Personalized Mathematics Education with Intelligent Recommendation

Personalized Mathematics Education with Intelligent Recommendation

Benjamin Trey

A Thesis in the Field of Software Engineering

for the Degree of Master of Liberal Arts in Extension Studies

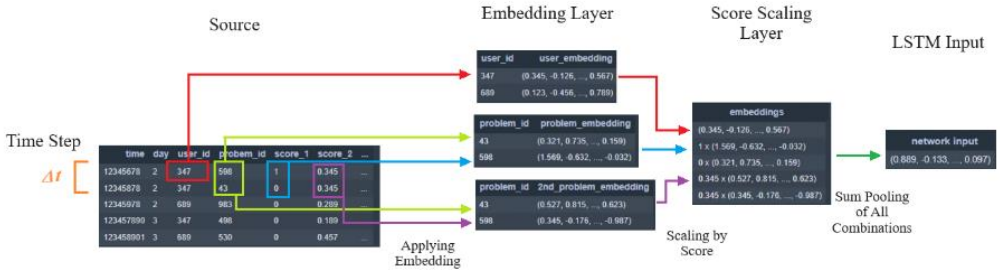
Harvard University

May 2024

Abstract

Differentiation, the process of individualizing education, remains a difficult endeavor that can overwork teachers and negatively affect students. Due to the tremendous upside of differentiation, this thesis examines the possibility of predicting student performance using Cross-Network LSTMs with additional data sources. After modifying the attention mechanism, the model proposed showed an improved performance in prediction with the inclusion of additional measurements already in the Junyi Academy Online Learning Activity Dataset. The modifications also showed the ability to separate problems into course units based on their similarity scores, and quickly make long term predictions. The development of this network as recommender system for differentiation is discussed as well as a working example using the AP Calculus BC curriculum.

Frontispiece



Dedication

To Bruce, Marcia and Holly, for their endless support.

Acknowledgments

Foremost I would like to thank my students for their constant inspiration. The incredible things they do always drive me to become better.

I first discovered neural networks because a student wanted to learn them. Without Shmuel Sokol I would never have started this journey. His lifelong dedication to learning, and desire to take on seemingly impossible tasks inspired this work more than both of us understand. I am grateful for his friendship and influence on me.

The instruction I received while conducting this research also deserves specific recognition. As I began the process of writing this thesis, the task seemed completely alien to me. I would like to thank my advisor Dr. Seiji Isotani for his advice and direction. He explained to me the underlying motivations driving each section and the way to present results. The advice he gave me showed his experience, and his ability to predict the direction of the project amazed me. He is truly an incredible researcher, and I owe a debt of gratitude for his help.

Similarly, I also owe my thesis director. The input from Dr. Hongming Wang provided wonderful guidance as I discovered how to complete this work. She knew how to make the right additions at the right time. Many of the relevant sections in this research only exist because of her suggestions.

In addition to help conducting research, many people also helped me gain the professional experience necessary to complete this project. I would like to recognize the opportunities given to me to become a better educator. Particularly I would like to

recognize Ms. Lee and Mr. Wang of Kinglee High School, Zhengzhou CN. Without their support I would not have developed into the educator I am today. The environment they cultivated allowed me to thrive under the guidance of wonderful, handpicked supervisors. They inspired me to dedicate myself to every class of students I teach, and to find success through creating success for others. Their contributions to the Zhengzhou educational community cannot be understated.

The individuals in charge of bringing the visions of Ms. Lee and Mr. Wang to life also deserve special recognition. Under the supervision of Sabrina Langlois and Dan Kemock I created my first website to differentiate in the classroom. They defended my methods of instruction and gave me the freedom to make the best choices for my students. The direction they gave helped me to trust supervisors and incorporate the suggestions of others.

Finally, I would like to acknowledge Proctorio for the opportunity they gave me. My internship gave me a chance to work with neural networks in an educational setting. It gave me the confidence to apply the methods of CS231n.

Table of Contents

Chapter I.

Introduction

Traditional k-12 education has the following disadvantages. First traditional education is unable to provide a personalized manner of teaching and learning. In traditional education, students are instructed in classrooms. Because there are many students, usually more than 50 students, in a classroom, teachers cannot teach students in a personalized manner. Second, traditional classroom education does not take different abilities and traits of individuals into account, resulting that the teaching schedule is too relaxed for the top students and too pushy for academically poor students. As the same homework is assigned to the whole class, top students finish assignments faster and need to seek more work to do, whereas academically poor students are frustrated with the task they are assigned. Third, traditional education system lacks a valid standard method to assess students' knowledge level by collecting their submitted exercises data.

—(Gong et al., 2018).

Any educator entering the classroom immediately realizes the size of the task in front of them. Each student thinks uniquely, with different interests, and has their own needs. To properly address the distribution of personalities a teacher will need to spin

many plates at the same time. Just the complexity of the task intimidates many educators from even attempting to modify the curriculum on the individual level.

When I first attempted to differentiate the curriculum in my classroom, I felt the same. It was in my first year as the head of the Mathematics Department at Kinglee High School in Zhengzhou, China, and due to articulation, my Algebra II class became composed of equal parts remedial and advanced students. The environment remained productive, and we did create a positive math experience, but towards the last quarter of the year I found the students either struggling, or desperately seeking a challenge.

Kinglee always remained a supportive environment for educators, and with the encouragement of my supervisors I decided to differentiate my classroom. I split the class into two parts and allowed students to choose their half of the class. One half worked on applied projects, and the other half reviewed problems. The projects scared away the remedial students, and the problems threatened the advanced students with tedious boredom. I did not think I could lecture both, so I gave only a brief introduction to the projects and allowed the students to problem solve their projects as they went.

At the time I decided to intervene, the class was in the unit for systems of equations and matrices. The project group worked on making websites that performed matrix operations. The other group worked on general linear algebra questions. I spent most of class time answering questions posed by either group.

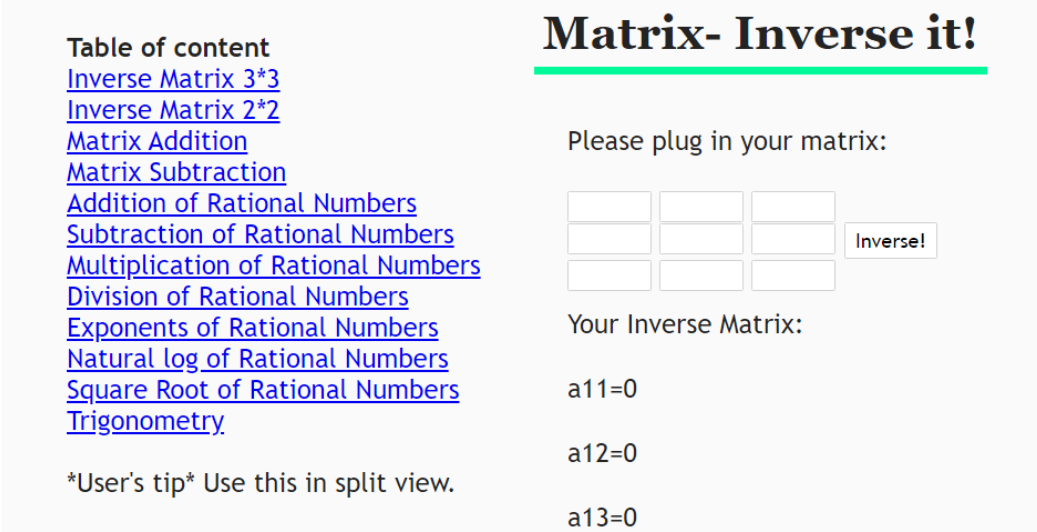


Figure 1. Matrix Inverses Widget, In Split View!

Finished Website Turned in by a Student. (Lin, n.d.)

At the end of the semester, I felt the two groups got what they needed. Everyone took the same final and did well. Some students got an introduction to applied math in front end development as shown in Figure 1. Other students got the extra reps they needed to master the material. Everyone created a positive experience with math, and many of the students in that class went on to pursue graduate work in STEM.

As a teacher I felt a great sense of pride in my work, but as a developer of curriculum I knew I created an unrepeatable course. Even when I wrote my unit guides, it became difficult to archive the final quarter. I created a course that did not scale and required extensive professional development for its instructor. After the course, no instructor chose to duplicate that semester.

Even if another teacher chose to replicate that curriculum, the semester did not thoroughly differentiate the material. The course did not offer much in the extent of

individualized education. Students chose if they wanted to do projects or problems. The choices did not address other preferences or optimize the success of the students. At best it made two groups based on the students' choosing, which may lead to additional difficulties.

Creating a choice does not even guarantee a student will make decisions to facilitate their learning in the best way. The student received work based on their preference, and making students happy is not always the best metric for education. Even with the extra work I put in, and the praise of my supervisors, my efforts could only offer an additional choice. It did not give students the opportunity to learn and explore the material in a unique way that was meaningful for them.

As in my case, differentiation puts many requirements on instructors. It adds an additional course the instructor must prepare. This will often require separate lectures, assessments, and materials. If a teacher does a modest differentiation of each course, it doubles their amount of preparation time. Traditional methods of instruction make individualizing math education difficult.

On the other hand, technology caters to millions of users on the individual level. News feeds, shopping suggestions, and targeted ads all operate catering to specific user profiles. All of us expect to get unique results when we log in. Social media sites operate elastically and readily adapt to an influx of complex data. The algorithms used by these platforms extend usage, and nurture habitual use.

The next time I considered using technology to differentiate my classroom. The ability to predict student preferences still remained outside of my skillset at that moment. Each student got problems from a problem bank with randomized parameters. Upon

submission of the problem set, the students got an immediate result of their grade with specific labels for the problems they got wrong. Although the different assessments ignored student preferences, a secondary effect of individualized education occurred. Students could not submit another student's solutions. Every student completed every assessment independently.

The first Algebra II class to receive individualized assessments became one the most successful classes I ever taught. I gave them a quiz on simple 3×3 matrix inverses without any instruction. Every student successfully inverted 3×3 matrices, and each student used a different method.

Even with the success of individualized assessment, students always gave some resistance to its use. I always needed to add individualized assessments carefully. The contrast between traditional instruction and individualized instruction felt completely jarring for most students. Students do not frequently encounter assessments created specifically for them. Even more rarely do students encounter assessments with the same adaptability as social media, as proposed in this project. Based on the results from my students, it appears differentiation of education on the individual level offers a wildly different experience compared to traditional instruction.

The combination of an under-addressed need, and stable widely used technology, presents a tremendous opportunity. Their intersection should lead to an increased usage, and research of both. The user experience, and more importantly the education of the individual, should benefit through promotion of interdisciplinary research.

1.1 Differentiation in Education

Differentiation, the individualization of education, aims to provide a learning environment for each student's interests, abilities, and learning styles (Deunk et al., 2018). Formative assessments should engage the students. Students with short attention spans should get problems with many smaller parts. Students that learn better through life experience should get problems with an opportunity to work independently. Students looking to explore a career should get problems from professionals in that career (Tomlinson et al., 2003). End of unit tests should allow students to demonstrate knowledge in a variety of ways. Differentiation finds the right curriculum for each student while accomplishing the same academic goals.

Of course no introduction to differentiation in education would be complete without the classic cartoon of animals taking a test. This cartoon exemplifies differentiation so well it exists in hundreds of variations throughout the internet. Some versions add a quote falsely attributed to Albert Einstein, redraw the picture, or add some text to give political context. Almost every introduction to differentiation finds enough utility in this cartoon to include it, and even sometimes add modifications.

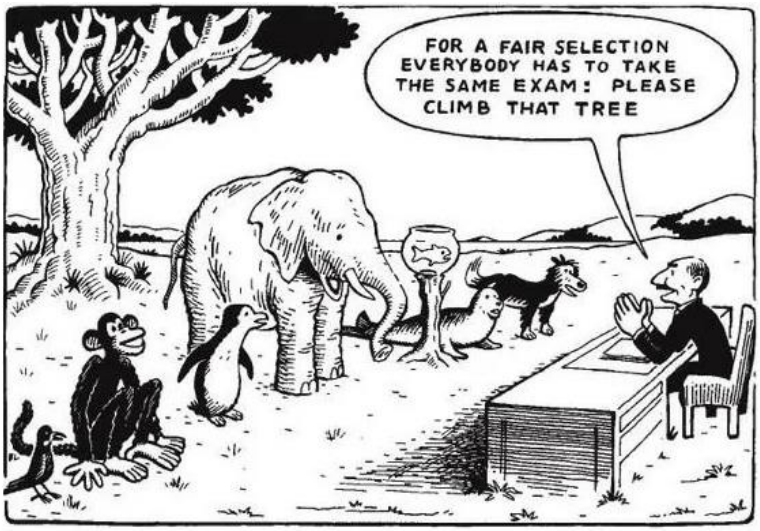


Figure 2. A Comic Frequently Used to Describe Differentiation in Education

An Instructor Asks a Group of Different Animals to Complete a Task Suited for One of Them (Linton, 2000)

The cartoon in Figure 2 shows an instructor asking different animals to climb a tree and highlights how individual differences can affect the measurement of ability. Instructors will often use this picture as an example of *reductio ad absurdum* when introducing differentiation. Asking a penguin to climb a tree seems ridiculous. If the method of examination does not address the differences of the individuals, the results carry as much weight as asking a monkey and a fish to climb a tree. Although this example highlights the absence of differentiation in examination, this example can also extend into instruction as well. If assessment does not address the differences of individuals, it will create underlying differences in achievement (Moon, 2005).

1.1.1 Potential Difficulties in Differentiation

To properly differentiate a classroom, an instructor will often create multiple assignments for each lesson with similar learning goals. Assessments should address each learning style, ability, and interest. For an instructor this means making resources for every student and the workload can quickly become overwhelming (*Differentiated Instruction*, 2014). To accomplish differentiation in a traditional setting, this often requires the grouping of students based on the observations of the instructor. Even with the best intentions, the act of grouping may introduce unintended consequences.

In the case of Circular Differentiation, bias may group students by race, ethnicity, gender, family social status, immigrant status or disability, instead of interests, ability, or learning style. In this case the act of grouping the students determines the students outcome (Schneider, 2018). Often the most at-risk students will need the most specialized education while attending schools without proper funding. As the study of differentiation goes forward, it needs to address how to group students in a way that most benefits them.

The documented difficulties in individualized education make it a hit or miss proposition. It greatly increases the responsibilities of teachers, and students could face lifelong setbacks. Even though these issues are difficult to overcome, these potential difficulties should not deter the future use of differentiation. With improved tools, and technologies, students could receive individualized education without an increased workload on teachers.

1.1.2 Addressing Language Usage with Differentiation

In addition to the problem of proper grouping, cultural effects in assessment give another problem differentiation should address. Differentiation should reduce the

differences created by the educational process. Standardized tests frequently exaggerate cultural differences in achievement. For example, a test may measure a cultural difference in achievement, but only when the test uses common language. These differences disappear when the test asks questions more formally. A theoretical explanation of this phenomena comes from the different use of common language within each group (Freedle, 2003). As the understanding of a question should not affect the measurement of proficiency, individualized instruction needs to adapt appropriately. Phrasing a question in a way that best communicates to the user looks like a great opportunity for differentiation.

1.1.3 Fundamental Work in Differentiation

Carol Ann Tomlinson summarized the problems in differentiation, and strategies to address them in the work *The differentiated classroom: Responding to the needs of all learners*. Here she laid the groundwork for the modern differentiated classroom. Many expanded on this work in specific areas, and mathematics received a full treatment in *Good questions: Great ways to differentiate mathematics instruction in the standards-based classroom*. by Marian Small. In an extension of Tomlinson's work to big data the *Recommender systems handbook second edition*, summarized the incorporation of recommender systems in the domain of education, or as literature refers to it individualized education (Tang et al., 2016a).

1.2 Recommender Systems

The previous section showed the incredible difference differentiation can make in education, but it did not address how to overcome the difficulty in implementation.

Individualized education requires too much effort from teachers, and mistakes can be costly. The primary problem of how to deliver content on an individual level is already solved in the technology sector in the form of Recommender Systems.

Recommender Systems (RS) take a user's interactions online and make recommendations for content. Social media and online platforms depend on this technology. Facebook, Tik Tok, Instagram, and YouTube rely on these technologies for user engagement, but also to give effective targeted ads. Without RS, deriving meaning out of the quickly accumulating, and bottomless volume of user data appears impossible.

RS works by summarizing a user's interactions mathematically. After summarizing the interactions, the RS can predict the user's preferences and present the user with content they will most likely interact with. If the platform prioritizes usage, the user will use the platform more frequently and longer. If the platform prioritizes development, the same mechanisms will attempt to maximize user development.

Similarly to differentiation, RS and the collection of educational data are well understood and researched due to their industrial use (Rodrigues et al., 2018; Sin et al., 2015). Discovered almost unintentionally by computer scientist Elaine Rich in 1979 while looking for book recommendations, RS attracted many corporate and institutional research groups. The possibility for commercial use attracted attention from traditional tech companies, which prompted Netflix to begin the Netflix Prize. Opening the research of recommendation engines to a public competition introduced many new ideas including causing some researchers to consider neural networks for prediction.

1.2.1 Influence of the Netflix Prize in Recommender Systems

With the influence of the Netflix Prize researchers used user ratings with Residual Neural Networks (RNNs) to create predictions for YouTube ratings. Their research focused on the contextual information by embedding it and then combining the embedding with the hidden state with element-wise multiplication. The long-term and short-term dynamics of the problems drew most of the focus from the team. They found considerable statistical significance in their predictions for recommendation systems with a downside of a long training time (Beutel et al., 2018).

RNNs also showed an ability to predict student behavior by using student written essays and user click data coming from Massive Online Open Courses (MOOCs). Given a prompt, Two Layer LSTMs can suggest the next word after given the previous text in an essay. For different levels of mastery, the model can train on segmented data based on grade level or ability. After training the model predicted the correct work choice 21% of the time, which is quite a high accuracy given the large vocabulary of the system.

In the same study researchers looked at predicting the clicks of students given 27 options. They found statistically significant predictions but not enough to claim a solution to the problem. For further research they suggested more detailed data including section number and problem id. In an extension of the research the same research team applied their methods to a different MOOC. In this case they compared the performance of several different models. The models showed a similar overall asymptotic performance while differing mostly in the number of epochs for training. According to the researchers one of the downsides came from the necessity of a GPU for the Python modules used (Tang et al., 2016b).

Previous work also showed the use of AI in mathematics education. The Classroom Learning Partner (CLP) used tablet-based software to help teachers understand how students visualized arithmetic. Students solved problems on their tablets and machine analysis routines categorized students based on solution method. In addition, the tablets took data from the students as they developed a solution. Measured data points included the order of steps and removed steps. This automated the need for a human to label a student's work, and improved feedback capabilities. Students received feedback based on the order of intermediate steps and removed steps. For the teacher it allowed a real time summary of how the students visualized arithmetic (Koile et al., 2016).

In a setting outside of education Perera and Zimmermann showed the ability of combining models in a Cross-Network LSTM Network. They used models trained on responses from independent data sources and combined them for input into a LSTM. Their results showed an improved accuracy in predicting the actions of users. Several opportunities exist to improve prediction based on the inclusion of contextual details into the models (Perera & Zimmermann, 2020).

As the primary goals of differentiation include incorporating students' interests, reducing the effects of cultural background on performance, and giving assessments that adapt to learning styles, a network that quickly includes additional data sources, appears like a prime candidate for differentiation. With a problem bank of sufficient complexity, a Parallel LSTM Network could take additional measurements and give users an experience that exactly meets their preferences.

1.3 Automated Assessment in Mathematics

Mathematics assessment comes in a variety of forms due to the prevalence of technology in the classroom. Through. automation of grading, modification of assignments between students, and gamification of education, technology usage in the classroom accelerated assessment in mathematics. Due to the available platforms assessments usually come in a few standard types, static problem sets, static problem sets with variable parameters, dynamic problem sets, dynamic problem sets with variable parameters, individualized problem sets, and individualized problem sets with variable parameters.

A Learning Management System (LMS) such as Canvas[1] will offer access to the problems as well as other course resources. The LMS also serves as a base of tools an instructor can use for assessment. The proprietary tools as well as plugins such as Proctorio allow for the creation of formative assessments and tests.

As shown later the presentation of the problem will become an important factor in the prediction of a student's performance. Hints, references, examples, and videos give the students enough resources to finish problems without a formal introduction. When predicting how a student will do on new problems, problems with videos attached do not see a drop off even if the content comes much later in a course. This will need more careful treatment later, but for now it serves as a testament to the importance of problem format.

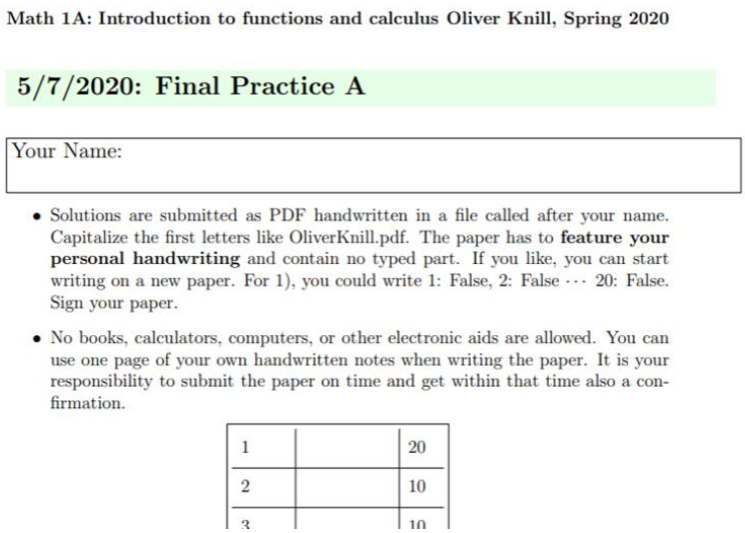
[1]Canvas is a Learning Management System used by many educational institutes to administer classes. https://www.instructure.com/canvas

1.3.1 Static Problem Sets

Static problem sets consist of identical assignments between all students. These can consist of problems out of a book, problems created by an instructor, or problems coming from an online platform offering assessment. The students may complete the assignment by turning in their work on paper or submitting answers through an online form. The students can receive feedback immediately after completing a problem, after completing all problems, or after submitting their assignment and their instructor reviews the assessment.

An example of an assessment with static problems would be the Final Practice from the course Math 1A Introduction to Functions and Calculus[2] offered at Havard College. All students in the class shared the same problems shown and submitted their answers by writing their answers into the answer blank. After reviewing the submissions an instructor could provide feedback to the students.

[2] Math 1A is the Harvard Introductory Calculus class. https://qrd.college.harvard.edu/classes/math-1a-introduction-calculus



Math 1A: Introduction to functions and calculus Oliver Knill, Spring 2020

5/7/2020: Final Practice A

Your Name:

- Solutions are submitted as PDF handwritten in a file called after your name. Capitalize the first letters like OliverKnill.pdf. The paper has to **feature your personal handwriting** and contain no typed part. If you like, you can start writing on a new paper. For 1), you could write 1: False, 2: False ··· 20: False. Sign your paper.
- No books, calculators, computers, or other electronic aids are allowed. You can use one page of your own handwritten notes when writing the paper. It is your responsibility to submit the paper on time and get within that time also a confirmation.

1		20
2		10
3		10

Figure 3. Math 1A Final Practice A Answer Blank with Partial Answer Blank *Instructions with Partial Answer Blank Shown (Knill, 2020)*

Figure 3 shows how the answer blank would appear in practice. Each answer blank refers to a set of questions, and different versions of the same exam could exist to vary questions between groups of students. This answer blank refers to one version of the assignment, and students with different versions of the assignment could not share work. However, because the problems cover the same topics different versions of the test will allow students to collaborate without directly sharing responses.

Problem 3) Matching or short answer problem (10 points). No justifications are needed.

a) (4 points) On the Boston Esplanade is a sculpture of **Arthur Fiedler** (1894-1979) a long-time conductor of the Boston Pops Orchestra. His head is sliced into thin slices. Assume that the thickness of each level is $h = 1.5$ inch and the area of each of the 100 slices k is $A(k)$. Which formula gives the volume of the head? (One applies.)

Formula	Check if true
$1.5[A(1) + \cdots + A(100)]$	
$\frac{1}{1.5}[A(1) + \cdots + A(100)]$	

Formula	Check if true
$1.5[\frac{1}{A(1)} + \cdots + \frac{1}{A(100)}]$	
$\frac{1.5}{100}[A(1) + \cdots + A(100)]$	

Figure 4. Math 1A Final Practice A Sample Problem

Sample Problem with Answer Blanks Shown (Knill, 2020)

Static problem sets serve as a great choice when an instructor needs to guarantee the student can attempt the problems without difficulties. Figure 4 shows an example of a static problem set. Students can do static problem sets without access to the LMS, the internet, or even electricity. They remain a stable fail-safe system from traditional education. Due to their simplicity and reliability, they remain very common even with other options.

However, some of the key criticisms of static problem sets come from the same characteristics that make them great. After a student receives the problem set, the problem set can go anywhere. The student can take a picture of it and upload it to a Discord server with their friends, they can upload it and it can become part of the

problem bank at websites such as Chegg[3] where professional tutors will solve it, or they can even take a picture of it where an app will turn the problem into Latex and search for the solution. Even prior to the widespread use of the internet, static assessments and their solutions could become part of a frat's test bank. Static problem sets struggle to adapt to modern technology.

1.3.2 Static Problem Sets with Variable Parameters

Static problem sets with variable parameters consist of shared problems between all students with different parameters for each student. These can consist of problems created by an instructor with specialized instructions, or problems coming from an online platform offering assessment. In the case of an instructor writing their own problems, a parameter may remain unset and later plugged in with a personal detail such as the last few numbers of a student identification number. This will sometimes automate grading as they can use the student's personal details in an excel file and compare the answers with the student's answers. Parameters often come from a small set of possible values giving an instructor the opportunity to verify automated grading operates correctly. Even with a small set of parameters for each problem, the likelihood of two students getting similar assessments remains very low with a modest number of parameters.

Students may exploit an assessment of this type. Given enough attempts and a small set of possible parameters, students can cycle through their values until they receive a problem containing parameters they desire. As some grading systems give the answers to problems after an incorrect attempt, a student just needs to continue incorrectly

[3] Chegg is an online education platform that specializes in providing solved homework problems. https://www.chegg.com

answering the problem until they get a desired set of parameters. The parameters they chose may come from a previous attempt that gave them the solution, or from another student's problem set that contains the answer. Also due to the need for the solution to come from a simple formula based on the parameters, a student may see the relationship and finish a problem though pattern recognition. Although these seem as far-fetched methods for completing assignments, students have used all of these methods in front of me.

As with the static problem set, students may complete the assignment by turning in their work on paper or submitting answers through an online form. As each problem contains a different answer, the need for an automated grading system becomes readily apparent. Also, as before with the static problem set, the students can receive feedback immediately after completing a problem, after completing all problems, or after their instructor reviews their assessment after submission.

An example of the static problem set with variable parameters would be a problem coming from the WeBWork[4] platform as shown in Figure 5.

[4] Webwork offers automated grading for the OpenStax textbooks. https://webwork.maa.org/moodle/

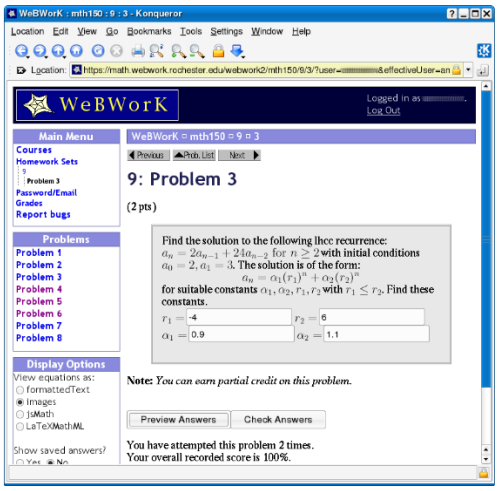


Figure 5. WeBWorK Problem

WeBWorK Problem with Answer Blanks and Variable Parameters (AlHaosului, 2007)

In this case students submitted their answers in the online form. An instructor may choose to give immediate feedback or wait until receiving all submissions. Immediate feedback improves the experience in formative assignments but may lead to academic dishonesty for test and summative assignments.

These problems promote a higher level of academic integrity. As the number of values of a parameter increases the likelihood of a student coming across their problem decreases. Even though these problems can dramatically reduce the prevalence of cheating they do require an automated grading system. For a problem of many parameters the grading system will need a function programmed to evaluate if the student submitted a correct answer. This will most likely require a software engineer to oversee. These types of problems cause the greatest number of problems in practice. For a small set of parameters some LMSs offer plugins that can check the student's answer against a table.

Problems of this type can create the most difficulty for a neural network. When the network predicts student performance a problem with many parameters can act like a family of problems. Some parameters may lead to a simple solution while other problems may greatly increase the difficulty.

For example, in mathematics the factoring of a polynomial can greatly increase in difficulty with the variation of a parameter. As a formative assesment, students first encounter factoring polynomials in Algebra I, and see it continuously as parts of problems until they finish Differential Equations. It appears as the solutions to an equation, the roots of a parabolic function, the denominator to a partial fraction integral, or the characteristic equation for a second order differential equation with constant coefficients. These potential uses of the problem span from middle school though undergraduate math courses.

Consider for example the second degree polynomial$y = x^2 + 5x + \beta$, for the parameter β. If the parameter takes the value 6, a student may choose to solve the problem by inspection and simply write the solution. If the value becomes a decimal, while the polynomial retains real roots, students may choose to use the quadratic equation. In the standard curriculum students will study math for two years between learning these methods of solution. Additionally, if the polynomial contains complex or irrational roots students may not be able to factor it during their entire time studying math in high school.

1.3.3 Dynamic Problem Sets

Dynamic problem sets consist of unique sets of problems for all students with identical parameters when repeated. These sets usually consist of problems coming from a problem bank with a predetermined solution. Students can get feedback after each problem, after submitting all of the problems, or after their instructor reviews their assignment.

Similar to static problem sets, students can usually find the problems in their assessment online or share them in online group chats. This becomes more difficult as the test bank increases. Individuals may also share their problems, but for a small group of individuals the intersection of problem sets reduces as the size of the problem library increases. If a summative assessment needs to reduce the uncertainty in difficulty between all the problems, such as a semester final, the assessment may use repeated problems sets.

1.3.4 Dynamic Problem Sets with Variable Parameters.

Dynamic problem sets with variable parameters essentially combine the two types of assessment considered previously. The problem sets will consist of unique problems coming from subsets of a problem bank. The individual problems each use different parameters for each instance. The grading mechanisms, difficulties in implementation, and benefits of usage, remain the same even when combined.

Assessments of this type reduce the possibility of students sharing solutions better than any other type discussed. When encountering problems of this type students will usually give up on answering questions dishonestly. They also create the most difficulty in ensuring the correct grading and require the most testing.

Even with the added difficulty of the delivering these problems, their potential for differentiation encourages their usage. The variable parameters can change the language, content, and presentation of a problem. In the domain of grammar exercises, NoRedInk exemplifies how to incorporate variable parameters to keep users studying.

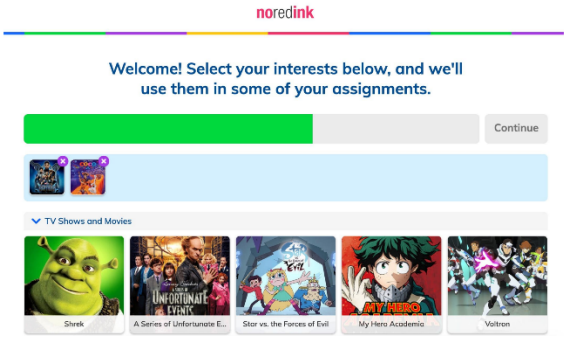


Figure 6. NoRedInk Prompting Students to Include Their Interests

*When Registering Users Get Asked a Series of Questions to Include Their Interests. (*How Do I Update My Interests? – NoRedInk Help Center*, n.d.)*

As shown above NoRedInk[5] does a great job of including student interests in assignments. During sign up a prompt asks students to give a list of their interests, and a list of friends' names. The students then receive grammar exercises with parameters modified specifically for their interests. Figure 6 shows the prompt a user encounters when they login to NoRedInk for the first time. After an introductory survey a database will associate variables with users to summarize their interest.

[5] NoRedInk is a grammar learning app. https://www.noredink.com/

Using predictive models offers an alternative to directly asking the user for information. This offers the benefit of requiring nothing additional from the user and starts improving recommendation immediately. Models can use IP addresses, networks connected to, and cross app data. Additionally, specific devices can give additional measurements to aid recommendation such as location data. Models can even use the proximity of other users throughout the day to determine common interests.

Outside of interests, these same measurements can also find what presentation of a curriculum a user will respond best to. Students may learn best with problems of the optimized length, difficulty, or type. These accommodations to learning style may even change between topics. Measuring the history of performance, and session measurements can address immediate and long-term needs.

Figure 7. Adaptive Junyi Problems

A Screenshot of The Junyi Academy LMS

Junyi Academy[6] does a great job of accommodating users by adjusting the difficulty of problems. Figure 7 shows how the Junyi Academy adjusts difficulty between groups of problems called exercises. Adjusting difficulty allows users to upgrade and get more difficult problems based on their performance. This addresses the need of students needing easier or more difficult problems.

This also appears to open the doors for models to make assessment adaptive to the individual. In addition to making problems with different difficulty levels, problems can alter language, and learning preferences, by introducing problem variations, and variable parameters. Just as Junyi Academy used variations in exercises between users, the modification in language used by Web Assign can also accommodate learners.

1.3.5 Hints, References, Examples, and Video

Along with the question, problems may contain links or references to additional resources. The hints and references may contain the topics used to finish the problem, or references to textbook sections where a student may review additional material. Links to examples may also exist that link to other parts of the learning platform or may toggle open additional parts of the GUI. Additionally, links may take the student to a video which may exist on a third-party platform.

Hints quickly guide the student and give them a level of control over the help they receive. When students try to quickly solve problems a great hint can encourage impatient students. Hints also create the front line of resources that will stop students from using websites with their homework solutions. They integrate seamlessly into

[6] Junyi Academy is a Taiwanese online education platform. https://www.junyiacademy.org/

learning platforms running on web browsers. They can open as small modifications in the website using JavaScript Events or React.

Additionally, they may come from a hover feature. When used properly a hover feature can complement a problem. They do not take up space, convey information quickly, and implement intuitive usage. Despite their benefits, hover features can also inadvertently open, and students lose the control to work independently. Hover features also hide in plain sight. A user will not know a hover feature exists until they encounter it, but after they find it, they can quickly integrate its usage.

Like hover features, modals also require the right situation. Modals[7] give a direct message that the user must address. This makes modals great for important messages like error messages. They guarantee the user addresses the message, but they also contain drawbacks. Due to their usage in giving error messages, users will negatively associate with modals.

When students learn several methods to solve similar problems, these references can guide the student when they plan a method of solution. Guiding the student becomes extra important when only certain methods will solve certain problems. Reaching a dead end while following instruction can lead to frustration and a reduction of motivation in a student.

Examples show a concrete problem a student may review when struggling with a formative assessment. They can appear as links, modals, or toggle open on an event. They often show the same problem with altered parameters. Videos may also accompany

[7] Modals come from the Bootstrap framework and appear like a popup. https://getbootstrap.com/docs/4.0/components/modal

problems with instruction. This happens more often when the learning platform offers a complete course.

1.4 Previous Work on Junyi Datasets

Junyi Academy provides large datasets of student performance on exercises. Their dataset includes measures of student demographics, problem details, and records of the student responses to problems. The datasets provide not only the basics required for recommendation systems, such as timestamp, user, and rating, but also includes additional measurements.

These datasets provide the basis for researchers in student modeling. They give the resources for training baselines, as well as the additional measurements that can develop new models. As this research uses the Junyi20 dataset it depends on the previous work done.

1.4.1 Should Neural Networks be Used?

Consumers now encounter neural networks daily. Their incredible performance leads many to believe in the coming of a broad restructuring of society. Even though almost all users have encountered an AI intelligence unimaginable just a decade ago, almost all users have also encountered an AI making comically absurd mistakes. They can hoard resources and underperform. Even in well researched problems, such as the self-driving car, creating a sufficient neural network can be difficult. Elon Musk is in his 10th year of promising a self-car. (*Elon Musk Promises for the Millionth Time That Tesla Will Achieve Full Self-Driving Soon We've Definitely Heard This One Before.*, n.d.).

When allocating the resources necessary for a neural network, a developer should consider if a simpler algorithm would perform better.

Even when a situation calls for neural networks, the question of the complexity needed still stands. RNNs should not immediately replace simple and efficient networks. Even the multi-layer perceptron can show success in modeling complex human behavior (Armstrong, 2017).

Due to the difficulty in designing, maintaining, and training complex neural networks, they look like they are overkill when machine learning (ML) techniques will do. This makes ML techniques great for education applications. Specifically, researchers frequently apply ML techniques to the Junyi datasets.

1.4.2 Usage of ML Techniques on the Junyi Datasets

ML shows a great ability to categorize problems in the Junyi Datasets. Researchers demonstrated this ability by using ML to create pairs of problems. Random forest, nu-SRV, and linear regression all matched problems pairs for exercises in a Junyi dataset. The pairs between problems focused on similarity, difficulty, and perquisite knowledge. It ran against human making pairs, and showed much improved performance (Tang et al., 2016b).

Researchers also found success with ML by breaking up data and including additional complexity to the model. Logistic Regression based models predicting student performance very well on the Junyi15 dataset. The model improved prediction without requiring additional data measurements by adding additional features. It also took advantage of the Junyi measurements coming from its LMS. The datasets record if the student watched the videos related to the question before responding. Including the if the

student watched the video improved performance. Additionally, the model trained on different moments in time after the student began the curriculum. This allowed the model to adapt to the students as they learn. (Schmucker et al., 2022).

1.4.3 Usage of Neural Techniques on the Junyi Datasets

To further model how students change as they learn, other researchers adapted an LSTM on the Junyi data. Two specific mechanisms control the learning and forgetting gates, which they call Gating-controlled Forgetting and Learning mechanisms for Deep Knowledge Tracing (GFLDKT) (Zhao et al., 2023). Other researchers investigated using Graph Neural Networks (GNN) on the Junyi15 dataset. In these methods the network makes nodes and edges to model student behavior (Ni et al., 2023). Both models showed improved performance compared to their predecessors. Furthermore, they provide evidence that neural networks can provide improvements on the Junyi datasets.

1.5 What Direction Does the Research Tell us to Go?

The ML techniques on the Junyi data sets showed student prediction changes as they go through the course. In addition, the Junyi datasets give additional measurements to improve prediction. The ML techniques also showed success categorizing problems by similarity, difficulty, and prerequisite knowledge. LSTM techniques showed an ability to update without segmenting the data. GNN showed a necessity to update the network topology to optimize prediction. Combining these three methods would require a neural network that changes as students learn and changes its structure to model students.

1.5.1 The Case for Cross-Network LSTMs

When trying to find a method that balances all the techniques cross-network LSTMs on the Junyi dataset seem up to the task. They can ingest many parallel data networks at once. The LSTM basis allows the learning and forgetting gates that modify as students learn more. The Junyi dataset gives multiple measurements that can lead to the development of additional mechanisms that improve prediction.

1.5.2 The Perera and Zimmermann Network

With a few modifications the Perera and Zimmermann Network makes for a great potential network for the Junyi dataset. It can directly take the data from the dataset and make predictions. It only requires a data point consisting of a user, an object ranked, and a score assigned to it.

The Perera and Zimmermann Network also contains an attention network which looks at similarity between ranked objects. A method is given that uses the cosine similarity between objects, but the authors also leave open the possibility to replace the cosine similarity with another method. As early work using ML on the Junyi datasets showed an ability to find similar problems, it looks like the similarity could use a categorization of like objects.

Chapter II.

Network Description

Is someone aware of a language model experiment where you keep all the 2022 goodies/data, except swap a Transformer for an LSTM?

—Andrej Karpathy Founding Member of OpenAI

The goal of this network is to create something that will adapt to different datasets coming from many applications in math education. Although a network can always include additional complexity to meet the specific details of a single dataset, in practice this does not lead to a more robust network. The design of this network will avoid adding bells and whistles to make short term improvements in prediction.

The primary way to ensure the network will make predictions beyond the scope of the data in this research comes from the choice of network. The network chosen here already shows proficiency in making predictions about user interactions on social media (Perera & Zimmermann, 2020). Adapting a proven network assures the network works in one instance outside of its current application and improves its probability of success.

Adapting the network will require some new techniques not used in the original implementation of the network. The primary modification from the original network will come from the additional data sources used. In the original paper the second data source came from an additional social media platform and measured a user's interactions with media. In the model proposed here, the second measurement will come from session, and database measurements made on the individual. The attention mechanism will also need

modification as the library of problems used in the dataset shows little overlap. To modify the attention mechanism, this network categorizes problems by topics, so the network may recognize similarity in sessions. Finally, as the scores in the networks represent categorical data, their values will reference a one-dimensional embedding to get scores.

The second way this network will lend itself to more general usage comes from the method of fitting the hyper-parameters of the model. The fitting of the hyper-parameters in this model will not try to prove some artificial point. The hyper-parameters will remain uniform thoughout the experiments, and the observed performance differences will come from modifications to the network.

Finally, the design of this network will emphasis on not requiring any rare, exotic, or expensive resources. The software and hardware discussed in this section drop these requirements by being widely available, and widely applicable. The software comes as open-source, and anyone can readily download it. The network was designed around hardware that some might consider e-waste but maintains a comparable performance with more expensive options. The network can also run on available virtual resources for quick start up and tear down. The combined effect of these considerations makes this network available to the masses.

2.1 Why Use an LSTM?

A quick search on the current use of LSTMs returns the debate over the relevance of LSTMs. Half of the articles state “LSTMs are Dead! Long live Transformers”, and the other half respond “No, LSTMs are not Dead!” For the purpose of this research LSTMs are a tool with advantages and disadvantages.

2.1.1 Combining Data Networks

The primary reason for using an LSTM is the ability to quickly combine data from different sources. In the usage case of math education, in addition to the problem, and the user, metadata such as the session length, time of day, and user background would give additional improvements to prediction. Cross-Network LSTMs combine data sources upon ingestion and offer a practical solution.

2.1.2 Long-Term, Short-Term Characteristics

In mathematics education, the performance of students depends on many factors. Some of these factors will depend on the student's background, previous experience in mathematics, natural ability, and intuition. These factors would influence a student's performance independently of recent study sessions and show a longer lasting overall influence. Other factors such as how long a student studied a certain topic, what unit they are on, and current responsibilities, would also influence performance, but these effects would appear more transient.

The Carnegie Mellon research group mentioned in the introduction noticed the importance of transient effects on performance. They trained separate models focusing on different time frames of the student and combined the results. Their models focused on students at the beginning of the curriculum and towards the end. The combined model recorded a very high AUC, but it required several models. Ideally one model should adapt to the students as they grow.

An LSTM appears as a natural solution, with unique qualities to help prediction. The weights can update after each step and move the embeddings of the students. The model used here updates once a day and would slowly change with the student. Instead of

breaking a student's data into separate sections and training the model, the data would get broken up on a daily basis.

2.1.3 Low Computational Overhead

LSTMs also give sequential predictions. Running through the daily data, adjusting the weights, and updating the embeddings, happens in a couple milliseconds under the hardware used in this experiment. After updating the network, the prediction comes out equally fast, and for certain metadata the prediction can happen once per day.

With the hardware used here, a class of 20 students with a time step each day, would require 5-10 seconds of computation throughout the entire year. This means that the hardware here could take on several districts of students and adequately meet their needs. In fact, the more resource intensive aspects would come from the delivery of the assessments to the student through an LMS.

2.2 Hardware

One of the primary concerns in the design of this network focused on building the network around the capabilities of the GPU. A network designed with this concern can achieve great performance without incurring a cost that would prohibit the use of the network by educational institutions. For this reason, using hardware and software to unlock the potential of the GPU will give an opportunity for wide usage with great efficacy.

2.2.1 GPU

The realistic time scale of latency in the updating of the weights, and the requirements of this the current implementation require a GPU. PyTorch readily incorporates CUDA for NVIDIA[8] GPUs through a specific installation of PyTorch for CUDA. Although this method quickly solves latency issues, it also depends on the usage of proprietary software. This could potentially limit the usage of similar methods in an academic setting, but at the time of this writing NVIDIA GPUs are plentiful and cheap. Additionally, Amazon Web Services offers EC2[9] instances that launch with PyTorch and CUDA. Currently virtual resources remain outside of the budget for most public schools with a g4dn EC2 instance coming at $0.63 per hour(*Add Service - AWS Pricing Calculator*, n.d.), but as better resources become available their cost should reduce. Although the future gives no guarantee of the availability of these resources, their absence would imply the development of better and more affordable alternatives.

This experiment used a NVIDIA GeForce GTX 1660 Ti. It is a common and affordable GPU, with 2.02% market share of Steam users (*Steam Hardware & Software Survey*, n.d.). It is a GPU commonly used in laptops for its low power consumption, high performance, and reasonable cost. This makes it an excellent prospect for investigating a network that a school would consider adopting.

In this experiment the GPU performed adequately with the ability to store the network, the problem embeddings, and the user embeddings. This data set in particular used 72 thousand students, 1.6 million data measurements, and 42 thousand problems.

[8] NVIDIA is a manufacturer of hardware. https://www.nvidia.com/
[9] EC2 or elastic computing instances form the backbone of Amazon's cloud-computing platform. https://docs.aws.amazon.com/ec2/

Outside of coding errors the GPU usage remained less than half of its maximum. These parameters imply a modest GPU could handle realistic data set in an actual application.

Figure 8. GTX 1660 ti GPU Externally and Inside of Thermal Piping

A Picture of the GPU Outside of the Laptop and the Under the Thermal Piping

The GPU and CPU come together and do not need any additional assembly. Figure 8 shows them combined into a singular laptop. This will ensure fewer issues with setting up the hardware and makes the network easier to adapt to new implementations. As this research focuses on not using exotic hardware, using a laptop that combines everything fits into the spirit of this research.

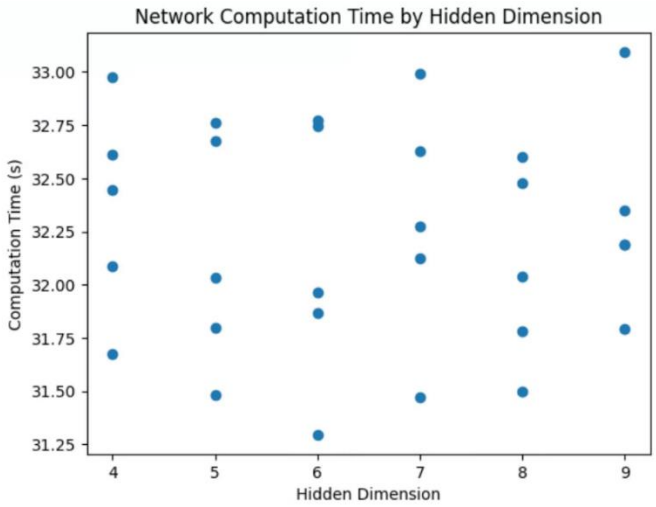


Figure 9. Time to Run Five Mini-Batches with Different Network Size

Time Independent Graph for the Runtime of the GPU

When running the network with different size embeddings, the time to run the network five times appears independent of component size. Running the network requires several matrix multiplications. As the network runs with several matrix multiplications the theoretical limit for the time complexity of the network should approach the time complexity of matrix multiplication, $O(n^{2.37})$. The miracle of the GPU spreads the calculations amongst many threads, and as shown clearly from the graph makes the runtime independent of the embedding size. Figure 9 shows no noticeable difference as the dimension of the embedding increases. A time complexity of $O(1)$ is the theoretical best result achievable.

2.2.2 CPU

As the implementation studied relies on CUDA, the CPU requirements remain minimal. The network and methods used were also coded with the intent of reducing the CPU dependency. The network could run on a CPU and did during testing, but the reduction of performance remained too great for practical use. This experiment used an Intel Core it-10705h @ 2.60 GHz. It is a low power CPU for laptops with six cores, and twelve threads. When paired with the GPU it performed adequately for the loading of large tensors in chunks, organizing forward passes, and storing data summaries.

2.2.3 Performance

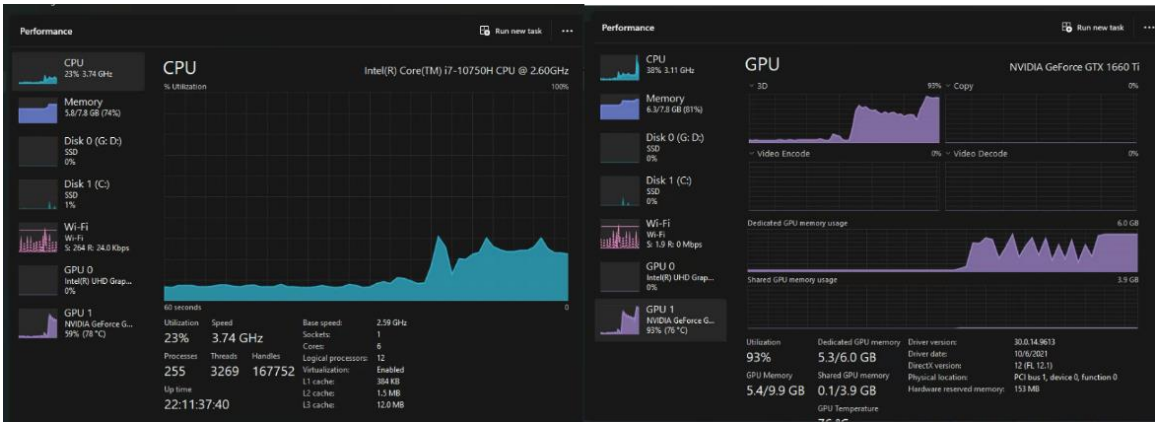


Figure 10. Usage of the CPU and GPU While Running the Network

Screenshot of the CPU and GPU Monitoring

When running the network on the GPU, the CPU appears to handle the load adequately. The GPU does exhaust its memory while running the network calculations

and will revert to a scheduler when this happens. Figure 10 shows the load on the CPU and GPU. The parallel performance of the GPU threads performed better than the CPU even in this case, and handle the load in a reasonable amount of time.

2.3 Network Model

The network used in these experiments will closely follow the work done by Perera and Zimmermann. Their network consisted of a LSTM that combined data sources and made predictions on YouTube videos. In the development of their network, they considered two problems with LSTMs. First LSTMs struggle to model non-linear behavior, and secondly, they do not update after the accumulation of additional user data. To address these issues, they suggested three additions to the standard LSTM. The first addition is an attention network that includes all the user's interactions and models the long-term user interests. The second addition gives a higher order combination of the data prior to entering the LSTM. These two alterations make quite interesting improvements when applied to mathematics education.

2.3.1 Context of the Network

The original network combined data sources to make predictions about YouTube videos. The network looked to predict if a user will like a YouTube video or add it to a playlist. The data sources they combined consisted of Tweets and posts on Google Plus. These datasets consisted of very sparse data, as users will not frequently interact with content they encounter on social media. The users averaged 26 interactions each. The timestep varied from one second to seven months. This could imply timesteps included as

little as one datapoint, but no information exists on the number of data points in a timestep.

2.3.2 Non-Linear Behavior

Prior to entering the LSTM cell, the data from this network gets combined through component-wise multiplication of *n*-dimensional arrays. If the data was just added together and put into the cell, a repeated measurement would amplify with the number of times it appears. This would send a larger input into the LSTM. Including higher order combinations of data allows the network to mitigate the increasing amplitude.

In a linear network, if a student attempted a problem multiple times, the signal would get bigger with each attempt. An array of *n* entries would represent an embedding for each user and problem in an *n*-dimensional space. These arrays would enter the LSTM directly. If a student attempts problems multiple times in a row it may not make sense to amplify the signal. Doing a problem multiple times may mean the student does not understand the material and the data network should not amplify the signal getting sent to the LSTM cell.

Adding the second order interactions removes the necessity to amplify the signal. If a student attempts a problem multiple times, the second order addition allows the network increase or reduce the signal going into the LSTM.

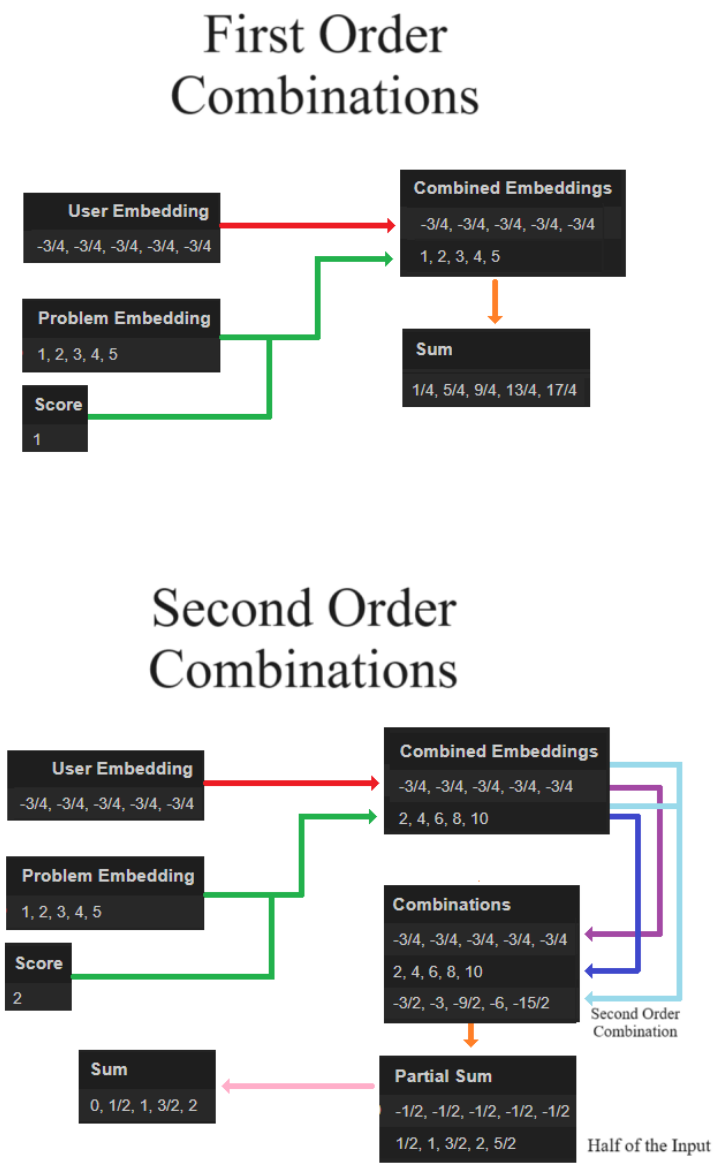


Figure 11. Usage of the CPU and GPU While Running the Network

Example of a Second Order Combination Reducing the Effect of a Problem Embedding

The example in Figure 11 shows a reduction of the signal due to the second order combination. For the specific problem, and user embedding above, doubling the score of the problem embedding reduces the magnitude of the same embedding. In the intermediate sum you can see the overall contribution from the problem embedding is half as large as in the first order combination.

2.3.3 Attention Mechanism

In a standard LSTM the hidden state must contain all the information of user's history. This network also uses a hidden state, but it also uses an additional cell and hidden state. This hidden state is calculated by using the previous hidden states, and weights them by their similarities to the interactions in the last time step. This additional, hidden state and cell state form the attention mechanism.

This addition will also benefit the network given the context of the application. Embeddings will give the characteristics of the users and the problems. How a user performs in the last week, semester, or unit will come from the attention mechanism. In the original context the attention mechanism looked for similar interactions by looking at if a user interacted with the exact same objects during a timestep. Modifications to the attention mechanism allow the network to look for if the user completes problems from the same topic.

2.3.4 Irregular Timesteps

Perera and Zimmermann added an additional modification to their model. They added the ability of the network to handle irregular timesteps. The third addition to the LSTM does not affect performance in this context. Although this network can handle

irregular timesteps they should not occur. The timesteps should consist of uniform lengths between periodic low demand hours of the day.

The problem this network addresses by taking in irregular timesteps though, should improve the performance of the network. While the network adds additional students, it will need to update the embeddings of the users and the problems. Allowing for irregular timesteps would let the network pick up students at different stages in the course.

2.4 Network Architecture

The network consists of a cross network LSTM. Previously this network used data coming from different sources and combined them (Perera & Zimmermann, 2020). In this network treats different measurements as data coming from different data sources. It does this by creating a specific embedding for each measured quantity. Incorporating different measurements in this method creates an increased performance without additional measurements.

2.4.1 Layers in Perera and Zimmermann

The network from Perera and Zimmermann consists of a data layer, cross-network topical layer, embedding layer, higher order interaction layer, and the prediction layer. I highly recommend looking at their diagrams and description of the network. The data layer consists of an array representing the user and raw data sources. The user array is a sparse array with the same dimension as the number of users. A single unitary entry representing the user. The cross-network layers consist of sparse arrays for each data source with entries corresponding the score a user gave to items. The embedding layer

gets the embeddings for the user and items. The network also uses the scores from the topical layer to amplify the embeddings in the embedding layer. The higher order interaction layer makes element-wise products of the combinations of all items. After summing these products, the result enters the LSTM which forms the prediction layer.

Most of these layers appear in the coding except for the cross-network topical layer. The cross-network layer uses the binary interaction vectors for each network and the user. As the interaction vectors only serve to supply indices to the embeddings, that layer does not exist in the current coding of the network. The data stores items as unique integers and uses those integers to get the embeddings. As the interaction vectors use memory in the GPU, their omission improves efficiency. However, they should remain in the diagram of network to improve the clarity of how the network performs forward passes.

The higher order interaction network requires the most computation prior to the data entering the prediction layer. It unintentionally acts as a gate keeper and will overwhelm the GPU if a mini-batch becomes too large. Due to its importance, it stands out as a modular component.

2.4.2 Embeddings of the Network

Embeddings addresses the problem of sparsity in the vectors. When examining an interaction vector composed of binary values, where ones indicate user interactions with the *n*th item, most of the vector will comprise of zeros. Given the dataset used for this network consisted of thousands of problems, a student will complete a small proportion of problems in any timestep. Instead of using these vectors directly the non-zero entries act as indices for embeddings.

During a forward pass the raw data comes into the embedding layer as integers referring to the student, and the problem. The data then enters the LSTM though the student and problem *n*-dimensional embeddings. The network can treat the dimension of the embedding as a hyperparameter. As the network consumes data and adjusts its weights, similar problems and students form clusters in the embeddings.

Given the randomized instantiation of the embeddings it is unlikely that any dimension will correspond to a recognizable characteristic for the student or problems. Like problems and students will group together for reasons in the data. In future work with a wider spectrum of problems the embedding could separate problems by student interest, learning styles, or assessment style.

Although embeddings make it difficult to find meaning in a particular dimension, this network initially ordered the problems on the unit circle in the first two dimensions. As problems came later in the course, their initial embedding fell further counterclockwise on the unit circle. Even though initializing the embeddings required data from the course, a problem's topic could give an estimate of when students should attempt it. This initial embedding allows the network to identify the topics of the problems quickly.

The implementation of this network holds the embeddings in a class object. The weights come from a parameter in the object used to reference the tensor. For *K* embeddings with dimension *k* the embeddings at a time step *t* appear as $E^{\mathrm{k},\,t} = \{{e_1}^{k,t}, {e_2}^{k,t}, \dots, {e_K}^{k,t}\} \in \mathbb{R}^{\mathrm{k}\times\mathrm{K}}$ where ${e_j}^{k,t} \in \mathbb{R}^k$. The problem and user embeddings will share the dimension *k* but differ in the number of total embeddings.

2.4.3 Additional Embeddings

For each additional network every problem gets an additional embedding. Although the embeddings could differ in dimensions the higher order combination of the embeddings require the same dimensions. For an additional number of networks *m*, the embeddings appear as $E^{\mathrm{k},\,t} = \{{e_1}^{k,t}, {e_2}^{k,t}, \ldots, {e_{mK}}^{k,t}\} \in \mathbb{R}^{\mathrm{k}\times\mathrm{mK}}$.

2.4.4 Higher Order Interaction Layer

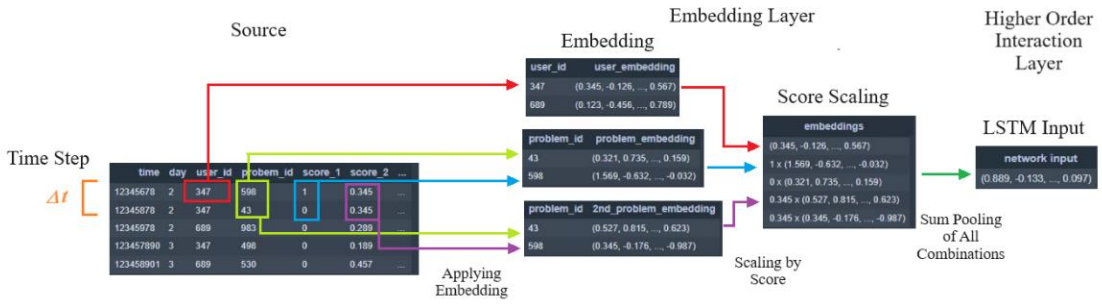


Figure 12. Combining Embeddings for LSTM Input

Diagram of How the Data Reaches the LSTM

Putting the embeddings into the network requires a final method to combine the data into a single array of length *k*. Although necessary, combining the data also allows the opportunity to model higher order behavior in the embeddings. The original authors of this network took inspiration from Factorization Machines (FM) for the idea of combining the data this way.

Combining the embeddings requires three steps. As shown in Figure 12 the first step in the network gathers all the embeddings related to the users' interactions for a time step. In an application in education, a time step of a day would allow the network to update with an appropriate latency to give students relevant daily problems. An embedding for a problem would appear in each network. An additional embedding comes from the user.

The second step involves combining the embeddings into a new set of arrays with the same length as the embeddings. The set includes all the embeddings individually. Then all combinations of two embeddings get put into the set. To put the combinations into the set, two embeddings get multiplied with the element-wise product resulting in one array that can fit inside the set. As combinations from a set do not include an element chosen twice, the new set does not include the elementwise product of arrays with themselves.

The third step simply sums the elements in the last set by component. The result gives an array with the same length as the embeddings and includes second order interactions. This array enters the LSTM cell with the same dimension as the embedding dimension.

Considering the first step formally, given *n* interactions a user makes with the network in a given a periodic time step Δt, the embeddings will give a tensor filled with embedding $E^{\alpha,t} \in \mathbb{R}^{n\times k}$. For each additional network their exists an additional embedding $E^{\beta,t} \in \mathbb{R}^{n\times k}$, $E^{\gamma,t} \in \mathbb{R}^{n\times k}$, … which will each contribute to the final array. In addition, the embeddings will include the user embedding $E^{u,t} \in \mathbb{R}^{k}$. Together all the embeddings combine to give $E^{t} = \left(\mathrm{E}^{\mathrm{u,t}} \middle| \mathrm{E}^{\alpha,\mathrm{t}} \middle| \mathrm{E}^{\beta,\mathrm{t}} \middle| \ldots\right)$.

Prior to combing the embeddings, each problem embedding gets multiplied by a score. For the problem embeddings, the score consisted of a one if the student got the problem right and continued, and a zero otherwise. The score used by the RS which inspired this network consisted of scores of zero to five stars. The design gives the network the ability to add additional granularity for scores. In the implementation of this network other options exist for scores, such as giving one point for completing a problem and two points for completing the problem and continuing.

This network operated on the assumption that problems and users acted as the primary contributions to the probability for the student to get a problem correct and continue. To reduce the dependency on the additional networks, the scores get reduced by an order of magnitude. For the raw hour of the day the scores,$h \in \mathbb{R}^k$, $h_{max} = 23$, and $h_{min} = 0$. When plugged directly into the network these large scores overwhelm the other contributors to the final probabilities. To lessen the effect of the hour scores, the scores get put through an additional one-dimensional embedding. Each value becomes an index for the embedding. The values in the embedding start as decimals less than one, so using them as scores reduces the behavior caused by the initial large values. Other scores, such as the minutes studying *m*, show a larger number of embedding indices that would get lost due to the variety of data encountered. For cases when the score values become large, this network chose to categorize them by bin. For the minutes studying, the scores came from bins of five-minute intervals.

The inclusion of an array of all ones simplifies the second step, as the summing step needs to include the individual embeddings. Let

$E^{*t} = \left(\mathrm{E}^{\mathrm{u,t}} \middle| \mathrm{S}_{\alpha} \odot \mathrm{E}^{\alpha,\mathrm{t}} \middle| \mathrm{S}_{\beta} \odot \mathrm{E}^{\beta,\mathrm{t}} \middle| \ldots \middle| \mathbb{1}\right)$ include the additional array with dimensions $(nm + 2, k)$ for *m* networks, where S_{σ} is a tensor composed of rows with repeated values corresponding to embeddings, and $\odot$ is the element-wise product. Combing two arrays give the number of combinations as $(nm + 2)(nm + 1)/2$. Call the list of all combinations of indices $\mathrm{c} = [(0,1),(0,2),\ldots] \in \mathbb{N}_0^{(nm+2)(nm+1)/2\times 2}$. Then the list of all combination becomes $C^t{}_j = E^{*t}{}_{c_{j,0}} \odot E^{*t}{}_{c_{j,1}}$.

The result of the previous two processes gives the input into the LSTM as

$$\mathrm{u}^{\mathrm{t}}{}_{\mathrm{j}} = \sum_{i=0}^{(mn+2)(mn+1)-1} \mathrm{E}^{*\mathrm{t}}{}_{\mathrm{c_{i,0}}} \odot \mathrm{E}^{*\mathrm{t}}{}_{\mathrm{c_{i,1}}}.$$

2.4.5 Attention Mechanism

A standard LSTM contains a hidden state that gets passed back into the LSTM before a forward pass. The LSTM for this experiment uses an additional hidden state that weighs previous hidden states on their similarity to the previous days. After calculating the similarity between the daily interactions, the previous hidden states get aggregated with similar hidden states gaining greater importance. The similarity comes from Softmax applied to the cosine similarity between the days. The arrays for each day come from sum of the hot coded arrays based on the topics of the problems. The topics for the problems exist as points in the problem embedding and a problem's topic depends what topic embedding the problem lies closest to. As the number of topics could increase arbitrarily the topics could represent, difficulty, unit, semester, interest, or learning style. The only issue with increasing the number of topics comes from the increased likelihood of no similarities between time step aggregations.

The cosine similarity of hot-encoded arrays falls between zero and one. After summing the hot-encoded arrays the cosine similarity between the resultant also remains between zero and one. This comes from the encoding that makes every component positive. After putting the Cosine Similarity in Softmax the greatest weight will differ from the smallest by a factor of *e*.

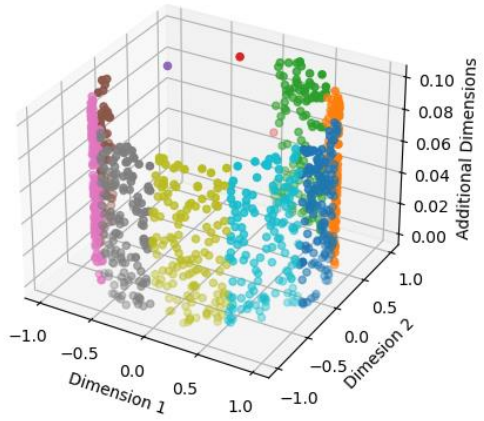


Figure 13. Initialization of Problem Embeddings and Their Topics

Showing the Initialization of Problem Embeddings to Ensure Ordered Topics

Basing the topic of problems on their location in the embedding makes fitting the topic embeddings difficult. To facilitate the fitting of embeddings, the topic are positioned on the unit circle, aligning with the first two dimensions of their embedding, and their angle is determined by the day of their initial appearance. Figure 13 shows the initial embedding of the problems in two dimensions. Assuming students learn with progression of units their order of first appearance would give an approximation of their topic. If the topics of the problems are already known, their assumed topic may place

them on the unit circle. The remaining dimensions of the embeddings allow for the fitting of weights to determine additional characteristics.

Consider a list of topics for a particular day $T_i = [t_0, t_1, \dots]$ where $t_j \in N_0$. Each topic will contain a positive integer as a label for certain problems. The hot encoding of a topic will appear as a sparse array. For a hot encoding function *h* with a domain of *k* topics, $h(t_j) = [0,0,\dots,1,\dots,0] \in R^k$. The sum of all hot encoding becomes the aggregation for the interactions in a particular time step as $d = \sum_j h(t_j)$.

The cosine similarity for two aggregations d_i, and d_j uses the *n*-dimensional dot-product to find the cosine of the angle between the two arrays, as $\mathrm{sim}(d_i, d_j) = \frac{d_i \cdot d_j}{|d_i||d_j|}$. Notice the geometric interpretation of the cosine similarity as the dot product of normalized vectors. For days without interactions the addition of a small parameter in the denominator avoids interruptions of computation $\mathrm{sim}(d_i, d_j) = \frac{d_i \cdot d_j}{|d_i||d_j|+\delta}$. Softmax retains its standard definition as the Boltzmann Distribution, $\sigma(z_i) = z_i / \sum_j \exp(z_j)$, where the states would include all previous states. The combination of these two functions gives the weight for a hidden state as $\alpha_j = \sigma\left(\mathrm{sim}(d_{last}, d_j)\right)$.

Given previous hidden states as h_j, the element-wise sum of the of the hidden states and the weights gives the attention array as a sum of the previous states $h_A^{t-1} = \sum_s \alpha_s h_s$.

2.4.6 LSTM Cell

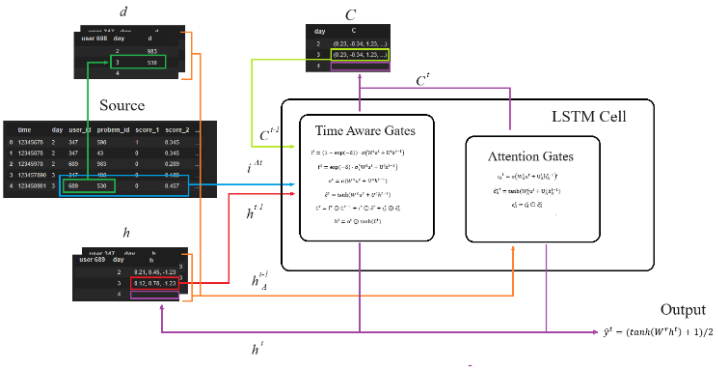


Figure 14. LSTM Cell

Diagram of the Operation of the LSTM Cell

The long term, short term, memory aspects of the LSTM make it a great candidate to model the performance of students. The hidden state holds the information about short term interactions, and in this application, it would hold information about a student's performance with their current material. The cell would contain weights for long term memory. In this case the weight would hold a student's learning style, demographics, and interests. Figure 14 summarizes the interactions of the network, but I would highly recommend viewing their original description of the Perera and Zimmerman network. Their presentation does a phenomenal job explaining the operation of the network.

After creating the inputs for the current state of the LSTM, the forward pass consists of a series of linear layers, and activation functions. Let σ be the sigmoid activation function. Then for the activation branch of the forward pass,

$$\mathrm{i_A}^{\mathrm{t}} = \sigma\left(\mathrm{W_A^i u^t + U_A^i h_A^{t-1}}\right)$$

$$\widetilde{\mathrm{c_A}}^{\mathrm{t}} = \tanh(\mathrm{W_A^c u^t + U_A^c h_A^{t-1}})$$

$$c_A^t = i_A^t \odot \tilde{c}_A^t$$

where $\mathrm{W_A^i, W_A^c} \in \mathbb{R}^{\mathrm{k\times k}}$ are linear layers without biases, $\mathrm{U_A^i, U_A^c} \in \mathbb{R}^{\mathrm{k\times k}}$ are linear layers with biases, and *k* denotes the dimensions of the embeddings.

The most recent leg of the of the of the LSTM gives the forward pass as,

$$\mathrm{i^t} = (1 - \exp(-\delta)) \cdot \sigma\left(\mathrm{W^i u^t + U^i h^{t-1}}\right)$$

$$\mathrm{f^t} = \exp(-\delta) \cdot \sigma\left(\mathrm{W^f u^t + U^f h^{t-1}}\right)$$

$$o^t = \sigma(W^o u^t + U^o h^{t-1})$$

$$\tilde{c}^t = tanh(W^c u^t + U^c h^{t-1})$$

$$\mathrm{C^t} = \mathrm{f^t} \odot \mathrm{C}^{t-1} + i^t \odot \tilde{c}^t + i_A^t \odot \tilde{c}_A^t$$

$$\mathrm{h^t} = \mathrm{o^t} \odot \tanh(\mathrm{C^t})$$

where $W^f, W^o, W^c \in \mathrm{R}^{h\times k}$ are linear layers without bias, $U^f, U^o, U^c \in \mathrm{R}^{h\times k}$ are linear layers with bias, and *k* and δ are hyperparameters.

In the Perera and Zimmerman network δ measured the length of the timestep for irregular timesteps. In a network with uniform timesteps δ takes on different meaning. It serves as a factor to reduce the importance of old data. It could however retain its original purpose if the network considered the days between student interactions. This network will not implement the original intent of δ, but this could form a basis for future improvements.

The biases also take on different meaning given the context of the network. Biases in the linear layers maintain group statistics in the final prediction. If they remain without dropout during training, the network will minimize the effects of the embeddings and use the biases as the primary effect of the linear layers. Leaving the biases in will improve performance but reduce the effects of the user and additional data sources.

The final output for the network comes from

$$\hat{y}^t = \sigma(W^r h^t)$$

where $W^r \in \mathbb{R}^{m \times h}$ is a linear layer with bias, and *m* predictions for the probability of getting the problem correct.

2.5 Software

Just as the hardware chosen can make the use of the network outside of research impractical, software shares an equal importance. The theoretical mapping between gradients, implementation of a GPU, and updating of weights, require a high level of expertise under the wrong implementation. The choice of software can limit the complexity of these problems by automating tasks.

This research chose software to prioritize the practical usage of this network. The software should be open source, capable of automating complexity, and easily used. Although TensorFlow, Keras, and MXNet could conduct this research, PyTorch showed the best implementation with remote resources, and the momentum in the industry.

2.5.1 PyTorch

PyTorch (Torch) [10] comes with robust libraries to handle the implementation of networks, fitting of weights, and usage of GPUs. It comes as open source, and it's widely used. It also holds the majority of market share for RS research, and it installs quickly in a variety of implementations. This makes it ideal for this application.

2.5.2 Tensors and Gradients

Torch uses Tensors as the primary object for holding data. Tensors exist as multi-dimensional arrays with size and shape determined upon initialization. Weights and embeddings will exist with this structure to hold values in their classes. Tensors exist on hardware, and tensors existing on different hardware can only interact in limited ways. Tensors on CPUs do not allow for basic linear algebra operations with tensors on GPUs, but they can pass as arguments for indices of GPUs. Tensors on GPUs use methods that parallelize on many threads and show excellent performance. The data stored in the tensors can consist of several types with binary representations, but Tensors cannot contain common data types such as string or char.

Tensors can contain a secondary data structure containing gradients for each value. Upon instantiation a passed parameter will determine if the tensor should maintain static values during network updates, or values that update with the network. The gradients of Tensors do the primary work when updating the weights and embeddings through the back pass.

[10] Pytorch is a Python module for incorporating ML and GPUs https://pytorch.org

2.5.3 Autograd Engine

Updating weights between forward passes relies on navigating a jungle of partial derivatives. Diagraming the network and creating the relationships between the gradients becomes an insurmountable task as the complexity of the network increases. Testing the effect of small alterations leads to hours of algebra to update the relationships between the gradients.

One of the huge upsides of using Torch is the availability of the Autograd Engine (*Overview of PyTorch Autograd Engine | PyTorch*, n.d.). Given a target, an input and a model, the Autograd Engine can update the weights in a model for most processes consisting of linear algebra, and neural network operations. The requirements for deriving the equations to update weights disappears and allows for quick changes to the network while in the developmental stages.

The process works by automating the chain rule from multi-variable calculus. The input and output for components come with gradients associated with their values. For the components PyTorch determines the relationships between the gradient of the input and the output. As the data gets passed from one component to another, a map holds the relationships between the inputs and the outputs.

Pytorch handles most RNNs well with the exceptions of LSTMs. The hidden state from a previous forward pass will link the next forward pass to the last one. This requires Pytorch to hold on to the mappings for all passes. Old mappings usually get discarded once a forward and backward pass finish, so keeping old mappings puts an additional strain on the hardware requirements. While running the hardware will show the effects of the old mappings by reducing the speed of forward and backward passes. To prevent this

feedback loop, the class LSTM in the nn module implements LSTMs and solves the problem behind the scenes. As this class does not allow for the addition of the attention mechanism, the network will need to use the other solution to the problem, detaching the hidden and cell state between forward passes.

2.5.4 Loss Functions

The loss function compares the target data and the calculation from the model when updating the weights. A loss function determines the difference between two arrays. If the loss function gives a high value between the two arrays, then the model did not give a good prediction. In the setting of the weights, the network will look to update weights in a way to minimize the value of the loss function called loss.

Although it seems like Root Mean Square Error (RMSE) should become a primary candidate for a loss function, measuring the loss is only one purpose of the loss function. A loss function also needs to give gradients that quickly update weights during minimization. RMSE would not quickly give a quick measurement as efficient code avoids the use of square roots. Additionally, RMSE does not arise from the form of its gradient. This means that it looks like a good measure of error but may not update weights efficiently.

As with network modifications, changing the loss function does alter the relationship between the gradients. As it comes in the last step of a forward pass, it modifies the mapping between the gradients minimally, but it still requires an updating to the mapping.

To fill the gradients after running through the data the Autograd Engine starts from the loss function and works backwards though the mapping. The initial gradients

from the loss function get calculated by looking at the relationship between the gradients of the input, weights, and output. After choosing an appropriate learning rate the mapping between the gradients updates the gradients of the weights all the way through the network. This makes the gradient of the loss function its primary attribute.

Looking at loss under this magnifying glass changes the type of loss functions a network should implement. This research used two primary loss functions, the Mean Squared Error loss function (MSE) due to its common usage in RS development, and the Binary Cross Entropy loss function (BCE) due to the network attempting to match binary values. The MSE comes with its standard definition $\mathrm{MSE}(x, y) = \frac{1}{N}\sum_{i=0}^{N}(x_i - y_i)^2$, and the BCE comes with the definition $\mathrm{BCE}(x, y) = \frac{1}{N}\sum_{i=0}^{N} y_i \ln(x_i) + (1 - y_i)\, ln(1 - x_i)$.

2.5.5 Optimizers

Once the mapping determines the relationship between gradients, updating the weights relies on optimizers. The torch.optim module automates this process, provides methods for updating weights, and includes classes of optimizers. Methods to assist in the tuning of the model include methods to dynamically adjust the learning rate and normalize batches. The engines to update the weights come in about a dozen varieties with use cases based on the data they encounter. Overall, the module supplies a plethora of tools to update the network.

As with determining the loss function, the obvious choice to update the weights does not always achieve the desired goals. When first encountering the problem, it seems reasonable to find the gradient, take a small step the size of the learning rate, and the loss should reduce. Subtracting the product of the gradients and the learning rate from the

weights does find local minimums but can struggle when the gradients become low. Optimizers that use momentum when updating weights, such as the family of ADAM optimizers, find improved weights even under these undesirable circumstances.

This study employed Stochastic Gradient Descent with momentum for the optimizer. The implementation of Stochastic Gradient Decent came from the SGD class in the torch.optim module. The learning rate and the momentum were specified from passed parameters upon initialization, and their values came from testing preliminary models prior to using the educational data.

2.6 Running Network

Running the network in simulations comes with considerations based on the purposed usage of the network. Choosing the research methodology depends more on finding a good simulation to catering for students, than finding the parameters that optimize the network. In addition, while fitting hyper-parameters, the requirements on the hardware need to define constraints for the model. The context of the usage will primarily make decisions usually reserved to optimize the predictions.

2.6.1 Time Step

In general, the time step should come from which values makes the best predictions on the data. As this network also comes with the context of mathematics assessment, the size and frequency of the time step should fit into this context. A large time step will not make predictions with the appropriate latency. An excessively small time step will not give enough time for the network to accumulate new data, and may become smaller than the granularity of the data.

This network will use the time step of a day. This doesn't come from any performance improvements of the network, but strictly from the needs of the application. Under the assumption students do math problems daily, the use of a daily time step will give new predictions with the same frequency as the students do problems, but also puts the least dependency on the physical resources while meeting demands.

In addition, for each time step the weights and embeddings should update. Depending on the resources available, the LMS may slow down while the network updates. An additional consideration of the time step should include at what time the network separates two different timesteps. When deciding where to separate timesteps low activity times should take precedence due to a reduced probability of separating one study session, and so users experience no unnecessary delays.

2.6.2 Hidden Dimension

For this network three parameters remain unchosen. The first parameter sets the dimension for student embeddings, problem embeddings, topic embeddings, and hidden state. These dimensions need to match, or the combinations of the embeddings will not go into the LSTM cell. The second dimension chooses the size of the cell state. These need to match between the LSTM cells but no other mechanisms in the network need to match these dimensions. The final dimension chooses the number of topics that will categorize the problems.

These parameters should be chosen to optimize the performance of the network, but the first parameter is the primary decider of the size of the network in memory. In addition, as the size of the embeddings increases, the training time also increases. Specifically, the size in memory increases as $O(n)$.

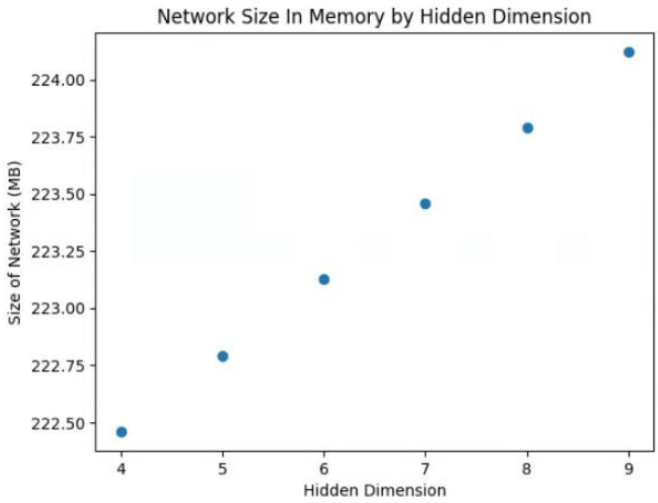


Figure 15. Network Size as the Embedding Dimension Increases

GPU Memory Allocation as the Embeddings Increase

When looking at the actual effect of increasing the embedding dimension the theoretical relationship between network size and embedding dimension appears accurate. Figure 15 shows how the memory grows with the increase of embedding dimension. The overhead however overwhelms the theoretical relationship. From the graph doubling the size of the embeddings increased the network size by less than 2%. Given the actual measurement of the network size a corrected space complexity would give $O(1)$. The practical limitations of fitting the embeddings would make this the theoretical best space complexity while at the same time the worse. The network size does not grow much because of the resources allocated to Pytorch upon instantiation. The best possible result comes because Pytorch takes so many resources to start.

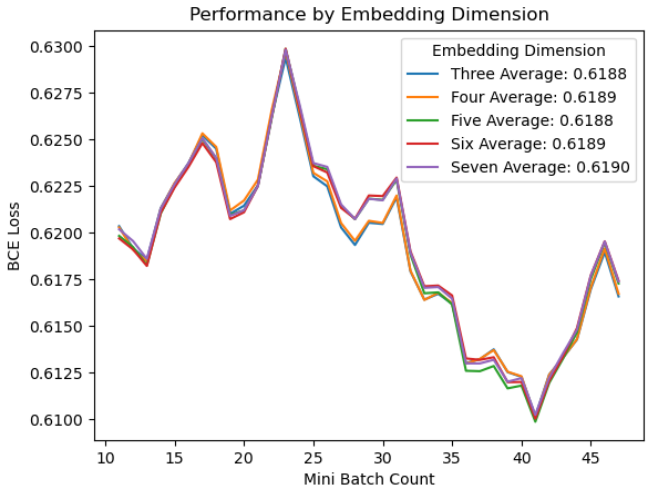


Figure 16. Loss for Various Values of Embedding Dimension

Comparing the Performance of the LSTM One Data Network with Several Embedding Dimensions

As shown above in Figure 16, after running through the data for two epochs the embedding dimension performs the best with the parameter for dimensions as three. Five dimensions also performs well but increases the size of the network in memory. For this graph the cell size remained set at three.

This cell trained with biases and no dropout on the biases. Biases reduce the influence of differences from the embeddings but gives an overall performance improvement. The similar performance of the different embedding dimensions comes due to preserving the accuracy of the group statistics. After determining the value for bias dropout, a preferred embedding dimension will appear.

As you change the first dimension, the performance of the network for each value for the size of the cell differs. Choosing the best parameter for the embedding dimension

does not guarantee the best overall performance. You can set these parameters in tandem using a relaxation technique, or a net, but the overall performance on this specific data set does not primarily concern this research.

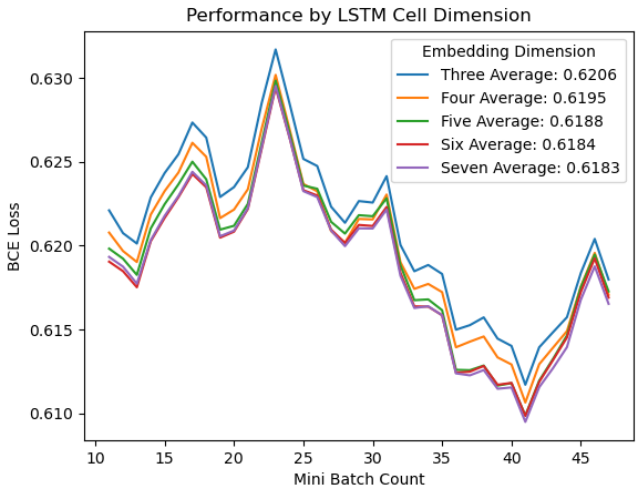


Figure 17. Loss for Various Values of Cell Dimension

Comparing the Loss for Different Cell Dimensions in the LSTM

For the embedding dimension of three, the cell dimension of seven appears to perform the best. Figure 17 summarizes the results. Five also performs well but the extra memory used for the extra dimensions does not matter for the cell dimension. As this network trained with biases changing the cell dimension changes the group statistics and creates a larger effect. In the second graph a larger difference in performance shows up in the graph.

2.6.3 Detaching Hidden and Cell Tensors

LSTMs use the hidden state outputs from previous forward passes in their current forward pass. If you consider the map relating the input and output of components, the output of components during previous forward passes will map to components in the current forward pass. This mapping becomes intricate quickly, and updating the weights may require updating the weights for every previous forward pass. PyTorch avoids these situations by not keeping maps between forward passes. This also means that when the hidden states try to map backwards, they will reach a dead end.

For cases with hidden states, the hidden states need to be specifically detached between the forward passes. The method detach() for a tensor in PyTorch removes a tensor from a graph. Running the detach method on the hidden and cell states will prevent them from interacting with the previous forward passes.

2.6.4 Epochs

When using biases the network sets its weights after two passes through the entire training set. After two passes reducing the loss becomes increasingly difficult. Each pass through the data gets cut up into mini-batches of users. The GPU specified earlier allowed mini-batches of 20 users. For each time step in the mini-batch the network runs a full backward pass and updates the weights and embeddings. Updating the network after each forward pass replicates its performance in practice.

For the training used here, each mini-batch passes through its entire set of data before starting another mini-batch. This replicates a small class of 20 students going through the curriculum before another group of students begins. An alternate path though the data would loop through each mini-batch for a time step. This would replicate many

classes of 20 students studying at the same time. The actual data collection happened over a period of less than a year so running though each time step would more accurately replicate the use of the network in practice. Although the training of this network did not choose the method most representative of its actual usage, the network does not encounter pairs of users, and problems in a different order. The weights in the LSTM cell however will be fit on the entire year of data before encountering the next mini-batch.

Chapter III.

Modifying the Attention Mechanism

. However, all parts of the user history are not equally relevant for recommendations in the current time step (e.g., due to seasonal preferences). Hence, the model computes attention scores to weigh the relevances of different parts of the user history.

— Dilruk Perera and Roger Zimmermann

To summarize the long-term behavior, an attention mechanism retains information from many previous timesteps. The importance of each time step depends on its similarity to the last time step of interactions. Similarity will depend on the context of the network application, but the larger the pool of problem the more complex it becomes to define similarity.

As the number of problems increases, the network will need more training time to determine the relationships between them. The training methods used for neural networks also do not guarantee an optimal solution, but only a solution that performs good locally around the initialization of the weights. This makes the initialization of the weights an important role in the overall performance of the network.

With the correct training time and optimizer settings, theoretically the network should achieve the same performance regardless of any initial values for weights, but this may not be the case in practice. Addressing these issues allows the network to weight previous timesteps appropriately, while retaining a modest training time.

3.1 Attention Scores

The scores coming from the attention mechanism signaled a need to modify it. The attention scores equally weighted all previous sessions independent of interactions. As no overlapping interactions occurred in any sessions the similarity scores between any two sessions remained zero. After applying Softmax the weights all became the same value.

Consider a network with three problems, and three sets of problems. Also let each set of problems consist of one interaction in the data network. If different interactions happen in each set then the weights will all take on uniform values. With many problems, the intercations will most assuredly consist of sparse vectors with unique interactions. This causes the uniformity of the weights. However, if the interaction vectors get categorized by topic before calculating the weights, then the weights take on different values.

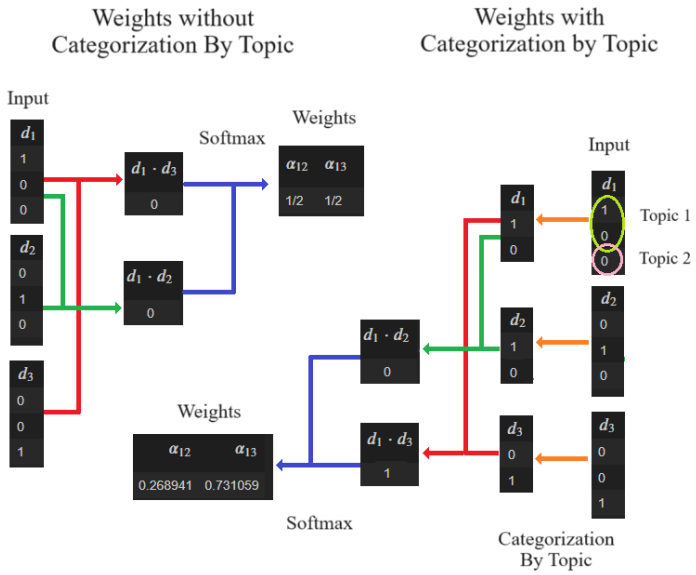


Figure 18. Grouping the Interaction Vectors by Topic

An Example of Categorizing the Interaction Vector to Add Relevance to the Similarity Scores

In Figure 18 you can see the calculation of the weights before and after categorizing the interaction vectors. Both inputs contain no overlap in the interactions, but grouping similar problems allows the network to modify similarity. Instead of weighting all sessions with the same similarity, different similarity scores come out of the network with topics.

3.1.1 Large Problem Databases

As the number of problems in a database increases the probability of encountering a problem multiple times approaches zero. This attribute of database size exists not only as coincidence, but also as a design for administering problems. Large databases give

students multiple opportunities to encounter material, provide unique learning experiences, and promote academic integrity.

The cosine similarity between days in these situations falls to zero. The use of cosine similarity in this network comes from the original context of the recommendation. As a user might frequently watch YouTube videos repeatedly, the cosine similarity between these days would not go to zero even if the platform gives many options for the user. With a large database of problems, a student returning to problems repeatedly would not come from the same type of behavior and implies an error in the recommendation system.

Remember the equations for the similarity scores $\alpha_j = \sigma\left(\text{sim}(d_{last}, d_j)\right)$, where $\text{sim}(d_i, d_j) = \frac{d_i \cdot d_j}{|d_i||d_j|}$, and d_i is the daily aggregation of interactions. For large a database of n items, $d_i \cdot d_j \approx 0$, and $\alpha_j = (1/n, 1/n, \ldots 1/n)$. These scores should represent the similarity between study sessions, but instead they start losing meaning. All the past states become equally important in the attention mechanism.

3.1.2 Daily Aggregations

To address the vanishing dot product this network uses a different daily aggregation to summarize a day as a vector. The primary difference comes from categorizing the interaction vectors, into categories called topics. The original cosine similarity did not differentiate for how many times a user interacted with the same problem during the same time step. When switching to topics it becomes almost inevitable a user will interact with the same topic many times during a time step. To

address this obstacle, the daily aggregation for this network consists of the sum of the topic vectors encountered from each problem.

When calculating the daily aggregation, the problems will get hot coded by topic. At the end of the time step the hot coded arrays will still get summed, creating a daily aggregation as $d_j \in \mathbb{N}_0{}^j$. The topic vectors will act the same as the interaction vectors and the network will still operate in its usual fashion.

This new definition of similarity shares many of the characteristics with the old cosine similarity. As the daily aggregations only consists of the sum of hot coded vectors, the cosine similarity stays between zero and one. It gives a value of one for identical vectors and zero for vectors that share no topics. It also gives a similarity score of one for study session that consist of the same topics in equal proportions. Sessions with high similarity scores will consist of the same ratio of problems but may differ in length.

3.1.2 Topics

The usage of the word topics refers to the different categories the problems can fall into. This comes from the idea that students would work on topics covering the same material as they work their way through a unit. In practice topics simply define the categories problems can fall into. For instance, a network with two topics might focus on separating problems by semesters, change in material, or difficulty. For example, AP Calculus BC would be a good candidate to try a network with two topics due to the transition from differential to integral calculus. Topics could also take values representing, chapters, lessons, or sections. In a network with 10 topics the labels could represent chapters or units.

The problems get topic labels from their location in the embedding space. The topics sit in the same space as the problems and a problem gets its label from the closest topic embedding. As they sit in the same space this means that the topic embedding must have the same dimension as the problem embedding. This assumes regions of the embedding space contain similar problems, which may not be true for all contexts of the network.

Consider a 3-dimensional space with problem embeddings in the space. This would corespond locations in a space as we commonly understand them. In higher dimensions the network will loose the analog to our perception of space, but we could still use this mental model. Topics would also sit in the same space. To label the problems we could just identify the closest topic to their locations as show below.

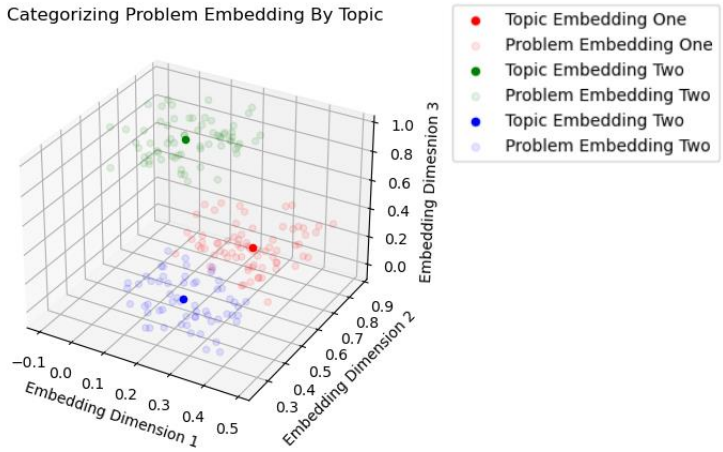


Figure 19. Finding the Topics of the Problems with the Topic Embeddings

Showing How the Topic Embeddings Categorize the Problem Embeddings with the Closest Topic Embedding

Figure 19 shows the categorization of problems by topic for a 3-dimensional embedding. The darker dots represent the topic embeddings, and the lighter dots represent the problem embeddings. The categorization depends on the closest topic embedding and the grouping of the problem embeddings becomes apparent.

3.1.3 Updating Topic Embeddings

Putting the topics in an embedding allows the network to use the Autograd feature to update the embeddings between forward passes. This allows the problem embeddings and the topic embeddings to move as appropriate. The positions could also update with K-Means between forward passes, and if the implementation of K-Means used the linear algebra operations available in PyTorch, the positions of the topics would properly update. This research did not focus on categorizing problems with K-Means, but it could prove a useful method going forward.

3.2 Initialization of Embeddings

Operating under the assumption that problems with similar locations in the embeddings share similar characteristics, the initialization of the problem embeddings allows the curriculum to pass data to the network prior to measuring interactions. If the locations get assigned randomly at the start, the network may never discover the relationship between problems and the curriculum. However, if the embeddings contain dimensions ordered by the curriculum while retaining randomly assigned positions in additional dimensions, the embeddings may describe the curriculum while allowing the network to encode additional relationships in the embeddings.

3.2.1 Randomized Initialization

Randomizing the problem and topic embeddings creates mostly noise. The attention scores appear uniform and do not weight similar study sessions higher. The noise is so severe that the network may not find the progression of problems throughout the course.

As the problem embedding will also keep the same topic label under small updates, the network also falls into the dilemma of vanishing gradients. Using the appropriate optimizer with momentum can address this problem, but it increases the difficulty of training the network.

With randomized weights fewer topic embeddings will give better results because of the noise created by increasing the number of topic embeddings. This does not necessarily lead to a better network, but a more limited network. It removes the possibility of topics embeddings predicting student interests and learning styles.

3.2.2 Initializing Weights Based on the Assumption of Curriculum

In this network's context, recommendations depend on the previous units. The material from the course comes in a sequential progression and creates a chain of dependency. A poor recommendation will not allow a student to demonstrate understanding due to the recommendation not coming at the right time. A good recommendation will not introduce material too far beyond the student's current progression though the curriculum. Paradoxically to find when a recommendation is inappropriate the network will need to learn from making bad predictions.

In addition, predictions will need to incorporate many aspects of assessment to maximize desired outcomes. Questions accomplish multiple goals such as, checking for

broad understanding, reviewing old topics, and creating a sense of trajectory when they contain material from multiple units. Some of the best questions contain material from multiple units and good assessments should use them. To make better predictions the network will need to find a user's progression in their current course.

The issue of poorly timed prediction doesn't come about due to the material coming from multiple units, but giving problems to students before they encounter the prerequisite material. Traditional assessment does not encounter these problems due to a sequential dependence of units. The curriculum will most likely come ordered in terms of units, or quickly separated into units, so labeling the problems prior to running network would combine the natural structure of the assessments with the network. Although this seems like a straightforward solution, precautions needs be taken so that the network can still draw its own inferences.

The data used for this network represented an academic year with increased density during the semesters. The problems came with a label structure, but the structure did not seem to give much help to initialize weights. The labeling also does not appear generalizable, and the methods used in this instance may not adapt to other data sets. For this reason, using an alternate method for labeling may benefit future networks.

A more generalizable estimate may come from the first occurrence of the problems as each problem usually comes with a first occurrence in the course. Using the first occurrence of the problem uses on a natural progression of topics of material. Assuming the problems came as part of a curriculum, their first appearance would approximate the primary unit of their content. In the problem embeddings, their initial placement should roughly group them by units with additional freedom to model

additional behavior. This would make the date of first occurrence a good measurement to initially label the unit of the problems.

Given an *n*-dimensional embedding where $n > 2$, the unit circle in first two dimensions gives an embedding for assumed topics. The topic embeddings for the *i*th topic embedding $e_i = (\cos(2\pi i/n), \sin(2\pi i/n), \ldots)$, and for the problem embeddings let $e_i = (\cos(2\pi day/365), \sin(2\pi day/365), \ldots)$ where the remaining dimensions get initialized with randomized numbers of a reduced magnitude. The radius of the circle may also serve as hyperparameter as the problems might quickly reduce their magnitude to fit their interactions with other components.

3.3 Performance of Topics

Given the categorization of the problems, the similarity scores should improve. As a student changes between topics, their performance should not rely as much on their results from pervious units and more on their results in recent units. Additionally, students should find the highest similarity scores with their most recent performance as it is more telling of their current mental state.

3.3.1 Performance

The hardware limitations of the network gave an upper bound for the number of topics. Using topics to represent lessons or weeks put too much strain on the GPU. The forward pass would stall when the topics went beyond ten. With limited topics the network would alternate labels to model the transitions between topics. Additional topics would show an improved performance when reviewing several topics for a summative assessment.

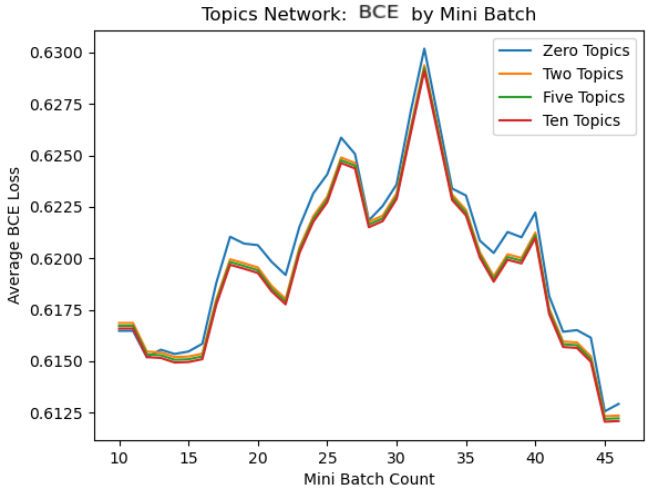


Figure 20. Loss for Models with Different Size Topic Embeddings

Showing the Performance Difference of the Networks with Different Numbers of Topic Embeddings

In terms of performance the addition of topics did not show a continued improvement in the network. Figure 20 summarizes the performance when modifying the embedding dimension. Increasing the number of topics beyond two does not show a visible difference in prediction on the graph. Changing the number of topics altered the fourth decimal of the prediction meaning it most likely consists of a fourth or fifth order correction to this prediction. However, differences in the way the network made the prediction did appear.

If you considered the average of the last 10 similarity scores, adding topics did weight the more recent sessions higher. It appears that categorizing the problems and then calculating the similarity scores affects how the network includes the pervious hidden

states. The addition of topics will allow for more relevant suggestions, compared to the network without categorized problems.

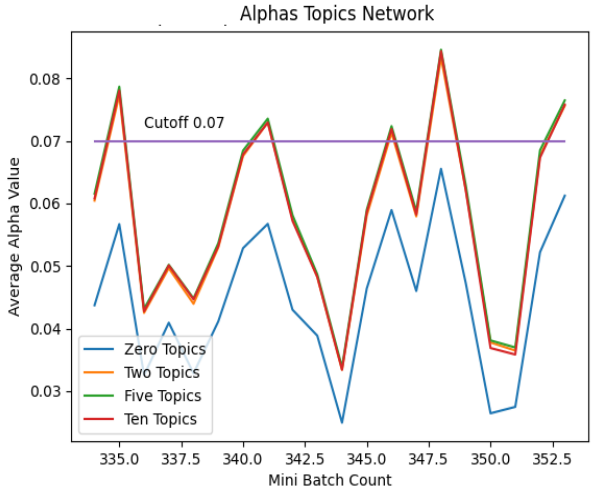


Figure 21. Last 10 Similarity Scores Average

A Graph Showing as the Number Topic Embeddings Increases the Weights of Problems in the Same Unit Increases

In looking at how the alphas distribute in practice, the average for the last 10 alphas values increased with the addition of networks. Figure 21 shows the shifting weights with the increase in topics. As this value increases the network puts more weight on the more recent measurements. The network with zero topics crossed the cutoff 2.8% of the time, the network with two topics crossed the threshold 13.6% of time, the network with five topics crossed the threshold 14.4% of the time, and the network with 10 topics crossed the threshold 13.6% of the time. The similarities between the number of networks

support the uniform performance of students though out the school year. We should expect to see this in the data.

SoftMax also adds an unwanted consequence of normalization. As the number of days increase the alpha values with similarity scores close to zero approach a $1/days$ behavior. To compare the values of alphas between forward passes their overall values were multiplied by a factor proportional to the number of days.

The network with zero topics also showed different values of alpha. The problem arrays contain sparse data, so the variation of alpha values seems surprising. This may come from days when students repeat problems or from days when the students did not complete problems, such as weekends.

3.4 Deciding to Add in Topics

3.4.1 Considerations

Adding topic embeddings depends on the performance of the attention mechanism. If the context of the network requires more importance on more recent data measurements, then the use of topics seems appropriate. Topics also can aid when items of recommendation come in set like units. Topics find similarity when sparse arrays would not. Finally topics aid when recommended items require content to be delivered in order.

Chapter IV

Data Source

We hope our dataset could empower the research of creating a better and personalized learning experience for students, and further encourage broader participation for contributing to the future of online learning from interdisciplinary experts.

—Junyi Academy

To ensure general applicability of the network, the data source should not contain anything that would skew the results. Primarily this would include the basic measurements needed for an RS. Ideally no measurable influence from external factors should appear. Furthermore, the data source should exhibit universal characteristics common to all datasets coming from math assessment. The dataset will require examination for any peculiarities that will cause any unwanted behavior when training the network.

The primary concern in a data source for fitting a network comes from the number of datapoints. Each datapoint will modify the weights and embeddings during one backward pass. As datapoints will contain different values of each measured variable several datapoints need to exist for each value of a variable to contain a distribution of measurements. After filtering the data, a sufficient amount of datapoints need to exist to fit the network.

The fewest measurements required for a datapoint in an RS would include the user, rated object, and rating. State of the art networks also include a measurement for the time when the rating occurs. The analog measurements in education would be student, problem, and correct answer. This dataset will also require additional measurements for the added data networks. These additional measurements could come from session details, or from database details about the user.

The measurement of the data also needs consideration. The students need enough time and incentive to complete the problems. The problems need the appropriate difficulty, and adequate student preparation. Additionally, the distribution of students needs to include distribution across many measurements including intelligence, intuition, experience, region, and gender.

For the dataset used, the trends in the data should guide the development of the network. Additions in data should improve the performance of the network when used as additional measurements. Measurements with little effect on the data should show less improvement, however the network can group students or problems together and find less obvious relationships.

4.1 Context of Data

Where the data comes from, how the data was collected, and what the data says, will primarily influence the design of the network and its usage. If students do problems online for practice, they will perform differently than if they do the problems for a grade. Also, as discussed earlier, the resources given with the problem will affect how well the students do. Depending on if hints, examples, or instructional videos come with the

problems, differences in performance should appear. Finally, the dependency of data on different variables will determine what variables to include in the second data network.

4.1.1 Source

To train the network the model ran on data from the Junyi Academy Online Learning Activity Dataset[11]. This data set consists of 16 million exercise attempts from 72k students in Junyi Academy from August 2018 to July 2019. Junyi Academy Foundation is a Taiwanese, non-profit, educational organization and maintains a mission goal to provide education for all children by using technology. The students consist of first through twelfth grade students from cities though Taiwan.

The questions the students answered on the Junyi Academy platform. They platform operates as stand-alone courses, supplement to in class instruction, or as an additional enrichment for students. Some students did get teachers from the Junyi Academy assigned to them, while others worked independently. Additionally, the platform offered formative videos, and badges to incentivize students.

The platform showed some adaptation to the students. First each student got a different problem set over a topic. Depending on the student's performance the LMS upgraded or downgraded students after the answered exercises.

4.2 Details of Data

The data comes in three files totaling about 2 GB. The files break the data into problem attempts, user information, and problem information, with the names

[11] https://www.kaggle.com/datasets/junyiacademy/learning-activity-public-dataset-by-junyi-academy

Log_Problem.csv, Info_UserData.csv, and Info_Content.csv respectively. The Junyi Academy released the data under the Creative Common License, CC BY-NC-SA 4.0. Figure 22 shows the summary of dataset on Kaggle.

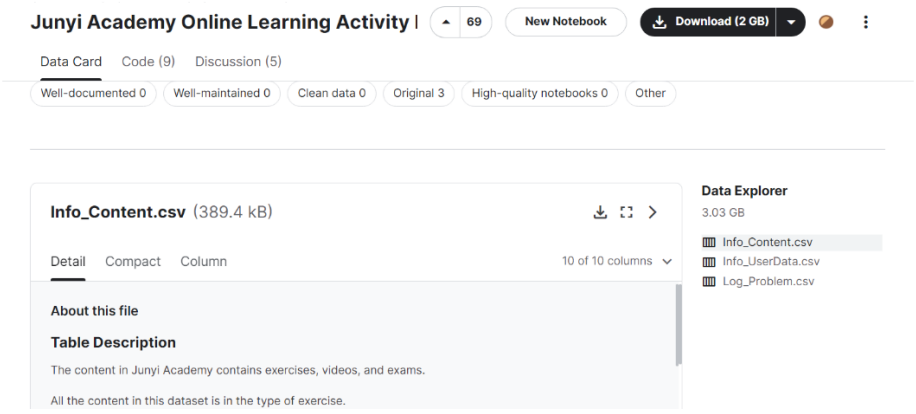


Figure 22. The Dataset on Kaggle

A Screenshot of the Dataset on Kaggle with the Files in the Dataset

4.2.1 Students

The students consist of elementary, middle school, and high school age students. Each user got a unique ID (uuid) consisting of a random alpha numeric string. The platform recorded the gender of students as “male”, “female”, “unspecified”, or “null” with the 55% of the users recorded as “null”. Users also identified with a city from the list of 20 cities and towns, spread throughout geographic, economic, and population demographics.

Figure 23. Pandas Data Columns from Loaded CSV Information File *Showing a Screenshot of the DataFrame Loaded into Pandas*

The snippet of code above in Figure 23 shows column names after loading the data from the csv file to pandas. The columns names represent the data from the columns, and the data type differs by column. PyTorch can only take in data represented numerically so it needed alternative representations of the data. This can be done by representing the sting in ascii as hexadecimal, or by using dictionaries.

The number of problems each student finishes also affects the ability of the network to make predictions. To ensure each student completes enough problems the density of problems attempted per student needs to be looked at.

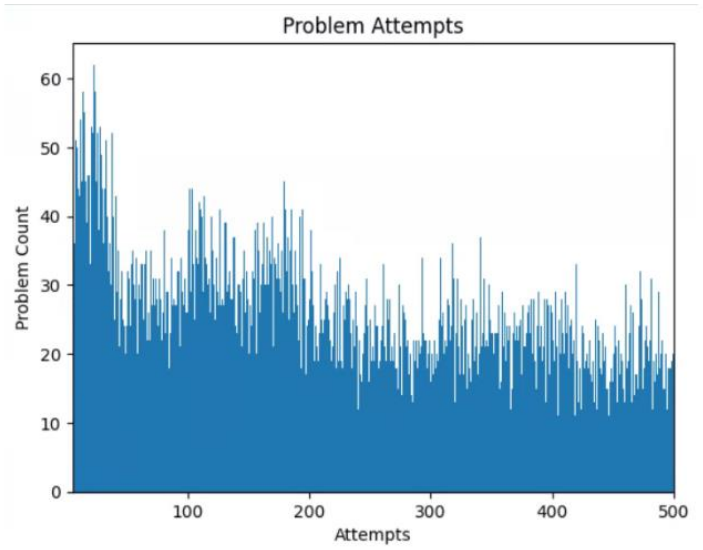


Figure 24. Problem Attempt Density

Problem Density for the Number of Times Students Attempted a Problem

After filtering the students that completed less than 100 problems, the data remains dense. As shown above in Figure 24, the number of students roughly decays exponentially with the number of problems finished. This leaves enough students for the training of the network.

Additionally, the distribution of students per year will show where the bulk of the data lies. This will help in the analysis of performance. It will show where a lack of data may lead to abnormal behavior of the network.

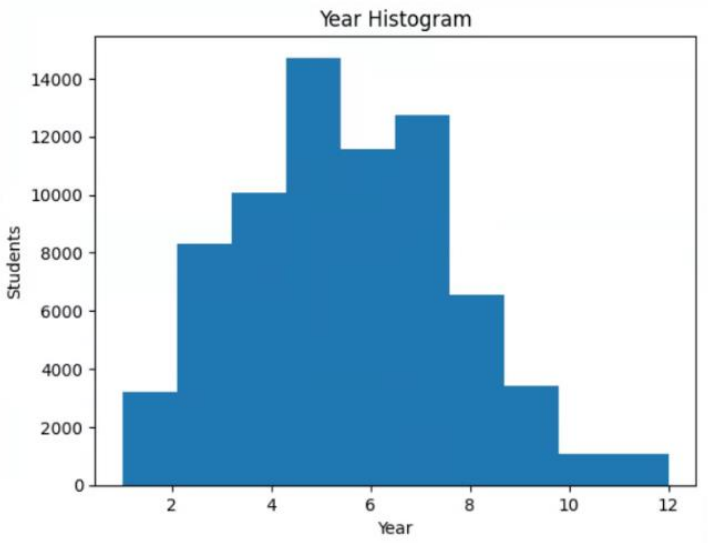


Figure 25. Histogram of Students by Year

Histogram of Students by Year for the Junyi Dataset

From the graph shown in Figure 25 the data peaks between 5th-7th grade. As the age of the students goes away from the central max the volume data drops off. There still exists several hundred students in each year, and several thousand students in the bins from 2nd-9th grade. With a median value of around 50 problems per student this would mean there are several thousand data points for each year.

4.2.2 Problems

The problems come from exercises, and exams. All measurements in the data come from a type of exercise with the structure shown in Figure 26. A unique problem id (upid) identifies each of the 25,785 problems in the dataset. Some of the exercises come as a collection of problems related to a certain topic. Each problem also comes with a topic id based on its content, its perceived difficulty, and its school level. The difficulty

levels come as “easy”, “medium”, and “hard”, and the school level comes as “elementary”, “junior”, and “senior”. The students get problems presented to them in groups that share a common content ID.

Figure 26. Pandas Data Columns from Loaded CSV Content File

Showing a Screenshot of the DataFrame Loaded into Pandas

The content that contains the problems also comes sorted in a 4 level ID structure. Each ID consists of a randomly generated alphanumeric string such as “aH0Dz0KdH9gio7rrcGRHvrmd9vcd/0WJbeEFB7qeUKA=”. Level 1 represents the subject, which in this dataset only consists of math. Every problem in the dataset has the Level 1 ID corresponding to math. The Level 2 ID more closely relates to the course the problem comes from. The example given in the documentation uses “Geomentry” as the string associated with the Level 2 ID. The Level 3 ID represents the unit, and the Level 4 ID represents the topic. Students get problems presented to them in groups that share a common content ID.

From the screenshot of the website, you can see an exercise consisting of several problems. Between the problems this website also uses videos to introduce the material. The multi-media play button links to videos explaining the problems, and the star button links to a problem.

Figure 27. Example of an Exercise Consisting of Several Problems

An Exercise Consisting of Videos and Problems (怎樣解題：數量關係 | 均一教育平台, n.d.)

As discussed earlier, the addition of videos affects the measurement of performance. If a student encounters a problem outside of what they know, the videos allow them to learn how to solve the problem. As shown in Figure 27 several videos introduce each problem. When the network predicts the probability of a student answering a question correctly, the probability should fall to zero as a problem falls later in the course. In the data it does not appear students attempted problems coming much later in the course before completing the introductory material. With the addition of the videos and the missing data for later problems, the network will struggle to make predictions for problems coming later in the course.

The time of the videos also comes next to link so students see the videos will concisely explain how to do the problems. Students will sometimes avoid videos due to the time it takes to watch them and look for examples that they can quickly modify to solve their problem. If the videos do not get too long, including the times should positively influence performance beyond the standard effect of videos.

The videos themselves do a good job of explaining the method of solution. They solve the problems with diagrams, algebraic relations, written language, and audio description. When trying to quickly, and clearly explain a problem to a group of students the combination of elements will allow students to learn in their preferred method while keeping a redundancy in case a student becomes confused in their primary method of learning.

Figure 28. Example of a Video Explaining the Material

Screenshot of a Video Coming from the Junyi Academy Exercises

The screenshot above from Figure 28 shows a video used in the exercises. This problem comes from the Algebra I curriculum, in the 6th year of instruction. The videos are public YouTube videos linked to a teacher's accounts. The videos come in Mandarin and the instruction style relies heavily on the diagramed answer.

The problems come as static problems coming from a problem library. The problems also use parameters and may change, but that is uncertain. The problems use blanks for responses, and grade immediately. After grading, mascots also encourage the students to keep trying. The format also includes a list of problems and video. This helps put in the appropriate videos next to the problems, but it also makes students start a new problem set to continue after the last problem.

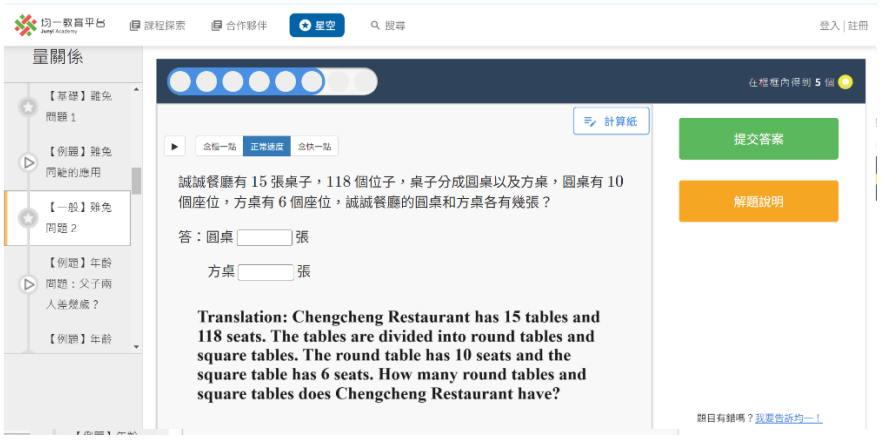


Figure 29. Example of a Problem in an Exercise

A Problem Coming from a Junyi Academy Exercise with Translation Added

Figure 29 shows a problem with an added translation. In the Common Core Curriculum the problems would come from Advanced Algebra I, or Algebra II. The exercise is labeled as “Elementary”, but the content falls a little above elementary.

To investigate how the problems appear to the students, a web browser can probe the server several times with different cookies. The problems do not require a user login so loading the problems with a different web browsers and cookies give different problem sets.

Figure 30. Example of Different Users Getting Different Problems

Two Different Final Problems Coming from the Same Exercise

After loading the same exercise twice two different final problems appear. Figure 30 shows an alternate question in the exercise. The problem library contains enough problems that it becomes difficult to determine if the problems will show up with different parameters. The problems did not show up on a Google search or Chegg. This exercise shows how to correctly use a library of problems. This guarantees the students worked independently but may have worked together on similar problems with different parameters.

Even though the exercises used a large problem library, the density of problems each student attempted remains high. Many students attempted several hundred problems. The data does not fall off very quickly. As Figure 31 shows a sufficient number of students attempted several hundred problems.

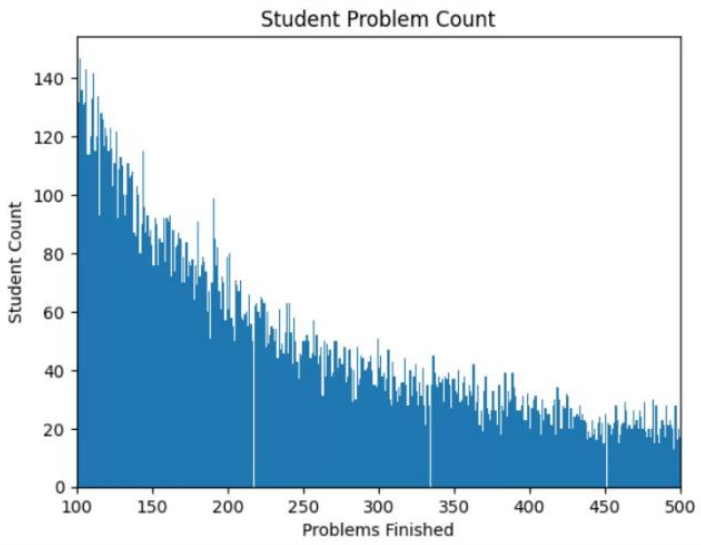


Figure 31. Density of Students Completed by Problems Completed

Density of Students by Problems Showing the Number of Students that Completed the a Certain Number of Problems

Problems occurred in the dataset most commonly less than five times. Each occurrence would correspond to an attempt to solve the problem by a student. After filtering problems with less than five occurrences, the number of problems appears to decay linearly as the number of attempts increases. Considering the weights will only use a portion of the data for fitting each problem needs multiple attempts to fit itself inside the embedding. Given this density of attempts there exists sufficient data to set the embedding of the problems after a few epochs.

4.2.3 Measurements

The log data of the problem attempts came as a CSV file with, string, Boolean, integer, and float data. Figure 32 summarizes the columns of the CSV. As Pytorch tensors only support numerical datatypes, the data will need a dictionary, or representation in hexadecimal to fit into a tensor. Each row in the CSV records one attempt for a problem in an exercise.

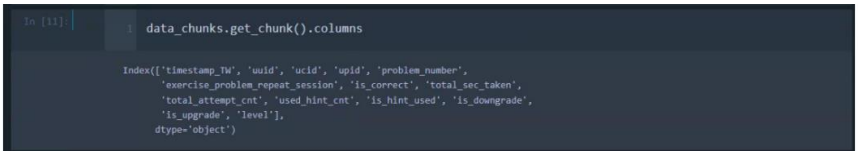


Figure 32. Pandas Data Columns from Loaded CSV Log File

Screenshot of a Video Coming from the Junyi Academy Exercises

When a student answers a question a 14 column measurement are inserted into the log problems data. The 14 measurements consist of Taiwanese timestamp, uuid, ucid,

upid, number of problems encountered in the exercise, the number of times the user attempted this problem during this exercise, if the user answered the question correctly, the total seconds taken, the number of times the user submitted an answer, and the if the hint was used. Rows consists of a comma separated values of strings. The timestamps use a granularity of 15 minutes, and the time spent on each problem used a granularity of seconds.

4.3 Trends in Data

For the dataset given, the analysis focused on session variables, and database variables. The network could easily use these variables as second data sources. Database variables could contain user information that the user embeddings might find difficult to record, and session variables record instantaneous measurements that are difficult to determine from user embeddings.

4.3.1 Session Variables

During a user session the platform can make measurements about the current exercise. For instance, the platform can measure length of session, how many problems the user attempted, or the time in the user's time zone. These measurements do not depend on the characteristics of the user and simply on the interactions performed on the network.

These characteristics measure universal attributes like fatigue, frustration, and time constraints. As this network also uses higher order interactions, the network may model characteristics that adapt to the individual. In practice these measurements come from pools of users which should show a distribution of the values. Even if the

aggregations of data do not reveal patterns, the network may naturally segment the data where trends do exist. In addition, user personality should dictate preferences in study sessions, outside of broad trends.

4.3.2 Time of Day

Time of day serves as a uniform measure of fatigue. Although particular students will perform better during different times of the day, the middle of the night should show a dip in performance for all users. As performance at certain times of the day differs by user this variable also makes for a great second order effect.

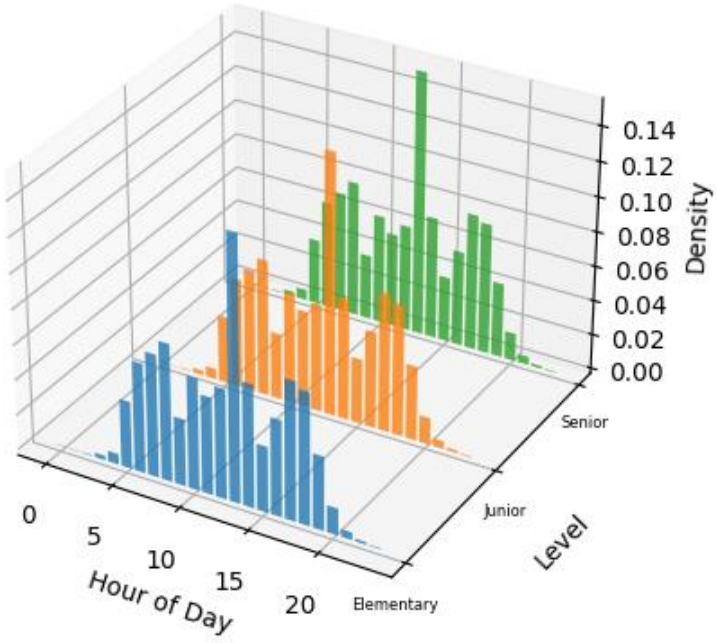


Figure 33. Histogram of Daily Activity by School Level

Graph Showing Density of When Students Work

Trends appear in the daily activity which reflect the circadian rhythms. Figure 33 shows these patterns. When plotted by level, the distribution of activity by hour looks identical across the different school levels. Activity begins around 5 am and dies down at 8 pm. Peaks in the data occur at 8 am, 12 pm, and 4 pm. The main peak at all three levels occurs around 12 pm.

As discussed in subsection 2.5.1 the time separating the timesteps should come from the low activity periods on the network. As the data shows separating timesteps between 10 pm and 3 am would reduce the probability of breaking a study session into two separate timesteps and reduce the load on the network when the network updates weight and embeddings.

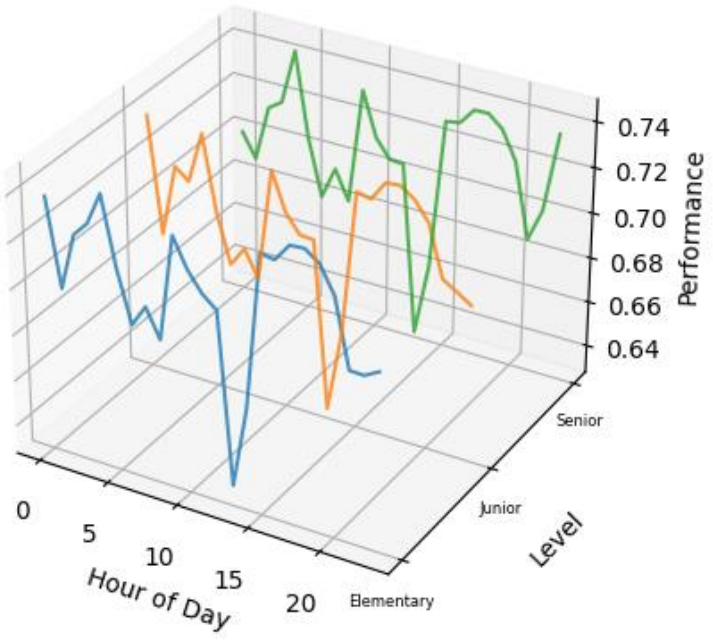


Figure 34. Student Performance by hour of the Day

Graph Showing the Performance as the Time of Day Changes

Additionally, the time of day affected the performance of the students. Figure 34 summarizes the measurements. Performance in this manner refers to the probability a student will get the problem correct and begin another problem within fifteen minutes. Similar trends exist between the different school levels, however in contrast to the previous day, data noticeable differences exist in the late-night behavior. The eldest students showed an increased performance at the end of the day compared to students in other levels. Throughout all levels, increased activity led to decreased performance.

4.3.3 Length of Session in Minutes

The number of minutes a student works differs from the other session variables. This measurement comes with an expectation of the shape of the data. In the first few minutes the data should look noisy with students attempting problems and then abandoning study sessions. After a period of a couple minutes where a student commits to a study session, students should start to leave session with a constant rate. The rate combines students finishing assigned work and distractions. If the rate remained constant the curves would appear as exponential decays.

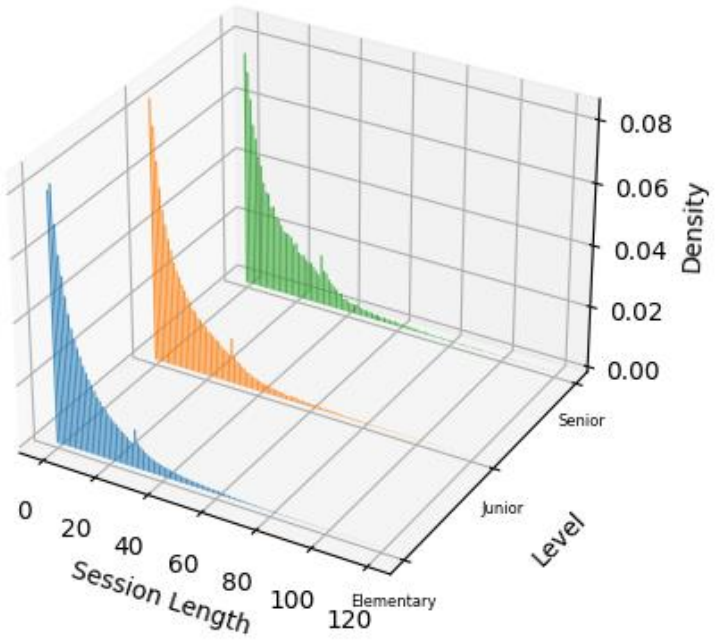


Figure 35. Histogram of Session Length by School Level

Density of Session Length in Minutes, Categorized by Level

Though out the levels of instruction the exponentially decaying behavior prevails as shown in Figure 35. Additionally, trends in session length appeared uniform throughout the different school levels. The session lengths show a finer granularity than the data due to their calculation. For the length of study session, the seconds the students took on each problem were added together, and the analysis used idle periods of more than a half hour as markers to end a study session. High school students showed a minor tendency to study for longer periods of time, but not enough to visibly skew the data. All levels showed a reduction in the number of study sessions lasting beyond 60 minutes.

There also exists a spike around 30 minutes at all levels. This implies many students ended their study sessions at 30 minutes. This might occur due to instructors

requiring students to do problems for 30 minutes, but it also might occur due to the way the graph splits up study sessions. If a student remained idle for 30 minutes the graph considered that period as the end of the study session. That would mean students who opened a problem and left it on their computer would fall in this bin on the graph.

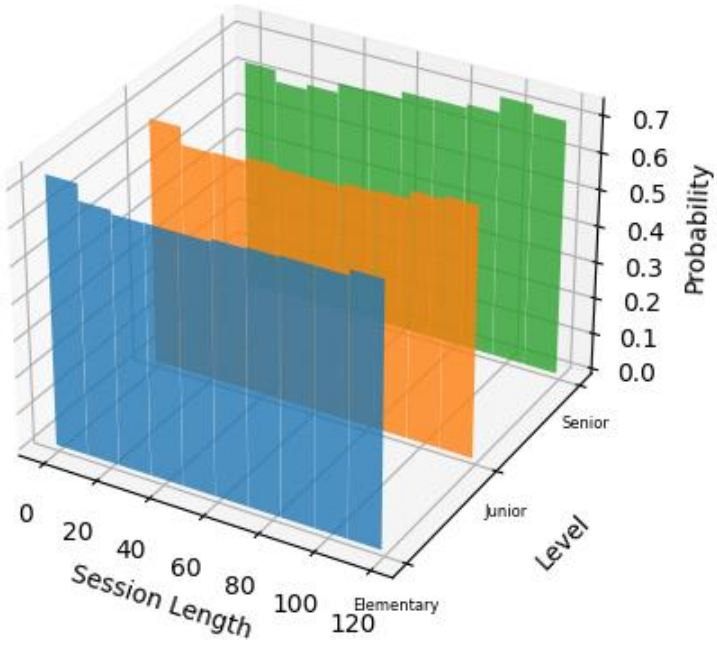


Figure 36. Probability of Correct Answer by Session Length

Probability of Correct Answer by Session Length, Categorized by Level

Although the data shows a clear distribution in the session length, the session length does not affect the student performance based on level. When the data gets further stratified by grade patterns emerge. In particular the 12th year students showed an increase in performance in longer sessions. As visible from Figure 36 almost no difference in performance exists as session length increases.

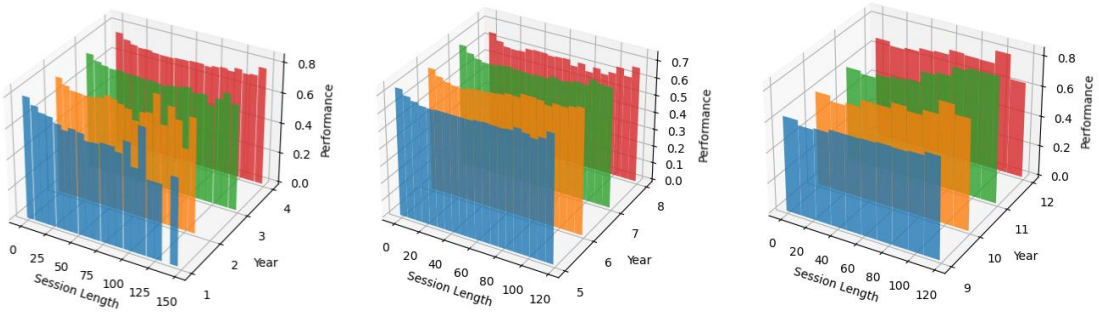


Figure 37. Student Performance of Each Grade by Session Length

Performance by Session Length Separated by Grade

The consistent performance, which includes the probability of a student not continuing, appears roughly constant throughout as a function of study time. This approximates the behavior seen in the pervious set of graphs which showed an exponential decay of the density. As seen in Figure 37 small differences exist between grades but they mostly approximate their averages by level displayed in Figure 36.

4.3.4 Number of Problems in a Session

One of the standard measurements in the Junyi datasets comes from how many problems a student completed in their current study session. The data records the number of problems a student encountered including their current problem. As before, an exercise consists of the group of videos and problems. With this definition the expectation is the value of problem number to peak at around a central value and quickly fall off. At earlier years the fall off might not get as extreme due to their exercises consisting of shorter problems.

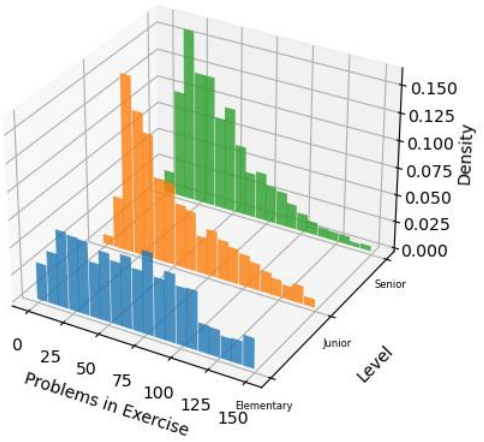


Figure 38. Problems Per Exercise Separated by Level

Problem Number in Exercise as Given by Junyi

When separated by levels patterns in the data appear. Figure 38 shows these differences. The fall off at the elementary level does become less severe. For the Junior and Senior level students the problems peak at around 20 problems with a more drastic fall off.

4.3.5 Database Variables

In addition to session measurements, static database variables can also guide prediction. In this context database variables could include gender, location, and additional socio-economic indicators. Unlike session measurements, database variables change infrequently, and may rely on information provided by the user. During the

session the network will not update or record database variables as it would with session variables.

These characteristics measure details specific to the user. In general, the session measure will describe the progression of the session, and the database measurements will measure the characteristics of the user. The user embeddings should include these details, but creating such an embedding requires extensive training. It therefore may improve performance to include these characteristics through a second network.

4.3.6 Gender

Gender comes as the first database variable examined. Database variables require the user to submit their gender at a certain point, so these values are frequently missing. Gender also traditionally shows differences amongst students, but this dataset does not guarantee these differences will appear.

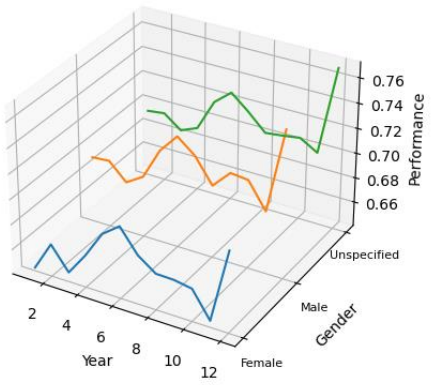


Figure 39. Performance Separated by Gender Separated by Year

The Average Performance of the Three Values Recorded for Gender Separated by the Student's Grade

When looking at the relationships between gender and performance, these graphs show the largest differences between curves out of any data encountered so far. Figure 39 summarizes these findings. Gender is not a particularly strong indicator of performance, as we will see later, but it shows the dramatic difference between session measurements and user measurements. The least predictive database variables will show larger differences in performance than the most predictive session variables.

As stated in the previous section, the dataset recorded gender very poorly. Most users did not make any record of their gender. It also appears the highest performing gender consisted of students choosing to not specify their gender. From this data even bad database variable measurements look like better predictors than session measurements.

4.3.4 City

City value also shows a great potential to affect performance. City value can include additional characteristics the user embeddings may not. City value can describe the proficiency of local schools, the influence on regional education programs, cultural attitudes toward education, or even household income levels. Considering performance differences in local schools exist at every level, users should also show differences in performance based on the local elementary, middle, and high schools. The regions or districts these schools fall into, and how they get their funds can also affect the performance of the students. This database variable can potentially measure many differences in educational opportunities

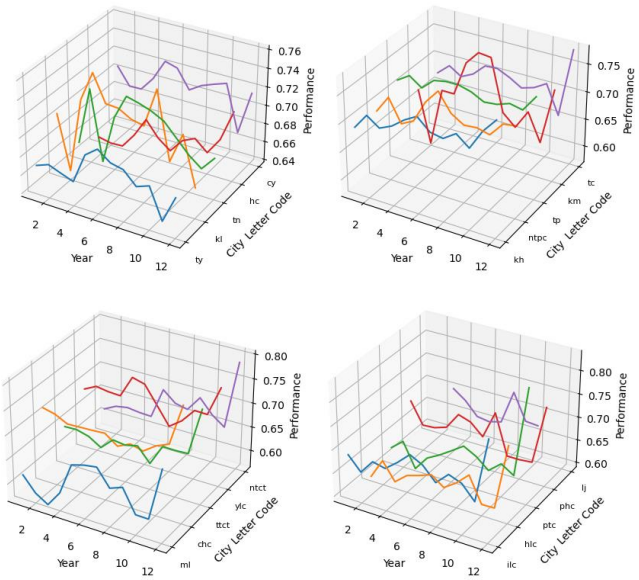


Figure 40. Performance Separated by City

Performance Separated by City, Two Letter City Codes

The city associated with each student also looks predictive of the performance. When plotted against each other the performance varies wildly between cities. Figure 40 shows large variations from city to city. When summarized by gender peaks appear around year six and year twelve. When split by city the any universal characteristics appear to disappear.

Chapter V

Performance of Parallel Networks

My psychiatrist told me I was crazy, and I said I wanted a second opinion. He said, "Okay, you're ugly, too."

—Rodney Dangerfield

Adding parallel networks acts like a second opinion. Each student gets a unique user embedding, and the student's user embedding should give the best prediction. That doesn't mean the user embedding can contain all the information available. Adding parallel data networks acts like a new individual privy to additional information and inputs additional measurements into the network.

For instance, the network without additional data sources does not contain any way to include session variables. Without the session variables the network remains blind to the current status of the student. It does not know how long a student has worked for, what time of day it is, or how many problems the student has attempted. Adding additional data networks allows for the input of these measurements without dramatic alterations to the network.

5.1 Adding Parallel Networks

In the Perera and Zimmerman network additional ingestion networks come from separate sources of data. These data networks usually consist of an independent data

source with little heterogeneity. The data networks added in this case will consist of different measurements coming from the same dataset.

5.1.1 Additional Embeddings

The instantiation of the LSTM object requires the passing of a dictionary containing the number of networks among other things. This allows the data when passed in to contain an additional column for the scores of an additional network. The network scales to as many additional networks as the hardware will allow to exist. As each row of data contains an integer representing the problem attempted, the data can identify additional embeddings for the same problem associated with a different measurement.

The additional embeddings come in during the higher order interaction layer of the forward pass and get combined with the original data. They take the same path after this step and get passed through to the same LSTM cell. Although it may seem more appropriate to create a complete second network and combine the results at a later stage, combining the data sources prior to entering the cell makes the addition of additional data sources simple.

The drawback from this architecture comes from the addition of networks after training. As the higher order interaction layer happens before entering the LSTM cell, the cell does not adapt to the new data combination coming into the cell. It will need to retrain with the additional data. However, during the retraining the network performs well when the number of data sources gets decided prior to fitting the weights. Given the quick training time of the network, reaching asymptotic performance after two epochs in some instances, the problem of adding data sources after training might not matter.

5.1.2 Network Hyperparameters

As throughout this work, the hyper-parameters used while combining networks were fit, but not with the intent to create a misrepresentation of performance. The results summarized do not come from the meticulous training of the network for a desired result, but from the apparent performance upon modest training. This network should adapt to other education sets, with the goal of improving mathematics education. The suggestions made should aid many instances of future networks, with datasets currently unavailable.

5.2 Separation of Data

Separation of data makes sure the network gets an opportunity to perform on data it did not train on. If the separation of data does not occur, no real measurement of the network accuracy exists. The network may also over-fit its weights to the training data, and this only becomes apparent when the network makes predictions of the test data. Separation of data remains an important step for the evaluation of the network.

5.2.1 Training, Testing, and Evaluation Data

The data for training, testing, and evaluating gets split by users into three groups after filtering. The splits consisted of 60% of users for training, 20% of the users for testing, and 20% for validating. Upon instantiation the LSTM object, which requires an int representing the number of users in a passed dictionary, creates three random lists consisting of the users. Prior to creating the lists, the LSTM object sets the random seed in PyTorch, so the results of the test remain deterministic.

The training data sets the weights, and the testing data compares different values of the hyper-parameters. For one set of hyper-parameters the training data will set the

weights, and the testing data will give accuracy to the network based on a particular loss function. After several cycles enough measurements should exist to determine the optimum hyper-parameters for the data. This research looks to determine the effect of parallel networks with secondary data source, and meticulously tuning the hyper-parameters may undermine these effects.

After fitting the hyper-parameters, the validation data works as a test to see how the network will act on new data. The values of the loss functions on the evaluation data will form a metric for the success of parallel networks.

5.2.2 Network Ingestion of Data

The network goes through more forward passes it will need to record the data it sees for the calculation of the hidden states, and the hidden states in the attention mechanism. Feeding all of data at once would give the smallest requirement on memory but creates some issues with assurance the network accurately makes predictions that it already holds in tensors. In order to accurately test the network, the LSTM object holds data passed through it separately, and between epochs its historical data gets reset. This prevents the network from making predictions on data out of order or referring to outcomes stored in the data it already encountered.

5.2.2 Mini-Batches

The number of data points for any given day will overwhelm the network if a forward pass contains all of them. The data then gets broken into groups of a few users and the network trains itself through the whole course on these users. Between each day the weights update in the same way the act during normal operation.

5.2.3 Filtering

To ensure adequate data exists to model problems and students, cutoffs for the minimum number of attempts for a problem, and the minimum number of problems a student completed filtered the data. These graphs used cutoffs for a problem of 20 attempts and 100 completed problems for a student. Some problems students completed got filtered even though the students actively completed problems. As the measurement of time fell into 15 minutes increments, a student could get a problem removed from a session without the network falsely recognizing the session as two separate sessions.

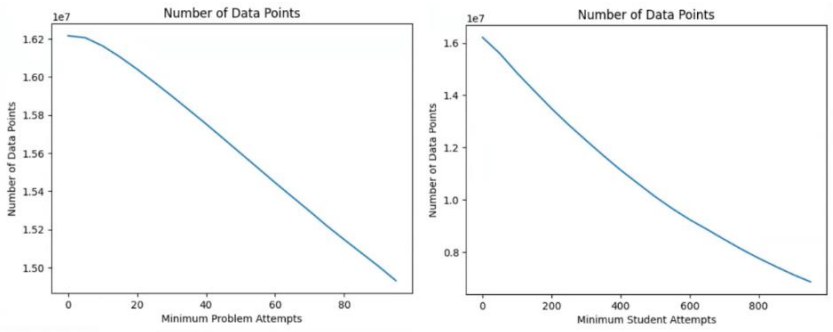


Figure 41. Data Points Remaining After Filtering

Reduction of Total Datapoints as the Cutoff for Minimum Number of Problems Increases

After the filtering it appears that putting cutoffs on the data does not dramatically reduce the size of the dataset. Figure 41 shows the effect of the cutoff. The cutoff at 20 problems appears to reduce the size of the dataset by less than one percent. When looking at filtering the problems by number of attempts, the default cutoff of 100 problems per student was used. Similarly applying cutoffs for the number of problems completed by

students does not dramatically affect the data. Applying the filtering does leave enough data to reasonably fit the weights for the network.

5.3 Session Data Parallel Networks

Session data comes from values stored in temporary variables and consists of measurements made during a session. The measurements take no consideration for the characteristics of the student using the learning platform. Predictions coming from a session should measure details universal to all students, such as fatigue, motivation, difficulty and learning style. Students using the network still get personalized modeling, but session details will not determine the geographic, socio-economic, or cultural, aspects of student performance.

Throughout these tests the embeddings used a size of five and the LSTM cell used a size of five. The networks included topic embeddings, so embedding the curriculum in the first two dimensions affected the performance of the networks with three dimensions. The networks will fit around the sixth year dataset due to the curriculum's similar structure to later grades, its large number of students, and position in the amongst the other years. Each network trained for two epochs on the training data, before running on the test data.

5.3.1 Performance of Session Variables

The session variables examined in this experiment included time of day, problem number in the exercise, and session length. These three variables indirectly measure fatigue and frustration. In the data, time of day showed a distinctive effect on the performance of the students.

When running the network, the biases of the linear layers allow for the network to change its performance and distinguish between data sources. Added biases allow the network to perform better on group statistics. Operating the network without biases allows the network to show differences between the two data source networks.

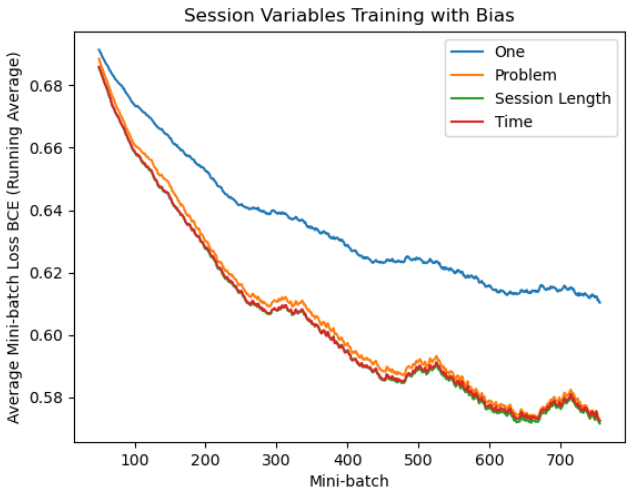


Figure 42. Training for Session Variables with Bias

Graph Showing the Difference Between Different Session Variable Networks During Training, the Linear Layers Operated with Bias

With biases the training curves become almost indistinguishable even after 700 mini-batches. Figure 42 shows them right on top of each other. Other experiments showed the data should start to separate before reaching 700 mini-batches. All of the two data source networks uniformly performed better than the one data source, but it becomes difficult to state one outperforms the others. The networks approach asymptotic performance after about 700 mini-batches. All of the curves dip and climb which implies

they struggle with the same mini-batches. After removing the biases different behaviors appears.

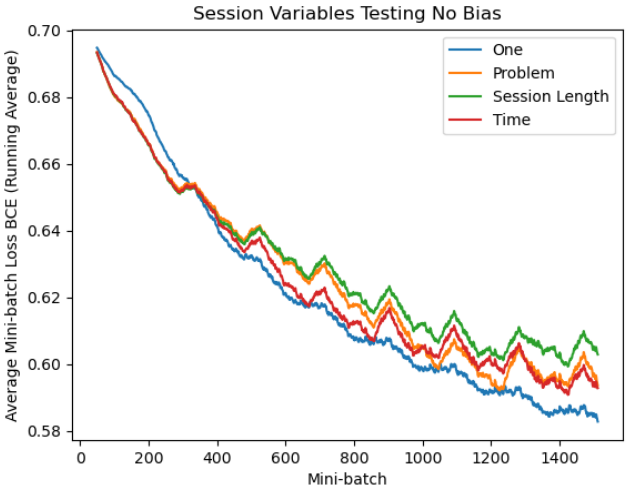


Figure 43. Training for Session Variables with no Bias

Graph Showing the Difference Between Different Session Variable Networks During Training, the Linear Layers Operated with no Bias

Without the bias the data visibly separates. Figure 43 shows a considerable separation between the performance of networks. In the first 400 mini-batches the loss values for the two data source networks stay together. After 400 mini-batches the differences in performance become apparent. On the training data the two data source networks do not perform better than the single data source network. Each network shows similar difficulties on the same mini-batches

After running through eight epochs on the training data, a clear distinction between the single and double data source appears. On the training data the single source

shows a better performance than the two data sources. The network appears to approach their asymptotic performance values.

When running the network on the test data the inclusion of biases should allow for an improved performance at the expense of distinguishing the networks. All of the networks with biases outperformed the single data source network, and all of the networks without biases performed worse than the single data source networks. With the biases the networks show the expected behavior.

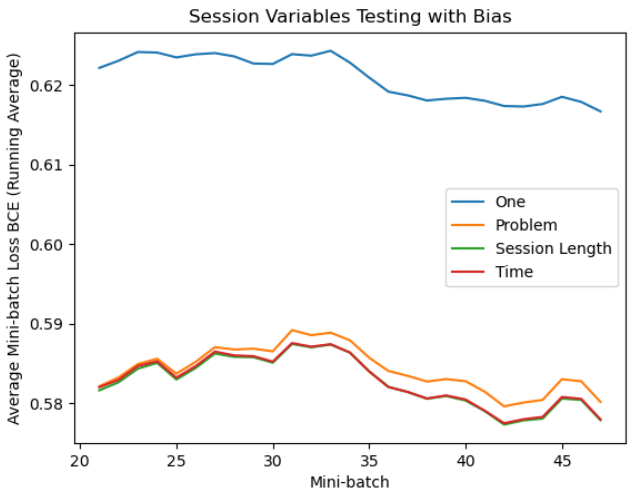


Figure 44. Session Variable Testing with Bias

Running the Trained Variable Networks with Biases on the Test Data

Uniformly the networks with biases performed better than the single data source network on the testing data. Figure 44 shows them with a much lower loss. The networks performed similarly to their performance on the test data. Their grouping also stayed

similar with the training data. The problem number network performed worse on the data, but the session length and time of day networks performed better than the problem number session.

The network without the bias should show a worse performance than the one data source network on the test data. It also should perform worse than networks with biases. At the expense of this performance the networks should show different performances on the test data.

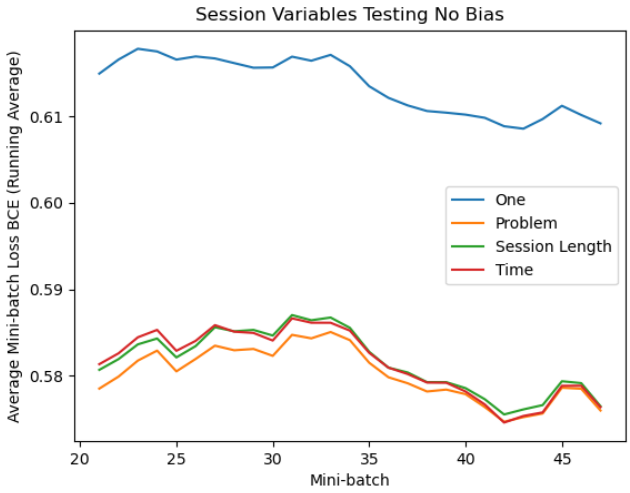


Figure 45. Session Variable Testing

Running the Trained Variable Networks on the Test Data

The results of the network on the test data show a similar performance to the network with the bias. Figure 45 summarized their results. All the networks with two data sources performed better than the network with one data source. The networks with two

data sources also uniformly struggled on the same mini-batches. Without the bias larger differences between the curves appear which matches the performance on the training data.

Based on the performance on the test data the networks with two data sources should outperform the networks with one data source using the metric of AUC. In previous work using state of the art logistic regression on pieces of the data showed an average AUC of around 0.8 (Schmucker et al., 2022). As the embedding dimension, hidden dimension, data sources, and bias dropout rate were not tuned on this data the previous value should serve as an upper bound of performance.

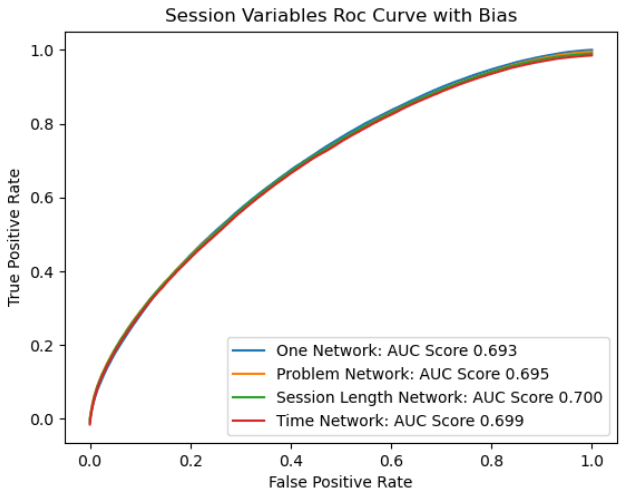


Figure 46. Session Variables ROC Curve with Bias

Comparison of All Session Variable ROC Curves and AUC Scores on Test Data

The ROC Curves showed a better performance for all second variable source networks, however Figure 46 shows little difference in their curves. The performance overall looks very similar. As the Autograd focused on minimizing the loss and not maximizing the AUC different training methods could improve this metric.

Without bias the previous loss graphs imply the AUC curve should show more difference between the networks while retaining similar performance. The one data source network should also change its performance as removing the biases will change the architecture of the network.

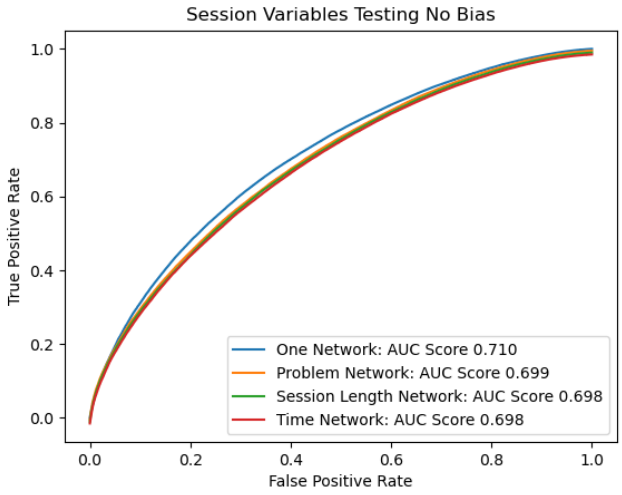


Figure 47. Session Variables ROC Curve without Bias

Comparison of All Session Variable ROC Curves and AUC Scores on Test Data

Without bias the one data source network performs the best. It performs better than all pervious networks. Figure 47 shows the single data source network with a much

higher AUC score than the other networks. The addition of bias also seems to group the networks together. The performance of the different networks shows less difference than previously.

When using networks with a second data source of session variables, removing the bias showed the best performance. Including biases reduces the loss while improving the AUC values. Using dropout on the biases may allow for differences between the networks while maintaining the increased performance.

5.3.2 Number of Problems in an Exercise with Bias

The data implies the number of problems a student completes in the exercise could affect the performance of that student. As a student works longer, fatigue, frustration, and probability of quitting should affect the probability of a student getting a right answer and continuing. These effects should also depend largely on the individual.

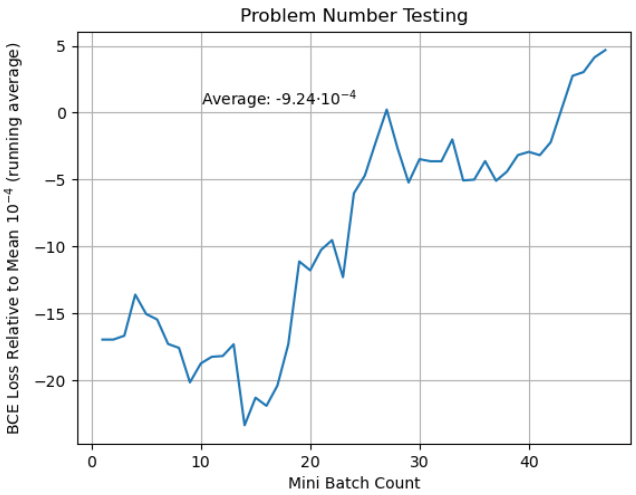


Figure 48. Test with Problem Number Second Network

Difference Between Problem Number Network and Average of All Two Data Source Session Variable Networks

After removing the average performance, the loss of the network with the problem number second data source appears lower than the average. The graph in Figure 48 displays an almost uniformly negative value. It shows the greatest increase in performance during the mini-batches where the two data source networks found prediction difficult. The improvement does not look universal as at some points its loss becomes greater than the average performance.

5.3.3 Session Length in Minutes with Bias

Although the data did not show any relationship between session length in minutes and a reduction in performance, individual students may exhibit behavior outside of the data summary. This second data network differs from the problem number because

the data did not show a difference in performance. As before, individual variations may exist, but not present themselves in the summaries of the data.

On the training data it showed a very similar performance to the other networks and approached the mean very quickly. Small differences in performance existed while training but after 50 mini-batches these differences broadly disappeared.

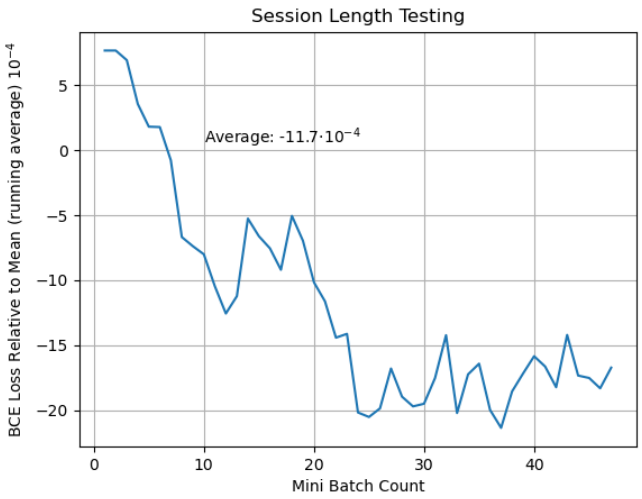


Figure 49. Test with Session Length Second Network

Difference Between Session Length Network and Average of All Two Data Source Session Variable Networks

On the test data this network showed the best performance of the networks considered. Figure 49 showed the lowest loss relative to the other networks. It performed best when mini-batches maintained a uniform difficulty of prediction. It did not universally perform better than the other networks but performed better on average. The

session length and problem number in exercise networks are relatively comparable with a percent difference of less than 25%.

5.3.4 Time of Day during an Exercise

From the data, time of day showed the greatest promise as a second data source. From the graphs in the previous chapter performance clearly dipped for all students in the middle of the day, and at the end of the day for younger students.

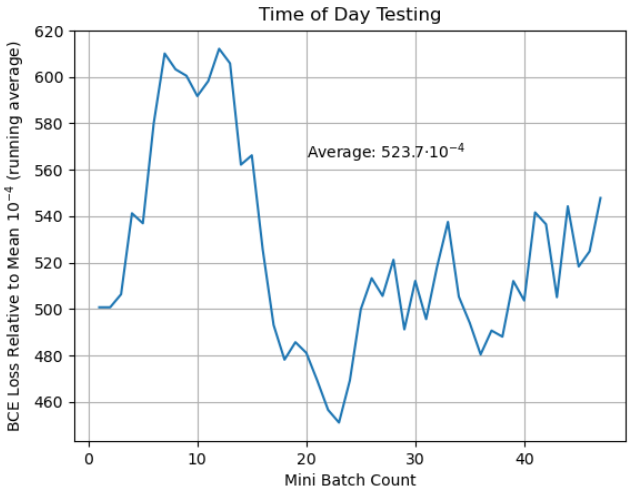


Figure 50. Test with Time of Day Second Network

Difference Between Time of Day Network and Average of All Two Data Source Session Variable Networks

Time of Day performed the worst of all the second data sources from session variables. Figure 50 showed an overall positive value for loss relative to all the other networks. Its difference from the average appears on a different scale than the others.

Although this network does not perform as well as the others, a second data source with the large variations seen in the data may need more specialized training than the other networks.

5.4 Database Variables and Parallel Networks

In the last section, using the situation during which a student attempted a problem clearly affected the performance. In this section the network used the characteristics of the users to determine their effects on performance.

Similar to the reduced scores used in the networks with session contextual data, the networks with user contextual data used reduced scores. The reduced scores rely on the same assumption, the main predictors of performance are the user and the problem they attempt.

The motivation for the research in the previous section focused on the ability to keep a student productively working given the context of a session. The network could give individualized suggestions to maximize the effectiveness of the session on the individual level. In this section the motivation doesn't look to maximize the individual but to get a uniform performance amongst all of the students. Based on individual characteristics problem selection can achieve the same curriculum goals but without the unintended consequences of asking a question in the wrong way.

5.4.1 Performance of the Database Variables

The database variables examined in this experiment included gender, and city. These variables indirectly measure complex differences between the users. In the data these showed the greatest differences in student performance.

As the session variables focused on attributes common to all individuals like fatigue, and networks with bias showed good group statistics, database variables show theoretical mechanisms for improvement and reduction for performance. Database variables could allow networks adapt to the individual and maintain good group statistics, or they may effect the networks ability to make good predictions on the group.

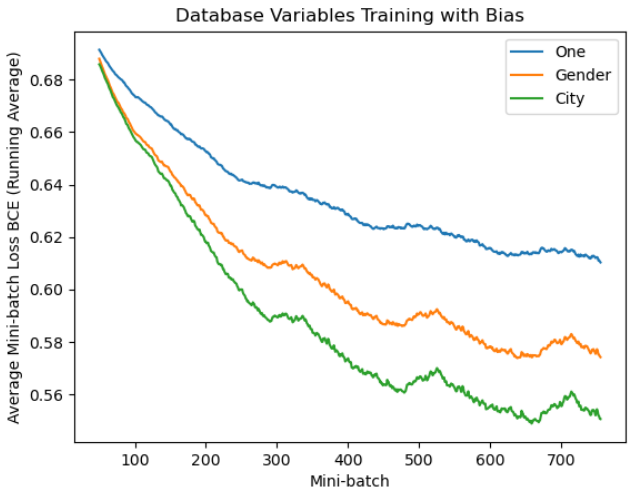


Figure 51. Training for Database Variables with Bias

Graph Showing the Difference Between Different Database Variable Networks During Training

On the training data the database variable networks showed a large difference in performance. Figure 51 shows the differences. Both networks performed better than the one data network, but additionally a large difference in the two networks becomes apparent between the two data source networks. The networks with session variables did not show a clear distinction between data sources.

The curves also separate very early in the training which did not appear in the session variables. Before 100 mini-batch a difference between the curves appears. A large difference in performance persists until the end of training. Given the difference in the training data the testing data should show a similar difference.

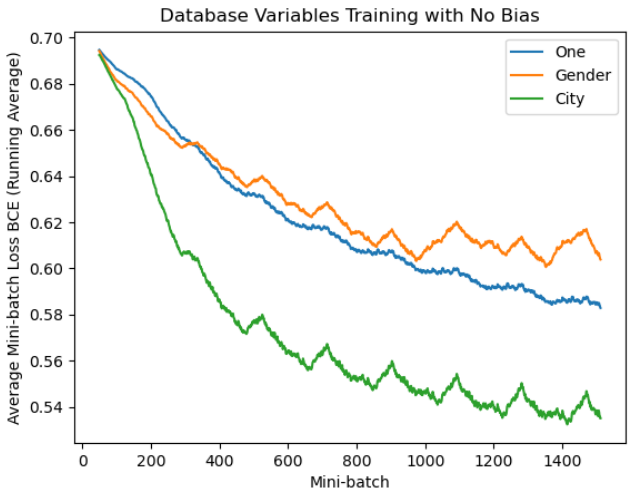


Figure 52. Training for Database Variables with No Bias

Graph Showing the Difference Between Different Database Variable Networks During Training

After removing the bias some consistencies seen in the training return. Figure 52 appears to keep the characteristics of Figure 51. The reduction in performance returns with a visible difference in between the two data source curves. The city network however, retains an improved performance even with the bias removed.

With the bias the two data sources should show an improved performance on the testing data with little difference between the performance of the networks. The session

variables showed that behavior throughout training, and testing. The differences they showed also only appeared after removing the bias.

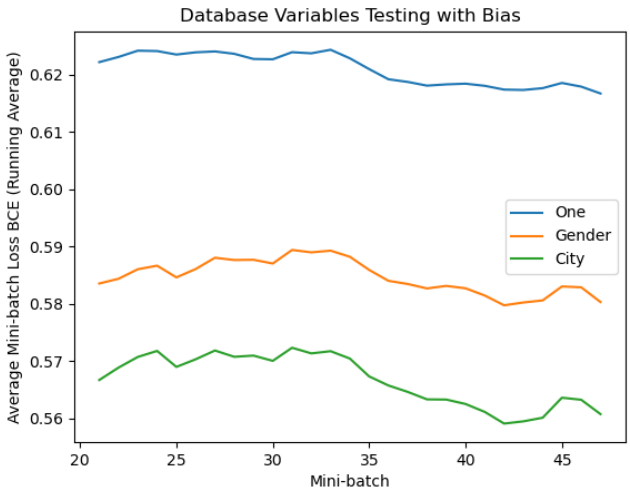


Figure 53. Database Variable Testing

Running the Trained Variable Networks on the Test Data

As before the two data source networks showed an improved performance on the testing data. In Figure 53 the second data source networks find much lower average values for loss. City showed a better performance that persisted though the training data. With no bias the difference between the networks differs from the session variables. As a second data source measurement of city looks like a unique metric.

Without the bias the session variables showed an improved performance even with the reduction in performance while training. On the testing data the database

measurements without bias should show similar results. The certainty of this expectation is less, due to the unusual performance of city measurements.

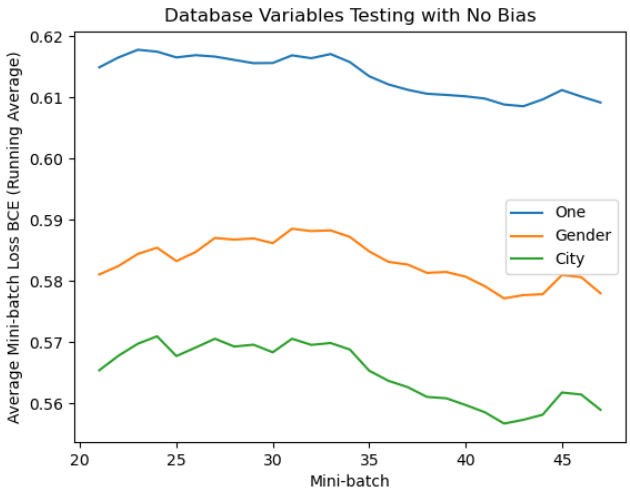


Figure 54. Database Variable Testing

Running the Trained Variable Networks on the Test Data

Without the bias the difference of performance on the testing data remains about the same as shown in Figure 54. The bias does not appear to affect the prediction of networks. Universally the two data source networks made better predictions with the same difference between the networks as seen in the session variables.

On the ROC curve the one data source performed better with the inclusion of a bias. Overall, the session variables showed the best performance from the one data source network and no bias. With a bias the one data source did not perform as well with the bias.

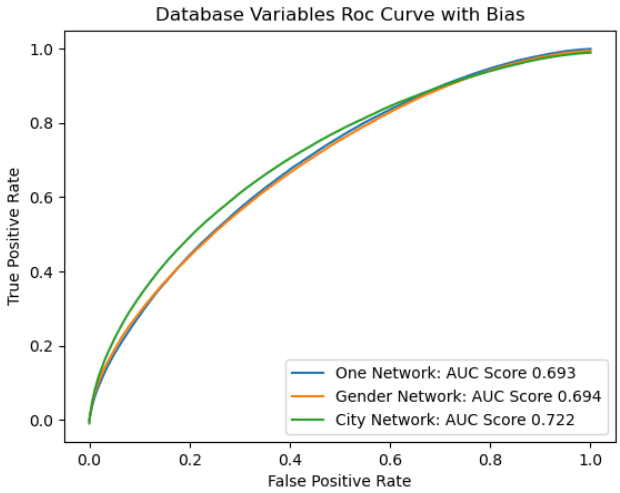


Figure 55. Session Variables ROC Curve

Comparison of All Session Variable ROC Curves and AOC Scores on Test Data

With the bias the city network shows a better AUC than any network looked at so far. In Figure 55 the first clear distinction in the AUC curves occurs. As before all the two data source networks performed better than the one data source. The difference of the curves in this situation however makes the database variables much different than the session variables.

With no bias the one data source curve should perform better than all of the curves. Prior to running the network on the city data, the one data source without the bias showed the best AUC. Before looking at the performance the expectation of performance remains unclear.

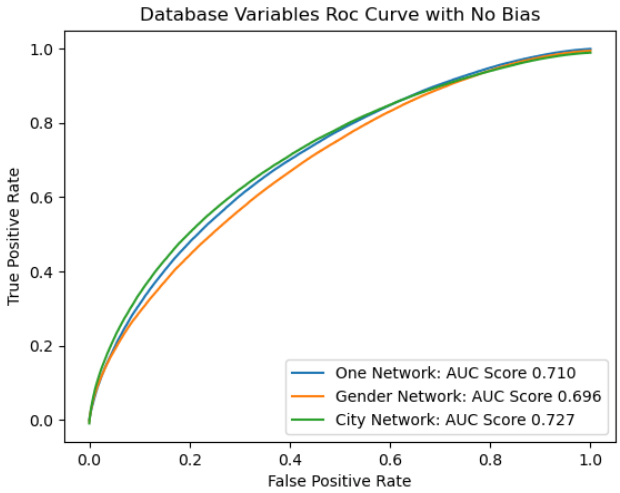


Figure 56. Session Variables ROC Curve

Comparison of All Session Variable ROC Curves and AOC Scores on Test Data

The City Network without bias showed an even a better performance than with bias. Figure 56 exaggerates the difference in performance first observed in Figure 55. Its AUC score rose above the one data source network and all of the other networks. It appears as a special measurement for a second data network, and clearly improve the prediction of the network.

5.4.2 Gender

A large portion of the data came with no indication of gender. Even with the reduced dataset, the possibility of improving the prediction of student performance makes the experiment worthwhile. Gender remained one of the most visible differences in performance across all grades.

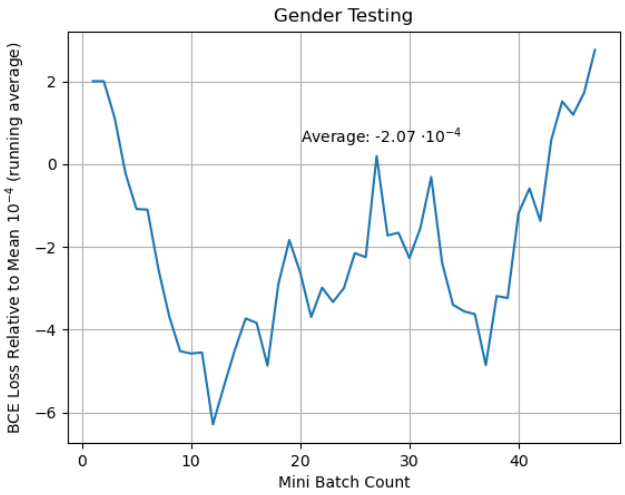


Figure 57. Test with Time of Day Second Network

Difference Between Gender Network and Average of All Two Data Source Database Variable Networks

Gender showed the smallest loss of the two database variables. In Figure 57 it maintains a negative value almost though out its domain. It did not uniformly outperform the average of the database variables, but its average was below the mean. Its improvement in performance does not follow the average performance.

5.4.3 City

The cities changed the data most dramatically out of any of the variables looked. The data shows enough intricacies that it looks difficult to model with any methods. Some cities performed better or worse on average, but when separated by city particular grades began to show differences.

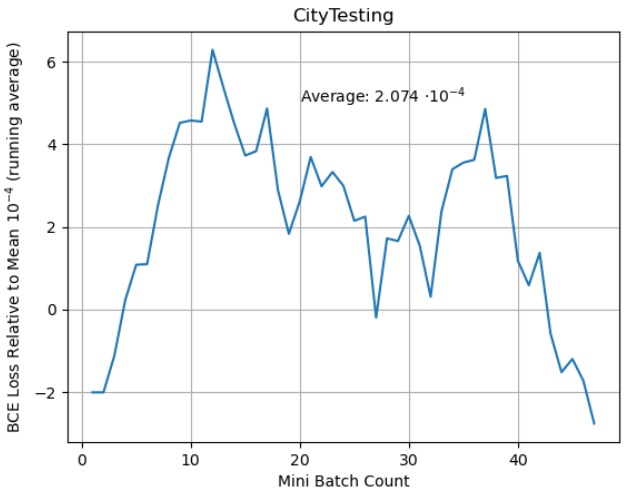


Figure 58. Running Average of Performance with City Second Network

Difference Between City Network and Average of All Two Data Source Database Variable Networks

City showed the lowest performance of the database variables, as shown in Figure 58. The increase in the loss function also follows the pattern, as the details of the dataset increase, the network struggles to reduce loss. Although it's tempting to compare the performance of the time network and the city network, a different average was subtracted from each loss. To compare the performance, a direct comparison will still need to occur.

5.5 More than Two Parallel Networks

One of the benefits of the network construction includes the ability to quickly include additional parallel networks without any additional coding. The LSTM object only requires the number of networks stated upon instantiation and additional data columns of data given to it during a forward pass.

The issue with increasing the network complexity becomes the training of the additional embeddings. The network may give better results, but it may come after several additional epochs of training. Allowing one network to receive extra training does not make it comparable to other networks.

5.5.1 Number of Problem and Time of Day Network

The specific coding allows the addition of networks seamlessly. After looking at the performance of different variables, curiosity would dictate looking at the performance of three variables. The session variables so far measured how long a student worked, and when they worked. For the three data source network, the two additional data sources should represent both categories of measurements.

In the previous experiments, as the complexity of the data increased the performance reduced. For the three data source network the performance should reduce under the assumption three data sources contains more complexity than two data sources. However, the three data source network contains an extra embedding. Adding an embedding showed the greatest increase in performance so far.

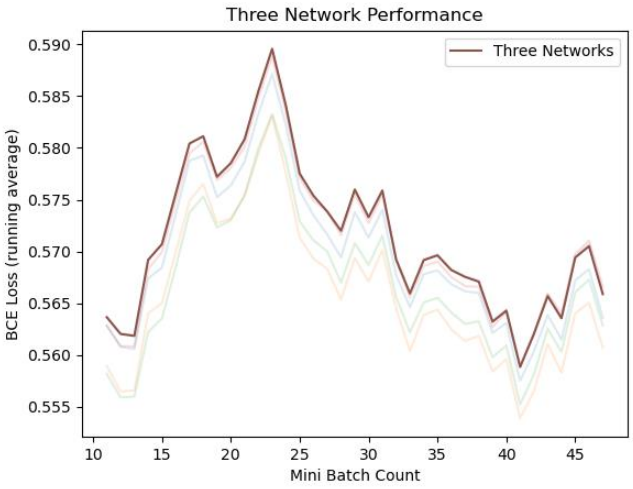


Figure 59. Test with Time of Day Second Network and Problem Number Third Network

Graph Comparing all Two Data Source Networks and a Three Data Source Network

The three data source network performed about the same as the two data source. Figure 59 only shows a minor difference. Adding one embedding may exhaust the benefits from adding embeddings. This may change when the data comes from two sources that represent the same measurement. The complexity of the three data sources however did make the three data source network perform worse than the two data source network.

5.6 Comparison of Networks

After removing the one network LSTM, the difference between the networks appears. Taking the average value for their loss gives a good comparison of which network performed best on the test data.

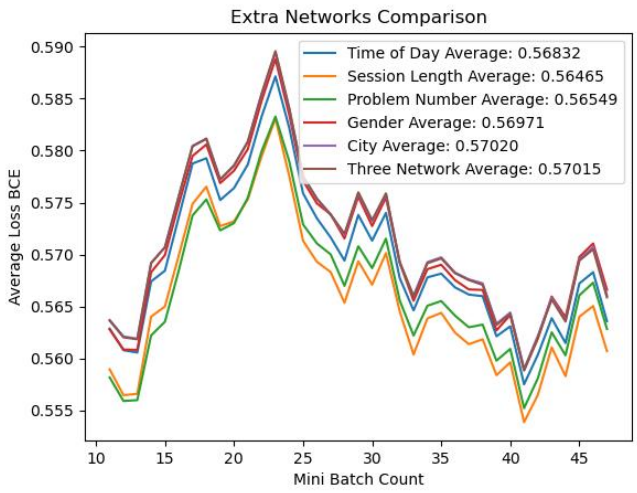


Figure 60. Test with Time of Day Second Network and Problem Number Third Network

Comparison of All Networks

The session length gave the best prediction on the data as a second data source. As speculated before, combining the session variables that the embedding struggles to encode did lead to the best prediction. Figure 60 confirms this suspicion. Students do form habits of how many problems they perform in a row, and the encoding can contain the overall effect, but that does not affect the prediction on one individual problem. The

second most predictive variable came from the problem number in the exercise. These both come from session variables describing the current stage of the session and not the long-term characteristics of the individual.

5.6.1 Complexity

Complexity of the data source remained the number one predictor of the network performance. As the data contained more complexity, the network reduced in predictivity. City contained some of the most complex data and performed the worst. When deciding which data source to use a consideration of complexity should be made, if the training time will remain the for all networks.

Chapter VI.

Implementation of Network in Practice

So I live in a friendly neighborhood and everyone really likes to talk to their neighbors. So This new guy moved into my neighborhood and I as a friendly neighbor introduced myself. "Hi I'm Norm McDonald" so he says something, something, I don't remember his name. He asks, "What do you do?" I say, "I'm a comic." and I ask him "What do you do?" and he says "I'm a professor of logic at University of something." I don't remember the name that well. So I ask "What does it mean ? I mean what do you do?" so he says "It includes syllogism of something, something," and everything's going over my head so he says, "Let me show you with an example."

—Norm Macdonald

The discussion so far consisted of abstract, highfalutin, complicated language, so the time has come to implement the network to recommend problems. Some issues will still require rigor, but the goal of this chapter exists to discuss how to use the network.

6.1 Cold Starting

Cold starting refers to the difficulty a network faces when it makes predictions for users for the first time. The network will struggle because it has no measurements on the users outside of session variables.

The weights for the problems and users will start randomly in the embeddings, so the network will not become predictive until it encounters a significant about of data. Setting the weights forms a feedback loop where putting data into the networks will alter the predictions and direct the creation of new data for the network.

The data this network ran on consisted of csv files of measurements. Simulating these files should set the weights. Given a model for students and a model for the problems, a few lines of code should be able to create the files. Modifying the models can create more realistic csv files but as the weights update once a time step, this seems unnecessary. After determining the number of timesteps needed to set a user's weights, the weights at this timestep in the simulation form a good initial value for a new user.

6.1.1 User Embeddings

When the LSTM encounters a user for the first time it will randomly place them in the embedding. The position of the user in the embedding will summarize the preferences and characteristics of the user from measurement. Due to its random placement the network will need to move the user's embedding though back propagation before the network begins to make accommodations. This will require the network to make measurements on the user while it operates in an unintelligent way. Depending on the length of this period the user may lose faith in the learning platform and stop attempting problems.

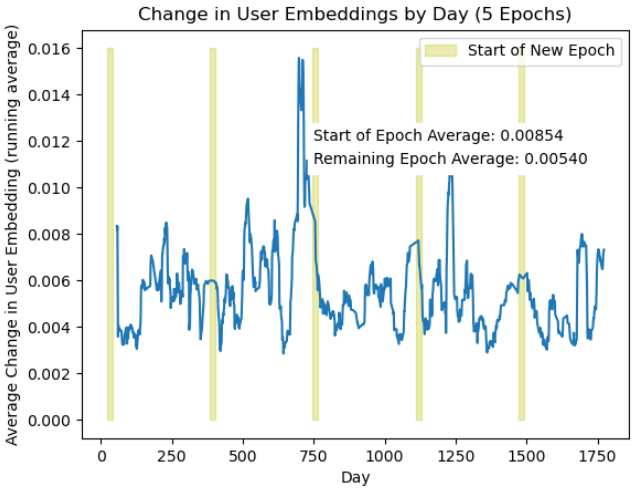


Figure 61. Test with Time of Day Second Network and Problem Number

Changing of the Weights for Each Forward Pass in the LSTM Network

Although the background noise remains large the user embeddings change more on average during a new epoch. Figure 61 shows the increased movement of the embeddings during the beginning of the epochs. The changes in the user embeddings increased 58% during the first 20 days compared to the change in the embeddings during the remainder of the epoch. The network encounters users for the first time at the start of the epoch, so an increase is expected. The change of the embeddings during the remainder of the epoch however appears quite surprising. The embeddings keep moving even after the initial period of encountering the new users.

6.1.2 Easing Cold Start with Additional Data Sources

Previously additional data sources appeared redundant as the user embeddings should summarize the measurements of the individuals. These variables however may

reduce undesired behavior during cold start. Upon acquiring a new user session and database variables may reduce the extent of the undesired behavior. For instance, the database variables measured in the dataset studied, would normally get encoded into the embedding. They were also pieces of information that a user would need to volunteer prior to using the platform. Including additional measurements prior to a cold start would help alleviate it.

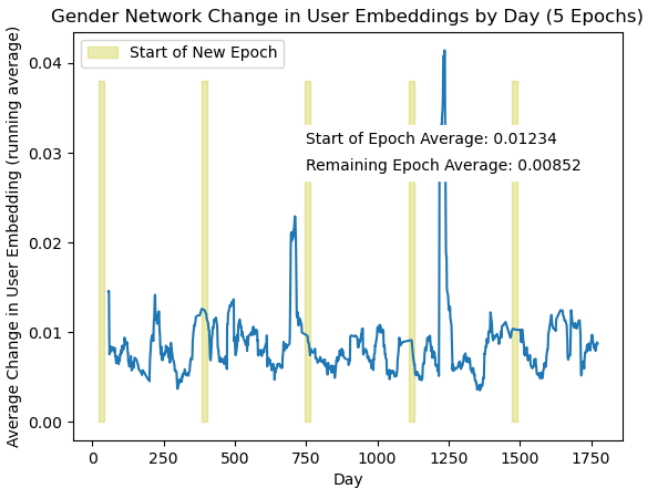


Figure 62. Test with Time of Day Second Network and Problem Number *Changing of the Weights for Each Forward Pass in the Gender Network*

When looking at gender as a second data source, the change in the user embedding increases more in the beginning of the epoch as well as the remainder of the epoch. Figure 62 summarizes these results. The effect of the second data source seems to increase the gradients throughout the epoch. The percent increase of the daily change in

the embeddings though remains smaller. Day to day the user embeddings change 45% during the first 20 days of the epoch.

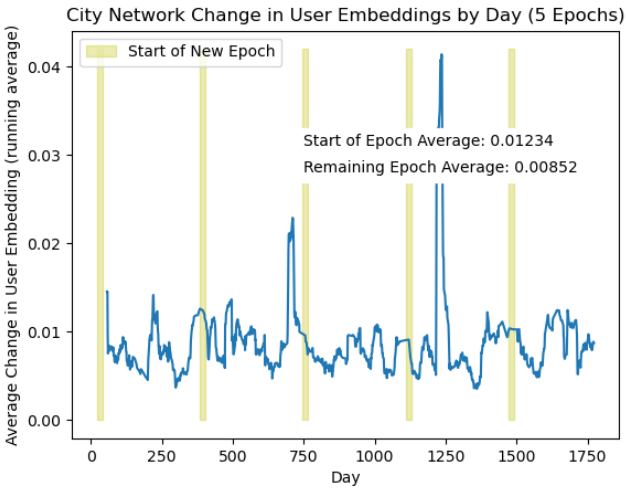


Figure 63. Test with Time of Day Second Network and Problem Number *Changing of the Weights for Each Forward Pass in the LSTM Network*

The city second data network shows a similar behavior to the gender network, as shown in Figure 63. Even with the large amount of noise in the data, the change in the embeddings decreases during the first 20 timesteps. It appears in all three networks reduce the change in user embeddings as time goes forward, but these changes do not go to zero. Continually changing embeddings could imply the development of a student so their continued movement does not necessarily imply the network struggles to fit the user.

From these networks it appears that the user embeddings take about 50 timesteps to settle down. To alleviate cold start, it may make sense to use this embedding for a new

user. With simulated data, after about 50 timesteps the user embeddings would reach values that could be used for the initial embedding of new users.

6.2 Predicting Standardized Test Scores

One of the interesting things about the initial period when user embeddings move around, is the long-term prediction after the user embeddings relax. The network predictions on problems toward the end of the course did not move around after the first 50 timesteps. The network understands the user very quickly and applies their characteristics to the remaining curriculum. Inside the first quarter of the school year the network could make predictions for the remainder of the year. This makes the network a good candidate for early intervention, and early estimations of standardized test scores.

6.2.1 Problems without Instruction

As discussed earlier, the resources given with a problem may alter the performance of the students. The data set used in this research only contained problems with videos. In addition, the students did not attempt problems out of order. In the beginning of the course the students did not jump to problems in the end of the course. No data to model a drop off was fed into the network. For the prediction to remain relevant in practice students need to receive the same level of instruction throughout the entirety of the course.

6.2.2 Prediction After *n* Days

After *n* days the prediction of performance in the previous days should converge to their asymptotic values. The model does not get any new data for the previous

problems, and the original prediction should stay static. The user embeddings do continue to move throughout the course, and they may change the prediction of previously measured data, but this should not lead to a large change.

To test the performance of the network on previously measured data, the network ran though one mini-batch and stopped periodically to make a prediction on the problems in the beginning of the course. One mini-batch roughly equates to one class of 20 students. To record the day the problems occurred this test used their average timestamp. As the dataset used a library of problems students could do them on different days or do different problems entirely. To find the predicted performance on a given day, this test collected all the problems with average timestamps on that day, and averaged all problems over all users.

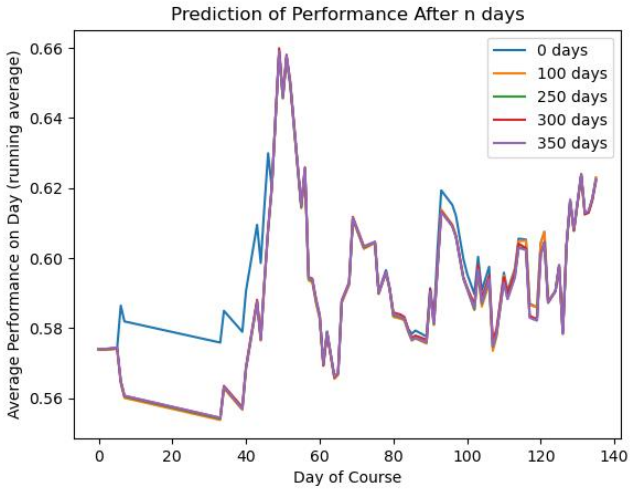


Figure 64. Test with Time of Day Second Network and Problem Number

Predicting the Performance of the Mini-Batch in the Beginning of the Course After n Timesteps

The ability of the LSTM cell to model the performance of students with no measurements is quite extraordinary. In Figure 64 the curves change very little after receiving more measurements. The predictions only show a minor difference between the predicted performance over the beginning of the course after 0 days and after 350 days. After going through the problems once the model maintains an almost uniform prediction on previous data for the remainder of the course. As the user embeddings move continuously, small differences in the prediction of first 100 days occurred during the later units of the course.

To test the performance of the network on long term predictions the same test occurred but with predictions over the entire course instead of the beginning of the course. As the network records more data, the later predictions should update as the course goes forward.

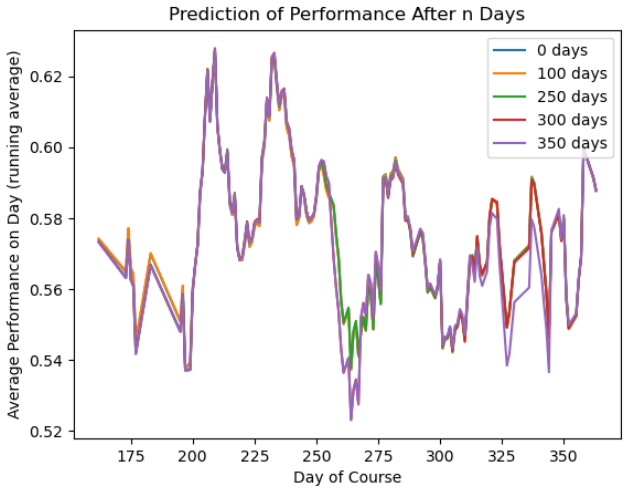


Figure 65. Test with Time of Day Second Network and Problem Number

Predicting the Performance of the Mini-Batch at the End of the Course After n Timesteps

The prediction of performance in the last units of the course does not change in a meaningful way as the network gets more data, as shown in Figure 65. In fact, the predictions before the network acquired measurements of the students and after the network made several hundred days of measurements do not differ significantly. As the LSTM cell trained on several hundred mini-batches of training data, it does a great job representing the average performance of the students.

When looking at the individual performance a similar pattern emerges. The network decides very early about the performance of the student in the later units of the course. As the performance of a student may stay uniform trough the entirety of the course these predictions do not seem unusual.

Figure 66. Test with Time of Day Second Network and Problem Number

Predicting the Performance of One Student At the End of the Course After n Timesteps

The predictions on the user appear like the predictions on group. The graphs fall right on top of each other in Figure 66. They update very little as the network collects data, and the prediction mostly depends on the LSTM Cell. The curve also matches the graph from the average group prediction. This implies the difference between the group and individual prediction does not differ dramatically.

To determine the difference the network gives to different users the background prediction needs removal. After subtracting the mean of the predictions, the distribution about the center of the curve appears.

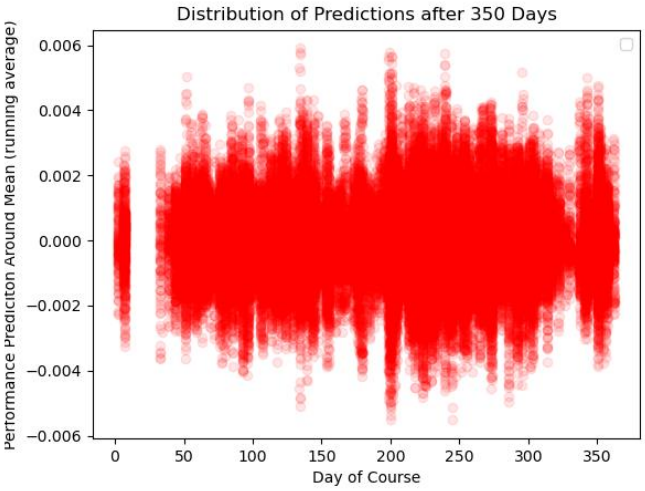


Figure 67. Test with Time of Day Second Network and Problem Number

The Distribution of the Prediction About the Mean n Days into the Course

In the distribution you can see the effect the user embedding on the prediction, as shown in Figure 67. For this mini-batch the prediction differed about 1% for the students. This makes the bulk of the prediction coming from the LSTM Cell and user embeddings contributing to a third order correction to the probability. This may seem like a small difference in prediction, but the user embeddings do not limit how large they can grow. If larger differences fit the data better the data better the network could accommodate them.

Due to the quick prediction of the network, this makes it an ideal candidate for early intervention. With about 50 timesteps of data the user embeddings relax and a prediction of student performance stays the same during the remainder of the course. Additionally, as the network can make predictions though the entire course, weighting the predictions by their probability of occurring on a standardized test gives a prediction for the performance of a student on that test.

6.3 Choosing Problems

The LSTM gives useful predictions and can find problems suitable for each student. These were some of the major concerns for differentiation. This network can find the best problems for the individual while not increasing the workload on an instructor.

Although the LSTM solves two of the major problems in individualizing instruction, it still does not give problems to the user. To give exercises to the user, the LMS will still need an additional algorithm to use the network.

6.3.1 Utility Functions.

Utility functions give scores to items based on predetermined priorities. They can focus on one priority or balance several priorities at the same time. For this application, improving the students score on an end of year test could provide a balance of prorates needed for the basis of the utility function.

Consider for example the AP Calculus BC multiple choice test. From the course description AP gives the distribution of problems. A simple function to choose the topic of the next question could consist of picking the earliest unit the student historically performed under a certain threshold and choosing a problem at random from a list determined by the LSTM. This should also allow a new user to quickly find the appropriate difficulty of the problem during their session. As the AP test consists of many problem types in each unit, the units could consist of even smaller subsections to assure even coverage.

Exam Weighting for the Multiple-Choice Section of the AP Exam

Units	Exam Weighting (AB)	Exam Weighting (BC)
Unit 1: Limits and Continuity	**10–12%**	**4–7%**
Unit 2: Differentiation: Definition and Fundamental Properties	**10–12%**	**4–7%**
Unit 3: Differentiation: Composite, Implicit, and Inverse Functions	**9–13%**	**4–7%**
Unit 4: Contextual Applications of Differentiation	**10–15%**	**6–9%**
Unit 5: Analytical Applications of Differentiation	**15–18%**	**8–11%**
Unit 6: Integration and Accumulation of Change	**17–20%**	**17–20%**
Unit 7: Differential Equations	**6–12%**	**6–9%**
Unit 8: Applications of Integration	**10–15%**	**6–9%**
Unit 9: Parametric Equations, Polar Coordinates, and Vector-Valued Functions BC ONLY		**11–12%**
Unit 10: Infinite Sequences and Series BC ONLY		**17–18%**

Figure 68. Test with Time of Day Second Network and Problem Number

Exam Weights from the AP Calculus Multiple Choice Test as Given By the Course Description

After looking at the table of weights in Figure 68, a simple utility function does not seem too farfetched. The table already determines the importance of different units. With the additions of algorithms combining everything together the

6.4 Maintenance

Once a timestep the network needs to go down while the weights update. While the network goes down the LMS will need to manage any incoming requests. Based on the length of the downtime the prediction may need to temporarily use a divert to another algorithm while the weights update.

6.4.1 Backward Pass

During the experiments run, the network updated once a timestep. It collected the data since the last update and used the data to update the weights. The network split the data into different timesteps every day at 3:00 TW. This allowed the network to learn about new users and set the user embeddings.

For the data from the dataset a mini-batch took about 30 seconds to run. This would consist of doing the forward passes, and backward passes required for 365 days with 20 users. For several thousand users this could become a lengthy update process. To assist in the process the LMS could queue the back passes and load problems based on a failsafe algorithm to keep running while updating. As the data showed a down period in the early morning, the queue should be able to handle updating the weights during this period.

6.4.2 Adding Users and Problems

Tensors exist to hold the embeddings for all the problem and users. As the LMS adopts users these embeddings will need to expand. The indices of the tensors refer to specific users which requires care when placing the user into the tensor. A similar problem occurs with the addition of problems.

The GPU also does not handle errors in the same way the CPU does. Small errors can cause crashes and the output will give only rudamentary error tracing. As the predictions stay stable between timesteps keeping a copy of the model in memory will help with redundancy.

6.5 Phaidu

The actual implementation consists of the webpage phaidu.com. It runs on a t2.large EC2 instance from Amazon AWS. It combines the network implemented in Pytorch with a webserver by using Django. A MySQL database handles the database needs of the website. Together these technologies allow a student to work their way through the AP Calculus BC curriculum.

6.5.1 EC2

In the development of this network keeping a GPU no longer remained economically feasible. Although AWS does offer GPU instances, their monthly cost for using an instance with a GPU as a web server makes it prohibitively expensive. Pytorch does however allow CPU only usage. This makes a good substitute until resources become available to incorporate a GPU. Depending on the device that stores the tensors Pytorch will operate on either a CPU or GPU. The only modification to network for GPU usage comes from compiling Pytorch with functionality for GPUs, and storing the devices on the GPU.

Several EC2 instance types can run webservers with a MySQL database. Incorporating Torch into the RS increases the disk requirements beyond the specifications of the lower tier options. The size of instances that can include Torch far exceed the needs of a simple webserver. This means the addition of Torch increases the cost beyond the cost of a basic server needed to run a website.

The upside of an EC2 instance however far exceeds the difficulties meeting the specifications of Torch. The disk keeps multi-regional redundancy, the network stays up

at a high rate, and the hardware quickly upgrades. Given the nature of Torch to quickly incorporate a GPU, EC2 instances make an ideal choice for hosting a Torch based RS.

6.5.2 Django

The current implementation of the network requires Python and the standard Apache, NGNIX, or Node.js servers will not directly integrate Python scripts. Php, and JavaScript allow options for running command line statements so it is possible to run scripts, but it would be simpler to remove communication between the server and the network. Using a different language for the Torch library would work but requires coding the network in another language. In the end the app needs to choose between tools front-end developers find more useful, or tools network researchers find more useful.

Django adequately performs basic webserver functions, so it makes the quickest jump from development to market. The network code can exist with the webserver and import itself while instantiating the server. This implementation of the app uses the LSTM network as a global variable created when the server defines the functions for rendering the webpages. This stores the network in memory while the server runs. To prevent interruptions in service the network could occasionally preserve itself using the pickle module of Python and reload its previous state when server restarts. Given the utility of Python, Django will allow the network to run with the server and survive interruptions in service.

6.5.3 MySQL

Using MySQL as a database allows for familiar methods of user verification, and security. It survives restarting the EC2 instance and makes for a robust option. There also

exists resources online for preventing MySQL injections and cross-site attacks. From a security standpoint, it is well researched and stable. As shown in Figure 69 a MySQL table can accommodate the need for storing login and user data.

```
mysql> describe Users;
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| email         | varchar(255)  | YES  |     | NULL    |       |
| password      | varchar(255)  | YES  |     | NULL    |       |
| resetPassword | varchar(255)  | YES  |     | NULL    |       |
| id            | int           | YES  |     | NULL    |       |
| lastProblems  | varchar(1024) | YES  |     | NULL    |       |
| randomInt     | int           | YES  |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
```

Figure 69. Description of Table Containing User Data

The Output of the User Table Description from the Users Database

```
mysql> describe Problems;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| id       | int          | YES  |     | NULL    |       |
| question | varchar(1024) | YES  |     | NULL    |       |
| answer   | varchar(255)  | YES  |     | NULL    |       |
| topic    | int          | YES  |     | NULL    |       |
+----------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Figure 70. Description of Table Containing Questions

The Output of the Problem Table Description from the Problems Database

Other properties of MySQL also make it ideal for the distribution of data. The segmentation of repositories into databases allows for MySQL to separate user

verification, and problem information. In this app separate databases housed the tables associated with problems, and the tables associated with users. The problem statements, solutions, topics, and ids came from rows in the Problems table. The user emails, passwords, ids, and history of problems came from the rows in the Users table. Figure 69 and Figure 70 show the summary of the tables.

In addition to segmenting data, after the incorporation of the GPU MySQL balances the workload between the CPU and GPU. Although Python allows for confining user data to Torch tensors the app would lose the benefits of MySQL tables. The tables may also contain enough data for the app to act independently if the network fails. Separating the login information and the networks seems like the best option.

To add a fallback in case of an interruption in service from the network, this app keeps the last 100 problems a student answered. When the network fails the table gives randomized suggestions outside of the last 100 problems. A rudimentary function measures the last topic the student worked on, and their proficiency in that topic. The algorithm uses this topic to choose the problem. This makes the MySQL tables a great option for a fallback when the network goes down. Together with Torch, MySQL can make an excellent addition for delivering problems, and running the website.

Chapter VII.

Conclusion

And to this end they built themselves a stupendous super-computer which was so amazingly intelligent that even before its data banks had been connected up it had started from I think therefore I am and got as far as deducing the existence of rice pudding and income tax before anyone managed to turn it off.

—Douglas Adams

7.1 Summary

When I began this research, I wanted to know if a neural network can give a student the best possible math education by including additional data networks. I wanted to know if a neural network can give a student exercises, they find relevant, exercises that address their cultural background, and exercises that fit their exact skill level. After running the network and looking at the results, it became clear the network could find the problems students preferred, but the network opened more opportunities for research than answered questions.

Overall, the network shows the potential for implementation as an RS in a mathematics education environment. It would perform better than other current options such as static problems, or problems randomly coming from a database. The network can

factor in many additional student attributes and reduce loss functions using these additional attributes.

The network also shows potential for additional applications such as a tool to predict early intervention situations and include student interests. With a trained LSTM cell, the network quickly predicted the course outcomes for the students. This would give administration tools to forecast end-of-year results in a few weeks. Early in the semester students could still move to a more appropriate level before progressing too far into a course. Additionally, adding student interests addresses the nagging concern of applicability for the individual. Students would see more of their day-to-day life intertwined with math and be able to make intelligent decisions.

The development of actual app shows the technology exists to incorporate neural nets in predictions. Including future hardware beyond what cloud resources will currently provide affordably gives a solid future that the network. The future of the network will improve with the adoption of future technologies.

7.2 Future Research

After seeing the network respond to data, the door really opens for the primary goals of differentiation. It remains an open question if a neural network can identify the interests of students and how student interest can improve their performance. It also is not known how a network would reduce the cultural differences between students. It could make predictions on what language a student will respond best to or incorporate an entirely different model. How the network will deliver content also remains open. Although this network used an LSTM other type of networks may perform better after investigation.

7.3 Data

Even with the network working on real data and in a real application, how it will react to different datasets remains unknown. The data contains too many resources for each problem. Every problem the students encountered came with videos. The network never modeled problems students should fail.

Given a problem sufficiently ahead of a student's current progress in a course, the network should predict a lower probability of answering correctly. The network never successfully modeled this drop off. As the LMS from Junyi Academy gave the students problems at the appropriate time, the dataset does not contain data to model this effect. This dataset does not contain enough opportunities for students to fail. The problems level up and level down to match the ability of the user, the problems come during the appropriate time, and with instruction.

The problems also come labeled by mathematical topics and not interests. They even lack variable parameters to modify content based on user interest. This makes differentiating by interest difficult. The ability of this network to apply a student's interest into the curriculum remains unknown. Additional work by either labeling the problems by interest, or by using additional data needs to take place to determine this network's ability to make these recommendations. Given the individual differences the network makes during predictions without biases, it remains a candidate for including student interests in the curriculum.

7.4 LSTM

The LSTM showed very quick developing long-term predictions with little variance between students, or slower developing long-term predictions with significant

variation between students depending on if the network included biases. With biases the network quickly fit its weights. After fitting the LSTM cell during training, the predicted performance did not change as the network encountered new students. The LSTM would give almost static long-term predictions even when the students completed more problems. Removing the biases showed the effect of more variation of prediction between students. The adverse effect of the larger spectrum of prediction came from the LSTM taking longer to learn to predict individuals. As the network ingested more data it changed its long-term predictions of the students. Using the network as an early intervention indicator may not give enough of difference between students at an early stage.

7.5 Potential Uses

This research concentrated on students learning Algebra I and applied the research to the AP Calculus BC curriculum. The network focused on predicting when a student will get a problem right and continue studying. After fitting the network, the application used the network as a recommendation engine, and predictive mechanism for early intervention.

As the network gets better it may not just be used as a problem delivery system but also to evaluate the ability of students. As it stands, a test focuses on a single data point to analyze a students' ability. When the network achieves significant predictive ability, it may know how well the student will do over all the potential problems. This will give a better measurement of the student's ability instead of just one test.

Appendix 1.

Glossary

ADAM – Adaptive Moment Estimation. An optimization algorithm that extends SGD by including momentum.

Attention Mechanism – A layer that allows the inclusion of the entire user history into the current forward pass while weighting different parts of the history differently.

Bais – A matrix in a linear layer added to the product of the input and the transpose of the weights. It acts analogously to the y-intercept in a linear model.

CPU - Central Processing Unit. A computer component that performs the majority of computation. CPUs excel at performing a few complex computations.

Differentiation – A teaching method that adjusts instruction, assessment, and examination to meet the needs of individual students. It's a process that ensures students with different abilities and needs have equal access to learning.

GPU - Graphical Processing Unit. A computer component that specializes in processing images and rendering of 3D computer graphics. A GPU can perform many linear algebra processes at the same time.

LMS – Learning Management System. A software application for organizations to institutions to host, manage, and track learning programs.

LSTM – Long Short-Term Memory. A recurrent neural network that is well-suited for sequence prediction tasks and excels in capturing long-term dependencies.

Momentum – The behavior of an optimization algorithm to continue to update weights based off of previous updates even when gradients become small.

Parallel Data Networks – Two or more data sources that get fed into a neural network.

SGD – Stochastic Gradient Descent. An optimization algorithm that updates the weights using Gradient Descent on a randomly selected portion of the data.

Topics – The category label the network gives to a problem.

Torch – Pytorch. A machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing.

Appendix 2.

Source Code

Link to Source Code.

```
import torch

device = "cuda"


class LSTM(torch.nn.Module):
    def __init__(self, tensor_data, hidden_1, hidden_2, hidden_3=0\
        , set_weights=False, bias=True, dropout=False):
        super(LSTM, self).__init__()
        # instantiating class
        self.bias = bias
        self.dropout = dropout
        self.dropout_rate = 0.6
        self.test_mode = False
        self.user_labels = tensor_data["users"]
        self.users = self.user_labels.shape[0]
        self.problem_labels = tensor_data["problems"]
        self.problems = self.problem_labels.shape[0]
        self.split_users()
        self.networks = tensor_data["networks"]
        self.tensor = torch.zeros(0, self.networks + 3, device=device)
        self.hidden_1 = hidden_1
        self.hidden_2 = hidden_2
        self.hidden_3 = hidden_3
        self.set_weights = set_weights
        self.epsilon = 0.000001
        self.delta = 0.000001
        self.day = 0
        self.days = tensor_data["days"]
        self.h = torch.zeros((self.users, self.hidden_2, self.days + 1)\
            , device=device, dtype=torch.float32)
        self.C = torch.zeros((self.users, self.hidden_2, self.days + 1)\
            , device=device, dtype=torch.float32)
        self.softmax = torch.nn.Softmax(dim=1)
        self.sigmoid = torch.nn.Sigmoid()
        if self.networks > 1:
            networks_sizes = tensor_data["network_sizes"]
            network_indices = [0, 0]
            for n in range(len(networks_sizes) - 1):
                network_indices.append(sum(networks_sizes[:n]))
            self.network_indices = torch.tensor(network_indices, device=device)
            self.network_embeddings = torch.nn.Embedding(sum(networks_sizes)\
                , 1, device=device, dtype=torch.float32)
        else:
            self.network_embeddings = torch.nn.Embedding(2, 1, device=device\
            , dtype=torch.float32)
            self.network_indices = torch.tensor([0], device=device)
        self.user_embeddings = torch.nn.Embedding(self.users, self.hidden_1\
            , device=device, dtype=torch.float32)
        self.problem_embeddings = torch.nn.Embedding(\
            self.problems * self.networks, self.hidden_1, device=device\
            , dtype=torch.float32)
        if self.hidden_3 > 0:
            self.topic_embeddings = torch.nn.Embedding(self.hidden_3\
            , self.hidden_1, device=device, dtype=torch.float32)
```

```
        if self.set_weights:
            self.problem_embeddings.weight.data /= 10
            self.topic_embeddings.weight.data = torch.zeros_like(\
                self.topic_embeddings.weight.data)
            self.topic_embeddings.weight.data[:, 0] = torch.cos(2 * 3.1415\
                * torch.arange(self.hidden_3) / self.hidden_3)
            self.topic_embeddings.weight.data[:, 1] = torch.sin(2 * 3.1415\
                * torch.arange(self.hidden_3) / self.hidden_3)
        self.WiA = torch.nn.Linear(self.hidden_1, self.hidden_2, bias=False\
            , device=device, dtype=torch.float32)
        self.UiA = torch.nn.Linear(self.hidden_2, self.hidden_2\
            , bias=self.bias, device=device, dtype=torch.float32)
        self.WcA = torch.nn.Linear(self.hidden_1, self.hidden_2, bias=False\
            , device=device, dtype=torch.float32)
        self.UcA = torch.nn.Linear(self.hidden_2, self.hidden_2\
            , bias=self.bias, device=device, dtype=torch.float32)
        self.Wi = torch.nn.Linear(self.hidden_1, self.hidden_2, bias=False\
            , device=device, dtype=torch.float32)
        self.Ui = torch.nn.Linear(self.hidden_2, self.hidden_2\
            , bias=self.bias, device=device, dtype=torch.float32)
        self.Wf = torch.nn.Linear(self.hidden_1, self.hidden_2, bias=False\
            , device=device, dtype=torch.float32)
        self.Uf = torch.nn.Linear(self.hidden_2, self.hidden_2, bias=self.bias\
            , device=device, dtype=torch.float32)
        self.Wo = torch.nn.Linear(self.hidden_1, self.hidden_2, bias=False\
            , device=device, dtype=torch.float32)
        self.Uo = torch.nn.Linear(self.hidden_2, self.hidden_2, bias=self.bias\
            ,device=device, dtype=torch.float32)
        self.Wc = torch.nn.Linear(self.hidden_1, self.hidden_2, bias=False\
            , device=device, dtype=torch.float32)
        self.Uc = torch.nn.Linear(self.hidden_2, self.hidden_2, bias=self.bias\
            , device=device, dtype=torch.float32)
        self.Wr = torch.nn.Linear(self.hidden_2, self.problems\
            , bias=self.bias, device=device, dtype=torch.float32)
        self.loss_function = torch.nn.BCELoss()

    def reset_tensor(self):
        self.tensor = torch.zeros(0, self.networks + 3, device=device)

    def split_users(self):
        torch.manual_seed(42)
        rand_users = torch.randperm(self.users)
        self.training_users = self.user_labels[\
            rand_users[: int(self.users * 8 / 10)]\
            ]
        self.validation_users = self.user_labels[\
            rand_users[int(self.users * 0 / 10) : int(self.users * 0 / 10)]\
            ]
        self.test_users = self.user_labels[\
            rand_users[int(self.users * 8 / 10) :]\
            ]

    def get_mini_batch(self, type, size=20):
        torch.manual_seed(42)
        if type == "train":
            users = self.training_users[\
                torch.randperm(self.training_users.shape[0])][ : size]
            self.training_users = self.training_users[\
                ~torch.isin(self.training_users, users)
                ]
        if type == "test":
            users = self.test_users[torch.randperm(self.test_users.shape[0])]\
                [ : size]
            self.test_users = self.test_users[\
                ~torch.isin(self.test_users, users)]
        if type == "validation":
            users = self.validation_users[\
                torch.randperm(self.validation_users.shape[0])][ : size]
            self.validation_users = self.validation_users[\
                ~torch.isin(self.validation_users, users)]
        return users
```

```
127
128     def get_users_indices(self, data):
129         return torch.cat((data, self.user_labels))\
130             .unique(return_inverse=True)[1][ : data.shape[0]]
131
132     def get_problems_indices(self, data):
133         return torch.cat((data, self.problem_labels))\
134             .unique(return_inverse=True)[1][ : data.shape[0]]
135
136     def get_problem_topics(self, data):
137         problem_embeddings = self.problem_embeddings(\
138             self.get_problems_indices(data))\
139             .repeat_interleave(self.hidden_3, dim=0)
140         topics_embeddings = self.topic_embeddings(\
141             torch.arange(self.hidden_3, device=device))\
142             .repeat((data.shape[0], 1))
143         chi_sqr = (\
144             ((problem_embeddings - topics_embeddings) ** 2).sum(dim=1)\
145             .reshape(data.shape[0], self.hidden_3))
146         return chi_sqr.min(dim=1)[1]
147
148     def alphas_topics(self, data):
149         # data format: time, day, user_id, problem_id, network, value,
150         # correct_and_continued
151         # embed new problems
152         new_problems_indices = self.get_problems_indices(data[:, 3]).unique()
153         unknown_problems = torch.arange(self.problems, device=device)[\
154             (self.problem_embeddings.weight.data[:, 0] == 0)[: self.problems]]
155         new_problems = new_problems_indices[
156             torch.isin(new_problems_indices, unknown_problems)
157         ]
158         days = data[:, 1].min() * torch.ones_like(new_problems, device=device)
159         self.problem_embeddings.weight.data[new_problems, 0:2] = (\
160             self.set_weights * torch.stack((torch.cos(2 * 3.1415 * days /\
161             self.days), torch.sin(2 * 3.1415 * days / self.days)), dim=1)
162             + (1 - self.set_weights)
163             * self.problem_embeddings.weight.data[new_problems, 0:2])
164         # returns tensor, rows are users, columns are days ago
165         topics_data = torch.cat((data[:, :3]\
166             , self.get_problem_topics(data[:, 3].int()).reshape((-1, 1))\
167             , data[:, 4:]), dim=1)
168         # problems, problem_indices = data[:,3].unique(return_inverse=True)
169         topics, topics_indices = topics_data[:, 3].unique(return_inverse=True)
170         max_day = int(torch.max(topics_data[:, 1]))
171         users, users_indices = topics_data[:, 2].unique(return_inverse=True)
172         # creating event tensor
173         event_tensor = torch.zeros(users.shape[0], topics.shape[0]\
174             , device=device)
175         day_indices = data[:, 1]
176         event_tensor[users_indices, topics_indices] += 1
177         event_tensor /= self.networks
178         # creating topic data
179         historical_data = self.tensor[torch.isin(self.tensor[:, 2], users)\
180             * torch.isin(self.get_problem_topics(self.tensor[:, 3]), topics)]
181         topics_historical_data = torch.cat((historical_data[:, :3]\
182             , self.get_problem_topics(historical_data[:, 3].int())\
183             .reshape((-1, 1)), historical_data[:, 4:]), dim=1)
184         # creating historic event tensor
185         historical_event_tensor = torch.zeros((max_day + 1) * users.shape[0]\
186             , topics.shape[0], device=device)
187         users_indices = topics_historical_data[:, 2]\
188             .unique(return_inverse=True)[1]
189         day_indices = topics_historical_data[:, 1]
190         topics_indices = (topics_historical_data[:, 3]\
191             .unique(return_inverse=True)[1]).int()
192         historical_event_indices = (users_indices * (max_day + 1) \
193            + day_indices).int()
194         historical_event_tensor[historical_event_indices, topics_indices] += 1
195         historical_event_tensor /= self.networks
196         # doting historic with last day
197         all_historical_indices = torch.arange(
```

```
            historical_event_tensor.shape[0], device=device).int()
        all_users_indices = all_historical_indices // (max_day + 1)
        historical_event_tensor[all_historical_indices, :] *= event_tensor[\
            all_users_indices : ]
        # summing
        alphas = torch.sum(historical_event_tensor, dim=1)
        alphas /= (torch.sqrt(torch.sum(event_tensor[all_users_indices, :]\
            , dim=1)) + self.epsilon)
        alphas /= (torch.sqrt(torch.sum(historical_event_tensor\
            [all_historical_indices, :], dim=1)) + self.epsilon)
        # reshape
        alphas = alphas.reshape(users.shape[0], max_day + 1)[:, :-1]
        alphas = torch.softmax(alphas, dim=1)
        # alphas = torch.nn.functional.normalize(alphas)
        return alphas

    def hA_tm1_topics(self, data):
        # data format: time, day, user_id, problem_id, network, value,
        #correct_and_continued
        users = torch.unique(data[:, 2])
        users_h_index = self.get_users_indices(users)
        days = torch.max(data[:, 1]).int() - 1
        alphas = self.alphas_topics(data)
        # h format: = users, hidden_1, days
        h = self.h[users_h_index, :, : days + 1]
        user_indices = (torch.arange(users.shape[0] * days, device=device\
            , dtype=torch.int64) // days)
        day_indices = (torch.arange(users.shape[0] * days, device=device\
            , dtype=torch.int64) % users.shape[0])
        h[user_indices, :, day_indices] *= (alphas[user_indices, day_indices]\
            .reshape((-1, 1)).repeat(1, self.hidden_2))
        hAtm1 = torch.sum(h, dim=2)
        return hAtm1

    def data_reshaper(self, data):
        # data coming format: time, day, user_id, problem_id, {network_scores}
        # data going format: time, day, user_id, problem_id, network, value
        h, w = data.shape
        return_data = torch.zeros(
            h * self.networks, 6, dtype=torch.float, device=device
        )
        # copy first four columns
        indices = torch.arange(0, self.networks * h, 1, device=device) % h
        return_data[:, :4] = data[indices, :4]
        # setting network label
        networks = torch.arange(0, self.networks * h, 1, device=device) // h
        return_data[:, 4] = networks
        # setting values
        values = torch.ones(h, self.networks, device=device)
        values[:, 1:] = data[:, (w + 1 - self.networks) :]
        return_data[:, 5] = values[torch.arange(h * self.networks) % h\
            , torch.arange(h * self.networks) // h]
        embedding_indices = (self.network_indices[networks] + (networks != 0) \
            * (return_data[:, 5]).int())
        embeddings = self.network_embeddings(embedding_indices).reshape(-1)
        return_data[:, 5] = (networks == 0) + (networks != 0) * embeddings
        return return_data

    def interaction_vector(self, data):
        # getting info
        users, user_indices = torch.unique(data[:, 2].int()\
            , return_inverse=True)
        problems, problem_indices = torch.unique(data[:, 3].int()\
        , return_inverse=True)
        network_indices = data[:, 4].int()
        # creating tensor of embeddings
        embeddings = torch.zeros(users.shape[0] * (problems.shape[0] \
            * self.networks + 2), self.hidden_1, device=device,\
            dtype=torch.float32)
        # filling embeddings tensor
        embeddings[user_indices * (problems.shape[0] * self.networks + 2)\
```

```
            + network_indices * problems.shape[0] + problem_indices\
            , :] = self.problem_embeddings(\
            self.get_problems_indices(data[:, 3]).int()\
            + network_indices * self.problems)
        ones_indices = (torch.arange(1, users.shape[0] + 1)\
            * ((problems.shape[0] * self.networks + 2)) - 1)
        embeddings[ones_indices, :] = torch.ones(users.shape[0], self.hidden_1\
            , device=device)
        users_embeddings_indices = ones_indices - 1
        self.get_users_indices(users).int()
        embeddings[users_embeddings_indices, :] = self.user_embeddings(\
            self.get_users_indices(users).int())
        # multiply by score
        embeddings[user_indices * (problems.shape[0] * self.networks + 2)\
            + network_indices * problems.shape[0] + problem_indices, :\
            , ] *= data[:, [5] * self.hidden_1]
        # permuting and dotting
        embeddings = embeddings.reshape(users.shape[0], -1, self.hidden_1)
        height = embeddings.shape[1]
        combinations = (height * (height - 1)) // 2
        combinations_vector = torch.zeros(users.shape[0], combinations\
            , self.hidden_1, device=device)
        combination_pairs = torch.combinations(torch.arange(height\
            , device=device))
        height_indices = torch.arange(combinations, device=device)
        combinations_vector[:, height_indices, :] += (\
            embeddings[:, combination_pairs[:, 0], :]\
            * embeddings[:, combination_pairs[:, 1], :])
        return combinations_vector.sum(dim=1)

    def forward(self, data):
        self.tensor = torch.unique(torch.cat((self.tensor\
            , data[: data.shape[0] * (1 - self.test_mode), :])), dim=0)
        data = self.data_reshaper(data)
        dropout = self.dropout * (torch.rand(1).item()) < self.dropout_rate
        users, counts = torch.unique(data[:, 2], return_counts=True)
        h_C_indices = self.get_users_indices(users)
        day = int(data[:, 1].max())
        i = self.interaction_vector(data)
        hA_tm1 = self.hA_tm1_topics(data)
        iA_t = self.sigmoid(self.WiA(i) + (1 - dropout) * self.UiA(hA_tm1)\
            + dropout * torch.mm(hA_tm1, self.UiA.weight.T))
        cA_t = torch.tanh(self.WcA(i) + (1 - dropout) * self.UcA(hA_tm1)\
            + dropout * torch.mm(hA_tm1, self.UcA.weight.T))
        cA_t = iA_t * cA_t
        h_tm1 = self.h[h_C_indices, :, day - 1]
        it = (1 - torch.tensor([-self.delta], device=device).exp()) \
            * self.sigmoid(self.Wi(i) + (1 - dropout) * self.Ui(h_tm1)\
            + dropout * torch.mm(h_tm1, self.Ui.weight.T))
        ft = torch.tensor([-self.delta], device=device).exp() * self.sigmoid(\
            self.Wf(i) + (1 - dropout) * self.Uf(h_tm1)\
            + dropout * torch.mm(h_tm1, self.Uf.weight.T))
        ot = self.sigmoid(self.Wo(i) + (1 - dropout) * self.Uo(h_tm1)\
            + dropout * torch.mm(h_tm1, self.Uo.weight.T))
        ct = torch.tanh(self.Wc(i) + (1 - dropout) * self.Uc(h_tm1)\
            + dropout * torch.mm(h_tm1, self.Uc.weight.T))
        C_tm1 = self.C[h_C_indices, :, day]
        Ct = ft * C_tm1 + it * ct + iA_t * cA_t
        self.C[h_C_indices, :, day + 1] += Ct * (1 - self.test_mode)
        ht = ot * torch.tanh(Ct)
        self.h[h_C_indices, :, day + 1] += ht * (1 - self.test_mode)
        output = self.sigmoid((1 - dropout) * self.Wr(ht) + dropout \
            * torch.mm(ht, self.Wr.weight.T))
        user_indices = (data[:, [2] * users.shape[0]] == \
            users.repeat(data.shape[0], 1)).nonzero()[: int(data.shape[0] \
            / self.networks), 1] problem_indices = self.get_problems_indices(\
            data[: int(data.shape[0] / self.networks), 3])
        prediction = output[user_indices, problem_indices]
        return prediction

    def loss(self, output, target):
```

```
340         return self.loss_function(output, target)
```

Appendix 3.

Working Example

Link to Working Demo

References

Add service—AWS Pricing Calculator. (n.d.). Retrieved February 4, 2024, from https://calculator.aws/#/createCalculator/ec2-enhancement

AlHaosului, I. (2007). *Screenshot of a WeBWorK session* [Electronic]. https://en.wikipedia.org/wiki/WeBWorK#/media/File:WeBWorK.png

Armstrong, E. (2017). *A Neural Networks Approach to Predicting How Things Might Have Turned Out Had I Mustered the Nerve to Ask Barry Cottonfield to the Junior Prom Back in 1997* (arXiv:1703.10449). arXiv.

Beutel, A., Covington, P., Jain, S., Xu, C., Li, J., Gatto, V., & Chi, E. H. (2018). Latent Cross: Making Use of Context in Recurrent Recommender Systems. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 46–54. https://doi.org/10.1145/3159652.3159727

Deunk, M. I., Smale-Jacobse, A. E., De Boer, H., Doolaard, S., & Bosker, R. J. (2018). Effective differentiation Practices:A systematic review and meta-analysis of studies on the cognitive effects of differentiation practices in primary education. *Educational Research Review*, *24*, 31–54.

Differentiated Instruction: Examples & Classroom Strategies. (2014, October 1). Resilient Educator. https://resilienteducator.com/classroom-resources/examples-of-differentiated-instruction/

Elon Musk Promises for the Millionth Time that Tesla will Achieve Full Self-Driving Soon We've Definitely Heard this one before. (n.d.). Retrieved February 11, 2024,

from https://futurism.com/the-byte/elon-musk-promises-tesla-full-self-driving-again

Freedle, R. (2003). Correcting the SAT's Ethnic and Social-Class Bias: A Method for Reestimating SAT Scores. *Harvard Educational Review*, *73*(1), 1–43.

Gong, T.-J., Yao, X., & Ma, W. (2018). GRE: An Adaptive and Personalized Exercise Model for K-12 Online Education. *Proceedings of the 2018 2nd International Conference on Education and E-Learning*, 48–54.

How do I update my interests? – NoRedInk Help Center. (n.d.). Retrieved February 1, 2024, from https://noredink.zendesk.com/hc/en-us/articles/205072369-How-do-I-update-my-interests

Knill, O. (2020). *Final Practice A, Math 1A: Introduction to Functions and Calculus* [Assignment]. https://people.math.harvard.edu/~knill/teaching/math1a2020/final/practice1.pdf

Koile, K., Rubin, A., Chapman, S., Kliman, M., & Ko, L. (2016). Using machine analysis to make elementary students' mathematical thinking visible. *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge*, 524–525.

Lin, T. (n.d.). *Algebra Playground: Polarize*. Retrieved February 1, 2024, from https://tengjuilin.netlify.app/archive/archive/other/inverse.html

Linton, B. (2000). *Climb that Tree*.

Moon, T. R. (2005). The Role of Assessment in Differentiation. *Theory Into Practice*, *44*(3), 226–233. https://doi.org/10.1207/s15430421tip4403_7

Ni, Q., Wei, T., Zhao, J., He, L., & Zheng, C. (2023). HHSKT: A learner–question interactions based heterogeneous graph neural network model for knowledge tracing. *Expert Systems with Applications*, *215*, 119334.

Overview of PyTorch Autograd Engine | PyTorch. (n.d.). Retrieved December 22, 2023, from https://pytorch.org/blog/overview-of-pytorch-autograd-engine/

Perera, D., & Zimmermann, R. (2020). *LSTM Networks for Online Cross-Network Recommendations*. https://doi.org/10.48550/ARXIV.2008.10849

Rodrigues, M. W., Isotani, S., & Zárate, L. E. (2018). Educational Data Mining: A review of evaluation process in the e-learning. *Telematics and Informatics*, *35*(6), 1701–1717. https://doi.org/10.1016/j.tele.2018.04.015

Schmucker, R., Wang, J., Shijia Hu, & Mitchell, T. M. (2022). *Assessing the Performance of Online Students—New Data, New Approaches, Improved Accuracy*. https://doi.org/10.5281/ZENODO.6450190

Schneider, B. (2018). *Handbook of the sociology of education in the 21st century*. Springer Berlin Heidelberg.

Sin, K., Muthu, L., & Bharathiyar University. (2015). APPLICATION OF BIG DATA IN EDUCATION DATA MINING AND LEARNING ANALYTICS – A LITERATURE REVIEW . *ICTACT Journal on Soft Computing*, *05*(04), 1035–1049.

Steam Hardware & Software Survey. (n.d.). Retrieved December 9, 2023, from https://store.steampowered.com/hwsurvey/videocard/

Tang, S., Peterson, J. C., & Pardos, Z. A. (2016a). Deep Neural Networks and How They Apply to Sequential Education Data. *Proceedings of the Third (2016) ACM Conference on Learning @ Scale*, 321–324.

Tang, S., Peterson, J. C., & Pardos, Z. A. (2016b). *Modelling Student Behavior using Granular Large Scale Action Data from a MOOC*.

Tomlinson, C. A., Brighton, C., Hertberg, H., Callahan, C. M., Moon, T. R., Brimijoin, K., Conover, L. A., & Reynolds, T. (2003). Differentiating Instruction in Response to Student Readiness, Interest, and Learning Profile in Academically Diverse Classrooms: A Review of Literature. *Journal for the Education of the Gifted*, *27*(2–3), 119–145. https://doi.org/10.1177/016235320302700203

Zhao, W., Xia, J., Jiang, X., & He, T. (2023). A novel framework for deep knowledge tracing via gating-controlled forgetting and learning mechanisms. *Information Processing & Management*, *60*(1), 103114.

怎樣解題：數量關係 | 均一教育平台. (n.d.). Retrieved February 1, 2024, from https://www.junyiacademy.org/course-compare/math-elem/math-6/math-grade-6-a/g06-menso6d