From Grading to *Demystifying the Computer*: My Exploration of Education and Idea Deconstruction

Written by Agustin Forero, MEng CS ’22 for Dr. Tapan Parikh

Independent Research (INFO 7900) at Cornell Tech, May 2022

Abstract

This report details my time working as an educator of computer science throughout the past few years. I discuss methodologies, experiences with different students and the various organizations I have worked for. I ultimately discuss my time working at the Roosevelt Island Senior Center, teaching senior citizens both how to use technologies and why they operate as they do. I close the report with some thoughts and reflections on these experiences.

1 Introduction

I have been lucky enough to be surrounded by teachers throughout the entirety of my life. My mother was a teacher for many years, teaching in parts of Texas and Chicago, ceasing to create a jewelry design business soon after I was born. Her mother, my grandmother, is also a teacher, herself raised by her teacher father in Patras, Greece. Looking back on how I was raised, these experiences certainly translated into how they addressed my siblings and I: they could put their foot down and enforce order when need be, but they always had a gentle, albeit meticulous, process to explaining things.

Teachers were also the highlight of my education before college. I looked up to the science teachers who made every student feel like a true chemist and biologist, or English teachers that asked us to answer difficult questions about ourselves and the media we consume. I especially admired history teachers: men and women who could spin dramatic stories that happened *here on Earth*, keeping the class at the edge of their seat in excitement, or short of breath with their hilarity. I owe an unknowable amount to these teachers, who so selflessly nurtured my classmates and I through some of the most formative years of our lives.

Something I did not quite appreciate until more recently, though, is that all of these people taught me how to *teach*. Teaching is a difficult and often exhausting task: you must place yourself in the shoes of your student, un-learning whatever assumptions, jargon and knowledge you have acquired on a subject. You cannot simply spew information from the perspective of a domain expert and expect a person to reciprocate. You must first break a matter apart piece-by-piece, finding the building blocks contained therein and assembling them back into a cohesive explanation.

This act of intellectual deconstruction is ofttimes extremely difficult: what if someone asks about a high-level subject? What if, while explaining this subject, it is revealed that a student does not understand one of the basic sub-topics that composes the subject? A teacher is asked to think on the fly, act as a confident arbiter of knowledge, and do it all with composition and grace.

This report, and the experiences detailed within, are a reflection on this difficulty, and how I have developed my own methods for teaching people around me. The age of my students has fluctuated with time, alongside the amount of knowledge with which they enter the classroom. I have taught in-person with a whiteboard and marker, as well as online over Zoom and Google Meet. My students have taught me just as much as I have taught them, and I feel all the more enriched from it. I only hope that my dabbling in education is one day continued through professorship or something similar. Until then, though, allow me to tell the story of how I accidentally fell in love with teaching.

2 Teaching at St. Olaf College

I graduated in May of 2021 with a BA in both Computer Science and History from St. Olaf College. It is a small liberal arts school in Minnesota, founded by Norwegian Lutherans in the 1870s. I greatly enjoyed my time studying there, even when the weather found new and creative ways to concoct the most slippery version of ice possible. You come to learn to walk like a penguin to avoid falling on your tailbone, but I am getting off-topic.

While at the College, I taught fellow undergraduates computer science in various capacities. I started by working as a grader for the College's introductory Python course, CS121, which would turn out to be the only class I was ever involved with. I would then work as a tutor for a semester, before working as a Supplemental Instruction (hereafter referred to as SI) Leader, similar to a teacher's assistant. Ultimately, I was the SI Mentor for the College's Math, Statistics and Computer Science department, teaching SI Leaders under me how to properly conduct their classes.

I did not expect to be so heavily involved with instruction on campus, as it initially grew out of a simple need to make money from a part-time job. Even still, my time instructing fellow students was one of the most important experiences in my undergraduate education, and it had an irrevocable effect on how I envision computer science and explain ideas to people.

2.1 Work as a Grader

During my sophomore year, I thought to become a grader for CS121. It seemed like a simple part-time job to fulfill while I went about my studies. All I would have to do was assign points to homework, leave feedback for students and host bi-weekly help sessions in the evenings. As it turned out, the job was not quite so simple: I would learn a few things about **idea deconstruction**, and how one explains concepts as an assemblage of smaller concepts.

Idea deconstruction is simply the act of viewing all ideas (in academia or otherwise) as a collection of smaller, accumulated ideas. Everything is derived, nothing is truly isolated and novel, and even the highest-level concept can be explained via a construction of lower-level concepts. This is not entirely a new concept *per se*, but is something I have become greatly familiar with over the past few years. Even the most inexperienced student can be taught something if idea deconstruction is utilized, and I have seen it successful across different classrooms and age groups.

Take, for example, the act of explaining the functionality of a neural network, as seen in **Figure 1**. Consider the following explanation, where I will italicize everything that may be interpreted as jargon.

> A *neural network* functions by changing the *weights* contained between the *neurons* held in its *layers.* It applies a *loss function* to measure its performance, then shifts *weights* around in order to heighten the *output* of neurons that contribute well to the model's *objective.*

In the context of Cornell Tech's Machine Learning Engineering class, the above explanation may be considered acceptable, but only because the students there already understand the terms italicized above. For a student just beginning their studies in computer science, this explanation would be complete hogwash. None of the terms make any sense, and each of them would easily necessitate its own paragraph of explanation. Though CS121 never dealt with neural networks, my experience with grading was my first experience with idea deconstruction: I could not just state issues with a student's code, but I had to instead work with them and at their level.

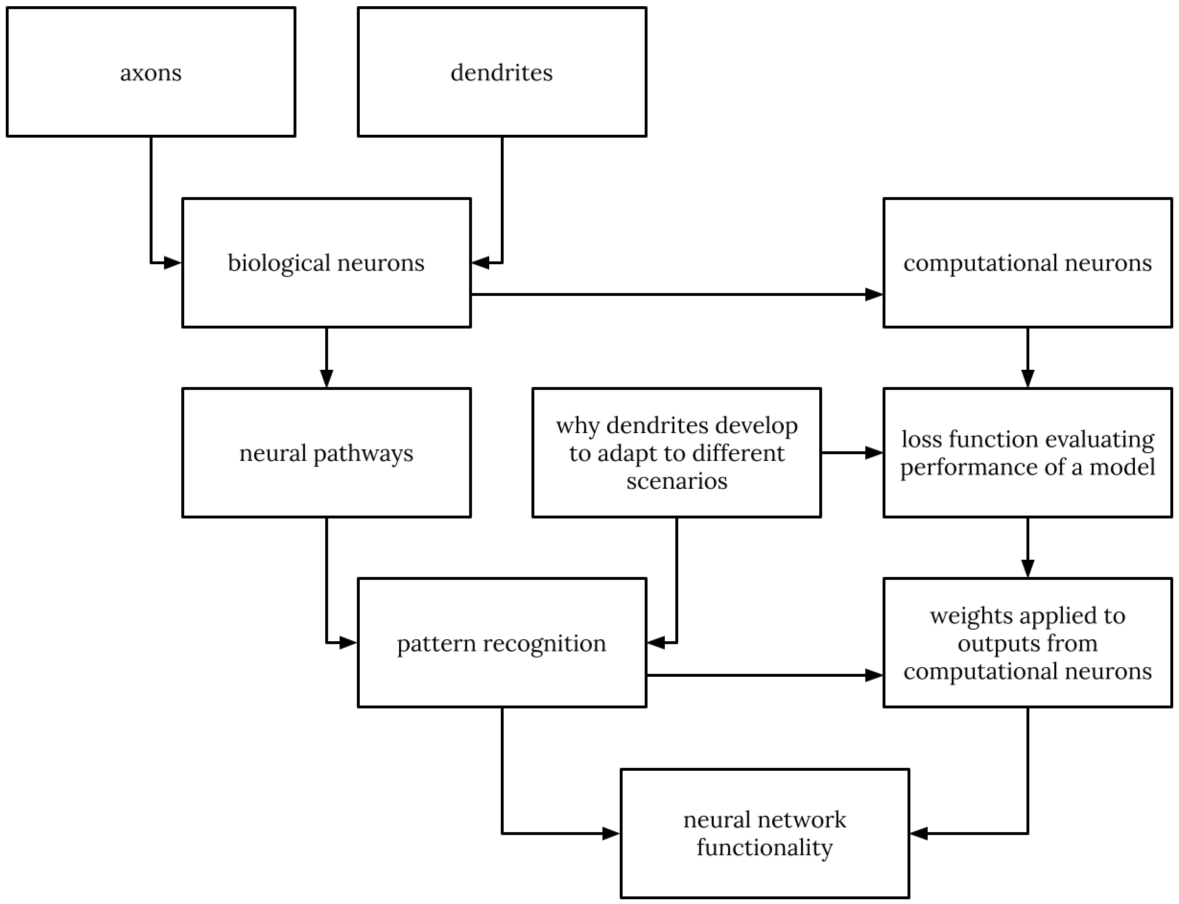
axons
dendrites
biological neurons
computational neurons
neural pathways
why dendrites develop to adapt to different scenarios
loss function evaluating performance of a model
pattern recognition
weights applied to outputs from computational neurons
neural network functionality

Figure 1: A chart detailing **idea deconstruction** applied to the functionality of neural networks. An arrow implies that it is best to teach something once this concept is cemented. For example,

$$\textit{biological neurons} \rightarrow \textit{computational neurons}$$

implies that it is easier to teach what a computational neuron is once the student understands neurons in the context of biology. Here, we can see how explaining the functionality of a neural network can be meticulously deconstructed into a series of smaller building blocks, themselves subject to this same deconstruction.

When grading assignments, I was sure to leave behind extensive comments detailing what went wrong and how they could fix it. Sometimes, the errors were extremely easy to fix: a hanging parenthesis, a missing tab, or a colon forgotten at the end of a for-loop. At other points, it was something more complicated: recursion depth exceeded due to the lack of a base case, an array being accessed one too many times, or a graphical error with the Tkinter library. These would each require much longer explanations, at which point idea deconstruction was a valuable tool to employ.

The same was for the homework help sessions. I would host two 1-hour help sessions every week, where I would help students work through their assignments (without explicitly giving them answers). A whole host of skill levels would come to the sessions. One student would show up knowing everything except fixing a quick, trivial issue. Others would appear in the middle of the object-oriented programming unit without knowing how to define a `Class`.

Each of these required a different approach, but the latter required a much more thorough explanation. Here, idea deconstruction would fit in naturally. To explain how to define a `Class`, I had to first explain how to define it syntactically, how it compares to more basic variables like `int`s and `float`s, what kind of real-life examples we could use it for, and when we would use it in Python. Struggling students usually understood these sub-topics on their own, and could more easily understand object-oriented programming once it was explained as a summation of these parts.

2.2 Work as a Tutor

I tutored for the first semester of my junior year. This was much more low-key than grading: I had only one student in CS121, and her and I met for only an hour each week. I did not have to grade any assignments or write any written reports.

Though it was not my first time working one-on-one with a student, this was still excellent experience with teaching in a more personal setting. My student would typically come to me with questions on the homework, and I would provide her everything aside from the answer itself. Teaching one specific person over and over is certainly a different environment than teaching *en masse*: you come to learn a specific student's mannerisms and preferred methods of communication. Does the student enjoy visual learning, or would a verbal explanation suffice? How effective is it to ask them to draw something on a whiteboard? What kind of cultural differences between you two have to be accounted for? These were not just questions for this particular student; in all settings where I have taught one-on-one, all of these concepts have surfaced.

Again, idea deconstruction surfaced here. This particular student had no prior experience with programming, so it was necessary to explain each subject as a construction of smaller parts. This was the case at the beginning of the semester with variables, just as it was at the end with recursion. Tutoring was not quite as involved as grading, but I still learned much while working it, and it certainly helped solidify my own communication and teaching skills.

2.3 Work as a Supplemental Instruction Leader and Mentor

For the second semester of my junior year, I worked as a Supplemental Instruction (SI) Leader. The role is a bit similar to a teacher's assistant, only instead of helping write exams and conduct class during the day, the SI Leader hosts extra class sessions at night (SI Sessions). I was expected to run at least two class sessions per week for one hour at a time, preparing a curriculum for each day depending on the current dealings of the class. Again, I worked for the CS121 class.

This was certainly the most involved of all the part-time jobs I had so far. It was nowhere nearly as free-form as grading or tutoring, which revolved around students coming to me with their individual questions or confusions. Instead, I was tasked with coming up with a lesson plan myself, filling out a 1-hour time slot with cohesive activities for the students to fulfill.

The activities were sometimes difficult to plan. The plan had to be cohesive enough that students at all levels could understand it, engaging enough that they would even *want* to interact with it and thorough enough to do the current topic justice. To start, this was fulfilled through team-based whiteboarding activities. I would group students together to solve a puzzle or task, and they would whiteboard solutions with markers. Students were tasked with designing bank systems, computerizing a beloved pet with Python classes and using recursion to manipulate strings. Things became much more difficult in March of 2022, when COVID-19 caused colleges nationwide to shut down, including my own.

After this point, conducting class was sadly much less personal. I could only afford to meet with students over Google Meet from my room in Illinois. I could no longer assign students to work in teams, as my Google Account was not privileged to the breakout teams feature, and because attendance had plummeted since the start of isolation. There was also just a lack of interpersonal connection: there is something invaluable about body language and slight facial expressions that are lost from online communication. Actions I had previously taken for granted, like high-fiving a student for getting something right, were no longer possible. The Google Meets fulfilled their tasks in the sense that we still met together to learn, but the virus reduced the classroom to something more coldly computerized and dull. I was absolutely not prepared to begin teaching online.

All the same, class continued, and I still employed idea deconstruction whenever possible. Whichever sub-topics composed a lesson were likened to real-life examples, and students were asked to explain concepts in their own terms. Considering the circumstances we were put in, alongside the innate stress attributed to that time in history, we still learned a lot together.

I continued on to my senior year as the SI Mentor for the Math, Statistics and Computer Science department. I oversaw all SI Leaders teaching Sessions in these fields. I watched their lectures to ensure they were delivering effective lessons, and advised them on handling different situations or professors. It was still not quite as involved as the SI Leader role itself, but it was exciting to work so closely with other instructors. Ultimately, throughout my time at the College instructing other students, I molded my own teaching techniques and strategies, and developed an understanding for how to properly practice idea deconstruction.

3 Teaching at Inspirit AI

For the summer after my senior year, I worked full-time as an Instructor at **Inspirit AI**[1]. The company hires students out of prestigious universities to teach high school students about artificial intelligence and programming. This was a younger demographic than I was used to and my knowledge of artificial intelligence was novice at best. Not just that, but I would have to capture and hold the attention of high school students over the internet. To say the least, the beginning of this endeavor was daunting.

To my advantage, all lessons were constructed preemptively by other employees of the company. I did not have to plan or schedule anything; it was all lay out for me to follow. Using these lessons, I taught small classrooms of roughly five to six students at a time, mostly beginners themselves. Their program was two weeks long: one to learn the basics of artificial intelligence (and programming if need be), and another to construct a project.

It is hard for me to detail many difficulties with the students themselves. Of course, it was not always easy to retain their attention once the lesson began, or to politely ask them to keep their camera on whenever possible. There were times when class morale was low and no one understood the concept at hand, even after a few attempts to explain it. All the same, I saw some incredible students that summer: people who went above and beyond what was expected of them, and who enriched the quality of education for their peers.

My best cohorts were those that learned as a collective. There were qualities of the students that enabled this: they were polite and pensive, but they were not afraid to raise their hand or stop the class to ask a clarifying question. My strongest classrooms often saw students answer questions on my behalf. There are few more cathartic things than seeing a group of students start out not knowing anything about coding or artificial intelligence, only to see them just a few days later taking charge and speaking like true programmers.

Again, the online medium is not the friendliest to learning. Finding a means to politely keep the classroom from devolving into a Zoom void[2] can be rather demanding, and students did not always come in eager to learn — it was clear some were simply there at the behest of their paying parents. All the same, I found my time at Inspirit AI to be extremely enriching, and the teaching methods I had learned in undergrad greatly improved my methods there. Students learned best when idea deconstruction picked complicated ideas apart. Figure 1's scenario of explaining neural networks isn't hypothetical: the chart really does detail how I explained it to these high school students.

[1]A link to their website can be found here.

[2]Zoom classes where everyone but the instructor has their microphone and camera off.

- Zoom
- Streaming services
- How to use Windows
- How to use the iPhone
- Google Drive
- The internet (safety, fake links, how it works, etc.)
- Tracking
- Scam protection
- How to use your email
- Women in computer science
- Ethics in computer science
- Artificial intelligence
- Algorithms
- Binary
- Video games about history

Figure 2: Some, though not all, of the topics covered in *Demystifying the Computer*. A full list of recorded lectures can be found at this link.

4 *Demystifying the Computer*

I taught undergraduates and high school students alike, but a whole new challenge was unveiled when I embarked on my project at the Roosevelt Island Senior Center, *Demystifying the Computer*. With the first two groups, I was working from an advantageous position I had previously taken for granted: younger demographics already knew how to use technology. In my practice of enforcing idea deconstruction, I had yet to consider technology itself to be a building block. In other words, I took technological literacy as a given. Teaching senior citizens how to use technology requires educators to step even further back in their field of expertise, coming to a standpoint where the devices themselves are unknown.

This class was essentially a culmination of everything I had learned about teaching thus far. I had experience developing curriculum from my days as an SI Leader, alongside explanatory skills acquired from Inspirit AI and tutoring. I knew I had the patience and domain expertise to hold together a strong class, and I was excited at engaging with a new age demographic.

The course was titled *Demystifying the Computer* because that is exactly what I sought to achieve: teaching seniors how to properly use the technologies that surround them every day, while also teaching the historical context and abstractions that help computers operate today. I wanted seniors to understand that computers are not just some mystical black box. In essence, they are just extremely robust calculators with many bells and whistles.

To this end, my approach to teaching the class was two-fold. On Mondays, our class would review various contemporary technologies, addressing any and all questions the seniors had to offer. Monday topics included how to use Zoom, Windows and email, how to protect themselves from phishing scams, how to identify fake links, and more. After class, seniors had a half hour to bring forth whatever technological issues they had, usually some small bug on their phone or a suspicious text message. Mondays were straightforward and relatively free from abstractions.

Wednesday lessons targeted my second objective of deeply contextualizing technology today. Seniors were taught how the architecture of the internet functions, the history of women in computer science, ethics of artificial intelligence and binary. Possibly my favorite lesson conducted over the 11-week period was a Wednesday lesson where I taught them the basics of Python programming. The seniors were greatly excited to become ”junior level programmers”, and they grasped onto syntax faster than many undergraduates.[3] **Figure 2** features a list of some topics covered across both Monday and Wednesday lectures, alongside the recordings I produced of the lectures.

To put it lightly, *Demystifying the Computer* was one of the most enriching academic endeavors I have had the pleasure of undertaking. I took many exciting classes at Cornell, ranging in topic from machine learning to design to privacy, but my independent study working with the Senior Center was surely the most intellectually stimulating and emotionally fulfilling. My students were easily the most

[3] A recording of this particular session can be found at this link. I am particularly proud of this session; I would highly encourage anyone to watch it.

active participants I ever instructed, and they never allowed me to sprint through confusing jargon without stopping to ask a clarifying question. If I asked the class to guess the answer to a problem, someone would invariably raise their hand with some sort of answer. Even if they got the question wrong, they would continue shamelessly participating. The tenacity of my students was astounding.

Seniors reacted differently to different concepts. I saw my students exhibit a full range of emotions from amazement to outrage, relief to fear. Most of the time, they were very happy to learn what they did: it was not rare for a senior to thank me for "enlightening them" to a concept at the end of a session, such as with the email lecture.[4] Other lectures invoked anger and discomfort on their behalf, thankfully not directed at me, but instead to the technologies themselves. This was especially the case with the tracking lecture,[5] where seniors expressed disdain towards draconian tracking policies employed by advertising agencies across the internet. The lesson on ethics in artificial intelligence conjured some uncomfortable gazes from the seniors, confronted with the reality of self-driving cars and automatic resume analyzers.[6] Generally speaking, I could see attitudes change over the course of the semester, and the students grew to voice exactly how they felt about the subject matter at hand.

I was also honored to develop a friendship with many of my students. Since Zoom took roughly half an hour to render my recordings after class was over, I usually stayed in the classroom for an extended period after the lecture had finished, and there were countless times where the seniors and I simply chatted about various subjects. Seniors would stay after to talk about a technology they found ethically objectionable, or another that they had difficulty navigating. Another time, we spent a good 20 minutes talking about the Will Smith Oscars fiasco. In wake of my upcoming move to Boston (where I will begin as a software engineer), the seniors advised me on how to find a place to live, and what to do once I get there. I was touched to get to know their names and interests, and my time spend with these students is like nothing I've experienced in academia.

As humbly as I can, I would consider *Demystifying the Computer* to be a striking success. I did not solve every technological issue the seniors were experiencing, nor did I turn them into hardened computer scientists. But throughout my weeks of instruction, the seniors that chose to dedicate so many hours to hearing my lectures certainly evolved a stronger sense of technological literacy. Simply from hearing them in conversation, I noticed that they had learned many of the concepts I aimed to teach them — detecting phony text messages and emails, understanding when and how they are tracked across electronic mediums, navigating their internet browsers, and so on. Each of these concepts were explained as a sum of smaller parts, drawing on my experience employing idea decomposition in the classroom.

5 Conclusion and Thanks

I owe a great deal to the senior citizens. Not only did they take time out of their days to hear me lecture, but they also provided incredible companionship during my second semester living in New York City. This Independent Research study was truly the capstone on top of my limited experience in teaching.

Demystifying the Computer was a synopsis of all skills I had acquired for effective instruction. I was tasked with teaching students unfamiliar with the domain I study for a living, placing myself in their shoes and utilizing idea decomposition to their benefit. As the same students repeatedly attended my classes, I discovered what each individual student was concerned with, and learned to reference their respective life experiences in class.

My journey dabbling in education was only a few years long and spawned from a simple work-study, but it has come to strikingly redefine my education within the discipline of computer science. I was taught to program and design software from the perspective of the engineer, but education instilled within me strong abilities in speaking and empathy. I would not be the same student without my experiences as an instructor, and I know this for a fact.

None of this would be possible without the guidance of the teachers in my life: those that instructed me during my childhood in Illinois, and the professors that taught me at St. Olaf and Cornell. They taught me how to teach, and showed me patience when it was unnecessary to do so. In wake of this

[4]Recorded at this link.

[5]Recorded at this link.

[6]Recorded at this link. I had the seniors engage with the MIT Moral Machine, which invoked some very passionate responses from the class.

journey, I must also thank my students — those that turned the classroom from a top-down dichotomy of teacher vs. student, into an open discussion of shared passion and intrigue. My own studies have been enriched from the presence of these people, and I am humbly thankful for this.