': False, 'weight': 15}
```

They identify that a priority package might still go in the morning even if it's very heavy—an unintended outcome. This prompts a redesign:

```
def setDeliveryTime(package):
    if package['priority']:
        return "morning"
    elif package['weight'] <= 10:
        return "morning"
    else:
        return "afternoon"
```

4.5.2 Example 2: Game Mechanics as Story (Design-First Storyboarding). Before coding a Rock-Paper-Scissors game, students storyboard the interaction:

- Who are the players?
- What events trigger the next move?
- How does the system declare a winner?

Students present the story visually before implementing any logic. This helps identify branching conditions and user interaction flow.

4.5.3 Example 3: Data Storytelling with Visualization (Narrative Reflection). After analyzing a dataset and creating a visualization, students write a brief narrative explaining:

- What the data shows
- What question their code answers
- What story the visualization tells

This aligns narrative structure with data reasoning.

4.5.4 Example 4: Explaining a Peer's Code (Peer Storytelling). Students swap programs and narrate each other's code out loud. Prompts include:

- What is the goal of the program?
- What are the key steps along the way?
- Where is a potential flaw in the story?

This activity strengthens code readability, testing comprehension, and fosters collaborative debugging.

5 LIMITATIONS AND FUTURE WORK

While promising, narrative pedagogy has limitations. Students with less fluency in English or those more familiar with mathematical abstractions may find storytelling unnatural at first. Additionally, there is a risk of oversimplifying technical concepts if too much emphasis is placed on metaphor over precision. Instructors must strike a balance between promoting narrative clarity and ensuring computational correctness.

This approach also requires additional instructional time and scaffolding that may not be feasible in all classroom contexts. Larger courses may face challenges integrating personalized storytelling activities or giving feedback on reflective work like journals.

Another limitation is the potential variance in how students interpret and construct stories. While diversity of thought can be an asset, it can also lead to misaligned mental models if not guided carefully. Further research is needed to explore how cultural and linguistic backgrounds influence narrative construction in programming education.

Future work will investigate how narrative practices scale in larger courses, how they integrate with peer programming and automated assessment tools, and whether they foster long-term improvement in code quality and debugging efficiency. Controlled studies comparing narrative and non-narrative cohorts could validate impact on learning outcomes. There is also opportunity for tool development—intelligent IDEs or tutoring systems that can prompt students to articulate their story during code construction.

Moreover, the emergence of AI and large language models (LLMs) presents an interesting new frontier. Narrative-based pedagogy may

inform the development of smarter educational prompts, explainable AI code assistants, and curriculum-aware LLMs that can co-construct stories with learners. By studying how students tell code stories, we may also improve how machines understand, generate, and teach code narratives—paving the way for more collaborative, human-centered computing.

6 CONCLUSION

While promising, the storytelling approach faces challenges, such as objectively assessing narrative quality and scaling it to large classes, which requires instructor training. Some students may initially find narrating code awkward, particularly if they lack confidence. Future research could quantify the approach's impact through controlled studies and explore tools to integrate storytelling into programming environments, as suggested by related work [2].

Storytelling is a universal human tool for reasoning through complexity. By treating code as a story, students learn to explain their thinking, debug more effectively, and write clearer programs. This paper presented a narrative pedagogy that maps programming constructs to storytelling elements—variables as characters, control flow as plot, and output as resolution. Such framing supports student understanding, especially when reinforced through narrative-based planning and story-first design strategies outlined.

We explored classroom practices that support storytelling, including reflective journaling, peer code narration, and live debugging as story deconstruction. These methods empower students to identify and resolve logic gaps, communicate their intent, and grow into thoughtful engineers. As software becomes more embedded in every discipline, helping students become reflective, communicative, and narrative-driven coders prepares them not only for technical success but also for collaborative, human-centered innovation.

As software becomes more embedded in every discipline, helping students become reflective, communicative, and narrative-driven coders prepares them not only for technical success but also for collaborative, human-centered innovation.

REFERENCES

[1] Ruven Brooks. 1983. Towards a theory of the comprehension of computer programs. *International Journal of Man-Machine Studies* 18(6) (1983), 543–554. https://doi.org/10.1016/S0020-7373(83)80031-5

[2] Paolo Ciancarini, Mirko Farina, Ozioma Okonicha, Marina Smirnova, and Giancarlo Succi. 2023. Software as storytelling: A systematic literature review. *Computer Science Review* 47 (2023), 113–120. https://doi.org/10.1016/j.cosrev.2022.100517

[3] Jean Griffin. 2016. Learning by taking apart: Deconstructing code by reading, tracing, and debugging. In *Proceedings of the 17th Annual Conference on Information Technology Education (SIGITE '16)*. Association for Computing Machinery, Boston, Massachusetts, 148–153. https://doi.org/10.1145/2978192.2978231

[4] Mark Guzdial. 2015. Learner-Centered Design of Computing Education: Research on Computing for Everyone. *Synthesis Lectures on Human-Centered Informatics* 8(6) (2015), 1–165. https://doi.org/10.2200/S00684ED1V01Y201511HCI033

[5] Dastyni Loksa, Lauren Margulieux, Brett A. Becker, Michelle Craig, Paul Denny, Raymond Pettit, and James Prather. 2022. Metacognition and self-regulation in programming education: Theories and exemplars of use. *ACM Transactions on Computing Education* 22(4) (2022), 1–37. https://doi.org/10.1145/3487050

[6] Norman Peitek, Sven Apel, Chris Parnin, and Andre Brechmannand Janet Siegmund. 2021. Program comprehension and code complexity metrics: An fMRI study. In *IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. Madrid Spain, 524–536. https://doi.org/10.1109/ICSE43902.2021.00056.

[7] Intan Satriani. 2019. Storytelling in teaching literacy: Benefits and challenges. *English Review: Journal of English Education* 8 (2019), 113–120. https://doi.org/10.25134/erjee.v8i1.1924

[8] Arto Vihavainen, Craig S. Miller, and Amber Settle. 2015. Benefits of self-explanation in introductory programming. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. Association for Computing Machinery, Kansas City, Missouri, 284–289. https://doi.org/10.1145/2676723.2677260

[9] Ernst von Glasersfeld. 1989. Cognition, construction of knowledge, and teaching. *Synthese* 80 (1989), 121–140.

Classroom Applications of Question Formulation to Support Problem-Solving in Computer Science

James Quinlan
University of Southern Maine
james.quinlan@maine.edu

Michael Todd Edwards
Miami University
edwardm2@miamioh.edu

ABSTRACT

Questions are an integral part of the teaching and learning process. As students ask questions, they explore complex ideas, challenge assumptions, and confront contradictions. Too often, however, students do not know what questions to ask. They are hesitant to reveal their misunderstandings in front of their classmates. Other students don't participate because they are not invested in the course content. This paper presents the Question Formulation Technique (QFT), a teaching method designed to provide computer science students with targeted instruction on question-posing. As students learn to ask better questions, they become more confident in their abilities as learners. In this experience report, we provide a framework and implementation process, highlighting key steps and potential outcomes. Drawing on instructor observations, we report improvements in student engagement and critical thinking, while also discussing the limitations of anecdotal evidence and outlining directions for future research. Through in-class examples, we discuss the method's strengths and limitations while offering sample prompts that can be adapted for classroom use.

KEYWORDS

inquiry, student questioning, computer science education

1 INTRODUCTION

Undergraduate computer science departments experience high failure and dropout rates in introductory programming courses (IPCs) [14]. If students drop out of an IPC, they are unlikely to enroll in subsequent computer science courses [27]. Too often, instructors in IPCs emphasize syntax and semantics over problem-solving and collaboration [13], discouraging students who may be interested in computer science but need more programming experience. Students without such experience must learn syntax and problem-solving simultaneously, a daunting task for many undergraduates.

How can we better support such students? One approach is to provide them with more meaningful opportunities to learn from each other and by asking (and answering) questions. Indeed, asking questions is central to learning, in general, [1, 5, 6, 10, 15] and is a key component in problem-solving [4]. When students ask questions, they tend to experience improved learning outcomes [2]. Moreover, students develop a growth mindset [7] towards learning. Asking questions helps students become more confident in their ideas and abilities. Students begin to see themselves as knowledge creators rather than passive participants [19]. When students hone their questioning skills, they are better prepared outside the QFT context to ask questions, for instance, at professional meetings and in interviews [16, 20, 23]. Given the many benefits of questioning, it is surprising how little time is devoted to question-posing in computer science classrooms.

How might you encourage your students to ask more thoughtful, computer science-related questions in the classes you teach? In the sections that follow, we discuss the application of Rothstein and Santana's Question Formulation Technique (QFT) [22] in computer science classrooms. QFT is a pedagogical method to promote and advance students' questioning skills. The technique has been successfully applied in various subjects, including English [6], biology [10], electrical circuits [16], and mathematics [5]. QFT is a versatile method that can introduce students to a new topic, assess their knowledge and understanding of previously taught content, or wrap up a topic to evaluate their growth. Research across educational settings has demonstrated the efficacy of QFT in promoting student inquiry, curiosity, and engagement. In mathematics education, Mannion [18] reported measurable improvements in students' performance on open-ended, written-response questions after exposure to QFT. Similarly, Summers et al. [24] found that undergraduate students developed questioning, creativity, and collaboration skills through regular use of the technique, as evidenced by assessment data. LeBlanc et al. [16] showed that QFT stimulated curiosity and technical question formulation in engineering courses. Research in college classrooms also highlights increased agency and engagement, with students demonstrating greater ownership of their learning [11]. Comprehensive reviews further support QFT as a research-based practice to engage learners in various subjects [15, 17, 26].

These findings suggest that QFT is a robust, evidence-based approach for enhancing learning outcomes across various disciplines. While the present report relies on instructor observation, the broader literature provides a formal evaluation of QFT's effectiveness in promoting active learning and inquiry.

In the following sections, we share how we have used QFT with undergraduates enrolled in an IPC. In Section 2, we outline the steps of the QFT framework, with a significant emphasis on constructing high-quality prompts (see Subsection 2.1). Student-centered steps are covered in Subsections 2.2, 2.3, 2.4, 2.5, and 2.6. In Section 3, we provide a brief discussion of limitations and offer implementation tips (see Section 3.2). In addition to the limitations, Section 3.1 includes ideas for future work to address the limitations.

https://doi.org/10.22369/issn.2153-4136/16/2/3

2 THE QFT FRAMEWORK

Six general steps comprise the QFT process [22, p. 4].

(1) *Design Question Focus*: The teacher designs a prompt for students. This is referred to as the "QFocus."
(2) *Generate Questions*: Students generate a list of questions about the QFocus.
(3) *Revise Questions*: Students revise questions, enhancing their readability and making them more conducive to further investigation.
(4) *Prioritize and Select Question*: Students discuss which questions seem most engaging.
(5) *Investigation/Implementation*: Students apply their question(s) to a classroom assignment such as a homework problem, a code experiment, or a lab project.
(6) *Reflection*: In small groups and whole-class presentations, students share their end products and discuss what they learned through the six steps of the QFT process.

Steps 2–6 may be shortened or altered to accommodate various needs or preferences. For example, students may work in groups instead of individually. Educational contexts, such as online distance learning situations or novel course scheduling, may also necessitate modifications. In the remainder of this section, we discuss the six steps in the QFT Process in greater detail, illustrating an implementation of QFT in an IPC classroom using a case study approach.

2.1 Design Question Focus

In the first step of the QFT process, the instructor designs and shares a question focus (QFocus) with students. As the Right Question Institute [21] notes:

> *The QFocus is a stimulus, a springboard, that students will use to ask questions. The QFocus can be a sentence, phrase, image, or situation that will be the "focus" for generating questions* (p. 3).

The QFocus is *not* a question itself—its purpose is to stimulate *student* questions. Since the QFocus sets the stage for the rest of the QFT experience, its importance cannot be overstated.

Early in our IPC courses a brief QFocus, "programming languages," helps us ascertain our students' pre-existing knowledge of programming languages. Later on, when we seek to deepen our students' understanding of more elaborate topics, we provide a QFocus that requires students to consider ideas from different vantage points. For instance, when our students study recursion, we present the QFocus "recursion and iteration" to students to encourage them to think more deeply about the relationships between recursion and iteration.

The QFocus should be narrowly focused to draw students' attention in specific directions. Vague or broad prompts make it difficult for students to formulate meaningful questions. Developing a suitable QFocus is challenging, so it's helpful to keep the following criteria in mind when designing them:

(1) The QFocus should not be a question.
(2) The QFocus should produce different lines of questioning.
(3) The QFocus should be simple, yet not overly simplistic [12].

Note that the QFocus can be graphical rather than purely textual. We've provided objects, images, word clouds, hashtags, and animated GIFs as QFoci. Consider, for instance, the flowchart depicted in Figure 1.

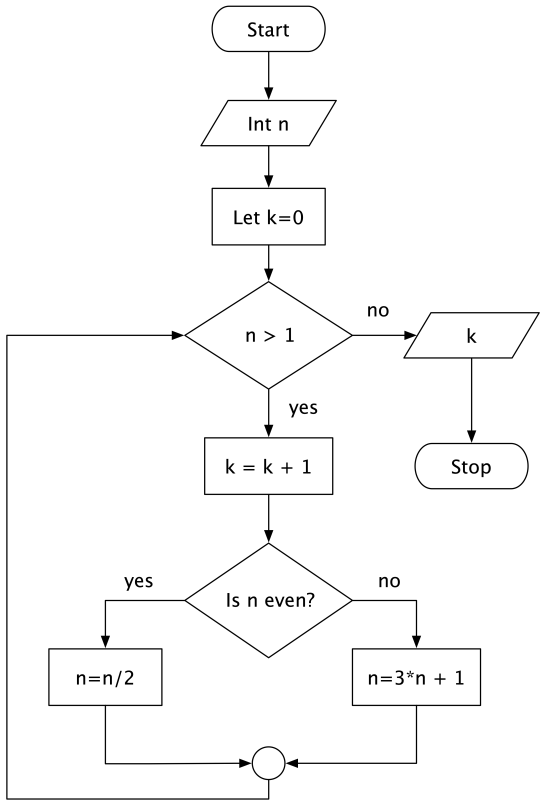

Figure 1: A flowchart of the Collatz conjecture as a visual QFocus.

This visually engaging representation is designed to capture students' attention and motivate them to think about coding and syntax, as well as the functionality of the code, its potential applications, and the process of revision.

Moreover, note the mathematical context embedded within the flowchart—namely, the Collatz conjecture. We find it beneficial to include mathematics in introductory computer science courses, similar to, but opposite of Friend et al. [9], who advocates for including computer science ideas in mathematics coursework. In general, we find it helpful to "hide" mathematical topics in our QFoci as this approach connects computer science ideas to content (e.g., mathematics), which is arguably more familiar to IPC students. Additionally, the method provides robust topics that lend themselves to student questioning.

Figure 2 illustrates a different visual QFocus designed to engage students in recursive thinking through the mathematics of the Tower of Hanoi puzzle. The prompt encourages students to consider how the problem might be solved using code. With this prompt, students have asked questions about loops, decision structures, syntax, recursion, and code efficiency. This is a great prompt for small groups of 3 to 4 students.

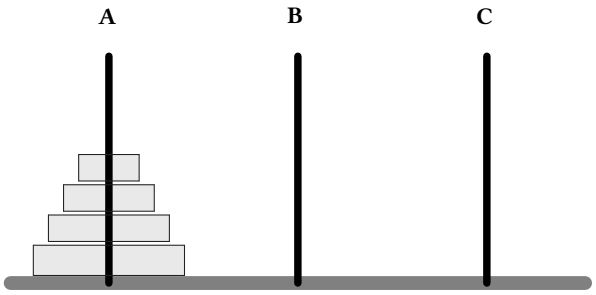

Figure 2: Move all disks from A to C without letting a larger disk be on top of a smaller disk, moving only one disk at a time.

2.1.1 More Examples of QFoci Prompts. The following is a list of QFoci we have presented to our students. The QFoci are presented in *italics* with brief commentary.

- *Increase in surface area of a sphere to increased radius.* This QFocus could be used to develop problem-solving and reinforce arithmetic operations in a specific programming language.
- *The Harmonic series,*

$$\sum_{k=1}^{\infty} \frac{1}{n}.$$

  Students are confronted with how to implement a summation with infinite terms and infinite loops. This might lead to questions regarding convergence, as well as round-off errors in floating-point systems.
- *The limit,*

$$\lim_{x \to 0} \frac{\sin(x)}{x}.$$

  This prompt encourages students to consider loops and syntax more carefully. Moreover, depending on the programming language (e.g., Python), students may need to import standard math libraries to implement their code.
- *Primes of the form* $4k + 1$ *and* $4k + 3$. In 1853, Russian mathematician Pafnuty Chebyshev hypothesized that there are more primes of the form $4k + 3$ than of the form $4k + 1$, up to the same limit [3]. We've used Chebyshev's conjecture to launch student discussions about modular arithmetic, cardinality, branching, and looping.
- *Sorting large datasets.* This QFocus can prompt students to formulate questions about algorithmic complexity, runtime efficiency, memory usage, and the practical trade-offs between different sorting methods.
- *Facial recognition software.* Although slightly outside the typical scope of an IPC, this prompt effectively engages students, as most have existing opinions or experiences, thus encouraging questions that extend beyond purely technical aspects of computer science.

2.2 Generate Questions

After presenting the QFocus, students typically work individually or break into small groups to generate questions. This step is the most crucial for students.

We provide 10 minutes to generate questions for the first few QFT exercises. After repeated practice, this step requires less time. Assigning this step as homework is another possibility, providing more time for students to generate questions.

We provide the following "rules of engagement" to our students *each* time we employ QFT, following [22].

(1) Ask lots of questions.
(2) Do not discuss, judge, or answer questions.
(3) Record questions precisely as initially stated.
(4) Change statements into questions.

We have found it beneficial to model words or behaviors that are "judging" in a whole-group setting before breaking students into smaller groups. Additionally, depending on your class, providing students with time to generate questions individually may be valuable before sharing them with others, as it can support equitable participation. This is particularly practical for students with limited English proficiency or less computer science experience.

We ask groups to generate at least ten questions. In our experience, the number of questions is somewhat inversely proportional to their quality. When we require too many, students tend to "pad" their list with trivial questions to meet the minimum requirement. We also encourage students to ask any questions that come to mind and remind them that the "best" questions will be selected from their list later in the process (see Sections 2.3 and 2.4).

Ultimately, students sift through the questions they generate and select a handful for further investigation. For instance, after examining the flowchart of the Collatz conjecture from Figure 1, students generate questions such as the following:

(1) Why study the Collatz conjecture?
(2) What is the Collatz conjecture, and how is it formulated?
(3) How can we store intermediate iterates?
(4) Which loop is appropriate for implementing the Collatz conjecture as code?
(5) Does the input need to be a positive integer?
(6) What if we do not input an integer?
(7) What is the code corresponding to the logic of this flowchart?
(8) What are the longest-known sequences?
(9) Can all flowcharts be reduced to code? Can a flowchart represent all code?
(10) What happens if the update formulas for n are changed?
(11) What if we made more than two update conditions? For example, what happens if we change from mod 2 to mod 3?

Note the variation in the questions. Some indicate previous experience with computer science concepts, while others aren't specific to computer science at all. Some will require significant work to answer, while others require only a one-word response, such as "Yes" or "No."

In the next section, we discuss the next step of the QFT Process—namely, revising questions. We provide tips to help your students assess and adjust first-draft questions for maximum impact. Ultimately, in subsequent phases of the QFT, each student will construct and select a question to explore in more detail.

2.3 Revise Questions

Revision leads to higher-quality questions. After the brainstorming phase, students revisit their questions for clarity. Additionally, we ask students to transform any "closed" questions into "open" ones. Using the following definitions, we explain that open questions are better suited for exploration and research.

- *Closed questions* have a single correct answer, such as 'yes' or 'no', or other factual information. For example, "Do we have to use a loop?", "Is there only one way to implement branching?"
- *Open questions* have multiple right answers and come in two flavors: interpretive and evaluative. *Interpretive questions* must be supported with evidence. For example, "Which code has the fastest runtime, and why?" *Evaluative questions* ask for opinions, beliefs, or points of view and have no wrong answers. "Do particular integers have common sequence characteristics?"

Open questions are preferable for teaching and learning because they elicit expanded thinking and processing of information [8]. We have our students classify their questions by placing an "O" next to their open questions and a "C" next to their closed questions [21].

Next, we model the transformation of a closed question into an open one. For instance, closed questions may be "opened" by adding 'how' or 'why' to the beginning. We ask students to brainstorm other techniques and post their list on the classroom whiteboard for reference.

At this point, students revise their questions. We collect students' revisions to understand their thinking process better and assess their understanding of open and closed questions. Consider, for instance, the fourth question about the Collatz conjecture.

> *Which loop is appropriate for implementing the Collatz conjecture as code?*

It is closed since this question can be answered with a simple response (e.g., "for loop" or "while loop"). However, a slight modification makes it *open*—namely, adding "how" to the query: "*How does one determine which type of decision structure is best for implementing the Collatz conjecture as code?*" Similarly, question 5, "*Does the input need to be a positive integer?*", can be converted to an open question by rephrasing it to be, "*why does the input need to be a positive integer?*"

2.4 Selecting Questions

Once students have revised their questions, they determine which will generate the most productive exploration of course concepts. We stipulate that individuals, or groups, generate a "top 5 list" of their best questions. For each selected question, students write a brief (1-2 sentence) rationale justifying its inclusion in the "top 5."

The following prompts may help students prioritize questions for possible exploration:

(1) *Is the question suitably narrow?* In other words, is the topic narrow enough that one could reasonably be expected to answer the question within the timeframe of the assignment?
(2) *Is the topic properly connected to the QFocus and the instructional objectives of our course?*
(3) *Will the topic help you strengthen your understanding of a particular area of computer science?* In other words, what capacity does the question have to push your learning and understanding of computer science?

We have students answer the prompts for each question in their "top 5 list" to help them make a final selection. Once they've selected a question, students are ready to begin the next stage of the process, namely Investigation/Implementation.

2.5 Investigation/Implementation

As University of Michigan notes [25], students could use their questions as a starting point to:

- Develop a lab experiment.
- Design a product or process.
- Write a research paper.
- Deliver a presentation.
- Prepare for an in-class discussion or debate.

In general, we've found it advantageous to introduce QFT with relatively short, low-stakes projects—such as writing pseudo-code for a short practice program or preparing for an in-class topic discussion. Using QFT in the first few weeks of class allows students to learn the process without worrying about a class grade.

For the Collatz conjecture, many of our introductory programming students focus on writing code to explore the topic (e.g., finding sequences of a particular length). Those with more coding experience often opt to learn more about the conjecture or compare various code implementations (e.g., "bit hacks").

In one class, we asked students to summarize the findings from their question in a "one-pager" (e.g., one page of code or a short research report of 250-500 words) and include a bibliography and inline citations from peer-reviewed sources. We've found that restricting their writing to one page encourages students to concentrate on the selected question without diverging into unrelated topics.

At the next class meeting, students share their one-pagers, highlighting key ideas and solutions they uncovered as part of the research process. As students share their findings, they gain confidence in their capabilities as self-directed learners and benefit from considering content from multiple points of view—rather than from the instructor's view alone.

2.6 Reflection

In the Reflection phase, students can produce a summary document showcasing what they (or their group) learned through the QFT process. Although our students typically present their results in writing, findings can be disseminated through oral presentations or coding demonstrations. In addition to sharing content with classmates, students often share thoughts about the QFT process. Typically, sentiments include recognition of classroom engagement and confidence in posing questions. As students share their questions and corresponding answers or solutions, they demonstrate their grasp of the topic, their ability to think critically, and their proficiency in applying programming concepts.

During the reflection phase, it's essential to consider group dynamics if working in a group. To ensure that all students are actively engaged, it is important to assign each group member a defined

role. This promotes balanced participation, encourages equitable contribution, and supports individual accountability. For instance, we assign a leader for each group, who submits a link to a single document (e.g., a Google Doc) for their group. This document consists of a cover page and one additional page for each group member (i.e., a one-pager for each group member). The cover page includes the following:

- The group's initial list of questions, along with revisions.
- Markings indicating questions that were initially open ("O") or closed ("C").
- A summary paragraph that discusses what the group collectively learned from the QFT experience.

Other roles have included a scribe to document group decisions and a presenter to share findings with the class. Colleagues who frequently incorporate group work into their instruction have also employed roles such as 'devil's advocate', 'prioritizer', 'diverger' (to encourage divergent thinking), and 'converger' (to support convergent thinking).

3 DISCUSSION AND LIMITATIONS

We observed several notable improvements in student learning and engagement after implementing the QFT in our introductory programming courses. Moreover, we also observed that students' ability to formulate questions extended beyond the QFT session to regular class discussions and problem-solving activities.

Active Learning and Engagement. Based on instructor observations, students responded positively to QFT sessions, showing increased engagement, frequently volunteering to participate, and expressing enthusiasm during group activities. This was evident in the volume and quality of student participation, as students took ownership by generating, refining, and exploring their questions about course topics. While some students offered positive comments informally, we did not implement a systematic evaluation or survey regarding the technique.

Development of Critical Thinking. Instructor observations revealed that students moved beyond superficial or factual queries to ask more analytical and open-ended questions. For example, students were observed debating the efficiency of different coding strategies, seeking to understand not only "what" but "why" a particular approach worked. This deepened their exploration of programming concepts and promoted higher-order thinking skills.

Curiosity and Intellectual Exploration. Students' curiosity was demonstrated by their willingness to pose follow-up questions and explore "what if" scenarios beyond the initial prompt. For instance, some students asked about alternative ways to solve a problem or proposed modifying existing code to test new hypotheses. These behaviors reflected a genuine interest in understanding the material and a desire to experiment with their learning.

Teamwork and Communication. QFT sessions fostered a collaborative classroom environment. Students worked together to develop questions, compare perspectives, and present their findings. Through group discussions and peer presentations, students practiced articulating their ideas clearly and responding to feedback, thereby improving their communication and collaborative problem-solving skills.

3.1 Limitations and Future Work

We acknowledge that the findings reported in this study are based solely on instructor observation and informal classroom feedback; no formal surveys, quantitative assessments, or systematic data collection were conducted during the implementation of QFT in our courses. As a result, while we observed improvements in student engagement, critical thinking, curiosity, and teamwork, these observations are anecdotal and should be interpreted with caution.

To more rigorously assess the impact of QFT, future research should employ comprehensive evaluation methods. For example, controlled experiments could compare students exposed to QFT with those who are not, using pre- and post-assessments to measure changes in questioning skills, critical thinking, and programming knowledge. Qualitative data from interviews, focus groups, or open-ended surveys can offer deeper insights into students' experiences and perceptions of QFT. Another promising direction for future research would be to replicate the analysis conducted by Summers et al. [24], which examined students' questions, feedback, and reflections to trace the evolution of their questioning throughout the QFT process.

By conducting such systematic assessments, we can more precisely determine the effects of QFT and further refine strategies to promote active, student-centered learning in computer science education.

Additional avenues for future work include exploring optimal QFT implementation in online or hybrid environments, investigating its effectiveness across programming languages and course levels, and integrating QFT with other active learning strategies. Another promising direction involves exploring how generative AI tools might assist instructors in creating QFoci or help students refine their questions during the revision phase.

3.2 Tips for Implementing

Based on our experience, we provide several implementation suggestions.

- *Allocate specific class times*: Dedicate regular periods or portions for QFT sessions. This can help establish a routine and signal to students the importance of question formulation. In a CS1 course, we allocated 15-20 minutes *every* class period for Steps 2, 3, and 4. Consider adopting a flipped classroom model if the time spent on QFT activities limits the coverage of course content.
- *Assign parts as homework*: Steps 5 and 6 of the QFT process can be assigned as homework. This can also maximize in-class time for more collaborative aspects, like improving or prioritizing questions.
- *Gradual integration*: Instead of implementing the full QFT process immediately, consider introducing it gradually throughout the course. Start with simpler QFT activities and build up to more complex ones as students become more comfortable with the technique.
- *Model the process*: Demonstrate the QFT process yourself by modeling how to formulate, improve, and prioritize questions related to the course content.

- *Encourage self-evaluation*: Incorporate opportunities for students to self-evaluate their questions, reflect on their questioning skills, and set goals for improvement.
- *Collaborate and share*: Encourage students to collaborate in small groups during QFT sessions, allowing them to share and build upon each other's questions.
- *Provide feedback*: Offer constructive feedback on students' questions, highlighting strengths and areas for improvement in their question formulation abilities.

REFERENCES

[1] Ester Aflalo. 2021. Students generating questions as a way of learning. *Active Learning in Higher Education* 22, 1 (2021), 63–75.

[2] William S. Carlsen. 1991. Questioning in Classrooms: A Sociolinguistic Perspective. *Review of Educational Research* 61, 2 (1991), 157–178. http://www.jstor.org/stable/1170533

[3] Pafnuty Lvovich Chebyshev. 1853. Lettre de M. le Professeur Tchébychev á M. Fuss sur un nouveaux théorème relatif aux nombres premiers contenus dans les formes $4n + 1$ et $4n + 3$. *Bull. Classe Phys. Acad. Imp. Sci. St. Petersburg* 11 (1853), 208.

[4] Christine Chin and Jonathan Osborne. 2008. Students' questions: a potential resource for teaching and learning science. *Studies in science education* 44, 1 (2008), 1–39.

[5] Paul Davis. 1994. Asking Good Questions about Differential Equations. *The College Mathematics Journal* 25, 5 (1994), 394–400. http://www.jstor.org/stable/2687504

[6] Ken Donelson. 2008. The Art of Asking Questions: Two Classes That Changed My Teaching Life. *The English Journal* 97, 6 (2008), 75–78. http://www.jstor.org/stable/40503416

[7] Carol Dweck. 2015. Carol Dweck revisits the growth mindset. *Education week* 35, 5 (2015), 20–24.

[8] Cornell University Center for Teaching Innovation. 2020. Using effective questions: Center for Teaching Innovation. https://teaching.cornell.edu/fall-2020-course-preparation/engaging-students/using-effective-questions

[9] Michelle Friend, Andrew W Swift, Betty Love, and Victor Winter. 2023. A Wolf in Lamb's Clothing: Computer Science in a Mathematics Course. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. ACM, 256–262.

[10] L. Goodman and G. Berntson. 2000. The Art of Asking Questions: Using Directed Inquiry in the Classroom. *The American Biology Teacher* 62, 7 (2000), 473–476. http://www.jstor.org/stable/4450954

[11] Martha Higginbotham. 2023. *Teaching students to ask questions: The role of question formulation technique in building agency and student engagement in the college classroom.* Ph.D. Dissertation. Rider University.

[12] The Right Question Institute. [n.d.]. Designing the question focus (QFOCUS) - right question institute. https://rightquestion.org/wp-content/uploads/2019/05/RQI-Resource-An-Introduction-to-QFocus-Design-PDF.pdf

[13] Sohail Iqbal and Om Kumar Harsh. 2013. A self review and external review model for teaching and assessing novice programmers. *International Journal of Information and Education Technology* 3, 2 (2013), 120.

[14] Sohail Iqbal Malik and Jo Coldwell-Neilson. 2017. Impact of a new teaching and learning approach in an introductory programming course. *Journal of Educational Computing Research* 55, 6 (2017), 789–819.

[15] Todd Larsen. 2020. Using Student-Generated Questions to Promote Curiosity and Student Learning. (2020).

[16] Heath J LeBlanc, Kundan Nepal, and Greg S Mowry. 2017. Stimulating curiosity and the ability to formulate technical questions in an electric circuits course using the question formulation technique (QFT). In *2017 IEEE Frontiers in Education Conference (FIE)*. IEEE, IEEE, New York, NY, 1–6.

[17] Cristo Leon. 2024. Empowering Inquiry: The Transformative Power of the Question Formulation Technique in Education. (2024).

[18] Jessica M Mannion. 2019. The effectiveness of the question formulation technique on open-ended, written response questions in mathematics. (2019).

[19] Barbara L McCombs and Robert J Marzano. 1990. Putting the self in self-regulated learning: The self as agent in integrating will and skill. *Educational Psychologist* 25, 1 (1990), 51–69.

[20] Kaitie O'Bryan. 2017. Using QFT to prepare students for new experiences. *Kaleidoscope: Educator Voices and Perspectives* 1, 4 (2017), 29–31.

[21] The Right Question Institute. [n.d.]. Experiencing the question formulation techniqueTM (QFTTM). https://condor.depaul.edu/tps/resources/level1/experienceqft.pdf

[22] Dan Rothstein and Luz Santana. 2011. *Make just one change: Teach students to ask their own questions.* Harvard Education Press.

[23] Luz Santana. 2015. Learning to Ask Questions: A Pathway to and through College for Students in Low–Income Communities. *About Campus* 20, 4 (2015), 26–29.

[24] Mindi Summers, Jordann Fernandez, Cody-Jordan Handy-Hart, Sarah Kulle, and Kyla Flanagan. 2024. Undergraduate Students Develop Questioning, Creativity, and Collaboration Skills by Using the Question Formulation Technique. *The Canadian Journal for the Scholarship of Teaching and Learning* 15, 2 (2024).

[25] LSA University of Michigan. [n.d.]. Question Formulation Technique - University of Michigan. https://sites.lsa.umich.edu/inclusive-teaching/wp-content/uploads/sites/853/2021/09/Question-Formulation-Technique-Draft.pdf

[26] Jackie Acree Walsh and Beth Dankert Sattes. 2016. *Quality questioning: Research-based practice to engage every learner.* Corwin Press.

[27] Susan Wiedenbeck, Deborah Labelle, and Vennila NR Kain. 2004. Factors affecting course outcomes in introductory programming.. In *PPIG*. 11.

Enhancing HPC Education through Virtual Cluster Administration and Benchmarking

Charles Ross Lindsey
Georgia Institute of Technology
clindsey8@gatech.edu

Jeffrey Valdez
Georgia Institute of Technology
valdez@gatech.edu

Aaron Jezghani
Georgia Institute of Technology
ajezghani3@gatech.edu

Will Powell
Georgia Institute of Technology
ajezghani3@gatech.edu

Richard Vuduc
Georgia Institute of Technology
ajezghani3@gatech.edu

ABSTRACT

The rapid advancement in high-performance computing (HPC) poses significant challenges for the HPC community. Current HPC training approaches often are too generic or too customized to local environments, limiting their applicability and impact. Often, these shortcomings are due to the limited accessibility, excessive cost, and specialized support necessary to provide HPC environments for teaching. To address these challenges, we introduce Virtual Cluster, a hardware-agnostic platform designed to provide an easy-to-configure, generalizable, and scalable approach to HPC system management for training and education in computational research alongside production system configurations. We implemented this platform in a virtually integrated project (VIP) course aimed at training undergraduates for HPC cluster building. Drawing from our experience from the VIP course, we advocate for the integration of more comprehensive educational and training approaches, such as HPC Virtual Cluster, to better support HPC.

KEYWORDS

HPC, virtual cluster, training, virtually integrated project

1 INTRODUCTION

Recent years have seen an acceleration in the development of novel hardware, software, and workflows in the field of High Performance Computing (HPC), and more generally for the broad spectrum of efforts under the blanket of Research Computing and Data (RCD). Fueled especially by the generative artificial intelligence (or generative AI) boom, vendors have halved their hardware release cycles [9], with each subsequent generation costing increasingly more [14] and requiring more electricity and heat dissipation to operate [7, 13]. Consequently, computing clusters across the spectrum of academia, industry, and government all face the mounting challenge of ever-increasing capital and operational costs, and thus necessitate capable researchers to efficiently utilize resources and administrators to optimally configure and support these systems.

https://doi.org/10.22369/issn.2153-4136/16/2/4

Naturally, computer science and engineering students have the strongest theoretical background to understand the nuances of HPC clusters, with core curriculum focused on systems architectures, networking, and parallel and distributed computing frameworks. The natural sciences have built upon a foundation of experimentation to find empirical validation for demonstrating theory as well utilizing HPC resources. For example, students can easily observe the principals of evolution in practice through nearly half a century of *E. Coli* cultures [10], or find validation for Einstein's Theory of General Relativity with the results from the LIGO collaboration [2]. However, current HPC training approaches at colleges and universities, particularly in undergraduate education, are lacking, either too generic, tailored to local environments and limiting their generalizability, or limited to graduate education. Part of the reasons for this have to do with cost for access to HPC environments, with, by necessity, require funding through research and provides little opportunity for training undergraduate students for future careers in research, ultimately hurting progress in the HPC research community. Although consumer-grade hardware is relatively affordable and thus accessible in a classroom setting, the specialized hardware in HPC clusters is much more costly, with entry-level costs for systems alone hitting $10k in 2009 [1], and support personnel and infrastructure tacking on considerably more.

These recent challenges leave HPC in a precarious state, especially with the onset of rapid architectural overturn in support of AI. End users looking to utilize HPC cluster resources to pursue research careers in science or engineering suffer because they lack expertise necessary and robustness in experience to navigate transitions into and throughout careers. HPC practitioners that stay on an academic path require time to adapt to the next system, which is problematic in the face of short evaluation cycles such as postdoctoral positions and pre-tenure evaluations. For example, a recent assessment of user activity on Georgia Tech research clusters found that since 2015, the average period of activity from first to last job submission is only 1.38 years (see Figure 1).

HPC practitioners that turn to industry may initially fare better due to robust onboarding and training into company workflows and teams, but, lacking foundational knowledge in HPC in their undergraduate education, may find themselves vulnerable to downsizing that has become more common with tech companies in recent years [12]. More urgent is the growing need for a workforce ready for the challenges of configuring, tuning, and maintaining HPC systems

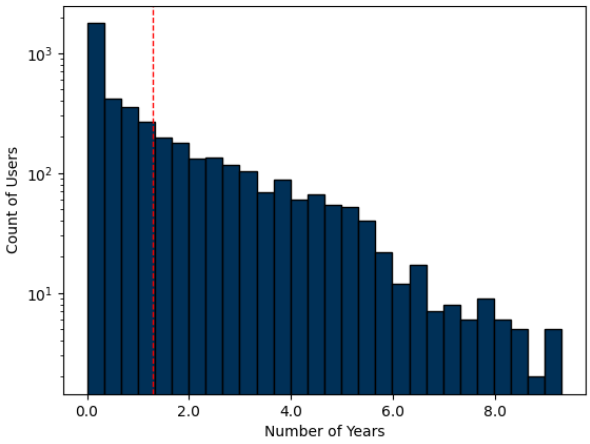

Figure 1: The average active period for users on Georgia Tech clusters was recently determined to be 1.38 years. Active period is defined as the time between first and last job submission, with the data spanning 2015 through present. The short period to engage researchers highlights the importance of effective training and education programs.

as part of their companies' Informational Technology (IT) infrastructure, which can affect whole organizations and their ability to execute computational research and development.

Consequently, this all points to a need for more comprehensive educational and training approaches in support of HPC. To address these challenges, we introduce **HPC Virtual Cluster**, a hardware-agnostic platform designed to provide an easy-to-configure, generalizable, and scalable approach to HPC system management for training and education in computational research alongside production system configurations.

2 VERTICALLY INTEGRATED PROJECTS

Since the late 1990s, the Vertically Integrated Projects (VIP) Program enhances core curriculum through research projects [8]. Ideally, VIP teams should be comprised of 10 or more students from multiple campus departments, and they should participate for three or more semesters to provide a spectrum of academic ranks and experience. Assessment is designed to facilitate these aspects of team composition, with a third of the students' final grades coming from documentation, teamwork, and accomplishments/contributions. Undergraduate students find the VIP program attractive because some colleges such as the College of Computing offer VIP in lieu of the traditional junior design course, while masters students can use it to satisfy elective credits. Currently, Georgia Tech hosts 97 VIP teams exploring topics such as technology-enhanced arts, AI-driven engineering, and Smart infrastructure based on expansive telemetry and automated intelligence.

2.1 Team Phoenix VIP

Originally, Team Phoenix was a participating team in the 2017 Supercomputing Student Cluster Competition (SCC). In Spring 2020, Team Phoenix was reborn as a VIP course under advisement by academic and research faculty from the College of Computing and the Partnership for an Advanced Computing Environment (PACE). Each semester, students progress through a series mini projects to further their knowledge of parallel and distributed computing on an HPC cluster, including resource management, building and configuring drivers and software, and benchmarking applications using various architectures and techniques. Students in the fall have submitted applications for the International Supercomputing High Performance Conference (ISC) SCC, while students in the spring have applied to the Supercomputing SCC and IndySCC competitions.

In addition to the assigned projects, students are exposed to the cluster design, procurement, and support processes. Vendors are brought in to discuss emerging hardware and platforms, and tours of the campus data centers are conducted to familiarize them with enterprise operations. A particular favorite among students is the "Build-a-Bear server dissection," where a production server is brought in and disassembled to foster an understanding of the interplay between hardware and software, as seen in the screen capture in Figure 2.

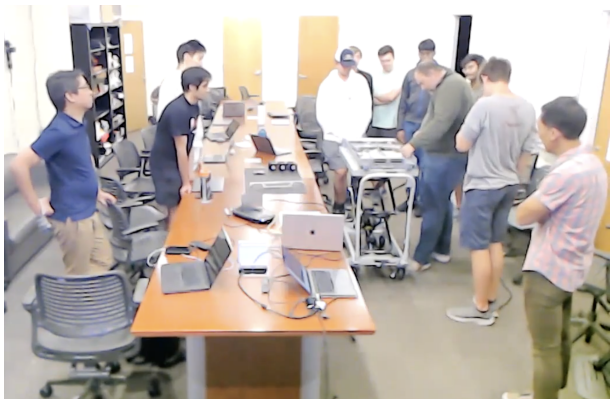

Figure 2: Screen capture of the hybrid class recording showing a class-favorite activity, "Build-a-Bear" Server Breakdown/Reassembly. Using a surplus V100 GPU server, students actively disassemble the server, review all of the components, and then reassemble the system.

2.2 The Missing Middle

Historically, the conversation of a "missing middle" in HPC largely centered on businesses or scientists who could have experienced a boon if given access to a cluster network [4]. The barrier to entry was insurmountable, unfortunately, for some potential users who *i*) could not acquire a supercomputer or purchase access, *ii*) lacked computing knowledge, and/or *iii*) failed to program a cluster to tackle their problem set [4]. In education, undergraduate students not only experience similar roadblocks but are faced with a lack

of curriculum necessary to bridge the gap between student and researcher.

Additionally, Team Phoenix was designed to address the "missing middle" [5] in education by filling in the gap in curriculum between foundational undergraduate courses in programming, architectures, and networking and specialized graduate topics such as AI methodologies, HPC systems, and advanced architectures. As a team-based, project-focused course, students collaboratively work through the process of building, configuring, and running applications for a stronger foundation in topics like parallel and distributed computing, communications frameworks, and software engineering. The multi-semester approach and promotion to leadership roles within the team provides further reinforcement, and the subsequent discourse facilitates a richer understanding of the underlying computer science principles.

Students enrolled in Team Phoenix are provided access to multiple cluster systems, one of which is detailed in this report. Once given access to a cluster, undergraduates are imparted with knowledge pertaining to Linux command-line fundamentals for system administration, use of build systems and compilers, job scheduling, file systems configuration, and other topics in HPC. Lastly, our cohort is provided hands-on instruction with scientific and benchmarking applications. Having met the wants of the historical "missing middle," Team Phoenix provides a workforce-development opportunity through participation in cluster competitions which test undergraduates on cluster building/management and utilization.

2.3 Current State of Team Phoenix

Since Team Phoenix's inaugural year, enrollment has increased 144%, from a total of 9 students to 22 in the present term, as presented in Figure 3. Furthermore, Figure 4 shows that retention rate among students, remains high with roughly 64% completing at least two semesters (one academic year). By building and retaining undergraduates, Team Phoenix has begun to fulfill its mission of building the "missing middle" by advancing course alumni to industry positions in Wall Street, Sandia, and NVIDIA. Additionally, students who have completed our VIP course have entered undergraduate or graduate HPC research at Georgia Tech.

In reviewing the demographics of students as reported in Table 1, a few trends quickly become apparent. Despite the multidisciplinary focus of the VIP program, and particularly the ubiquitous nature of computational research across all fields of science, students enrolling in the VIP are almost exclusively CS or CSE majors. Furthermore, females comprise roughly 10% of the total enrolled students, compared to the average of 20% in CS and Engineering degree programs [3], while minority participation sits well below averages, with Black/African American and Hispanic enrollment amounting to 3% and 6% compared to 8% and 12%, respectively [11]. In particular, it would be beneficial to develop and implement strategies to more effectively recruit and promote diversity within the Team Phoenix VIP.

3 INSTRUCTIONAL CLUSTERS

There are multiple HPC clusters that are available at Georgia Tech which the students have access to as students in the VIP Team Phoenix course. All clusters listed here use similar job schedulers (Slurm) in a Linux environment, and can be accessed using a shell environment or Open On-Demand. Table 2 summarizes the available node counts and hardware on each of the clusters.

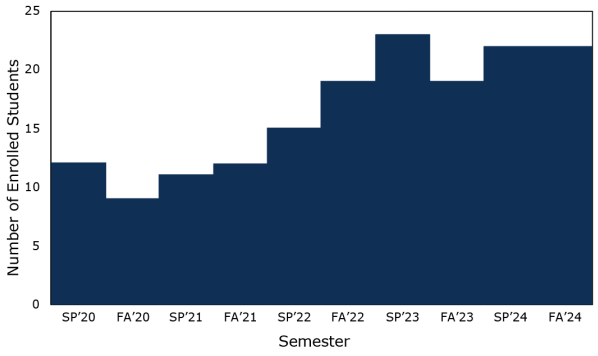

Figure 3: Team Phoenix VIP course enrollment by semester. Since its creation in the Spring 2020 semester, enrollment has increased 144% to 22 students in the current semester.

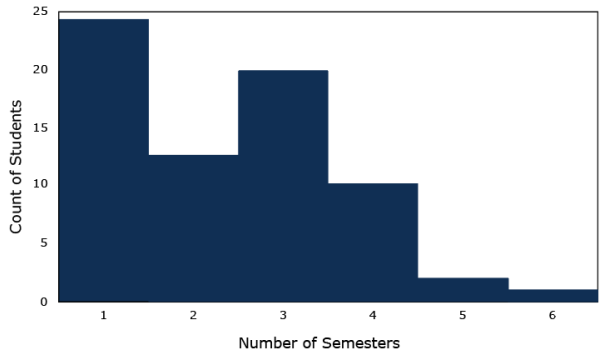

Figure 4: The count of students by numebr of semesters enrolled in the Team Phoenix VIP course. The vertically-integrated aspect of the VIP course hinges on students returning for successive semesters, and to date, Team Phoenix has seen roughly 64% come back for at least one additional semester.

3.1 PACE-Supported Clusters

The two clusters used by the VIP Team Phoenix course students include the Instructional Cluster Environment (ICE) and Phoenix clusters. The ICE cluster resources offer an educational environment that matches that of our research clusters and provides thousands of graduate and undergraduate students at Georgia Tech opportunities to gain first-hand experience with HPC. The Phoenix cluster is our research cluster.

For the VIP course, students typically use the ICE and Phoenix clusters to install and run examples in parallel computing, including running the LINPACK benchmark and parallel computing examples. However, using these clusters does require continual maintenance by the PACE team through Maintenance Periods and other updates,

Table 1: Breakdown of enrolled student demographics in Team Phoenix VIP.

Category	Number of Students			
Total Student Count	69			
Student-reported Gender	M: 62	F: 7		
Student-reported Race	Asian: 38 Two or more: 5	White: 31 Unknown: 2	Hispanic: 4	Black or AA: 2
Student Major	Comp Sci: 52 Elec Eng: 1	Comp Eng: 6 Comp Media: 2	Math: 4 Bio: 1	Mech Eng: 1 Elec Comp Eng: 1

and requires lectures and workshops throughout the semester for proper cluster usage.

3.2 CRNCH-Supported Cluster

The Rogues Gallery is a heterogeneous test bed of HPC servers maintained by the Center for Research into Novel Computing Hierarchies (CRNCH) for research in novel computing environments. For VIP Team Phoenix uses the Rogues Gallery for access to the Frozone cluster, a set of nodes which include two A100 GPUs. During the "Build-A-Bear" workshop, students actively disassemble and reassemble a GPU server, and afterword test HPC and GPU benchmarks on the cluster that the server is part of for the VIP course.

3.3 VIP Cluster

The Pho cluster is a test bed of HPC servers purchased using Georgia Tech's technology fee grant for instructional courses to provide VIP Team Phoenix course students hands-on experience with setting up and cutomizing their own cluster environment. This system is a 9-node system that includes two GPU nodes, each with an A100 GPU; two high-memory nodes with 1.5 TB 2933 MHz DDR4 RAM; two Intel Optane nodes with 4x128 GB Optane Persistent Memory 100 series modules; two high-storage nodes with 12x4TB hard drives; and one general-purpose node.

4 BUILD-A-CLUSTER ASSIGNMENT

4.1 KVM Cluster Virtualization on the Pho Cluster

For the Build-A-Cluster assignment, we used Kernel-based Virtual Machines (KVMs), an open-source virtualization included with Linux, for cluster virtualization on the Pho cluster. This use of KVMs allows for flexible deployment of cluster nodes for periodic refreshes (i.e., changes between users between academic years and semesters), quick configuration updates for new projects or assignments during the semester, or quick resets of cluster environments in case of issues.

Each of the 9 nodes in the Pho cluster included common hardware, with differences in hardware between nodes to add additional capabilities to the cluster. All nodes in the cluster include the following hardware: dual-socket Intel Cascade Lake Gold 6226R (2.9 GHz clock speed, 16C/32T), 192 GB 2933 MHz DDR4 RAM, Cisco VIC 1387 with 40 GB Ethernet, and 960 GB SSD, mirrored SW RAID. The two storage nodes include an additional 12x4 TB hard disk drive for additional storage. The two high-memory nodes include 1.5 TB 2933 MHz DDR4 RAM instead of 192 GB. The two Optane nodes include an additional 4x128 GB Optane PMem 100 series memory. The two GPU nodes include an additional NVIDIA A100 GPU for each node. Centos Stream 8 was the Linux distribution installed on all of the nodes.

The KVM images built for HPC instruction were installed on each of the nodes and configured to utilize CPU, memory, and storage resources in the hardware, with some differences depending on the node type. Each of the KVMs was installed with Centos Stream 8, and configured to use all of the CPU resources (32 cores across dual-socket CPUs) and 50 GB of disk space. The memory used across the nodes varied depending on the node type (84 GB to 1360 TB). For this initial pilot, the GPU nodes were not configured to use the NVIDIA A100 GPUs but will be used in future projects.

For accessing the KVMs when running on the network, Mac addresses assigned on KVM setup were recorded so that they could be reused and re-mapped to a set virtual machine hostname and IP address on the network upon subsequent re-installations.

4.2 Alternatives

Compared to other solutions, such as the configurable cloud environment for large-scale HPC research, Chameleon Cloud [6], our KVM cluster virtualization has multiple advantages. First, our approach is on-demand and can be reconfigured as needed. Second, because KVMs are part of Linux, no special infrastructure is required to support it, and so any available hardware can be leveraged. A corollary to this detail is that as much as older hardware can be utilized, if newer hardware becomes available, it can also readily be supported. Thus, for Team Phoenix, or any other group looking to replicate this work, they can just as easily utilize surplus servers as well as the cutting edge systems supporting research.

Furthermore, the lightweight nature of the approach lends itself naturally to providing a secure sandbox in which students can build a cluster from scratch, without risk of irreversibly breaking the system. Ultimately, the intent is to run this setup atop a production cluster, which is discussed in greater detail in Section 6, to provide a sustainable solution that can accommodate the bursty nature of instructional cluster use while leveraging available cycles from critical infrastructure.

4.3 Build-A-Cluster Project in HPC Virtual Cluster

This Build-a-Cluster project that we assigned as the last project for the Team Phoenix VIP course for the Spring 2024 semester,

Table 2: A summary table of the clusters accessible to students in the Team Phoenix VIP.

Cluster	Node Count	CPU Cores	GPU Count
Phoenix (PACE)	1,407	36,104	V100 (120), RTX6000 (148), A100 (32), H100 (32), L40S (64)
ICE (PACE)	117	4,352	V100 (40), RTX6000 (8), A40 (4), A100 (8), H100 (160), L40S (32), MI210 (4)
Rogues Gallery (CRNCH)	61	2,772	A30 (8), A100 (17), H100 (1), H200 (6), MI210 (2), Max1100 (2)
Pho (COC)	9	288	A100 (2)

we sought to provide students with an HPC Virtual Cluster using KVMs on Linux servers in order to prepare for Supercomputing (or High-Performance Computing, HPC) student competitions. Our goal for this project was for students to get hands-on experience with:

- Working on hardware (or KVMs) on the HPC Virtual Cluster that would be similar to working on multi-node clusters used in the real world or student cluster competitions.
- Installing, configuring, and optimizing software typically used for essential infrastructure for an HPC cluster (Ansible, Slurm scheduler, compilers, MPI).
- Installing, configuring, and optimizing scientific and simulation software used for HPC research.

For this last Spring semester, we divided the class into two teams of six students to work on two separate HPC Virtual Clusters. For this initial pilot, an HPC Virtual Cluster was comprised of a 3-node cluster corresponding to 3 KVMs (i.e., pho-storage1-vm, pho-highmem1-vm, and pho-optane1-vm for Team 1). Each KVM was built from scratch, and storage mounted from an NFS partition on a corresponding bare-metal storage nodes. MAC address assigned from the initial creation of the KVM were noted for hostname and IP address assignment and for future re-installation of the KVM.

We divided this project into different sections, where we would start by 1. introducing the students to the HPC Virtual Clusters they would be working with, continue to software infrastructure setup and job scheduler installation on the 2. login/head and 3. compute nodes, and then move to 4. installing software, which included installing GCC, Open MPI, Intel oneAPI MKL, and HPL. Other optional steps that we wanted to include but realized we did not have time for in Spring 2024 included 5. installing Lua and Lmod to load software configurations and 6. installing Ansible and developing Ansible playbooks to repeat the initial setup and software installation steps after completing all of the previous steps one time manually.

5 ASSIGNMENT OUTCOMES

For the Spring 2024, both teams encountered challenges due to time constraints near the end of the semester, so we were not able to proceed past the "Initial Setup for Login/Head Node" and "Initial Setup for Compute Nodes" of the original project plan. However, we were not able to get to the "Install Additional Software" step as we intended with the original time frame. While we were not able to get to the point of installing software like HPL for LINPACK benchmarking, we did have other projects with HPL and Mr. Bayes earlier in the Spring 2024 semester. In this upcoming Fall 2024, we are starting at the beginning of the semester with an updated version of the Spring 2024 project that should include the optional steps, including installing additional software, utilizing Lua and Lmod to reload software configurations with 'module,' and utilizing Ansible for quick rebuilds of the virtual cluster.

6 FUTURE WORK

In the future, we have additional ideas we plan to implement to improve this virtual cluster approach to HPC instruction. The first idea we plan to implement is using Slurm on the bare-metal machines to launch job batch files that deploy the virtual clusters on-demand. Using the host cluster Slurm's suspend & resume framework, which was originally intended for cloud node management, we can launch our minimal VM images or a full production machine as needed. In this way, we can operate parasitically atop the production cluster, providing a more sustainable solution to support Team Phoenix and other classes that may want to explore cluster building projects.

The second idea we plan to integrate is a section on file systems for HPC systems, and how selecting different approaches and optimizing file systems could improve HPC benchmark performance. We plan to add mini project explaining the current state-of-the-art for file systems in HPC clusters, with a workshop to illustrate some of the setup involved with our current file system in the virtual cluster. In the future, we could also look into what would be required to add newer parallel file system hardware to the Pho cluster that students could set up, configure, and test HPC benchmark performance.

The third idea would be to look at newer, more advanced hardware to utilize with this virtual cluster approach, perhaps even to optimize it for better deployment in instructional settings such as this. Currently, the Pho servers use Cisco remote system management that makes it difficult for us to reconfigure the bare-metal servers easily. With the aforementioned suspend & resume framework, we could look at using the novel hardware available in the Rogues Gallery, or even consider a larger scale project using ICE compute nodes.

7 CONCLUSION

A myriad of hurdles present a barrier for entry into HPC, a field crucial to research in academia and industry in areas that include natural sciences, engineering, and computing. These roadblocks include,

1. Introduction to HPC Virtual Cluster
2. Initial Setup for Login/Head Node
   (a) Mount network mountable storage system (already completed)
   (b) Set up user accounts
   (c) Set up Slurm (DB and controller on login/head node)
   (d) Turn off firewall
3. Initial Setup for Compute Nodes
   (a) Mount NFS storage system (already completed)
   (b) Set up user accounts
   (c) Set up Slurm clients (on login/head node and other 2 compute nodes)
   (d) Turn off firewall
4. Install Additional Software
   (a) Install GCC (compiler)
   (b) Install Open MPI (for MPI support)
   (c) Intel oneAPI MKL (for BLAS support)
   (d) Install HPL (for Linpack benchmark)
   (e) Install additional software used in student cluster competition (Optional)
5. (Optional) Install Lua and Lmod to use 'module' to load software configurations
6. (Optional) Install Ansible and develop Ansible playbooks to rebuild the cluster from scratch

Figure 5: HPC Virtual Cluster Project Sections

but are not limited to, *i*) access to HPC infrastructure, *ii*) maintenance support consisting of advanced and niche knowledge, and *iii*) costly hardware. To fill this knowledge and personnel gap within the HPC industry, higher education institutions should invest in the training and teaching of supercomputing concepts for their students. To overcome traditional and academic challenges prevalent in HPC, we believe the **HPC Virtual Cluster** provides a hardware-agnostic platform designed to provide an easy-to-configure, generalizable, and scalable approach to HPC system management for training and education.

REFERENCES

[1] 2009. Is HPC going to cost me a fortune? https://insidehpc.com/hpc-basic-training/is-hpc-going-to-cost-me-a-fortune

[2] B. P. Abbott and et al. 2016. Observation of Gravitational Waves from a Binary Black Hole Merger. *Phys. Rev. Lett.* 116 (Feb 2016), 061102. Issue 6. https://doi.org/10.1103/PhysRevLett.116.061102

[3] Sapna Cheryan, Allison Master, and Andrew Meltzoff. 2022. There Are Too Few Women in Computer Science and Engineering. https://www.scientificamerican.com/article/there-are-too-few-women-in-computer-science-and-engineering/

[4] Nicole Hemsoth. [n. d.]. Engaging the Missing Middle in HPC. https://www.hpcwire.com/2010/06/07/engaging_the_missing_middle_in_hpc/

[5] A. Jezghani, J. Young, W. Powell, R. Rahaman, and J. Coulter. 2023. Future Computing with the Rogues Gallery. In *2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE Computer Society, Los Alamitos, CA, USA, 262–269. https://doi.org/10.1109/IPDPSW59300.2023.00051

[6] Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S. Gunawi, Cody Hammock, Joe Mambretti, Alexander Barnes, François Halbach, Alex Rocha, and Joe Stubbs. 2020. Lessons Learned from the Chameleon Testbed. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*. USENIX Association.

[7] Tobias Mann. [n. d.]. A closer look at Nvidia's 120kW DGX GB200 NVL72 rack system. https://www.theregister.com/2024/03/21/nvidia_dgx_gb200_nvk72/

[8] Stephen Marshall, Edward Coyle, James V Krogmeier, Randal T Abler, Amos Johnson, and Brian E Gilchrist. 2014. The vertically integrated projects (VIP) program: leveraging faculty research interests to transform undergraduate STEM education. In *Transforming Institutions: 21st Century Undergraduate STEM Education Conference*.

[9] Nvidia. 2023. NVIDIA Investor Presentation October 2023. https://investor.nvidia.com/events-and-presentations/presentations/presentation-details/2023/NVIDIA-Investor-Presentation-October-2023/default.aspx

[10] Elizabeth Pennisi. 2013. The Man Who Bottled Evolution. *Science* 342, 6160 (2013), 790–793. https://doi.org/10.1126/science.342.6160.790

[11] Parth Sarin. 2023. Levers for Improving Diversity in Computer Science. https://cset.georgetown.edu/article/levers-for-improving-diversity-in-computer-science/

[12] Emily Sayegh. 2024. The Great Tech Reset: Unpacking the Layoff Surge of 2024. https://www.forbes.com/sites/emilsayegh/2024/08/19/the-great-tech-reset-unpacking-the-layoff-surge-of-2024/

[13] Anton Shilov. 2024. Intel's 1500W TDP for Falcon Shores AI processor confirmed — next-gen AI chip consumes more power than Nvidia's B200. https://www.tomshardware.com/pc-components/gpus/intels-1500w-tdp-for-falcon-shores-ai-processor-confirmed-consumes-more-power-than-nvidias-b200

[14] Anton Shilov. 2024. Nvidia's next-gen Blackwell AI Superchips could cost up to $70,000 — fully-equipped server racks reportedly range up to $3,000,000 or more. https://www.tomshardware.com/pc-components/gpus/nvidias-next-gen-blackwell-ai-gpus-to-cost-up-to-dollar70000-fully-equipped-servers-range-up-to-dollar3000000-report

Facilitating Academic Research with FPGA Support in a University Data Center

Jeevesh Choudhury
Arizona State University
jchoudh3@asu.edu

Thomas Jennewein
Arizona State University
tjennewe@asu.edu

Gil Speyer
Arizona State University
speyer@asu.edu

ABSTRACT

Field Programmable Gate Arrays (FPGAs) offer a practical solution that balances computational power with energy efficiency, which could address the growing demand for sustainable high-performance computing (HPC). Moreover, because they can be reconfigured and optimized for specific applications, FPGAs open up numerous possibilities for adaptive, high-performance workloads. However, the substantial expertise required to deploy FPGA designs has traditionally been daunting, requiring proficiency in Hardware Description Languages (HDL) such as SystemVerilog or VHDL. To address this accessibility barrier, the field has shifted toward high-level synthesis (HLS), which allows developers to program FPGAs using familiar languages like C++ and Python — mirroring the evolution seen in GPU programming.

In this paper, the resources available on the Sol HPC cluster at Arizona State University (ASU) [8] and the strategies employed to support and encourage researchers and instructors working with these nodes are examined. The practical challenges of using FPGAs, the integration of tools and libraries in the development workflow, and efforts to lower the expertise threshold required for effective use are explored. By sharing this experience, the aim is to contribute to the growing body of knowledge around accessible and sustainable FPGA development in HPC environments.

KEYWORDS

FPGA, HPC, Facilitation

1 INTRODUCTION

With the advent of the exascale era of high-performance computing, accelerators have become vastly more expansive and heterogeneous and often include various novel architectures besides GPUs, such as FPGAs, Vector Engines (VEs), Wafer-Scale Engines (WSEs), and Intelligence Processing Units (IPUs). Among such accelerators, FPGAs have often served as the platform to design, prototype, experiment, and deploy novel architectures that accelerate applications while leveraging a balance between computational speed and power efficiency. The versatility of FPGAs allow them to be used both as computational and network accelerators, as shown in Microsoft's Project Catapult [12], and they can be connected in varying configurations as shown in Lant, et al. [9] to exploit various compute advantages.

Due to the FPGA's low power and programmability features, implementation of edge computing designs has gained wide popularity. In the context of the data center, this use case would seem out of place. However, at a university, the development of projects on a reliable, well-supported datacenter platform can become an attractive alternative to local labs for both research teams and FPGA course instructors.

Nevertheless, deploying FPGAs in HPC clusters comes with its own set of challenges and impediments as illustrated in Chen et al. [4] and for scalability, FPGAs need to be connected to the data center network directly as shown in Weerasinghe et al. [13]. Another barrier to using FPGAs and similar heterogeneous accelerators is that their workflow and software stack is significantly dissimilar and non-standardized when compared to GPU or CPU workflows. Furthermore, depending on the vendor, these accelerators work with different software stacks as well, in spite of them implementing a similar workflow methodology.

Broadly speaking, development flow on an FPGA can be classified as:

- **Register-Transfer Level (RTL) Design Development**: This is the classical development flow on an FPGA where a researcher writes hardware description language (HDL) code using Verilog or VHDL which is synthesized onto the FPGA through an appropriate development suite such as Xilinx Vivado or Intel Quartus Prime. HDL Design requires greater expertise and is more relevant to VLSI engineers and microarchitecture researchers. FPGAs facilitate ASIC prototyping by serving as a platform to design, test, and optimize custom integrated circuits. This development workflow can possibly exploit cloud FPGAs with the important caveat that the final test on the FPGA of a compiled design bitstream may not be viable; an FPGA board often needs a complete power cycle to update the onboard design, and power cycle privileges may not be granted to every user.
- **C/C++ Application Development**: With advancing technologies, FPGA programming has been shifting to a paradigm similar to that of GPU programming. This development workflow splits the design into two code files: the kernel code that runs on the FPGA and the host code that runs on the host CPU and leverages the kernel code. High Level Synthesis (HLS) compilers convert C/C++ code into FPGA designs, thereby improving the approachability of FPGA programming. A user would only need a base understanding of C code to get started with this workflow, and the complete

https://doi.org/10.22369/issn.2153-4136/16/2/5

flow consists of a minimum of four steps to get a design running on the FPGA. These can be specified as:
(1) Compile the host code using any C/C++ compiler.
(2) Compile the kernel code into an HLS package.
(3) Link the HLS package into a binary file.
(4) Execute the compiled host code with the binary file.
This flow is easily deployable on HPC clusters, and applications made using this workflow do not require any heightened privileges for the user on the FPGA nodes.

- **Hybrid Development**: The aforementioned development flows can also be combined at various stages. Designs made in HDL code can be packaged into HLS modules and further linked to a design binary, thereby creating a combined workflow where traditional RTL projects can be deployed on HPC clusters with the C/C++ development flow without having to provide users increased administrator privileges on the node.

The tools required for these workflows vary, resulting in a suite of software tools, drivers, and resources available to the user for a comprehensive testbed environment. The development methodology is modular with multiple intermediate results in the design process, thereby implying that all stages need not be compiled, executed, and tested on the same resource, resulting in an inherent parallelism available to the workflow itself.

2 FPGA AS A SERVICE (FAAS)

With the increasing heterogeneity of HPC systems, FPGA resources have become popular as an extension to the pool of available resources offered by cloud services. Project Catapult by Microsoft and Amazon's AWS EC2 F2 instances are two examples of this growing trend. As mentioned earlier, the inherent heterogeneity of the FPGA node configuration requires a suite of software resources and modules to go from design to deployment. Creating and sustaining this development environment as a resource for a multitude of online users has its own challenges, but an added incentive to deploying such an environment on the cloud is that these resources are often self-contained suites that offer a wider breadth of features. Furthermore, they are designed in a modular fashion, enabling system administrators to upgrade or downgrade these integrated development environments (IDEs) as required. Therefore, the primary classification that inhibits cross-compatibility between IDEs generally boils down to the choice of software offerings. Intel and AMD are the prominent vendors in this field, as of this writing, and hence, the choice is between them as their FPGA software products are functionally similar.

In terms of performance benchmarks, the FPGA as a Service (FaaS) model has been evaluated to show better performance for compute-intensive workloads compared to CPU and GPU implementations while also leveraging relatively less power consumption as evaluated in Perepelitsyn et al. [11].

2.1 Compute and Software Resources

ASU's Sol has FPGA nodes available through the SLURM scheduler. However, while the OpenCL workflow from design to deployment is straightforwardy available to all users, full re-programmability of the cards and driver access is restricted by administrator permissions to prevent users from accidentally bricking the hardware. Providing elevated permissions to FPGA researchers is one possible solution for this. Bare metal servers are the ideal method for deploying FPGA resources on the cloud, as they allow researchers to fully exploit the re-programmability of FPGA hardware, and the possibility of such special access nodes on Sol is being explored.

Currently, Sol has two FPGA nodes with the following FPGAs:

(1) Alveo U280 with an AMD UltraScale+ XCU280 FPGA and 8GB of HBM2 memory [1].
(2) Bittware 520N-MX with an Intel Stratix 10 FPGA and 16GB of HBM2 memory [3].

Since the two FPGAs are from separate vendors, the requisite software environment for development on either can be easily provisioned for each, with AMD Xilinx Vivado being one of the available development environments and Intel Quartus Prime as its complement.

2.1.1 *AMD Xilinx Resources*.

(1) *AMD Xilinx Vivado*: Vivado is a comprehensive suite for the traditional FPGA workflow using HDL, and it supports Verilog, SystemVerilog, and VHDL. On Sol, Vivado 2022.1 Standard Edition is available as a module which can be loaded on any node and used accordingly. Vivado is a comprehensive IDE for RTL design, and most of its components can be used from any node on Sol to work from RTL design to bitstream generation. All boards supported in the Standard Edition are available for development with the edition on Sol. Consequently, within the cloud working environment, Sol offers the capability of offloading the design and testing of experimental VLSI architectures in a datacenter environment to researchers, thereby reducing the need for local machines required to run the same software.
While, with some effort, RTL design does allow users to use the full reconfiguration of an FPGA, significant domain expertise is necessary to competently exploit the resources available on an FPGA. ASU's Sol allows researchers to design, develop, and generate their custom architecture bitstreams with Vivado on any node, but deploying the bitstream on the Alveo U280 is restricted due to the partial reconfigurability feature of the board. As such, any design would require a debug core to be interfaced into them so as to not brick the Alveo U280 once its onboard design memory is updated. The card also needs a power cycle to update itself once a bitstream has been deployed, and, therefore, as a security measure, full reconfigurability is not granted to all users.
Since Vivado is compatible with various other FPGA design platforms, it is always plausible for researchers to simply use the software on any node on Sol to develop a design for any other FPGA of their choice. Any design files can always be easily transferred between the cloud and another local environment.
(2) *AMD Xilinx Vitis*: Vitis is an IDE designed for working with HLS using C/C++. The Vitis workflow involves splitting the code into the CPU/Host component and the Kernel/FPGA component, akin to specifying the CPU and GPU kernels

while working with HIP or CUDA code. The host code is compiled using a g++ compiler while the kernel code is compiled using Vitis v++ which translates the C/C++ implementation into an FPGA binary. The v++ compiler maps the code to a specific platform, and hence a .xpfm or .xsa platform file must be included when running the compiler command. This platform file can be the provided default for a specific board, or it can be custom made through Vivado. Vitis has various sub-components as well which can be used for timing analysis, code debugging, platform generation, and more. Figure 1. shows how the CPU and FPGA codes interact with each other.

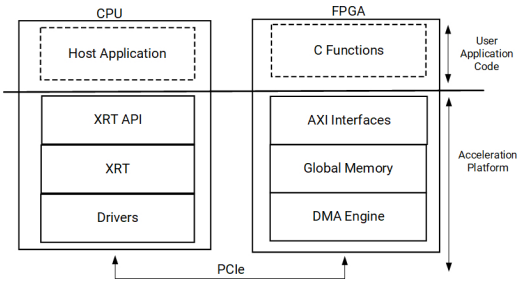

Figure 1: CPU-FPGA Code Interaction in Vitis [6]

(3) *Xilinx Runtime (XRT)*: XRT is a software stack composed of a mix of userspace and kernel driver components. It contains the collection of API keys that allows the host system to successfully communicate with PCIe based FPGA accelerators. The tools provided in this software stack not only help in synthesizing code to be deployed on the FPGA but also allow users to monitor and test the FPGA directly through a bash shell without having to go through Vivado or Vitis. Figure 2. shows the software stack structure of the XRT libraries.

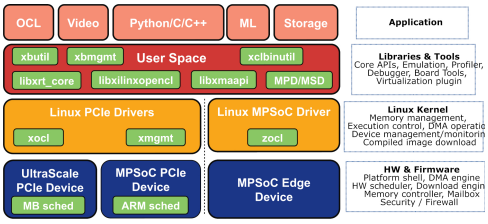

Figure 2: XRT Software Stack [14]

Figure 3. shows the software stack for XRT on Alveo based platforms such as the Alveo U280.

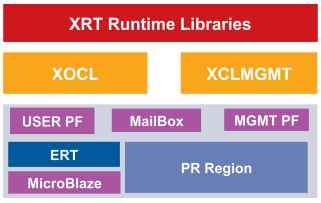

Figure 3: PCIe Stack for Alveo Platforms [14]

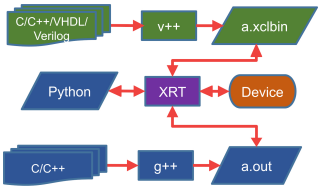

Figure 4: Compilation flow with XRT [14]

Figure 4. shows the compilation and execution workflow when working with XRT libraries to run kernels on the FPGA. It illustrates how the host code working on the host system CPU interacts with the compiled kernel binary deployed on the FPGA platform.

2.1.2 *Intel Altera Resources.*

(1) *Intel Quartus Prime*: Intel Quartus Prime is the comprehensive FPGA development suite available for Intel/Altera FPGAs. On Sol, there are multiple versions of Quartus Prime that are available, ranging between the Standard, Pro, and Lite edition. The recommended version is Intel Quartus Prime Pro 23.4 as that is the version that works ideally with the Bittware 520N-MX hardware that is available on Sol. Quartus Prime is similar to Vivado in that it boasts a comprehensive suite of tools, such as Quartus Programmer, Platform Designer, Timing Analyzer etc., which are designed to accommodate the many possible aspects of designing on Intel FPGAs. Unlike Vivado, which has separated off the Xilinx HLS components onto Vitis, Quartus Prime comprehensively includes the Altera HLS Compiler which acts as the working suite for compiling higher level code into its RTL equivalent for Intel FPGAs. The HLS Compiler is available in the Pro and Standard editions of Quartus Prime with the primary difference being the devices supported by those editions. Questa is also provided with Quartus Pro, which serves as an effective HDL simulator for behavioral verification of RTL modules.

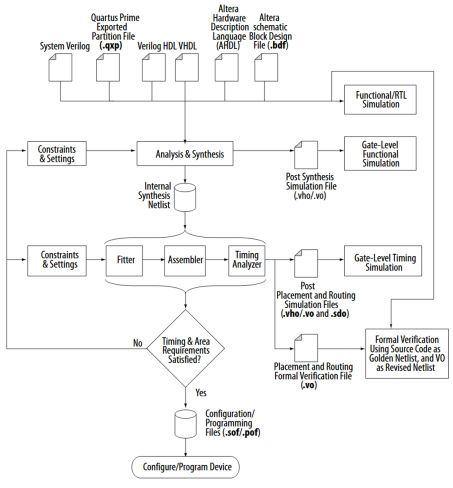

Figure 5: Intel FPGA Workflow [7]

Figure 5. shows a comprehensive workflow and the various file types involved in different stages of design and compilation for an Intel FPGA. While dissimilar in some aspects, the overall design philosophy and process is quite reminiscent of working with Xilinx FPGAs and thereby suggests that, while the tools may be different, RTL or C++ designs could possibly be ported from one vendor platform to another as long as the hardware resources supported the design.

(2) *Intel FPGA SDK for OpenCL*: OpenCL support for Intel FPGAs is provided through this package made available by default on the Bittware node on Sol. The Altera OpenCL (AOCL) utility can diagnose, program, and validate the FPGA, and it provides easy access without having to resort to working through Quartus. While AOCL has become a legacy tool as of writing this article, it is still readily compatible with many HLS design tools that allow extended programmability of the Intel FPGA. It is analogous to the XRT stack on Xilinx FPGAs and offers similar utilities for real-time hardware diagnostics.

(3) *Intel OneAPI*: Intel's OneAPI is a unified programming framework to write code for Intel FPGAs, CPUs, and GPUs. It is the current framework for programming Intel FPGAs through HLS built around C++ with SYCL. Multiple versions of OneAPI compilers and libraries are available on Sol with OneAPI 2023.2.1 being the most recent one. The OneAPI base toolkit is available as a module while its FPGA add-on is pre-installed on the Intel FPGA node, allowing for a seamless workflow with OneAPI tools.

Overall, both the AMD Xilinx and the Intel Altera resources provide tools for implementing the same design workflow and philosophy. Both vendors provide tools to facilitate RTL designs with Verilog and VHDL, and they also provide for the HLS design workflow which is much simpler and easier to learn with its similarities to GPU kernel programming.

2.2 Workflow

As mentioned above, modern FPGA programming is gradually shifting towards, and becoming unified with, GPU programming, often even using similar libraries, tools, and compilers. While the intricacies may be vendor specific, the overall design philosophy tends to remain the same, with the primary differences arising with the choice of coding language. In this section, we will attempt to codify the workflow based on HDL and C++ to create an abstract but comprehensive route from initial design to final deployment.

2.2.1 ***RTL Development Workflow.***

(1) *HDL Block Design*: The preliminary stage of any RTL design project begins with an abstract overview of the many functional modules to be implemented and how they should be interconnected and controlled. This stage involves a very high-level overview of the possible functional blocks, the state machines required for each block or groups of blocks, and the pipeline stages that will be incorporated into the design. These designs are then described in an appropriate HDL as per the discretion of the user. This design is then verified functionally and behaviorally with an associated testbench to ensure that the design is working as intended.

(2) *Synthesis*: Using an appropriate compiler, the HDL code prepared in the earlier stage is used to formulate a gate-level representation of the same code. This *gate level netlist* is a low level design using only primitive circuit components, representing the code with simple logic gates. It is then further optimized for timing, power, and performance according to the constraints and requirements as defined by the user. Functional and behavioral verification are usually executed using the earlier testbench once again to ensure that the translation from RTL code to the gate-level netlist did not create any behavioral anomalies.

(3) *Implementation*: Once the gate-level netlist is compiled and functional verification is done, the next step is to map these primitive components to the physical resources that are available on the FPGA. This step often uses a platform file or a board support package which contains the configuration and details of various blocks available on the FPGA for which the design is being implemented. *Place and Route* is the primary phase of this step where;
   - *Place:* Logic cells, required by the design, are placed at appropriate points on the FPGA board mapped by the compiler and,
   - *Route:* The connections for the placed logic cells are mapped to each other to form a complete circuit to implement the overall design.

   The placement and routing of the circuit is done in congruence with an appropriate platform and constraints files. Various verification methods pertaining to Design Rule Checks (DRC) are carried out in this phase as well to ensure that the designed circuit is in accordance with the physical resources available.

(4) *Timing Analysis*: Static Timing Analysis (STA) is the standard method for timing closure which analyzes the propagation delay through all relevant timing paths. In this context, slack is the margin by which the timing is met or violated. Setup slack indicates that the signal is not propagating too slowly, while hold slack signifies that the signal is not arriving too early. Positive slack implies that the circuit is meeting the timing constraint and that a faster clock can be achieved depending on the magnitude of the slack, while negative slack implies a timing violation. Achieving the highest operating frequency (lowest clock period) while maintaining timing closure is the ultimate goal of this step.
   Implementation and Timing Analysis are repeated iteratively until the timing violations in the circuit have been eliminated. Once all relevant timing paths have met timing and constraint requirements the design is ready to be exported to a bitstream.

(5) *Hardware Validation*: The implemented design is then exported to a device-specific binary configuration format which is used to program the physical hardware. Verification of the design is done by testing out the programmed FPGA in a real-time environment with an appropriate testbench. This step includes various design-specific tests to ensure that the hardware is performing well under various conditions and that there are no discrepancies between the simulated and implemented design and the final design on hardware.

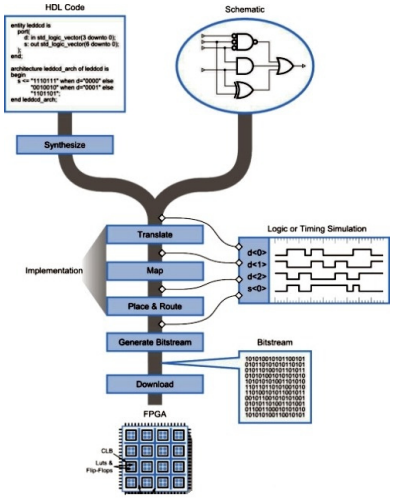

Figure 6: RTL Development Workflow [5]

Figure 6. shows an abstract and comprehensive overview of the aforementioned RTL workflow.

2.2.2 *C/C++ Application Workflow*.

(1) *C/C++ Algorithm Design*: First and foremost, the overall algorithm for implementing a design is formulated. This involves separating out the steps of the algorithm which can be computed on a host machine (usually with a multi-purpose CPU) and the steps that can be computed as an FPGA kernel. Depending upon the complexity and depth of the design to be implemented, the interaction between the host and FPGA kernel can range from simple function calls to more complex multi-stage data transfer and streaming operations. Once the overall algorithm is codified, functional verification is carried out with an appropriate testbench to ensure that the code is working as needed. This results in two primary components of the code:
- The host code running on the CPU and,
- The kernel code running on the FPGA

The kernel code is optimized further and then later recompiled to a full binary configuration which is deployed onto the FPGA in the proceeding steps.

(2) *HLS Optimization*: Once the algorithm is verified functionally, various HLS directives and pragmas are employed to optimize the kernel code for efficient hardware implementation. These optimizations include selecting appropriate data types (fixed-point vs. floating-point), loop pipelining and unrolling, array partitioning, dataflow optimizations, and interface specifications. The goal is to maximize throughput while minimizing resource utilization and meeting timing constraints.

(3) *RTL Generation and Co-Simulation*: After the functional verification and optimization of the kernel and host algorithm is done, the kernel code is compiled into an RTL module description. This is a synthesizable RTL representation of the algorithm which is useful in describing the datapath, control logic, and interfaces in a more hardware-centric manner. Co-simulation is then performed using the software testbenches to verify that the generated RTL produces identical functional behavior to the software reference, ensuring no behavioral discrepancies were introduced during the HLS compilation process.
Once it is verified that there are no functional anomalies, power, performance, and area analysis of the design can be conducted. HLS optimization and RTL generation and co-simulation can be carried out iteratively until the desired performance is achieved. Thus, these two steps act as a design exploration stage of the algorithm where performance bottlenecks, timing estimates, pipelining techniques, and memory partitioning and data flow can be tested and evaluated before the implementation of the full design.

(4) *Export and Testing*: Once the kernel module has been sufficiently tested and optimized, it can be compiled into a complete binary configuration for the FPGA. It is noteworthy that binaries from the HLS workflow are not interchangeable with those from the RTL workflow. HLS-generated binaries typically include additional metadata, such as kernel interfaces, scheduling, and pipelining information, providing more insight into the implemented design. In contrast, RTL workflows provide direct control over hardware implementation details. While HLS workflows offer higher-level abstraction and include runtime frameworks that facilitate simpler system integration, the resource efficiency of either approach depends primarily on the design quality and optimization effort rather than the source methodology itself.
Generation of the binary in this workflow takes a significant amount of time, typically above 3 hours. This is often because the HLS compilers carry out similar steps as in the RTL workflow, such as place and route, timing analysis, area reports, and resource optimizations, effectively going through the implementation and timing analysis steps repeatedly until an optimal design is achieved. The design summary and performance reports are also generated and exported during this step, which can be used for deeper analysis of the generated binary.
After the binary has been generated, it can be leveraged by the host code to be deployed onto the FPGA. Therefore, the host code can be a testbench to run simulations on the generated design to verify that functionality and performance have been preserved over the whole process, or it can be the primary algorithm interface to implement the overall design on the FPGA.

3 DISCUSSION

- **Cross-Compatibility of Workflows**: The workflows elaborated upon in Sections 2.2.1 and 2.2.2 are meant to be abstract overviews of the whole process. Each step is often self-contained and with various possible implementations. With the increasing diversity of FPGA tools and resources being made available to researchers, cross-compatibility between workflows is also becoming more straightforward. AMD Xilinx Vivado supports this cross-compatibility by

virtue of the IP Packaging Tool built into it, which can package an RTL design into a kernel module and which can be compiled into an application binary through Vitis HLS. While this approach can be employed to port older and simpler RTL designs to HLS binaries, the complexity of porting these designs mirrors the complexity of the RTL design itself.

- **Open-Source Resources**: Apart from the AMD and Intel tools mentioned in Sections 2.1.1 and 2.1.2, there are also many open source tools available to develop on FPGAs as well. Two open source tools deployed on Sol are OpenHLS [10] and ScaleHLS [16]. Both of these libraries are available as mamba environments on Sol. OpenHLS is a project that translates PyTorch neural network models to synthesizable RTL code and supports the Alveo U280 on the Sol cluster. The ScaleHLS project aims at compiling PyTorch code to its HLS C++ equivalent compatible with AMD Vitis.

- **Pre-Compiled Binaries**: To help researchers ease into the development workflow on Sol FPGAs, some pre-compiled projects have been made publicly available. These include:
  - Simple vector addition projects for the AMD and Intel nodes.
  - Custom and original implementations of the traveling salesman problem based on AMD's Vitis Tutorials [15].
  - An OpenCL implementation of a 2D FFT Accelerator for the Bittware 520N-MX based on Bittware's white paper [2].

  These projects contain the compiled kernel binaries, summary reports of the design, and host binaries to easily run the designs on the FPGA.
- **Run Time Benchmarking**: An important factor to consider when providing FPGA resources for the research community is program run time and card utilization. Table 1 presents some initial run time benchmarking and utilization for some basic programs.
  - The vector addition program simply adds two vectors of size 65536 to each other.
  - The prime number program calculates all prime numbers between 0 and 4095 and returns them in an output vector.
  - The triangular numbers program calculates the first triangular number to have over 500 divisors.
  - The second triangular numbers program was an attempt to solve the problem with a different method. It involved storing all divisors in an array and then iterating through the array when checking new numbers. This led to incredibly slow performance and the program did not finish in emulation or on hardware, even after eight hours.

	Kernel linking time (emu.)	Kernel Linking time (HW)	Comp. time (emu.)	Comp. time (HW)	Util.
Vector add. (65536 array)	13m 5s	1h 36m 49s	115.35s	9052μs	0%
Prime #s (4096 array)	13m 2s	1h 59m 0s	543s	5634μs	0%
Triang. #s	0h 41m 43s	4h 24m 11s	12849s	102524μs	~1.5%
Triang. #s (Array method)	0h 41m 48s	1h 22m 16s	DNF	DNF	N/A

Table 1: Some performance numbers on the Alveo U280 with the XRT Compilation Flow

4 RESULTS

To summarize, FPGAs have emerged as reconfigurable accelerators for specialized workloads in datacenters. They offer a unique balance between performance, flexibility, and energy efficiency tailored to specific applications. They perform well on compute-intensive tasks where parallelization and custom datapaths provide significant advantages. Despite these advantages, FPGAs have faced adoption challenges with long development cycles being a major hurdle. However, recent methods such as HLS and frameworks like XRT and OneAPI have been enabling wider deployment of FPGAs by allowing researchers to work with a wider arsenal of programming languages.

The codified workflows presented above offer a consistent platform to implement different projects as well as to reproduce results for similar projects. The workflows offer an accessible on-ramp for the onboarding of researchers onto FPGA projects on Sol and the versatility of the workflow imparts the capability for auto-tailorization of the flow with continued development.

The primary dichotomy between the workflows, as of this writing, arises not because of the overall development methodology with C/C++ and RTL, but rather due to the difference in vendors of the FPGAs. Open source tools are democratizing this barrier slowly but pragmatically, staying within one vendor's development environment does not seem to hamper the interchangeability of workflows significantly.

The datacenter FPGA landscape continues evolving with advances in memory integration (HBM2 and above), higher-speed interfaces (PCIe 5.0, CXL), multi-processor platforms (AMD Versal and Zynq MPSoCs), and improved development tools. As workloads become more specialized and energy efficiency becomes increasingly critical, FPGAs are positioned to play an expanding role in heterogeneous datacenter architectures, complementing CPUs and GPUs in optimized computing solutions.

ACKNOWLEDGEMENTS

The authors acknowledge Research Computing at Arizona State University for providing resources that have contributed to the results reported within this paper.

REFERENCES

[1] Advanced Micro Devices, Inc. 2024. Alveo U280 Data Center Accelerator Card User Guide. https://docs.amd.com/r/en-US/ug1314-alveo-u280-reconfig-accel.

[2] BittWare. 2021. *Accelerating 2D FFTs Using HBM2 and oneAPI on Stratix 10 MX*. Technical Report. BittWare. https://www.bittware.com/resources/hbm2-2d-fft-oneapi/ White Paper.

[3] BittWare, Inc. 2024. 520N-MX FPGA Accelerator Card. https://www.bittware.com/products/520n-mx/.

[4] Fei Chen, Yi Shan, Yu Zhang, Yu Wang, Hubertus Franke, Xiaotao Chang, and Kun Wang. 2014. Enabling FPGAs in the cloud. In *Proceedings of the 11th ACM Conference on Computing Frontiers (CF '14)*. Association for Computing Machinery, New York, NY, USA, Article 3, 10 pages. https://doi.org/10.1145/2597917.2597929

[5] Bruno Da Silva, An Braeken, and Abdellah Touhafi. 2018. FPGA-Based Architectures for Acoustic Beamforming with Microphone Arrays: Trends, Challenges and Research Opportunities. *Computers* 7, 3 (2018). https://doi.org/10.3390/computers7030041

[6] Héctor Gutiérrez Arance, Luca Fiorini, Alberto Valero Biot, Francisco Hervás Álvarez, Santiago Folgueras, Carlos Vico Villalba, Pelayo Leguina López, Arantza Oyanguren Campos, Valerii Kholoimov, Volodymyr Svintozelskyi, and Jiahui Zhuo. 2025. Porting MADGRAPH to FPGA Using High-Level Synthesis (HLS). *Particles* 8, 3 (2025). https://doi.org/10.3390/particles8030063

[7] Intel Corporation. 2024. *Intel® Quartus® Prime Pro Edition User Guide*. Intel Corporation. https://www.intel.com/programmable/technical-pdfs/qpp-ugs.pdf Document ID: 766292.

[8] Douglas Jennewein et al. 2023. The Sol Supercomputer at Arizona State University. In *Practice and Experience in Advanced Research Computing (PEARC '23)*. Association for Computing Machinery, New York, NY, USA, 6. (in press).

[9] Joshua Lant, Javier Navaridas, Mikel Luján, and John Goodacre. 2020. Toward FPGA-Based HPC: Advancing Interconnect Technologies. *IEEE Micro* 40, 1 (2020), 25–34. https://doi.org/10.1109/MM.2019.2950655

[10] Maksim Levental, Arham Khan, Ryan Chard, Kazutomo Yoshii, Kyle Chard, and Ian Foster. 2023. OpenHLS: High-Level Synthesis for Low-Latency Deep Neural Networks for Experimental Science. arXiv:cs.AR/2302.06751 https://arxiv.org/abs/2302.06751

[11] Artem Perepelitsyn and Vitaliy Kulanov. 2025. Methods of Deployment and Evaluation of FPGA as a Service Under Conditions of Changing Requirements and Environments. *Technologies* 13, 7 (2025). https://doi.org/10.3390/technologies13070266

[12] Andrew Putnam, Adrian M. Caulfield, Eric S. Chung, Derek Chiou, Kypros Constantinides, John Demme, Hadi Esmaeilzadeh, Jeremy Fowers, Gopi Prashanth Gopal, Jan Gray, Michael Haselman, Scott Hauck, Stephen Heil, Amir Hormati, Joo-Young Kim, Sitaram Lanka, James Larus, Eric Peterson, Simon Pope, Aaron Smith, Jason Thong, Phillip Yi Xiao, and Doug Burger. 2014. A reconfigurable fabric for accelerating large-scale datacenter services. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*. 13–24. https://doi.org/10.1109/ISCA.2014.6853195

[13] Jagath Weerasinghe, Francois Abel, Christoph Hagleitner, and Andreas Herkersdorf. 2015. Enabling FPGAs in Hyperscale Data Centers. In *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*. 1078–1086. https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.199

[14] Xilinx. 2021. Xilinx Runtime (XRT) Documentation. https://xilinx.github.io/XRT/2021.1/html/index.html

[15] Xilinx. 2022. Vitis-Tutorials. https://github.com/Xilinx/Vitis-Tutorials

[16] Hanchen Ye, Cong Hao, Jianyi Cheng, Hyunmin Jeong, Jack Huang, Stephen Neuendorffer, and Deming Chen. 2022. Scalehls: A new scalable high-level synthesis framework on multi-level intermediate representation. In *2022 IEEE international symposium on high-performance computer architecture (HPCA)*. IEEE, 741–755.

A Novel 3D Recurrent R-CNN for Medical Imaging Feature Detection: A Case Study for Coronary Calcium Detection

Vikas Sarvasya
University of Pennsylvania
sarva@seas.upenn.edu

Robert Gotwals
North Carolina School of Science and Math
gotwals@ncssm.edu

Liam Butler
Wake Forest University School of Medicine
liabutle@wakehealth.edu

ABSTRACT

Deep neural networks (DNNs), when trained on high-quality, noise-free images, often underperform when applied to images with distortions or noise [22]. A key challenge lies in the architecture of convolutional neural networks (CNNs), where the layers are structured to prioritize features detected at the final layers, treating earlier features as latent or redundant [5]. This design hinders the ability of CNNs to effectively detect small, dynamic, and complex structures in images, thereby limiting their adaptability to image variations [5].

The primary objective of this study was to develop a novel network architecture capable of accurately detecting the bounding boxes of coronary arteries and subsequently calculating a calcium score. The model's bounding boxes were validated against manually annotated arterial outlines, and the calcium score was compared to clinician-adjudicated values. The proposed network introduced an innovative propagation mechanism, coupled with various derived algorithms and modifiers, to mitigate the impact of motion distortions on coronary artery tracking and detection.

Results demonstrated exceptional performance, with the testing set achieving an average Mean Squared Error (MSE) of less than 2%, and a deviation of less than 1.5 pixels for each coordinate within a 128x128-pixel image. The calcium score derived from the network's bounding boxes exhibited a strong correlation of $R^2 = 0.921$, with a region of interest (ROI) accuracy of 88% across all calcium score ranges. In contrast, a standard CNN used as a control struggled, yielding an $R^2 = 0.0091$ and an ROI accuracy of just 10%.

This student research project presents a pioneering network that utilizes specialized algorithms and propagation techniques to accurately identify small, dynamic structures in non-gated chest CT scans. The model's ability to provide reliable calcium scores enhances the clinical utility of chest CT scans, offering a promising tool for improving the diagnosis of coronary artery disease and optimizing the management of cardiac disease risk [19].

KEYWORDS

Deep neural networks, Convolutional neural networks, Bounding box regression, Backpropagation algorithm, CT scan, Coronary Artery Calcium score, Artery detection, Distortion

https://doi.org/10.22369/issn.2153-4136/16/2/6

1 INTRODUCTION

Coronary calcium is calcium that builds up in coronary arteries. A substantial buildup of calcium deposits (calcium plaques) is called atherosclerosis and can narrow arteries and can be brought about by high blood pressure, obesity, diabetes, and high blood cholesterol [19]. A Coronary Artery Calcium (CAC) score of 0 shows that the individual has no coronary calcium buildup while an increasing calcium score (with virtually no limit) increases the risk of cardiovascular disease (CVD) [6]. Atherosclerosis is "the underlying cause of about 50% of all deaths in westernized society due to heart attacks, stroke, and peripheral arterial disease" [19]. CAC scores help clinicians assess risk of, amongst other cardiac diseases, heart attacks and stroke [19]. When CAC scores are moderate-high, clinical attention for disease prevention and/or treatment would be required. Cardiac gated computed tomography (CT) scans are conventionally used to calculate CAC. Gated CT scans may require specific drugs to slow down a patient's heart rate and imaging of the heart is done only during the mid to end diastolic phase, when cardiac movement is minimized [6]. This ensures that the gated CT contains no motion distortions and clearly highlights important features, such as coronary arteries [6]. This means that gated CT scans correct coronary motion and provide accurate values for the CAC score, since it is extremely easy to detect the coronary arteries [6].

Gated CTs are much more expensive than non-gated CTs [17] due to the use of drugs to modulate the heart rate. Consequently, cardiac gated CT scans are done very infrequently and only on patients who have already been diagnosed with some measure of CVD [7]. Roughly a quarter of patients who have heart attacks suffer from sudden cardiac death [16]. Of these patients, 80% have atherosclerosis, making it the most reliable indicator of cardiac arrest [26]. Therefore, it is imperative that better predictive tools are available in a larger group of patients to detect coronary atherosclerosis. Unlike gated CTs, non-gated chest CTs are more readily available and are routinely performed for a much wider range of pathologies other than coronary artery disease [13]. For example, chest CTs are used for detecting pulmonary embolism in patients presenting to the emergency room for trouble breathing.

If these non-gated CTs could be accurately evaluated for coronary artery disease, there would be less undiagnosed CVD patients, who could then be prescribed risk-reducing treatments and decrease the rate of instantly fatal heart attacks [13]. Non-gated CTs do not use drugs to modulate the heart rate and simply take cross-sectional images sequentially. Because images are taken during different stages of the heartbeat, these images contain motion distortion of the heart. In non-gated CTs, the coronary arteries are still in the field of view of the image, but the distortion of the image precludes

accurately identifying the arteries, making calculations of a CAC difficult [14]. This reduces the overall usability of these types of CTs for such cases. When non-gated CTs are used, the CAC score for these imaging scans is manually assigned into three categories: mild, moderate, and severe, rather than providing a numerical value [9]. This creates some issues, mostly due to variation and bias between physicians [9]. This unstandardized system also doesn't account for important factors such as age, gender, or race, making it unfeasible to use for diagnosis and treatment [8]. A mild CAC score for an elderly woman can be severe for a man in his 20s. There is a need for automated accurate coronary artery detection and CAC quantification, which can be achieved by the new 3D CNN's [24]. This is helpful for risk prediction and identifying patients who are at high-risk of cardiac diseases and require risk-reducing treatments [14].

Artificial Intelligence (AI) has made substantial advancements in its application within the medical field due to an increase in data availability. Deep learning methods have gained prominence in the use of medical imaging methods, such as magnetic resonance imaging (MRI) and computed tomography (CT) to develop AI-based prediction and detection models of multiple cardiovascular diseases. However, there remain caveats in their accuracy and generalizability, mostly due to the quality of the images used as inputs.

The literature into neural networks and brain structure suggests that this can be improved by altering the network to allow stacking maps on top of each other [1, 15]. Through each layer of a CNN, the filters contained in its nodes are applied to images to produce feature maps. By creating multiple sections of CNNs on top of each other, each layer will produce a set of stacked feature maps that can be analyzed sequentially and whose differences can be used to identify small and specific features. By using this approach, the CNN has more data it can train on (such as maps applied with sharpened kernels and sequentially subtracted images) but can process and find more features and increase the detail of examination when detecting features in an image [10, 21]. The overarching aim of this project is to address common issues in image regression within CNNs, which are distortion and variance in quality, size, and shape [11].

Traditionally, 3D Convolutional Neural Networks (CNNs) have been designed to handle RGB images by splitting the channels and training three distinct networks to learn features from each channel independently. This approach allowed each network to focus on a different set of criteria. However, by extending this methodology to more complex structures—such as networks with multiple stacks—our model can process grayscale or single-channel images effectively. This strategy increases the model's versatility, allowing it to handle not only color images but also those from modalities like MRI scans, which typically produce grayscale images.

This student research project proposed a novel backpropagation algorithm which would allow for the training of the 3D matrices/filters that build convolutional layers. This algorithm would advance current scientific knowledge on how to scale up stacked networks and how to isolate specific changes throughout numerous different variables used, allowing for more specific detected features and kernel changes. This novel method is superior to the current 2D CNNs for medical image analysis because of their inability to adapt to different complex structures while 3D CNNs will be able to make accurate arterial detection possible despite motion distortion in non-gated CTs [5].

In this project, we seek to apply a novel algorithm to radiological scans to isolate image sections of increased coronary calcification and output a calcium score. More specifically this research addresses the following aims:

(1) Develop a novel backpropagation neural network algorithm to allow 3D image processing
(2) Mathematically explain how this recurrent convolutional neural network (RCNN) can be integrated with this algorithm
(3) Maintain a higher accuracy level for the 3D CNN compared to 2D CNNs

2 METHODS

The data for this experiment consisted of 76 non-gated chest CT scans containing over 8,000 images in total. 53 of the 76 scans contained coronary calcium, while 23 did not. The 76 scans were divided into 80% training dataset and 20% testing dataset for 60 training scans and 16 testing scans. Of the training dataset, there were 41 with coronary calcium and 19 without. Of the testing dataset, there were 12 with coronary calcium and 4 without.

2.1 Backpropagation Algorithm

2.1.1 Forward Pass and Loss Function. The forward pass in a CNN involves propagating data through the layers, where each layer consists of weights (filters) and biases that are adjusted during training. The output of each layer is a feature map that represents certain patterns or features in the input data. The loss function, typically quantified using cross-entropy or mean squared error, measures the deviation between the predicted output and the true label of the input data [3]. In this case, the loss function is represented by Equation 1, shown below.

$$L = \frac{1}{XYZ} \sum_{x,y,z=1}^{X,Y,Z} \left(v - P_{xyz}\right)^2 \tag{1}$$

where x, y, and z are the input variables (pixels or features), L is the loss value, v is the result of the forward pass, and P is the predicted value. The deviation from the true values is calculated by computing the partial derivatives of the loss function with respect to each of the input variables, shown in Equation 2.

$$\frac{\delta L}{\delta x}, \frac{\delta L}{\delta y}, \frac{\delta L}{\delta z} \tag{2}$$

These derivatives indicate the sensitivity of the loss function to changes in the input values, providing crucial information for adjusting filters and weights during backpropagation. Next, we have the convolution operation. The central operation in CNNs is the convolution, where an input matrix (image) is convolved with a filter to generate an output matrix (feature map). The convolution is performed by multiplying corresponding elements of the input matrix and filter, summing these products, and sliding the filter over the image to generate the output matrix O, calculated by Equation 3 [4].

$$O_{11} = \sum_{i=1}^{n} \sum_{j=1}^{n} X_{ij} F_{ij} \tag{3}$$

where n is the size of the convolution kernel, F is the filter ,and X is the input matrix/image. A similar convolution operation is repeated for the rest of the input matrix, generalizing the previous equation to make Equation 4.

$$O_{kl} = \sum_{i=1}^{n} \sum_{j=1}^{n} X_{i+k,j+l} F_{ij} \tag{4}$$

The convolution operation extracts local features from the input image, which are then passed through activation functions like ReLU to introduce non-linearity and allow the network to learn complex patterns.

2.1.2 Backward Pass. The backpropagation algorithm relies on the chain rule of calculus to compute gradients of the loss function with respect to the network parameters. Specifically, we need to calculate the derivative of the loss function with respect to the output of the convolution operation. This can be achieved by summing up the product of the partial derivatives of the loss with respect to the individual components of the output.

$$\frac{\delta L}{\delta O} = \sum \left(\frac{\delta L}{\delta v} \cdot \frac{\delta v}{\delta O} \right) \tag{5}$$

In Equation 5, v represents the loss function, and O is the result of the convolution applied to the image. These partial derivatives help in adjusting the weights and biases of the network during training, ensuring that the network learns from the error at each layer. With these variables, we derive the gradient with respect to the filter. Then, we compute the gradient of the loss function with respect to the filter. The derivative provides the information needed to adjust the filter weights. The gradient is calculated as the sum of the input values multiplied by the gradient with respect to the filter in Equation 6.

$$\frac{\delta O}{\delta F} = \sum X_{ij} \cdot \Delta f(x, y, z) \tag{6}$$

This step calculates how much the output of the convolution operation changes when the filter is modified, guiding the network to adjust the filter for better feature extraction in subsequent iterations.

The modification of the backpropagation algorithm to improve CNN performance is well-supported in the literature. Several studies have explored enhancing the convolution operation and backpropagation process to optimize feature extraction, especially in the context of medical imaging [6, 7]. A major challenge in medical image analysis is the presence of noise and distortions due to factors like patient movement and varying scan quality. One effective approach to overcoming these issues is with image augmentation techniques, which artificially expand the training dataset by applying transformations such as rotation, translation, and scaling to the input images [15]. Image augmentation helps prevent overfitting, improves generalization, and ensures that the network learns robust features even in the presence of distortions.

Moreover, research has shown that incorporating additional layers in CNNs, such as 3D convolutional networks, can significantly improve the model's ability to handle complex structures in medical images, like the detection of coronary artery calcium (CAC) in CT scans [9]. By stacking filters and applying them sequentially across multiple layers, the network can better capture hierarchical patterns in the data and identify smaller, more specific features. The proposed modification of the backpropagation algorithm and convolution operation is consistent with these findings, as it aims to enhance the network's ability to process noisy, distorted images with high sensitivity and specificity.

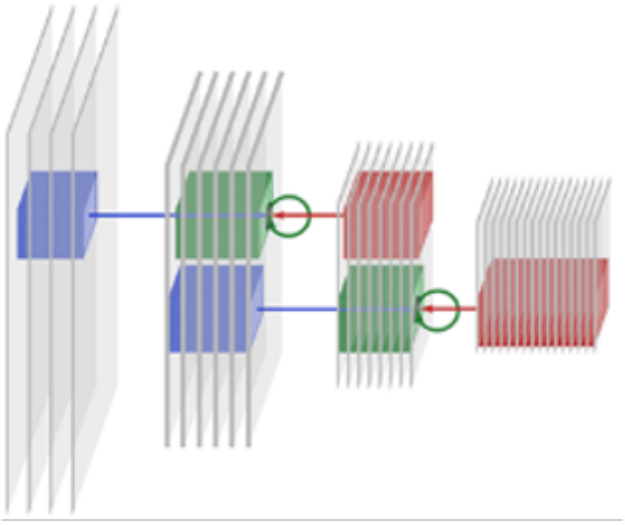

Figure 1: Types of Propagation. The blue cubes indicate forward propagation, which generates predictions. The red cubes indicate backpropagation, which is the method used by neural networks to learn and improve. The green cubes represent lateral propagation, or exchange of information between slices in the same layer, leading to augmentation and feature addition in consecutive image slices.

2.2 Network Structure

The second objective of this research centers around the development and functional integration of a novel Recurrent RCNN architecture. This new structure is designed to fully exploit all three spatial dimensions (x, y, z), which significantly enhances its ability to process and analyze complex 3D medical images. This three-dimensional approach is detailed in Figures 1 and 2, where each individual line represents a separate network processing one slice of the image set, and each "dot" in the broken line corresponds to a complete layer of the network, encompassing the entire x, y, and z dimensions.

Prior work in 3D convolutional neural networks (3D CNNs) has demonstrated the potential of extending traditional 2D CNNs to three dimensions for tasks such as medical image segmentation, object detection, and classification. Algorithms like 3D U-Net and 3D ResNet have significantly advanced volumetric data analysis [25].

These models utilize 3D convolutions to capture spatial relationships across adjacent image slices, which is essential for understanding structures in volumetric medical data. However, these earlier models typically focus on applying convolution layers independently across slices, without fully exploiting inter-slice information or the potential benefits of recurrent connections [3].

The proposed 3D Recurrent CNN structure, in contrast, integrates not only the spatial dimensions (x, y, z) but also introduces a novel recurrent mechanism to enable lateral propagation, a feature not commonly seen in traditional 3D CNNs. As shown in Figure 1, the red arrows represent the outward flow of data from the central image, while the blue arrows indicate the inward movement of data during backpropagation. This highlights the conventional forward and backward propagation steps within the training process, which are fundamental to most CNN architectures [1]. However, unlike standard models, the recurrent structure incorporated into our design allows for information to be shared laterally across neighboring nodes. This lateral propagation facilitates more complex interactions between adjacent layers, enabling the network to leverage contextual information from both the current slice and neighboring slices in the 3D volume.

The green cubes in Figure 1 illustrate the development of the recurrent state, where the core innovation of our approach is implemented. This recurrent mechanism allows the network to continuously refine its understanding of spatial relationships across the 3D volume, which is particularly valuable in medical imaging where subtle spatial dependencies between structures are crucial for accurate diagnosis. Previous work has explored recurrent networks in the context of medical image analysis, showing that recurrent connections can improve the network's ability to maintain long-term dependencies and capture contextual information across time-series data [25].

Beyond forward propagation, the new RCNN architecture employs lateral propagation between nodes during the prediction process, enabling a more robust exchange of information across layers. This allows the model to leverage techniques traditionally not possible in single-image or 2D CNN-based models, such as Intersection over Union (IoU) analysis, parity coloring, and feature overlap detection. For example, IoU, commonly used in object detection, can be more accurately applied in 3D volumes by taking advantage of the network's enhanced ability to assess spatial overlaps across slices. Similarly, parity coloring assigns higher values to pixels that are closer to corresponding pixels in the original image, further enhancing the model's sensitivity to fine-grained spatial patterns. Feature overlap, identified through Region Proposal Networks (RPNs), enables the network to focus on potential object areas, improving its detection performance in complex, highly dimensional datasets [20].

While previous algorithms for 3D CNNs have established a strong foundation for volumetric image analysis, the introduction of recurrent connections in our proposed RCNN structure offers a new approach to capturing and refining spatial dependencies across the entire 3D volume [1]. By combining forward and lateral propagation, as well as advanced image set techniques like IoU, parity coloring, and feature overlap, our model provides a more robust framework for handling the complexities of medical image data.

2.3 Modifiers

While the new backpropagation algorithm ensures that the network structure operates correctly while training on the stacked maps, the network training has not yet been optimized. One of the main issues with using 2D CNNs to detect these kinds of small, complex objects is that CNNs can only detect a singular feature or match a singular criterion for what it detects. Detecting a coronary artery in a chest CT scan requires the identification of several criteria to determine it as an artery, as shown in Figure 3 [24].

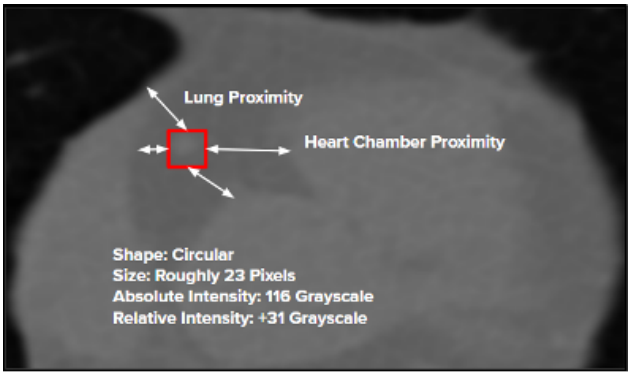

Figure 2: Arterial Conditions. Due to the complexity and variance of arteries, all the conditions labeled in the figure must be considered to detect an artery for feature detection.

2.3.1 Events with States & Plane Recurrence. Events with States is a mathematical framework designed to explore how sets of numbers and probabilities influence each other. In the context of this research, the technique treats each image layer as a separate set and works across five layers, represented as Images = (A, B, C, D, E), where each component corresponds to a distinct section of the network. As the algorithm progresses through these sets, a similarity coefficient is computed (Eq. 8). Here, A and B represent two sample images, and $i$and j are the pixel coordinates of these images. The calculation follows these key steps:

$$\text{Logarithmic Normalization: } \frac{1}{2}\left(\frac{A_{ij} + B_{ij}}{255}\right)$$

The term computes the average pixel intensity of corresponding pixels from both images, normalized by the maximum intensity value (255 for standard 8-bit images). Applying the natural logarithm (ln) ensures that brighter regions of the image receive higher precedence. This helps highlight important features, especially in medical imaging, where lighter structures (like bones or lesions) are often crucial for analysis.

$$\text{Absolute Pixel Difference: } \frac{A_{ij} - B_{ij}}{A_{ij} + B_{ij}}$$

The absolute difference measures the pixel-level variance between the two images, revealing how much the pixel intensities differ. This term is then divided by their sum to normalize the difference, so that variations are on a similar scale for all comparisons. Squaring this difference amplifies larger discrepancies more, ensuring that significant differences between images (such as the

presence of abnormal features) are emphasized. Once the pixel-level similarity has been calculated for each pair of images, the total similarity coefficient for the image set is obtained by summing the individual pixel-wise similarities and normalizing the total number of pixels. Where N is the total number of pixels across all layers (images), and is the similarity measure between corresponding pixels in images AA and B, the coefficient becomes:

$$\sum_{i,j=0}^{N} s(A, B, C, D, E) \tag{7}$$

This coefficient quantifies how similar the images are and is used as a feedback mechanism during training. During the forward pass of the network, the calculated similarity coefficients are aggregated for each image. This approach helps the network learn more complex interrelationships between different layers and variations of images. In addition to forward propagation, plane recurrence facilitates the exchange of information between layers during backpropagation, refining the model's understanding of spatial features across the 3D image. This recurrence ensures that the network doesn't treat each slice in isolation but instead considers the relationships between slices, promoting a more holistic understanding of the image data.

The determinant of the filter matrix is key to establishing connections between nodes in different layers. During backpropagation, the filter matrices are adjusted by calculating their determinants to ensure that high-importance features are transferred effectively between layers. The relationship between nodes can be mathematically modeled.

$$F_{l,adj.} = \frac{F}{||F||}\left(\sum_{k=1}^{5}(-1)^{l-k}\frac{||F_k||}{2^{l-k}}\right) \tag{8}$$

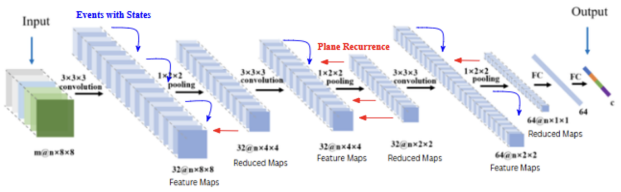

Figure 3: Network Architecture. The base layers are similar to an extended CNN architecture. The convolutional filters and feature maps are 3D, and there are modifiers applied to the network structure.

By adjusting filters based on the determinant, the network can more accurately identify and pass significant features across layers, enhancing the detection and segmentation of complex structures in medical images.

2.3.2 Calcium Score Calculation. The original images were in DICOM format, they were extracted and then converted into PNG to allow for better manipulation. This was done by setting the window level (center pixel value) and width (range of pixel intensities). Then, only the desired intensities on the DICOM image are saved. By narrowing the range, finer details are shown to give a more detailed image [17].

The Hounsfield Units (HU) of DICOM images are often used for CAC scoring. Since such values cannot be obtained from converted PNG images, a new calcium scoring algorithm must be developed based upon two values: threshold (t) and increment (i). The usual calcium score algorithm is as follows: each lesion of calcium detected is assigned a certain scaling factor, based on its maximum intensity. If the maximum intensity was between 130 HU and 200 HU, it was assigned a factor of 1 [12]. Between 200 HU and 300 HU was given a factor of 2, between 300 HU and 400 HU was given a factor of 3, and anything above 400 HU was assigned a factor of 4 [12]. For HU values, it has an equivalent threshold t = 130 and varying increment i = 70 or 100. To find the t and i values for PNG, we had to test out all possible combinations to find the optimal one.

2.3.3 Network Validation. The network was validated with 76 non-gated chest CTs, of which the first 60 were designated as the training dataset and the last 16 as the testing dataset. These images were non-contrast and originally 512 by 512 pixels. They were later shrunk down through max pooling to 128 by 128 pixels, which is the size they were input into the network. The images were all taken from the same source (Wake Forest Baptist Medical Center) and had standardized values for the kilovoltage peak (KVP) of 120. The field of view was also noted for each scan for accurate calcium scoring.

For validation of the bounding boxes, each image from each patient was taken and annotated for the location of the Left Anterior Descending (LAD) artery, the artery that was detected in this project. Four-pixel values were assigned per coordinate: the Left X, Right X, Top Y, and Bottom Y, with the location of the artery being determined by these four measurements and the tuple (0, 0, 0, 0) being given if the artery was not located. After validation with bounding boxes, the network used the calcium scoring algorithm on its generated bounding boxes to create a calcium score that was validated against a manually calculated one.

This project was undertaken using the Python programming language, and the files were held in two places: an Ubuntu 20 virtual machine stored on a local disk and a personal account to the Bridges2 Pittsburgh Supercomputer. The virtual machine was used for most of the debugging, data analysis, data curation, and initial writing of the code, while these files were transferred to Bridges2 to utilize its parallel processing capabilities. The library mpi4py and its submodule Message Passing Interface (MPI) were used to implement parallel processing with 80 processors on the Python network scripts, allowing the network to run at six seconds per image sample.

2.4 Standard Network

The standard CNN used as a control was built using the Keras/TensorFlow libraries. It used the optimizer Adam, trained on the training set for 20 epochs and had the following architecture:

```
Input Layer
Convolutional Layer (64 Filters of 5x5) with ReLU Activation
Max Pooling (2x2)
Convolutional Layer (16 Filters of 5x5) with ReLU
Max Pooling (2x2)
```

```
Convolutional Layer (4 Filters of 5x5) with ReLU
Max Pooling (2x2)
Flattening Layer
Dense (Fully Connected) Output Layer
```

3 RESULTS

3.1 Results of the Modified Network

3.1.1 Bounding Box Accuracy. To evaluate the spatial accuracy of the model, we assessed predicted bounding box coordinates using two primary metrics: the Mean Squared Error (MSE) and absolute coordinate-wise deviations from the ground truth.

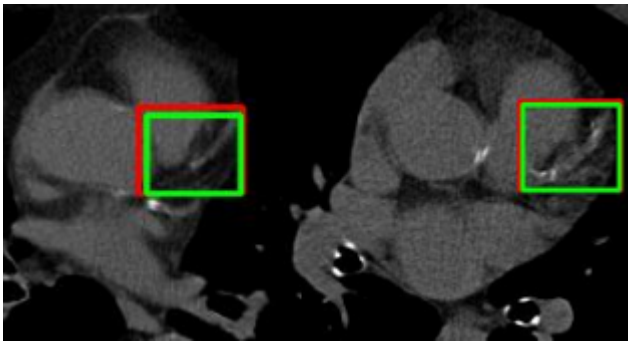

Figure 4: Annotated & Predicted Arterial Bounding Boxes in the Absence of Calcification. The green boxes are the annotated boxes, and the red are the network-generated predictions.

Coordinate deviations were defined as DLTX (Left X), DLTY (Top Y), DRBX (Right X), and DRBY (Bottom Y). As expected with neural network optimization, the accuracy of bounding box predictions improved progressively over the course of training. Final correlation coefficients reached 0.87 for DLTX, 0.80 for DLTY, 0.97 for DRBX, and 0.96 for DRBY, indicating high spatial alignment between predicted and annotated regions. Linear regression analyses further demonstrated the reliability of the model's predictions. The regression slopes were calculated as follows:

Left X: $y = 1.0083x$
Top Y: $y = 1.0136x$
Right X: $y = 0.9955x$
Bottom Y: $y = 0.9931x$

These results indicate that the predicted coordinates consistently deviate from the ground truth by less than 1.5%, with the largest deviation observed in the Top Y coordinate. The mean deviation across all four coordinates was within 1% of the ideal $y = x$ line, underscoring the model's high precision in spatial localization. On average, this is a two-pixel difference in each measurement. This is a variance of eight pixels in the perimeter and 74 pixels in the area, or an 11% decrease in the predicted area.

On a per-image basis, the average positional error between the predicted and true bounding boxes was approximately three pixels. Considering the input image resolution of 512×512 pixels and that the region of interest (i.e., the LAD artery) typically occupies a 100×100 pixel subregion, this error represents a highly accurate localization. Moreover, the model demonstrated robustness across variations in cardiac orientation, field of view, and anatomical differences, consistently producing bounding boxes within two pixels of ground truth in most cases. Visual examples of this performance are shown in Figure 4, where the model's predicted boxes closely overlap with manually annotated references, even under varying image conditions.

3.1.2 Calcium Scoring Accuracy. Beyond anatomical localization, the model's clinical relevance was assessed through automated calcium scoring, using predicted bounding boxes as the basis for score calculation. Based on the method described earlier, threshold and increment parameters were set to t=141 and i=25, respectively. Predicted calcium scores were compared to ground truth values obtained through a validated algorithm, with results summarized in Figures 7 and 8. The null hypothesis for the data is that they are correlated significantly, and not just because of random chance.

Clinically, calcium scores are stratified into risk categories: 0, 1–10, 11–100, 101–400, and >400. The model accurately placed over 90% of test cases into their correct risk category. Among misclassified cases, the average percent error remained below 20%, suggesting that even when a score was placed outside its true bin, it remained clinically proximate. Correlation analysis between predicted and true calcium scores yielded an R^2 value of 0.921 with a p-value of 0.86, indicating a strong and statistically robust relationship. Since the p-value threshold for significance is 0.05, we fail to reject the null hypothesis.

Together, these results highlight the model's high spatial precision in localizing coronary artery calcification and its ability to generate clinically meaningful calcium scores. The minimal pixel-level error in bounding box regression, combined with strong agreement in score stratification, supports the utility of this approach for automated cardiovascular risk assessment.

3.2 Comparison to Standard Network

3.2.1 Bounding Box Comparison. The standard network had an average bounding box deviance of 25 pixels per measurement, and 100 pixels in the perimeter. This led to an average difference of 1200 pixels in the area, an increase of 171%. While the modified network had a net decrease in area, indicating that the network was too precise with its detection, the standard network was unable to properly identify the complex structures surrounding the artery and often defaulted to simply encompassing all of the nearby structures, including valves, coronary veins, and the aorta. These structures can also contain calcium, resulting in diseases such as aortic stenosis, and muddle the measure of the arterial calcium.

Since the coronary arteries have a general area where they move throughout during a CT scan, the standard network was able to create bounding boxes in a reasonable region. However, it struggled to follow and detect longer segments of the artery. It would recognize the artery well at the beginning but was unable to track it throughout the slices and instead identified alternative structures that moved into the place where coronary arteries were before. For example, the Left Anterior Descending artery (LAD) splits in the lower slices, with the main segment moving to the dorsal side of the heart, while the circumflex (CX) branch takes its place. The standard network would continually identify the CX branch in the

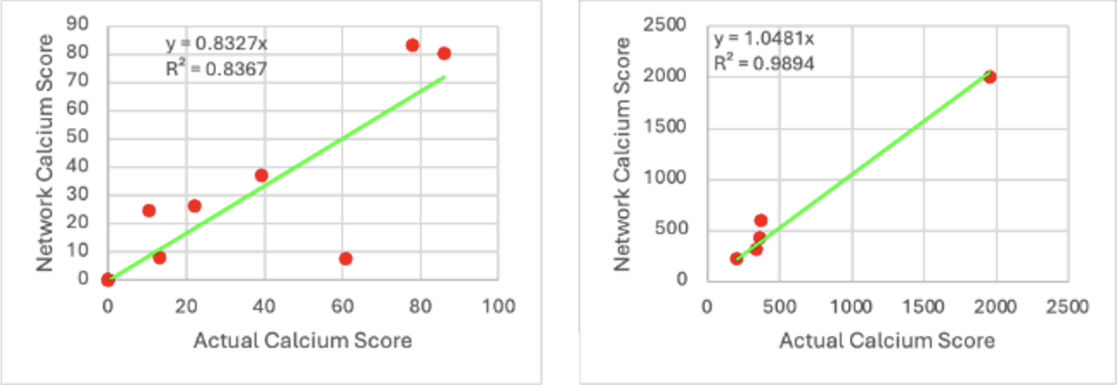

Figure 5: Actual v. Network Calcium Scores. The left graph shows calcium scores under 100, generally defined as mild or moderate. The right graph shows calcium scores above 100, defined as severe. The scores under 100 are 84% accurate, while those above 100 are 99% accurate. The scores are in the correct bin more than 90% of the time.

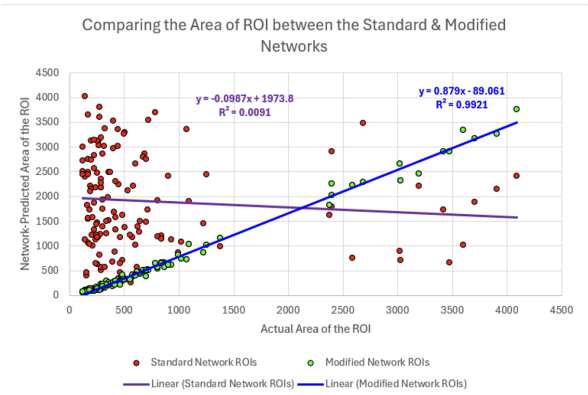

Figure 6: Analyzing the area of ROIs generated by the Standard and Modified Network. The Standard Network had the tendency to inflate bounding boxes, especially for small, precise arteries and therefore has a correlation close to 0, indicating that it was highly inaccurate. The Modified Network has a high correlation of 0.992 but does deflate the bounding boxes to 87.9% of their original size.

Scan	Actual Score	Predicted Modified Score	Predicted Standard Score
83	369	566.932	1898.117
84	361	425.294	610.465
85	0	0	23.204
86	338	318.859	480.779
87	0	0	4.397
88	86	80.124	142.551
89	1954	2004.065	2783.224
90	0	0	2.395
91	205	226.524	451.141
92	13	7.987	7.368
93	78	83.304	166.157
94	61	7.471	174.446
95	22.1	26.448	56.637
96	0	0	0
97	10.3	24.782	40.183
99	39.3	37.079	102.235

Figure 7: The arterial calcium scores of the scans, accompanied by the generated values from each network. Of the 14 scans in the testing set, the modified network fails to classify the score into the correct bin in 2 scans and the standard fails in 10 scans.

LAD section, and was unable to correctly identify the CX section during the CX detection process.

3.2.2 Calcium Scoring Comparison. Unlike the modified network, the standard network also fails in classifying scores into bins. The network frequently outputted a 100+ score for below 100 scores, a major issue since 100 is often used as the threshold for determining severity, especially in older patients. The network also fails for the lower scores: it is incredibly important to correctly determine 0 scores, since the presence of any arterial calcium is highly detrimental for younger patients and give false diagnoses for congenital heart disease or require unnecessary further imaging or procedures.

3.2.3 Reasoning for Improvements. The modified network demonstrated significantly improved performance in both artery detection and coronary calcium scoring compared to the standard architecture. These improvements were particularly evident in cases involving motion-distorted data, where the standard network consistently underperformed due to its inability to robustly account for anatomical variability and motion artifacts.

Specifically, the standard network exhibited a tendency to accurately localize arterial bounding boxes within the initial few slices; however, its performance degraded rapidly in subsequent slices. This inconsistency suggests a failure to model the temporal and spatial continuity of arterial structures across the imaging volume.

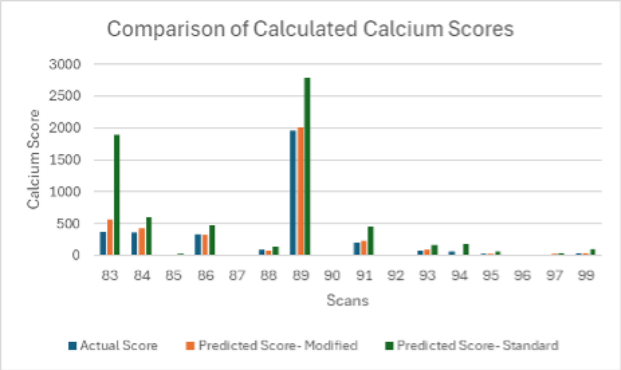

Figure 8: Comparison of Calculated Calcium Scores. The standard network had an average calcium error of 90.7%, compared to the modified network error of 24.7%. Correlation analysis between the standard score and actual score gave an R^2 value of 0.814 and a p-value of 0.408. Since the p-value threshold for significance is 0.05, we reject the null hypothesis.

In contrast, the modified network leveraged plane recurrence to maintain coherence across slices, enabling it to more reliably track arterial trajectories even in the presence of significant motion.

A key limitation of the standard convolutional neural network (CNN) approach lies in its reliance on structural uniformity. CNN filters are typically optimized to detect patterns that are consistent in size, shape, and intensity—an assumption that fails in the context of cardiac imaging where arterial geometries are often distorted due to cardiac motion. This inherent nonuniformity contributed to the standard network's inability to detect and delineate arteries accurately.

To address these challenges, the modified network incorporated the Events with States mechanism, which facilitates dynamic modeling of structural similarity across a five-slice window. This approach allows the network to infer and model patterns of motion-related distortion, thereby improving its ability to localize arteries despite significant inter-slice displacement. While the network does not generate a full parametric mask of the distortion, it effectively contextualizes the motion and compensates for it during inference.

Furthermore, the Events with States module enhances spatial awareness by prompting the network to evaluate the anatomical context surrounding candidate arterial regions. By incorporating constraints such as proximity to cardiac and pulmonary structures, relative intensity, and geometric orientation (as illustrated in Figure 3), the network applies a more rigorous spatial validation process. This not only increases detection precision but also reduces false positives in regions with overlapping intensities or ambiguous morphology.

4 DISCUSSION

This study attempted to identify coronary arteries using novel 3D networks and propagation algorithms. In addition to accurately identifying the coronary arteries, it automatically calculated a calcium score. Additionally, it correctly identified patients with calcium scores less than 100 and greater than 100. The results from this research demonstrate the potential of such 3D networks. These networks are capable of filtering through noisy and cluttered data and detecting objects, such as arteries, despite changes in orientation, position, and field of view, and can even account for distortion [20]. The innovative techniques described in this research as well as the novel structure that account for this accuracy are byproducts of the expansion to three dimensions modeling. We also show that the network can improve its results by using several images at once. This network is capable of accurately detecting bounding boxes of small, mobile, complex image components and can have substantial and important use in the medical field for quick and reliable analyses as part of routine care, helping in clinical decision making and health management.

While the bounding boxes around arteries (in each image) provide an extremely high level of accuracy, there can be improvement in the calcium score algorithm. Most of the observed limitations are due to the use of a new calcium score calculation algorithm since the network requires PNG images instead of full-resolution DICOM files. In a clinical setting, the network would run on the DICOM images taken straight from the scan, so the calcium score calculation could be more accurate due to its overall higher resolution. Consequently, this research can be further enhanced in a large-scale study with higher Graphical Processing Unit capacities, which can allow the network to continue learning using DICOM files. However, the accuracy achieved in this research when using PNG files has some merits. Predominantly, using the current model with PNG images allows for many more scans to be passed through the algorithm and generate calcium scores simultaneously. This is because the chest CT scans used are somewhat cheaper and much more commonly performed compared to cardiac CT scans, which are currently used for calcium score estimation. Not only does this research showcase a potential tool for clinicians to use while diagnosing a patient, but it also expands to a more general patient population that could receive a calcium score test, reducing potential underdiagnosis. Since chest CTs are taken on patients often reporting cardiac, pulmonary, or abdomen-related issues or symptoms [23], the calcium score could reveal an underlying disease or unnoticed atherosclerosis, helping clinicians find problems fast and more accurately, enabling them to provide more effective care [16].

The precision of the calcium scores maintaining the same numerical bin and the accuracy reducing the average percent error means that this network not only succeeds in detecting arteries within the image but can also identify hotspots of dense calcification [8]. These capabilities not only enable it to provide a useful calcium score but use many of the factors used in the calcium score calculation to identify parts of the artery that require the most focus and decide on which coronary bioimplants can be used for maximum effect. For example, two cases may have similar calcium scores, but different distributions of fatty plaque. A more spread-out plaque (possibly reducing its visibility) distribution along a long segment of the artery would most likely warrant the use of a stent to minimize the partial blockages and maintain the long-distance blood flow [18]. However, if the plaque was highly concentrated

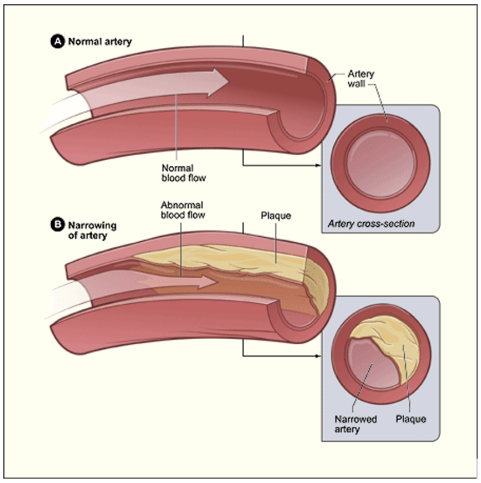

Figure 9: The Effect of Atherosclerotic Plaque on Arterial Blood Flow. Plaque buildup causes the artery to narrow, forcing blood flow to be constricted, increasing blood pressure. Even small deposits of plaque can cause disruptions in blood flow.

in a few completely blocked portions, bypass surgery would be much more effective [18]. The detection of the artery bounding boxes can be used beyond calcium scores to create 3D models of the artery through the chest and find narrow points or measure rate of blood flow, which can be used to prescribe a variety of life-saving medications such as Statins, blood thinners or beta blockers.

The implications of this research extend far beyond just CT scans and could be instrumental in revolutionizing computer vision applications in general practice. While the work focused on CT images, the approach can be generalized to a wide variety of imaging modalities, such as MRI, X-ray, and even ultrasound. In medical imaging, several common challenges, including distortion, obstruction, variance in shape, and size, can significantly degrade the quality of automated diagnoses. The algorithm developed in this research can help address these issues, particularly through transfer learning. By training on a diverse set of images, the model improves its robustness, making it more capable of handling variations inherent in medical imaging. A particularly promising extension of this research is the application to MRI scans, where images are taken from multiple angles (360 degrees) around the organ of interest.

In such cases, enabling the neural network to process images from any angle greatly expands the dataset and enhances the depth of learning during the training phase. This adaptability leads to more accurate models capable of interpreting complex anatomical structures from various perspectives, resulting in improved diagnostic reliability. In cases involving rare diseases or conditions with limited patient samples, a small dataset can hinder the network's ability to make informed decisions [22]. However, by increasing the dataset's variance through angle diversity and data augmentation, the model's robustness improves, allowing it to make more reliable and precise predictions [22]. This approach also facilitates the development of multi-modal models, capable of integrating different imaging techniques, such as combining CT with MRI or ultrasound, further enhancing diagnostic capabilities [24].

An essential feature of the proposed model is its ability to handle multiple images at once without slowing down the training process. The system can incorporate various perspectives simultaneously, making it more efficient while expanding its knowledge base. This multi-image processing approach allows different sections of the network to specialize in different aspects, such as varying angles or distinct image features, contributing to a more refined and accurate understanding of the data. The speed and scalability of training are maintained even as the dataset grows, which is particularly crucial in real-world applications.

This research also demonstrates the potential of combining multiple network structures to address challenges like artery detection and calcium scoring in chest CT scans. These tasks have traditionally been difficult to automate due to the complexity and variability of the anatomical features. The novel network architecture proposed here—by integrating different image perspectives and applying advanced backpropagation techniques—has shown promising results, particularly in bounding box regression accuracy. This suggests that the network, although still in its early stages, holds significant potential for real-world clinical applications, where precision in detecting subtle features like arterial calcifications can directly impact patient care.

One of the core innovations of this research lies in its attempt to guide neural networks in navigating complex environments by incorporating advanced image segmentation techniques [5]. The primary goal was to assist the neural network in recognizing specific patterns rather than relying solely on trial-and-error learning [5]. Through the implementation of advanced training algorithms, we provided the network with concrete guidance, helping it better understand the intricacies of arterial structures in CT scans.

While the methods employed here are promising, they represent only a fraction of the possible algorithms that could further optimize artery detection. Future work could involve exploring the potential of hybrid models that combine CNNs with other machine learning techniques, such as reinforcement learning or attention-based models, which are already showing promise in other areas of medical imaging. Additionally, cross-validation with other algorithms could provide a clearer comparison of performance, ensuring that the most efficient and accurate methods are chosen for clinical deployment.

This research also introduced a new optimizer and a modified backpropagation algorithm to improve the training process. The flexibility of the network structure allowed for adjustments between different stack sections and creates opportunities for fine-tuning performance for specific imaging tasks. Furthermore, the incorporation of parallel processing methods enhances the computational efficiency of the network, enabling it to scale across larger datasets and handle more complex tasks in real time.

The strong Pearson correlations across all four bounding box coordinates and sub-1% average deviation from ground truth trends support the spatial robustness of the network. Moreover, the regression slopes approaching unity confirm that the model is not systematically under- or over-estimating any specific boundary. These findings compare favorably to prior literature in anatomical detection using CNNs or object detection networks, which often

exhibit reduced localization precision when applied to small or irregular targets like coronary arteries [13]. Our network's superior accuracy may be attributed to task-specific training, optimized loss functions, and the constrained spatial variability of the LAD within the dataset.

Importantly, the model's performance remained consistent despite physiological and positional variations, including changes in heart orientation, image noise, and field of view shifts across test images. This robustness suggests strong generalization capacity, an essential feature for clinical deployment where real-world data often present with high variability.

From a clinical perspective, automated coronary calcium scoring remains a crucial yet underutilized component of cardiovascular risk assessment. Traditional methods rely on manual segmentation and measurement, which are time-intensive and subject to interobserver variability. Our model demonstrated high concordance with conventional calcium scoring methods, with an R^2 value of 0.921 and over 90% bin-level accuracy in clinical calcium score categories. These results align with or exceed those reported in other automated calcium scoring systems, particularly in terms of stratifying risk within accepted clinical thresholds (e.g., 0, 1-10, 11-100, 101-400, >400) [2].

Notably, the model maintained an average score deviation of less than 20% even when misclassified, highlighting its potential utility not only in screening but also in longitudinal risk monitoring. These findings are significant, given the growing recognition of calcium scoring as an independent predictor of cardiac events, particularly in asymptomatic patients or those with equivocal risk profiles based on traditional metrics [3].

In summary, this work lays the foundation for a new generation of neural networks capable of handling a diverse range of medical imaging challenges. The adaptability of the network, combined with the ability to process multi-perspective images and apply advanced training methods, promises to improve the speed, accuracy, and scalability of medical diagnoses, particularly in environments where data is sparse or complex.

5 LIMITATIONS AND FUTURE WORK

The limitations of this study include training on only one artery: the Left Anterior Descending (LAD) instead of all four available arteries: the RCA, LAD, CX, and LM. Training on these arteries would make the model more robust and versatile. The second limitation is that the proof-of-concept model architecture was developed to run on PNG, not DICOM images. While the process of window leveling tried to ensure that the PNG image kept most of the important cardiac features in the DICOM images, testing on the DICOM images is still necessary to either increase the models' accuracy and/or assess whether there are any differences when using PNGs vs DICOMs.

Additionally, the dataset used, while diverse in anatomy and image orientation, was limited to a single imaging modality and may not reflect the full range of acquisition protocols seen in broader clinical practice. Another limitation was inaccuracy in the ground truth annotations: they were manually curated and, while reviewed by experts, remain subject to human bias. Incorporating multi-institutional datasets with broader demographic and imaging diversity could further strengthen model generalizability.

Future directions for this project include expansion to all four coronary arteries, a larger dataset of chest CT scans, the addition of more convolutional layers, and the implementation of new techniques to further optimize the network. These adjustments would improve and optimize the network; however, it is computationally expensive. To further improve the network, images can be augmented through the application of convolutional kernels such as sharpening or modulating the relative intensity of surrounding structures, the network is able to explore in a wider range of images and environments to become better optimized and robust [10].

On CT scans, there is also the possibility of calculating the risk of valve disease. This type of network would specifically be helpful in datasets with continuous data, such as a functional MRI in the medical field. Beyond medicine, this could have wide-ranging applications in real-time detection and imaging by utilizing its third dimension to analyze images over time [25]. Another focus is integrating this detection module into a larger end-to-end diagnostic framework capable of processing raw DICOM images, performing artery segmentation, calcium quantification, and risk prediction without the need for manual intervention.

6 CONCLUSION

This project created a three-dimensional R-CNN to address common issues with 2D CNNs with respect to distortion and variance in size and shape. Through techniques such as new optimizers or events with states as well as alterations in the network structure and propagation, a five-section 3D CNN was created to train on non-gated chest CT scans, which contain distorted coronary artery motion, making it an example of the improvement of 3D CNNs over 2D CNNs. The network was used to identify bounding boxes of arteries in the chest CT scan and use those boxes to calculate a coronary artery calcium score. Both the network-generated bounding boxes and calculated calcium scores showed high accuracy, making this network capable of detecting small complex structures and providing meaningful clinical info. Applications of this network include datasets with multiple continuous images, such as a functional MRI or real-time camera data, especially in a setting with high variance and movement. The expansion of an AI network to 3D analysis using stacked images would further advance medical imaging analysis knowledge by providing information on how to analyze hyper-specific differences between consecutive images to quantify both shared and isolated features. The stacked feature maps may lead to new experiments with deriving different feature maps in the same convolutional layer. This leads to better feature isolation and a wider range of proposed features.

7 STUDENT REFLECTION

This project provided an opportunity to develop both technical and organizational skills. One of the main challenges was managing all aspects of the project independently, from planning and research to execution and problem-solving. Time management became crucial, as balancing the workload with other commitments required careful prioritization. A key solution was setting clear milestones

and breaking the project into smaller, manageable tasks, which helped maintain focus and ensure steady progress. The experience enhanced my ability to work autonomously, making me more confident in my problem-solving and decision-making abilities. From an educational perspective, it was a valuable exercise in applying theoretical concepts to practical situations. Specifically, I was inspired by concepts I learned in my classes for Multivariable Calculus and Linear Algebra, both taught by Dr. Michael Lavigne. This project demonstrated my initiative and capacity to handle complex tasks independently.

ACKNOWLEDGEMENTS

This work used Bridges-2 at Pittsburgh Supercomputing Center through allocation CHE160071 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296 [2].

REFERENCES

[1] Luís A. Alexandre. 2014. 3D Object Recognition Using Convolutional Neural Networks with Transfer Learning Between Input Channels. In *Intelligent Autonomous Systems 13 - Proceedings of the 13th International Conference IAS-13, Padova, Italy, July 15-18, 2014 (Advances in Intelligent Systems and Computing)*, Emanuele Menegatti, Nathan Michael, Karsten Berns, and Hiroaki Yamaguchi (Eds.), Vol. 302. Springer, 889–898. https://doi.org/10.1007/978-3-319-08338-4_64

[2] Shawn T. Brown, Paola Buitrago, Edward Hanna, Sergiu Sanielevici, Robin Scibek, and Nicholas A. Nystrom. 2021. Bridges-2: A Platform for Rapidly-Evolving and Data Intensive Research. In *Practice and Experience in Advanced Research Computing 2021: Evolution Across All Dimensions (PEARC '21)*. Association for Computing Machinery. https://doi.org/10.1145/3437359.3465593

[3] Giuseppe Pio Cannata. 2025. Backpropagation in fully convolutional networks (fcns). https://towardsdatascience.com/backpropagation-in-fully-convolutional-networks

[4] Kevin Clark. [n. d.]. Natural language processing with deep learning [Lecture Notes]. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/lectures/lecture5.pdf

[5] Laya Das, Abhishek Sivaram, and Venkat Venkatasubramanian. 2020. Hidden representations in deep neural networks: Part 2. Regression problems. *Computers & Chemical Engineering* 139 (2020). https://doi.org/10.1016/j.compchemeng.2020.106895

[6] Benoit Desjardins and Ella A. Kazerooni. 2004. ECG-Gated Cardiac CT. *American Journal of Roentgenology* 182, 4 (2004), 993–1010. https://doi.org/10.2214/ajr.182.4.1820993

[7] Alexey Dosovitskiy, Jost Tobias Springenberg, Maxim Tatarchenko, and Thomas Brox. 2017. Learning to Generate Chairs, Tables and Cars with Convolutional Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* 39, 4 (2017), 692–705. https://doi.org/10.1109/TPAMI.2016.2567384

[8] Philip Greenland, Michael J Blaha, Matthew J Budoff, Raimund Erbel, and Karol E Watson. 2018. Coronary calcium score and cardiovascular risk [review]. *Journal of the American College of Cardiology 72.4 (2018): 434-447.* 72, 4 (2018), 434–447. https://doi.org/10.1016/j.jacc.2018.05.027

[9] Harvey S. Hecht, Paul Cronin, Michael J. Blaha, Matthew J. Budoff, Ella A. Kazerooni, Jagat Narula, David Yankelevitz, and Suhny Abbara. 2017. 2016 SCCT/STR guidelines for coronary artery calcium scoring of noncontrast noncardiac chest CT scans: A report of the Society of Cardiovascular Computed Tomography and Society of Thoracic Radiology. *Journal of Cardiovascular Computed Tomography* 11, 1 (2017), 74–84. https://doi.org/10.1016/j.jcct.2016.11.003

[10] Alex Hernández-García and Peter König. 2018. Further Advantages of Data Augmentation on Convolutional Neural Networks. In *Artificial Neural Networks and Machine Learning – ICANN 2018*, Věra Kůrková, Yannis Manolopoulos, Barbara Hammer, Lazaros Iliadis, and Ilias Maglogiannis (Eds.). Springer International Publishing, 95–103.

[11] Hossein Hosseini, Baicen Xiao, Mayoore Jaiswal, and Radha Poovendran. 2017. On the Limitation of Convolutional Neural Networks in Recognizing Negative Images. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 352–358. https://doi.org/10.1109/ICMLA.2017.0-136

[12] Abdul Rahman Ihdayhid, Nick S. R. Lan, Michelle Williams, David Newby, Julien Flack, Simon Kwok, et al. 2023. Evaluation of an artificial intelligence coronary artery calcium scoring model from computed tomography. *Cardiac* 33 (2023), 321–329. https://doi.org/10.1007/s00330-022-09028-3

[13] Ivana Išgum, Annemarieke Rutten, Mathias Prokop, Marius Staring, Stefan Klein, Josien P. W. Pluim, Max A. Viergever, and Bram van Ginneken. 2010. Automated aortic calcium scoring on low-dose chest computed tomography. *Medical Physics* 37, 2 (2010), 714–723. https://doi.org/10.1118/1.3284211

[14] Ivana Išgum, Annemarieke Rutten, Mathias Prokop, and Bram van Ginneken. 2007. Detection of coronary calcifications from computed tomography scans for automated risk assessment of coronary artery disease. *Medical Physics* 34, 4 (2007), 1450–1461. https://doi.org/10.1118/1.2710548

[15] Scott P. Johnson, J. Gavin Bremner, Alan Slater, Uschi Mason, Kirsty Foster, and Andrea Cheshire. 2003. Infants' perception of object trajectories. *Child Development* 74, 1 (2003), 94–108. https://doi.org/10.1111/1467-8624.00523

[16] Nicole Karam, Sophie Bataille, Eloi Marijon, Muriel Tafflet, Hakim Benamer, Christophe Caussin, et al. 2019. Incidence, mortality, and outcome-predictors of sudden cardiac arrest complicating myocardial infarction prior to hospital admission. *Circulation: Cardiovascular Interventions* 12, 1 (2019), e007081. https://doi.org/10.1161/CIRCINTERVENTIONS.118.007081

[17] Andrew Murphy. 2023. Cardiac gating (CT). *Radiopaedia* (2023). https://radiopaedia.org/articles/cardiac-gating-ct?lang=us

[18] Khurram Nasir and Miguel Cainzos-Achirica. 2021. Role of coronary artery calcium score in the primary prevention of cardiovascular disease. *BMJ* (2021). https://doi.org/10.1136/bmj.n776

[19] Roma Pahwa and Ishwarlal Jialal. 2023. *Atherosclerosis*. Statpearls Publishing. https://www.ncbi.nlm.nih.gov/books/NBK507799/

[20] K. Raju, B. Chinna Rao, K. Saikumar, and Nalajala Lakshman Pratap. 2022. *An optimal hybrid solution to local and global facial recognition through machine learning*. Springer International Publishing, 203–226. https://doi.org/10.1007/978-3-030-76653-5_11

[21] Jia Shijie, Wang Ping, Jia Peiyi, and Hu Siping. 2017. Research on data augmentation for image classification based on convolution neural networks. In *2017 Chinese Automation Congress (CAC)*. 4165–4170. https://doi.org/10.1109/CAC.2017.8243510

[22] Daniel Soukup. 2020. The necessity and pitfall of augmentation in deep learning: Observations during a case study in triplet learning for coin images. In *Proceedings of the 9th International Conference on Pattern Recognition Applications and Methods - Volume 1 (ICPRAM 2020)*. SciTePress, 387–394. https://doi.org/10.5220/0008910303870394

[23] Conrad Wittram, Michael M. Maher, Albert J. Yoo, Mannudeep K. Kalra, Jo-Anne O. Shepard, and Theresa C. McLoud. 2004. CT angiography of pulmonary embolism: Diagnostic criteria and causes of misdiagnosis. *RadioGraphics* 24 (2004). https://doi.org/10.1148/rg.245045008

[24] Guanyu Yang, Yang Chen, Xiufang Ning, Qiaoyu Sun, Huazhong Shu, and Jean-Louis Coatrieux. 2016. Automatic coronary calcium scoring using noncontrast and contrast CT images. *Medical Physics* 43, 5 (2016), 2174–2186. https://doi.org/10.1118/1.4945045

[25] Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. 2017. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics* 28, 1 (2017), 162–169. https://doi.org/10.21629/JSEE.2017.01.18

[26] Douglas P Zipes and Hein JJ Williams. 1998. Sudden cardiac death. *Circulation* 98, 21 (1998). https://doi.org/10.1161/01.CIR.98.21.2334

Volume 16 Issue 2

ISSN 2153-4136 (online)