The Pennsylvania State University
The Graduate School

AI IN EDUCATION: EFFECTIVE MACHINE LEARNING METHODS TO IMPROVE DATA SCARCITY AND KNOWLEDGE GENERALIZATION

A Dissertation in
Information Sciences and Technology
by
Jia Tracy Shen

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

Aug 2023

The dissertation of Jia Tracy Shen was reviewed and approved* by the following:

Dongwon Lee
Professor of College of Information Sciences and Technology
Thesis Advisor, Chair of Committee

Amulya Yadav
Assistant Professor of College of Information Sciences and Technology
Committee Member

Clyde Lee Giles
The David Reese Professor of Information Sciences
and Technology
Committee Member

Rebecca Passonneau
Professor of Electrical Engineering and Computer Science
Committee Member

*Signatures are on file in the Graduate School.

Abstract

In education, machine learning (ML), especially deep learning (DL) in recent years, has been extensively used to improve both teaching and learning. Despite the rapid advancement of ML and its application in education, a few challenges remain to be addressed. In this thesis, in particular, we focus on two such challenges: (i) data scarcity and (ii) knowledge generalization. First, given the privacy concerns of students or students' behavior differences, it is common to have missing data in the education domain, which challenges the application of ML methods. Second, due to varying data distributions across education platforms and applications, ML methods often struggle to generalize well over unseen data sets. Therefore, this thesis proposes effective *data augmentation* methods to combat the challenge (i) and investigate *transfer learning* techniques to solve the challenge (ii). More specifically for the challenge (i), we provide simple to complex solutions to augment data by: (a) optimizing statistical time lag selection to reduce matrix sparsity and improve original model performance by 32% in classification tasks and 12% in regression tasks; and (b) developing deep generative models (i.e., LSTM-[L]VAEs) to generate missing data to improve original model performance by 50%. For the challenge (ii), we employ transfer learning to improve model generalization and enable knowledge transfer from other domains to the education domain in three approaches: (1) a comparison approach; (2) a TAPT (Task Adapted Pre-train) approach; (3) a DAPT (Domain Adapted Pre-train) approach. Approach (1) first demonstrates the effectiveness of the transfer learning and then compares the transferability saliency between different models (i.e., LSTM vs. AdaRNN vs. Transformer) and transfer learning methods (i.e., feature-based vs. instance-based). It discovers that the Transformer model is 3-4 times more effective than other model structures and feature-based method is up to 5 times superior to its counterpart in transferability. Furthermore, Approach (2) leverages the shared semantic and lexical extractions from the pre-trained general language model and forms a TAPT BERT model to adapt to the particular education tasks. It surpasses the original general language model by 2%. Finally, Approach (3) further trains on the general language model but adapts to a large mathematical corpus to form a DAPT model. It is demonstrated to improve prior state-of-the-art models and BASE BERT by up to 22% and 8%, respectively.

Table of Contents

List of Figures

List of Tables

Acknowledgments

This material is based upon work in part supported by NSF awards 1940076 and 1915801. K12/Stride does not release student-level data to external research, but does use such data to continually improve the learning experience for students served in K12-powered schools. The authors are grateful to K12 for allowing its internal researchers to align research questions of general academic interest and publish some of those learning.

The findings and conclusions do not necessarily reflect the view of the funding agency.

Dedication

This work is a fruit of countless and arduous sacrifices from a mother and a full time worker. I would like to thank my husband Mark, without his help on sharing the house chores and care for our daughter, I wouldn't make this far myself alone! I am greatly thankful to my adorable daughter Zoë, who started this journey with me since she was 8-month old. Thank you for making mommy's life never boring and ever challenging, which truly propelled me to aim high, stay low and achieve solid. Your dearest snuggle is the biggest motivation that I can do both PhD and full-time job at the same time! I would also like to thank my advisor Dongwon Lee, who gave me so much support and understanding in my situation and made research not that scary for me! Thanks also goes to all my lab-mates who don't hesitate to give advice and help on my research journey! In addition, I want to thank all our friends who helped to watch Zoë on the days when I have to catch up work and research. Finally, I want to thank my parents –Guanshui and Meiqin, especially my dad, without your open-mindedness, I wouldn't be able to come overseas for education and achieve so much! Thanks also to my brother Keqi, without his care for my parents, I wouldn't be able to take on this journey worry-free.

Chapter 1 Introduction

1.1 Background

In today's age, the increasing adoption of AI technologies in education is drastically transforming the way we teach and learn. More and more AI technologies are developed and implemented to improve the teaching effectiveness, to reduce the cost of administrative processes, to enable personalized learning and etc. Many of these AI solutions utilize machine learning (ML) including deep learning (DL) techniques to conduct either supervised or unsupervised learning for various problems and have achieved the state-of-the-art (SOTA) performance. While these methods are important and effective, there appear two major issues that challenge the development of AI technologies in education domain: (i) data scarcity; (ii) knowledge generalization.

In education domain, often times, it is either difficult to collect more data or simply not possible to acquire missing data. Take student surveys for example, their response rates are notoriously low due to the fact that surveys are not mandated in their nature and hence limited data is collected. Another example about in-feasibility of obtaining data is that many students' learning records could not be tracked due to regular sick leaves and even so under Covid pandemics. With the limited or unreliable data, models built from it tends to overfit, a modelling error that occurs when a model tries to fit all the data points available in the training data set [2]. This poses great risks on the performance of deep learning models, which then is inclined to learn extensively the pattern from the limited data set and is not able to generalize well on unseen data sets. A common solution from the

data space is *data augmentation* [3], a method used to artificially generate data from the available dataset [2].

As we already notice that data scarcity can lead to poor model generalization, this thesis extends the generalization challenge to be a *knowledge generalization* problem in the education domain. By *knowledge*, we mean the *learnings (common features)* from trained models rather than the education knowledge that students need to learn. *knowledge generalization* can appear in two scenarios: (a) we generalize knowledge/learnings from a source model that is trained on a limited data set to an unseen data set; (b) we generalize knowledge/learnings from a larger/more general domain to a special or focused target domain/task. Scenario (b) is often seen in the case where some features can be shared from other domains to the education domain but has not been or yet difficult to be done. For example, education domain texts share certain characteristics with the general language texts and hence a NLP model trained on general language can be leveraged to predict tasks in the education domain. Traditionally, (a) is considered as model generalization whereas (b) is considered as knowledge transfer or often heard as transfer learning. *Transfer learning* defined by Pan et al. is an approach that aims to help improve the learning of a target predictive function using the features learned in a source domain or task [4]. Therefore, empirically, transfer learning is suitable to be used for both Scenario (a) and Scenario (b) and knowledge generalization can be considered as the goal of the transfer learning approach.

With that, this thesis presents effective data augmentation methods to tackle the data scarcity issue and validate multiple transfer learning methods to improve the knowledge generalization challenge in the education domain. This chapter will summarize the proposed technical contributions of the thesis in the following sections.

1.2 Data Augmentation

There are various ways of conducting data augmentation but can be generally summarized into two groups: online and offline data augmentation. In online augmentation, data is augmented at training time so that there is no need to store the augmented data [5]. In offline augmentation, data is augmented in the preprocessing phrase and stored on the disk [6]. This thesis focuses on the offline

augmentation. In the category of offline augmentation, depending on the type of data sets, methodologies varies to a great extent. This thesis mainly works with numerical (especially time series) data generation, thus, we introduce two relevant approaches and elaborate on them as follows.

Approach I: For numerical data such as time series data in the education domai, it typically contains time-varying variables such as question difficulty, attempts for the question, assessment duration and time-invariant variables such as gender, grade level (see in Figure 1.1). A simple way to increase the data volume is create time lags to duplicate the data and meanwhile considering the auto-correlation effect from the previous time lags. In practice, if a student has L time steps in his learning history, full lags (L-1 steps) will be created to augment data (see in Figure 1.2). Although effective in the sense it can create hundreds of copies of its own data when the data set has hundreds of time steps, it introduces non-negligible *matrix sparsity* in the training. Matrix sparsity is referred to a sparse matrix with lots of zeros caused by the missing data when fed into time series deep learning models such as RNN-based models (e.g., LSTM, GRU) and attention-based sequence to sequence models (e.g., Transformer) which normally require 3D matrix input. To minimize the sparsity, Chapter 2 hypothesizes time lags that passes a certain threshold might not influence the current results anymore and hence attempts an embarrassingly simple yet very efficacious optimization approach by looping through all the time lags to identify the best time lag via the validation of classic Knowledge Tracing models such as DKT [7], NPA [8]. We are able to boost the original model performance by up to 32% in classification task and up to 12% in regression tasks after pruning the time lags.

Approach II: Although the method introduced in Chapter 2 produces superior improvement, it could be computationally costy, especially when the time lags are in hundreds. Therefore, in Chapter 3, we introduce effective deep generative frameworks where we build LSTMs into VAEs structures to amplify data based on the latent features extracted from the existing data. In this method, we posit the education time series data as longitudinal data where time-invariant subject descriptors co-exists with time varying variables. We hypothesis there exists a relationship between the subject descriptors and the latent features. Therefore, we leverage a module from the existing Longitudinal VAE frame work, namely, a multi-output additive Gaussian Prior (GP), to extract such relationship. Furthermore,

		var1	var2	var3	...	d_var1	d_var2	d_var3	...
ST0	t0	...	...	...	...	...	...	...	...
	t1	...	...	...	...	...	...	...	...
	t2	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...
	t_L	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...
ST_x	t0	...	...	...	...	...	...	...	...
	t1	...	...	...	...	...	...	...	...
	t2	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...
	t_L'	...	...	...	...	...	...	...	...

Figure 1.1: An illustration of typical education data. 'ST' stands for a student; 'var' stands for regular time-varying variables; 'd_var' stands for time-invariant description variable. The blue indicates that the variable values are the same for the description variables for each students.

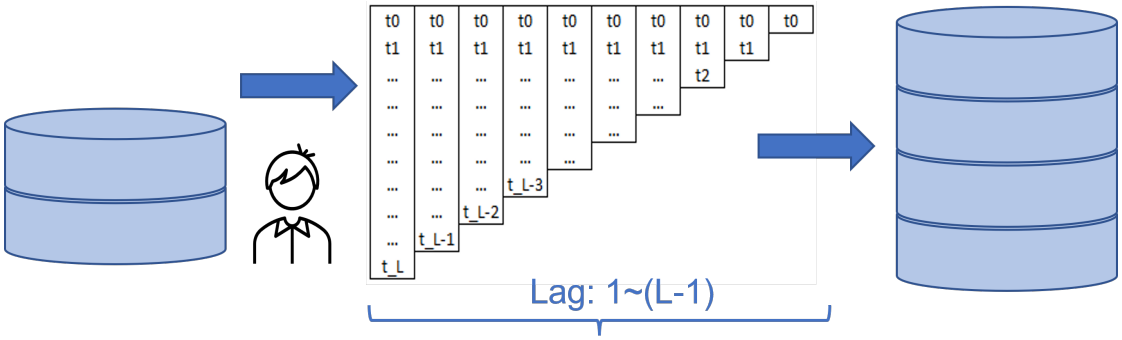


Figure 1.2: Creating Lags for Student '0' with L-1 lags.

we propose a subject-based training where we split and impute data by subject id to reflect the longitudinal nature of the education time series data and it achieves a satisfactory results. We are able to boost original model performance by 50% in average RMSE. Furthermore, a robustness measuring experiment is conducted to demonstrate that only 10% of the generated data is needed to boost original model performance if models are small to medium size.

1.3 Knowledge Generalization

To tackle the two scenarios of knowledge generalization, we employ transfer learning extensively to meet the ends. To explain transfer learning further, we define $\mathcal{D}$ as a domain that consists of a feature space $\mathcal{X}$ and a marginal probability distribution P(X) over the feature space, where $X = \{x_1, ..., x_n\} \in \mathcal{X}$. Given a domain, $\mathcal{D}$=$\{\mathcal{X}, P(X)\}$, a task $\mathcal{T}$ which consists of a label space $\mathcal{Y}$ and an objective predictive function $f(\cdot)$ (denoted by $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$) that is typically learned from the training data $\{x_i, y_i\}$ pairs, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. The function $f(\cdot)$ can also be seen as the conditional probability distribution function of P(Y|X). Therefore, given a source domain $\mathcal{D}_\mathcal{S}$, a corresponding source task $\mathcal{T}_S$, as well as a target domain $\mathcal{D}_\mathcal{T}$ and a target task $\mathcal{T}_\mathcal{T}$, the objective of transfer learning is to allow us to learn the target conditional probability function $P(Y_T|X_T)$ in $\mathcal{D}_\mathcal{T}$ with the information gained from $\mathcal{D}_\mathcal{S}$ and $\mathcal{T}_\mathcal{S}$ where $\mathcal{D}_\mathcal{S} \neq \mathcal{D}_\mathcal{T}$ or $\mathcal{T}_\mathcal{S} \neq \mathcal{T}_\mathcal{T}$ [4].

Based on the aforementioned settings, transfer learning can be categorized into three groups: (1) inductive transfer learning, where $\mathcal{D}_\mathcal{S} = \mathcal{D}_\mathcal{T}$ but $\mathcal{T}_\mathcal{S} \neq \mathcal{T}_\mathcal{T}$; (2) transductive transfer learning, where $\mathcal{D}_\mathcal{S} \neq \mathcal{D}_\mathcal{T}$ but $\mathcal{T}_\mathcal{S} = \mathcal{T}_\mathcal{T}$; (3) unsupervised learning, where we solve unsupervised tasks when $\mathcal{D}_\mathcal{S} = \mathcal{D}_\mathcal{T}$ but $\mathcal{T}_\mathcal{S} \neq \mathcal{T}_\mathcal{T}$; This thesis focuses on the first two settings which are the correspondence to the two scenarios of knowledge generalization. Inductive transfer learning corresponds to scenario (a) (i.e., model generalization), under which knowledge is generalized between different tasks but within the same domain whereas transductive transfer learning corresponds to scenario (b) (i.e., knowledge transfer), under which knowledge often exists in different domains. Below we introduce the proposed transfer learning methods for the two scenarios in detail.

Scenario (a) Approach: Under inductive transfer learning setting, there are four common methods to adopt: instance transfer, feature-representation-transfer, parameter-transfer and relational-knowledge-transfer. Instance transfer assumes certain parts of the source data instances can be reused by $\mathcal{D}_\mathcal{T}$ or $\mathcal{T}_\mathcal{T}$. Feature-representation-transfer aims to find good feature representation that reduces difference between $\mathcal{D}_\mathcal{S}$ and $\mathcal{D}_\mathcal{T}$. Parameter-transfer assumes there exists shared hyperparameters or priors between $\mathcal{D}_\mathcal{S}$ and $\mathcal{D}_\mathcal{T}$ to benefit knowledge transfer. Relationship-knowledge transfer builds mappings of relational knowledge between relational $\mathcal{D}_\mathcal{S}$ and $\mathcal{D}_\mathcal{T}$. To improve model generalization in Scenario (a), this thesis adopts the

instance- and feature-based methods and compares their efficacy in Chapter 4. More particularly, we apply instance-based method by training a Maximum Mean Discrepancy (MMD) loss function to minimize the distance between $\mathcal{T}_{\mathcal{S}}$ and $\mathcal{T}_{\mathcal{T}}$. We apply the feature-based approach by freezing layers (or weights) of source model to maintain the good feature representation from $\mathcal{T}_{\mathcal{S}}$. We discover feature-approach is more effectively at reserving the features from the $\mathcal{T}_{\mathcal{S}}$ and can be applied to $\mathcal{T}_{\mathcal{T}}$ to improve model generalization. The feature-based method via weights-freezing outperforms the instance-based method featuring MMD method by up to 10% (approximately 5 times) in average RMSE across different models. In addition, Chapter 4 also observe Transformer has superior transferability over other time series models such as LSTM and AdaRNN (a GRU based model) with a margin of 4-7% in average RMSE, about 3-4 times more effective.

Scenario (b) Approach: In Scenario (b), knowledge/features exists in a domain $\mathcal{D}_{\mathcal{S}}$ that is different from the education domain $\mathcal{D}_{\mathcal{T}}$ but there exists common knowledge/features between $\mathcal{D}_{\mathcal{S}}$ and $\mathcal{D}_{\mathcal{T}}$ that can be learned to apply onto the similar tasks in the education domain. For example, in the field of NLP, many language models that are pre-trained on news data such as the Wall Street Journal, Wikipedia share semantic representation with the content in the education domain. Therefore, to predict textual tasks in the education domain, we could leverage these pre-trained models from general language to save training time and cost. Chapter 5 introduced TAPT (Task-adapted Pre-trained) model that inherits the pre-trained weights from a general language model (i.e., BERT-base) and adapts to the education specific tasks. It obtains performances that exceed existing SOTA model of about 2.3%. However, in the process of transferring knowledge from general language to specific domain texts, we observe it is difficult to adapt general language to a specific type of texts such as mathematical texts which is rich in equations and symbols. Thus, this thesis hypothesizes that a language model trained on domain specific texts (i.e., mathmatical texts) could better fit $\mathcal{T}_{\mathcal{T}}$ in $\mathcal{D}_{\mathcal{T}}$. Chapter 6 introduce DAPT (Domain-adapted Pre-trained) model, namely MathBERT which surpasses the best prior model by up to 22% and even out-performs the general language model by 2-8%.

Illustration: To further differentiate the above approaches, we call Scenario (a) and illustrated in Figure 1.3 whereas we call Scenario (b) approach as Approach (b) illustrated in Figure 1.4. Approach (a) does not involve massive pre-training as

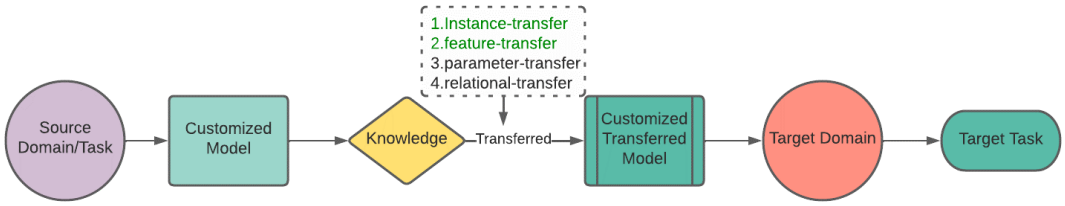


Figure 1.3: An illustration of Approach (a)

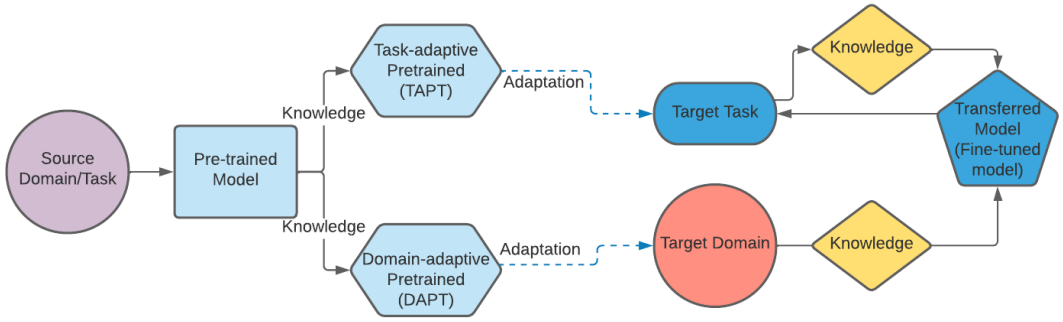


Figure 1.4: An illustration of Approach (b)

Approach (b) but simply develops a customized model (source model) and extracts the knowledge via either instance-based or feature-based transfer learning methods and finally apply on the $\mathcal{T}_{\mathcal{T}}$. Chapter 4 utilizes this approach to boost model generalization. Approach (b) has two variants: TAPT and DAPT models, both of which are further trained from pre-trained general language models. The difference between TAPT and DAPT is that TAPT during further-train only adapts to $\mathcal{T}_{\mathcal{T}}$ but DAPT adapts to $\mathcal{D}_{\mathcal{T}}$ with much more corpora. Thus, TAPT can only be applied to specific $\mathcal{T}_{\mathcal{T}}$ that it gets trained on whereas DAPT can be applied to multiple tasks that $\mathcal{D}_{\mathcal{T}}$ can cover. To predict $\mathcal{T}_{\mathcal{T}}$, they both will go through a fine-tunig process where the last layer of the model will be trained according to the specific task to gain target outputs.

Chapter 2
Augment Data via Time Lag Optimization

2.1 Introduction

Knowledge tracing (KT) is a modeling task whose goal is to trace the knowledge state of students based on their past exercise performance and predict how students will perform on future interactions. It is usually formulated as a supervised sequence learning problem: given a student's past exercise interactions $\mathcal{X} = \{x_1, x_2, ..., x_t, \}$, predict the probability that the student will answer the next exercise correctly, i.e., $p(r_t = 1|q_t, \mathcal{X})$. Input $x_t = (q_t, r_t)$ is a tuple containing the exercise q_t, which students attempt at the timestamp t, and the correctness of the students' response r_t. KT models consider r_t as the observed time series variable with t as the time step and assume a hidden knowledge state at each time step. To predict a student's future knowledge mastery (i.e., knowledge state) is essentially to predict r_t. Therefore, KT models can be considered as single-variate time series models.

Knowledge tracing data presents two unique characteristics. First, it comprises sequences which have various lengths. This sequence is students' past exercise interaction in the form of question-answer pair. For example, student A has 34 exercise questions answered in the last month whereas student B could have 150 exercise questions answered assuming student B takes additional exercises to master one concept. Second, there always exists a good number of time steps missing for certain percentage of students for various reasons and it is hard to impute. In

the example of temperature data, it is safe to impute missing daily values with the adjacent values or simply weekly average. However, it is risky to impute students' responses using the adjacent responses. For example, a student could be answering an addition problem at the current step but answering an algebra problem for next. Based on the two traits, a common way to augment the data volume in the knowledge tracing field is to apply statistical time series modeling technique, namely creating time lags. The classic practice is to multiply the data by taking its -1 to $-(L-1)$ lags to achieve the purpose of volume increasing. In addition, due to the vast adopting of temporal neural network structure such as RNN, attention mechanism, when constructing time lags for training, data have to be processed as a fixed length sequence. Thus, long sequences will cut short to the length of the fixed sequence length and short sequence will get padded for "0" (see the illustration in Figure 2.1). If we have many records of short sequences, the training data noticeably will be mainly filled up with "0"s and get sparse matrices. Therefore, it still introduces data scarcity in the training data even after we augment the data via creating time lags. To get rid of such side effect and optimize the classic statistical data augmentation approach for knowledge tracing data, we hypothesize that including less time lags or finding the best performing time lag could effectively reduce matrix sparsity (i.e., the training data scarcity) and increase model performance. We validate the performance after reducing time lags through classic KT models: DKT [7] and NPA [8].

Thus, this chapter attempts to answer the following research questions (RQs):

- RQ1: Does reducing time lags increase model performance in general?
- RQ2: Can reducing time lags improve the performance of models regardless of the sequence length and data size?
- RQ3: Does it exist a certain range of time lags that are more optimal than other time lag groups during optimization?

Through the answers to the above RQs (AQs), this chapter makes contributions as follows:

- To overcome the data sparsity in the education data, we propose to find the best time lag to reduce the presence of sparse matrices

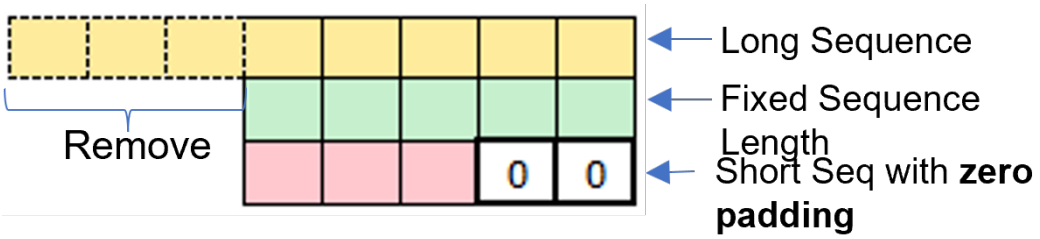


Figure 2.1: An illustration of KT data sequence aligning process.

- We discover reducing time lag has a significant effect on small to medium size data but mild on huge/long sequence data in boosting model performance
- There exists a certain range of time lags that have better performance than other time lags group

2.2 Related Work

2.2.1 Knowledge Tracing Models

In the field of knowledge tracing, researchers have long treated the exercises that students take during their learning trajectory as temporal sequence and hereafter build a number of effective time series neural networks such as DKT [7], DKVMN [9], NPA [8], SAKT [10], SAINT [11]. Researchers have done benchmarking evaluation on all the aforementioned algorithms on the same data sets [1]. Based on such evaluation results, we choose DKT and NPA models as our validation models because they obtain the best performances for majority of the public KT data sets. The DKT model is built on top of LSTM structure with an additional embedding layer that factors in the total number of questions a knowledge tracing data usually contains. The model first aligns all the exercise sequence to be a fixed length vector and uses LSTM cells to capture the hidden state information that is summarized from the past exercise sequence guided by what type of questions a student answers. This information then can be used to predict the next question response assuming next questions are in the same testing system. The input of the model is a question response pair and the sequence of question will be

[1] https://github.com/seewoo5/KT

converted to temporal information when fed into the LSTM structure. The nutshell of this algorithm is simply an LSTM model that fits on a single-variate time series knowledge tracing data. The NPA model is a complex version of adopting both LSTM and attention layers that also rely on the question number information to form the embedding layers. It utilizes bi-LSTM layer that takes in a matrix multiplication of question and response embedding to exhaustively learn the characteristics revealed from the question and response pair instead of only questions. An additive attention layer is applied afterwards to generate a score to indicate which question the new question assembles the most. The model although complex is effective in many settings and a good bench-marking algorithm to choose.

2.2.2 Time Lag Selection

In time series forecasting, the determination on which time lag to choose is critical to the forecast accuracy as not every past step has the impact on the current step. Many optimization approaches have been proposed to find the appropriate time lag to boost model performance. However, no single method has been proven to perform best in all prediction models and all time-sereis data. There are mainly three groups of approaches: (i) massive experiments; (ii) statistical approach - autocorrelation function; (iii) heuristic algorithms. Researchers have developed customized RNN-based structures to choose what time lag fits the best [12–14]. These methods although effective are time and resource intensive. In addition, these methods work the best for the continuous time series data such as air pollution, meteorological observation data. The heuristic [15–17] algorithms, although work effectively, only work for a specific domain and task such as finance, neuroscience, meteorology, business, which causes an application barrier. A recent study [18] that utilizes Genetic Algorithm to optimize time lag selection seems to be effective. However, it only works for data sets with short sequence length below 50, beyond which the computation costs spiral. [18] conducted comparison studies to examine the benefit and weakness of each approach and demonstrated the combination of experiment and heuristic algorithm seems to work the best but again for the continuous time series data. However, because our data is not continuous, the method is not applicable and we opt for the experiment approach.

2.3 Experiments

2.3.1 Problem Setup

Many of the existing KT models rely on the neural network's memory mechanism executed by memory cells (e.g., LSTM), attention mechanism (e.g., Transformer). Therefore, in the pre-processing, data is usually cut into segments of the equal sequence lengths before feeding into the neural network. This sequence length $\bar{L}$ normally is calculated as the mean of all the data sequences from various time steps: $\bar{L} = \frac{1}{N}\sum_{t=1}^{N} L_t$. KT models then will look $L-1$ (L is the total time steps) lags back to increase data volume and also for the past information to make prediction on the future sequences. The input at time step t (X_t) will present two situations after pre-processing. When $L_t < \bar{L}$, $X_t = \{\underbrace{x_1, ...x_t}_{t}, \overbrace{0, ...0}^{L-t}\}$, in which, '0' is a padded value and there are $L-t$ zero values in the input sequence. When $L_t > \bar{L}$, $X_t = \{x_1, ... \underbrace{x', ...x_t}_{L}\}$, in which, there is no zero padding and the input is cut short to extract the last L lags of the sequence (see in Figure 2.1 for details). Therefore, if the gap between L_t and $\bar{L}$ is huge or there are many $L_t < \bar{L}$ instances, we can imagine the input will contain large quantity of zero padded values and hence lead to data sparsity in the training process. Although we can not control the gap between L_t and $\bar{L}$, we can control the instances when $L_t < \bar{L}$. We use % of short sequences to measure how inclined the data set to have sparse matrices (see the % in Table 2.1). Instead of looking back L lags which is the full length of the set sequence, we hypothesize that not every step of looking back in the time series data plays the equal role in predicting the future sequence. We call the looking back step as "lag step" and set the desired lag step to be $\tilde{L}$ which meets $\tilde{L} < \bar{L}$ and is sufficient to predict for the future sequence and achieve equivalent or even better performance. To find such $\tilde{L}$, we adopt an embarrassingly simple but efficient method by looping through all the time lags till $\bar{L}-1$ and obtain the KT model performance via the neural network architectures of DKT [7] and NPA [8] which are the representative work of utilizing LSTM and attention mechanism in the single variate KT models.

Table 2.1: Six Data Sets Details. All ASSISTments data from ASSISTments platform [1]

Metric	ASSISTments2009	ASSISTments2015	STATICS	JUNYI	K12 Geom	K12 Alg2
# of Input	1	1	1	1	8	8
Target Variable	0 or 1	0 or 1	0 or 1	0 or 1	[0,1]	[0,1]
# Questions	110	100	1223	722	1,430	1,154
Avg. Seq Length	80	34	576	100	150	150
# Users	4,151	19,840	333	247,796	3,298	2121
# Instances	325,637	683,801	189,297	25,925,992	494,686	336,412
% Short Sequence	60.42%	66.56%	45.65%	59.63%	45.06%	47.43%

.

2.3.2 Data sets

We choose six data sets to evaluate the effect of time lags (see details in Table 2.1). They are four single-variate public data sets (i.e., ASSISTments 2009 [1] [2], ASSISTments 2015 [1][3], Junyi[4], STATICS [5]) and two multi-variate private data sets (i.e., K12 Geometry, K12 Algebra) including multiple educational information such as test attempts, attempt duration, item difficulty, question type. The prediction on the four public data sets are binary classification tasks whereas the prediction on the two private data sets are regression tasks with target variable as score rate.

The four public KT data sets are chosen due to their varying sequence lengths (34-576 time steps) and data size (from thousands to millions). This way, we can examine whether or not the effect is different for various-length sequences and sizes of data sets. The two private data sets we choose is medium-sequence length (150) and medium sized data sets (around 500k). The number of questions will used as a hyperparameter in embeddings of DKT and NPA architectures. The fixed length of the data set sequence $\bar{L}$ tells the average time steps per student in each data set, which also affects how many loops we will need to go through to find $\tilde{L}$. The two private data sets contain eight variables: "question_order", "total_attempts", "test_attempts", "assessment_duration", "question_difficulty", "item_difficulty", "usmo_difficulty", "usmo_id". "USMO" stands for 'Universal Standard Master Objective', a K12 internal naming convention to measure the objective level be-

[2]https://sites.google.com/site/assistmentsdata/home/2009-2010-assistment-data?authuser=0

[3]https://sites.google.com/site/assistmentsdata/home/2015-assistments-skill-builder-data

[4]https://pslcdatashop.web.cmu.edu/Project?id=244

[5]https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=507

tween questions and standards (skills). The attempts and duration measures are extracted from K12 platform log files. The question/item/usmo difficulty are a result by dividing the total score from users who answer the question/item/usmo correctly from the total score obtained by all the users from the current data set. Although the number of instances and average sequence length in each data set seem to be a decent size, the percentage of short sequence is quite large, about 50% of almost every data set are short sequences, indicating how many "0"s we introduce into training (see in Table 2.1).

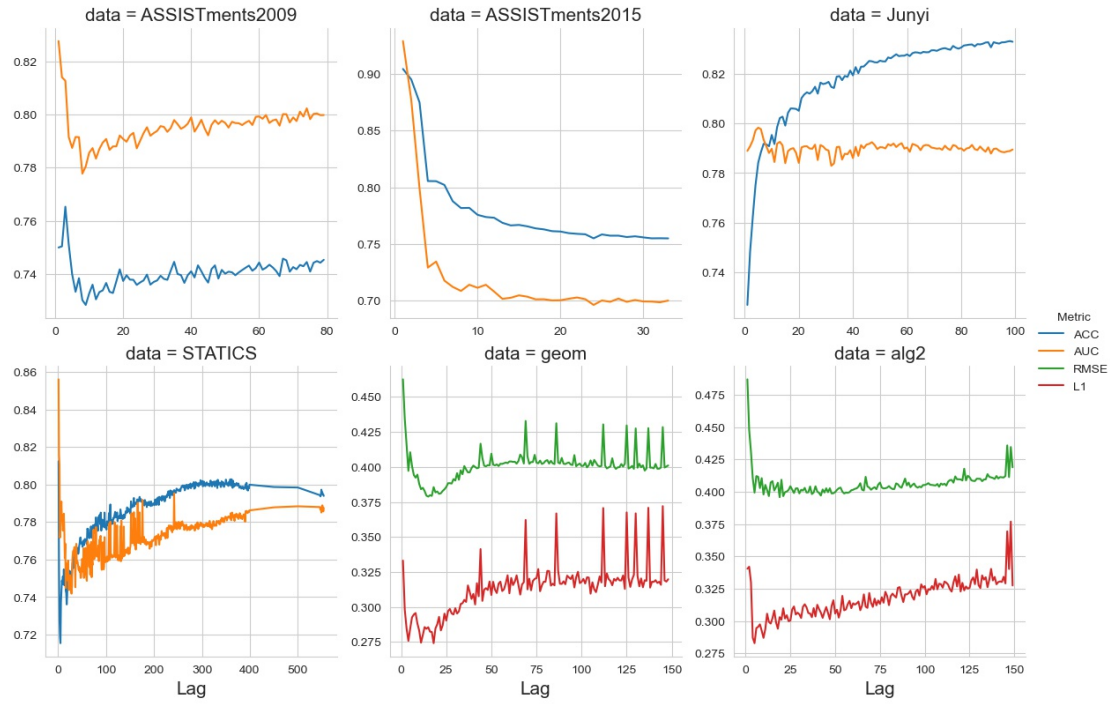
The first four data sets are binary classification problem to predict if the student will answer the next question correctly. We use Accuracy (ACC) and Area Under the Curve (AUC) to evaluate these data sets. The two private data sets are regression problems with target variable as score rates, which is a value in a range of 0-1 obtained by dividing the earned score of each individual question from the total scores that a student obtained in the data set. We use Root Mean Square Error (RMSE) and Least Absolute Deviation (L1) to evaluate these data sets. The choice of these evaluation metrics is consistent with the evaluation metrics in general KT field.

2.4 Results

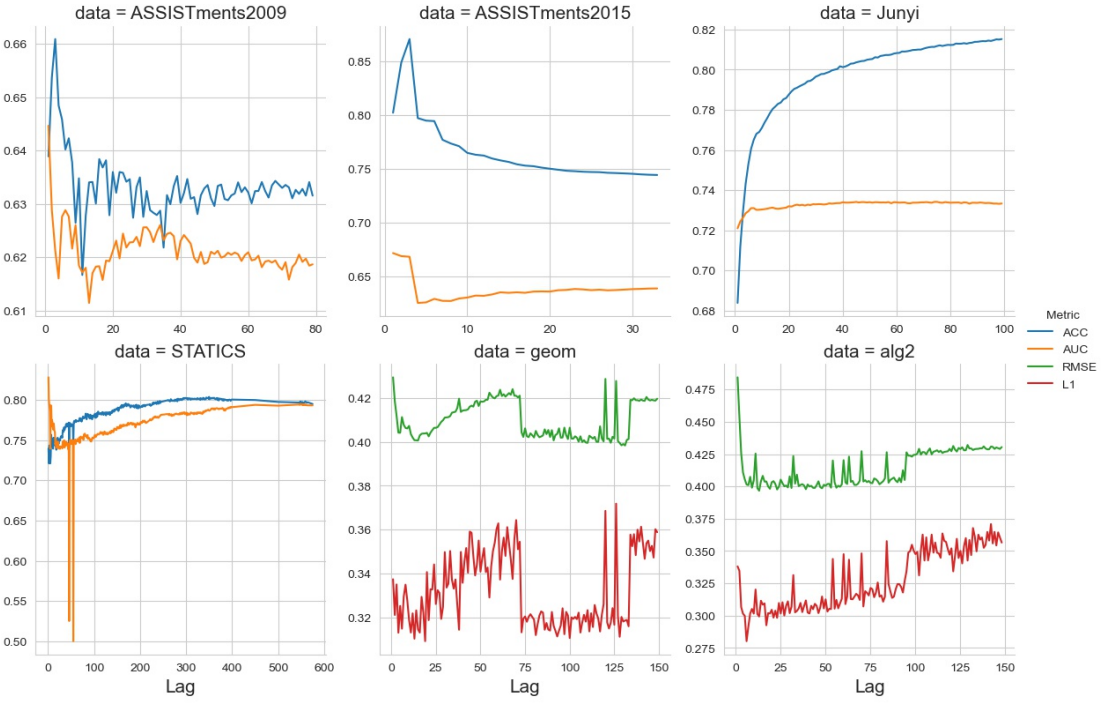
2.4.1 AQ1: Better Performance with Less Time Lags

After determining the data sets for experiments, we choose two single-variate KT models (i.e., DKT and NPA) to evaluate if reducing time lags will increase model performance (RQ1). In these two models, time lag is the only variable included and it can avoid the influence from other variables. We run experiments for every time lag till the maximum number of $\bar{L} - 1$ lags and present the performance change in Figure 2.2. The figure demonstrates that as the time lag becomes bigger, the ACC/AUC metric goes down or becomes plateau for the binary classification data sets such as ASSISTment2009, ASSISTments2015, Junyi and STATICS. We only obtain STATICS data every 50 lags after $l = 400$ to be computationally frugal and we do not foresee big spikes between 400 and 575 based on the trend.

We then present the best performance lag (l) from each data set and compare with the full lag ($\bar{L} - 1$) performance in the Table 2.2. We are able to get better



(a) DKT Performance Across Data Sets By Lag



(b) NPA Performance Across Data Sets By Lag

Figure 2.2: DKT and NPA Model Performance Across Data Sets By Lag.

Table 2.2: Performance Comparison Between Best Lag vs. Full Lags Across Data Sets. The subscripts indicate the relative improvement from the full lag performances (Negative means no improvement). * indicates significance. All performances are five random seeds average. FL: full lag.

Data Name	DKT		
	Best/Total Lags	ACC	AUC
ASSISTments 2009	FL=79	0.7428	0.7997
	3/79	0.7627_+2.68%*	0.8151_+1.92%*
ASSISTments 2015	FL=33	0.7555	0.7016
	1/33	0.9063_+19.96%*	0.9264_+32.04%*
Junyi	FL=99	0.8302	0.7842
	45/99	0.7823_-5.77%*	0.7968_+2.60%*
STATICS	FL=575	0.7950	0.7864
	1/575	0.7979_+0.37%	0.8650_+10.00%*
	Best/Total Lags	RMSE	L1
K12 Geometry	FL=149	0.4011	0.3215
	13/149	0.3849_+4.05%*	0.2837_+11.75%*
K12 Algebra II	FL=149	0.4146	0.3373
	13/149	0.4045_+2.43%*	0.3110_+7.80%*
Data Name	NPA		
	Best/Total Lags	ACC	AUC
ASSISTments 2009	FL=79	0.6322	0.6178
	5/79	0.6486_+2.60%*	0.6276_+1.59%*
ASSISTments 2015	FL=33	0.7555	0.7016
	1/33	0.8023_+7.79%*	0.6706_+4.97%
Junyi	FL=99	0.8150	0.7332
	75/99	0.8116_-0.42%*	0.7339_+0.10%
STATICS	FL=575	0.7962	0.7929
	1/575	0.7667_-3.70%*	0.8149_+2.78%
	Best/Total Lags	RMSE	L1
K12 Geometry	FL=149	0.4191	0.3541
	4/149	0.4092_+2.36%*	0.3106_+12.28%*
K12 Algebra II	FL=149	0.4296	0.3623
	21/149	0.4149_+3.42%*	0.3248_+10.34%*

performance with less time lags. For example, for ASSISTments 2009 data set, the best DKT performance we find is at $l = 3$ and best NPA performance at $l = 5$ with 2.68% and 2.60% increase in ACC respectively with significance. Similarly for ASSISTments2015 data, both models' best-performing lags is $l = 1$, which is same as what we see in Figure 2.2 with decent improvement of 19.96% and 7.79%

respectively with significance in ACC and 32.04% and 4.97% in AUC. For the two private data sets, we obtain the best performance at $l = 13$ for DKT and $l = 4$ for NPA models in K12 Geometry data and $l = 13$ and $l = 23$ for Algebra II data set comparing to the model of full-lag at $l = 150$. We observe the best performing lags are much shorter but has much higher performance than the full lag models in these two private data sets. For example, the DKT model at $l = 13$ for K12 Geometry data outperforms the full lag model by 4.05% in RMSE and 11.75% in L1 with significance and 2.36% and 12.28% for NPA model with significance. Similarly, the DKT model at $l = 13$ for K12 Algebra II data set surpasses the full lag model performance by 2.43% in RMSE and 7.80% in L1 with significance and 3.42% and 10.34% for NPA model with significance.

2.4.2 AQ2: Mild Effect on Long Sequence or Large Data Set

In Table 2.2, ASSISTments 2009 and ASSISTments 2015 are considered as small to medium data sets ($<$ 700k), where we observe significant improvement in model performance after we reduce the time lags. Meanwhile, we do not see huge improvement on the large data set (i.e., Junyi) and the long sequence data set (i.e., STATICS). However, we still find decent increase in AUC with 2.60% (sig.) in Junyi and 10.00% (sig.) in STATICS for DKT model and +0.10% (sig.) in Junyi and +2.78% (not sig.) in STATICS for NPA model. The reason why the increase is not huge for both the long sequence and large data might be due to that % of short sequences are lower than the small to medium size data sets with 45.65% and 59.63% respectively compared to above 60% for the other two data sets. Furthermore, given these two data sets are imbalanced data, AUC usually is a preferred measure to assess prediction robustness. Thus, the positive improvement from AUC is still a promising signal that we can use less time lags instead of full lag data to train for large or long sequence data sets. To answer RQ2, we observe that reducing time lags do not always work for all the data sets regardless the sequence length and data size. We observe significant improvement of ACC on small to medium size data but only discover a mild improvement of AUC on long sequence or large data sets.

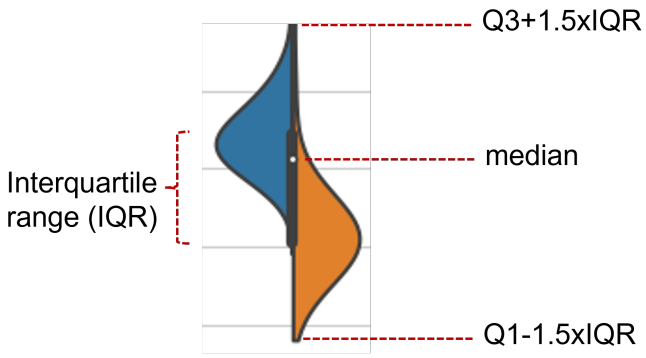


Figure 2.3: Illustration of A Violin Plot

2.4.3 AQ3: Relationship between Lag Group and Model Performance

To answer RQ3 (whether certain lag groups perform better than others), we create violin plot (see the understanding of a violin plot in Figure 2.3) in Figure 2.4 and 2.5 to demonstrate all the model performances in terms of distribution by ACC (blue) and AUC (orange) across the four public data sets. Given each data set has different lag sequence, we plot lag groups via 20% increments: $\leq$ 20%, 20-40%, 40-60%, 60-80%, $\geq$ 80%. For STATICS and Junyi data sets in Figure 2.5, we observe the higher lag group center points of both ACC and AUC distributions are higher than the ones from the lower lag group, especially lag group 40-60% having the highest mean of all. Note the white dot is median whereas the mean is the center point of the blue/orange distribution. The phenomenon that large or long sequence data sets have better performances at higher lag indirectly infers that the mild effect on the large/long sequence data sets. For small to medium size data sets such as ASSISTments2009 and ASSISTments2015 in Figure 2.4, it is noticeable that lower lag groups especially group $\leq$20% have higher point of means for both DKT and NPA models. In addition, they have much longer tails (where the maxima-highest performance is) than the higher lag groups demonstrated by the orange/blue dotted lines in Figure 2.4, indicating the highest performance appear more often in lower lag groups. When evaluating both DKT and NPA models, we keep all the training and model hyper-parameters same for both the data sets. Therefore, the model performance increase is mainly due to the time lag difference. Thus, we answer RQ3 that for small to medium size data set, $\leq$20%

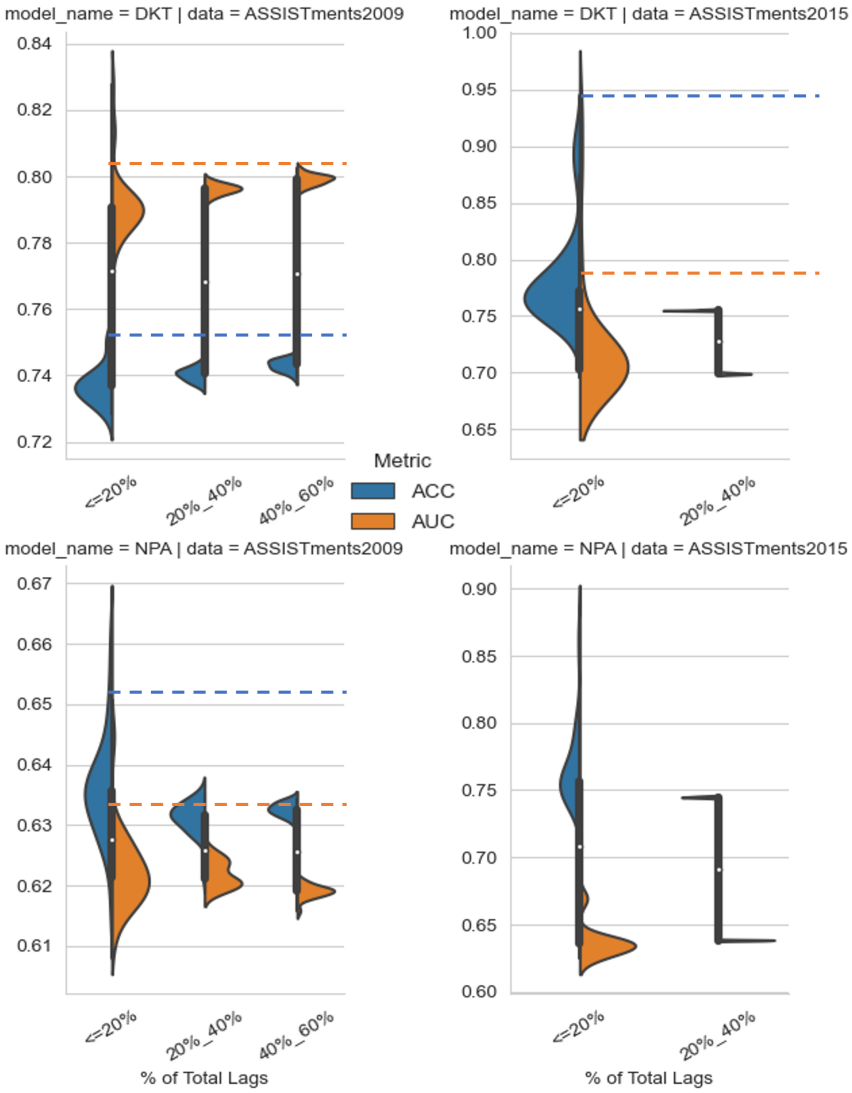


Figure 2.4: Small to Medium Size Data Sets Classification Task Maximum Performance by Lag Group. The orange/blue horizontal line are plotted to compared the maxima (highest point) on y axis for the orange/blue distribution. The figure is scaled by each category's total sample for a fair distribution comparison.

lag group have better performance than others and 40-60% lag group seem to be superior than other groups for large or long sequence data sets. In practice, 20% lags are about 7-14 time steps for ASSISTments data sets, which equates about 1-2 days of learning journey if students take 7 questions each day. For STATICS and Junyi, 40-60% lag group are about 250 time steps, which equates 30 days of learning journey (based on 7 steps/day). This indicates the the past month

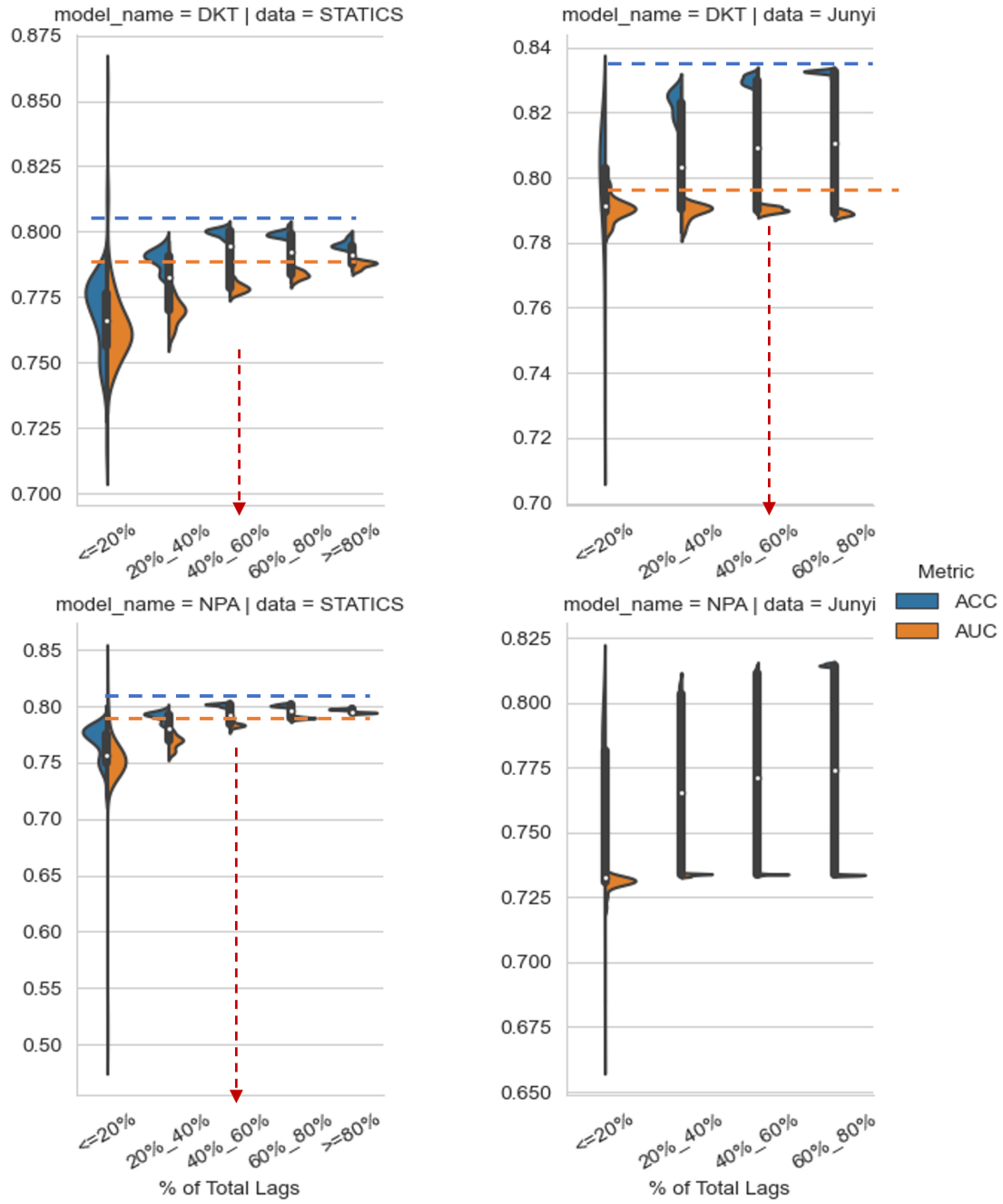


Figure 2.5: Long (STATICS) /Big (Junyi) Sequence Data Sets Classification Task Performance by Lag Group. The orange/blue horizontal line are plotted to compared the maxima (highest point) on y axis for the orange/blue distribution. The red dotted arrow points to the lag group 40-60%.

learning have more impact on the future question answering for students taken courses in STATICS and Junyi data sets.

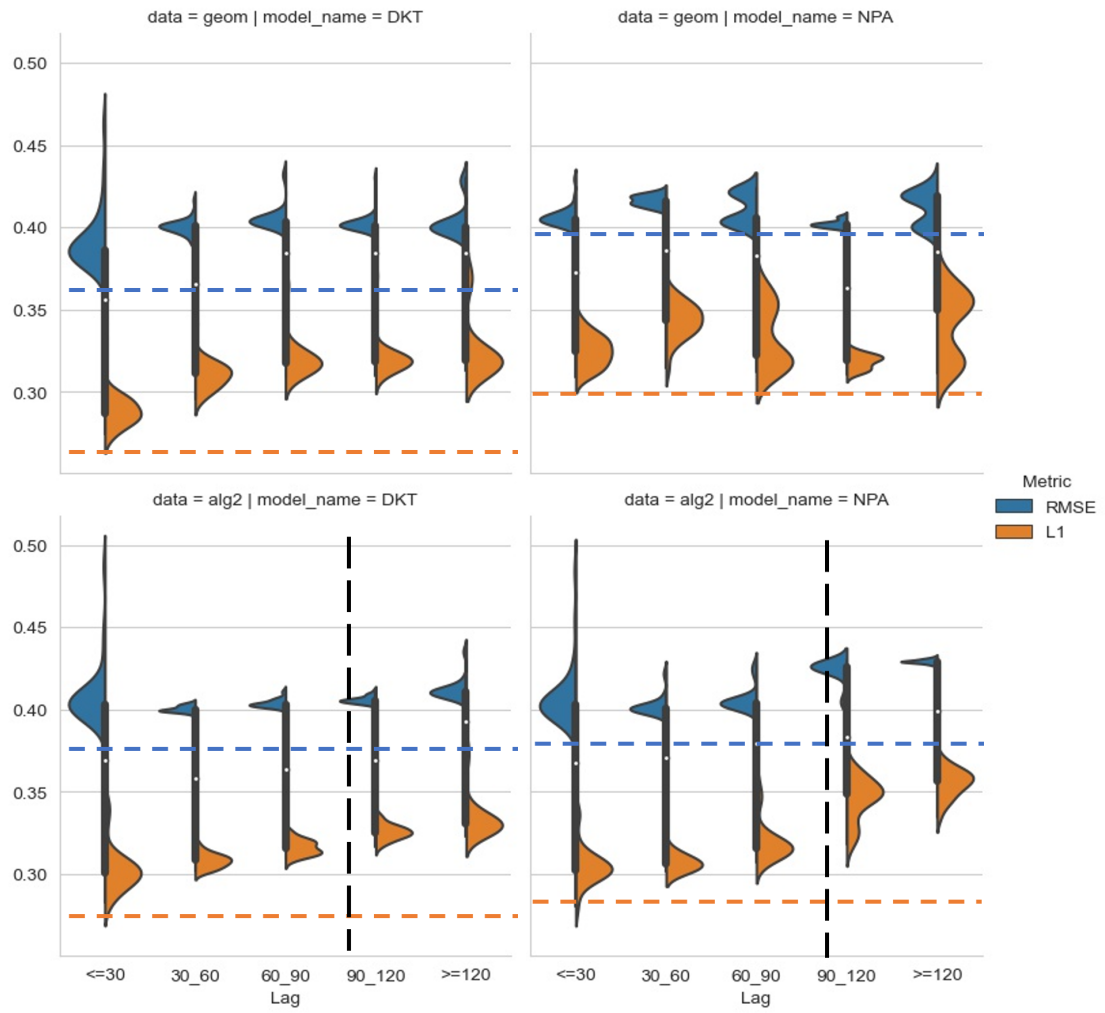


Figure 2.6: Regression Task Performance by Lag Group in Violin Plot. The orange/blue horizontal line are plotted to compared the minima (lowest point) on y axis for the orange/blue distribution. The figure is scaled by each category's total sample for a fair distribution comparison.

We plot Figure 2.6 to show the performance distribution in terms of RMSE and L1 by 30 lag increments for the two private data sets. Although the two private data sets have multiple variables, we only include the score rate and the sequential order in the training, which makes it comparable to other single-variate data sets. In the figure, we observe an increasing trend in means of RMSE and L1 distributions as the lag groups increase intervals. For example, lag group of ≤ 30)

has much lower RMSE and L1 error than higher lag groups (e.g., 90_120, ≥120) demonstrated by the orange/blue dotted lines. We also notice the distribution centers (i.e., mean) go much higher after lag 90 (see the black dotted line in Figure 2.6), which indicates lags $\leq$ 90 days have better performance than lags $\geq$ 90. According to the K12 Geometry and Algebra II data sets, the average questions that a student take per day is 8 and 9 respectively. This means about 3 (when $l = 30$) to 10 (when $l = 90$) days of the past question answering could well infer a student's score rate for the next question.

2.5 Limitation

The above experiment although exhaustively looped over all the time lags to examine which particular time step can drive the best performance. We are not able to run experiments on other classic KT models such as DKVMN [9], SAKT [10], SAINT [19] due to the limitation in computation resource. We are also aware of other large KT data sets such as EdNet [20] and NeurIPS 2020 Challenge Data set [21] which can be interesting to test on because the fixed sequence length could be much longer and the data size is much larger. Therefore, if we are given much ampler resources and time, we would also like to run experiments on the aforementioned KT models and data sets to see if our results hold.

2.6 Summary

In this work, we called out the data sparsity that exists during training even after the data set is increased by time lags. We then proposed training with less time lags to reduce sparsity. We demonstrated that reducing the time lags can greatly increase model performance for a margin of up to 32% of AUC in classification tasks and up to 12% of RMSE in regression tasks with significance for small to medium data sets but mild effect for the large or long sequence data set. In addition, we discovered the past learning information of less than 10 days has more impact on students' performance for small-to-medium data sets (i.e., ASSISTments and Geom and Alg2 data sets) and about 1 month of past learnings affect their next engagements with tests for longer sequence data sets (i.e., STATICS and Junyi).

Chapter 3
Augment Data via Subject-based VAEs

Knowledge tracing (KT) as a student modeling technique has been widely used to predict and trace students' knowledge state during their learning processes. In recent years, with the huge success that deep learning has brought to the field, there are many KT algorithms that can predict individuals' knowledge state to a decent extent. However, the *sparseness* of students' exercise data represented by *missing values* still limits the models' performance and application [22]. About half of the existing publications use public data sets [23], which can not be available for huge amount due to administration cost. Researchers could opt for other private data sets that however may not even have the sizable volume as the public data sets. Besides, many deep learning algorithms including the state-of-art (SOTA) KT algorithms need huge and diverse amount of training data to obtain decent performances. On the other hand, it is unavoidable to see the missing values in KT data because of two reasons: (i) data is missed completely at random (MCAR) where the probability of missing data is independent on its own value and on other observable values [24]. For example, due to COVID, we have many students missing exams; (ii) the data is missed not at random (MNAR), which indicates the reason for a missing value can depend on other variables but also on the value that is missing. For example, if a student performs poorly on the English subject and often miss exams in other subjects, his missed records in English quizzes can be attributed to other known reasons. Moreover, KT data is a type of longitudinal data, all collected repeatedly over time for each *subject* (i.e., student). Such data contains both dependent and independent variables. For example, the dependent

variables in KT data can comprise time-varying measurements per *subject* (e.g., response correctness, time taken per question), whereas independent variables are time-invariant *subject descriptors* (e.g., grade, gender, gifted or not) (see the illustration in Figure 3.1). Analyzing such data is challenging as it often includes high-dimensional time [in]variant variables with missing values. Despite that missing data in KT field is ubiquitous and poses challenges on achieving better model results, there are very few studies researching on effective approaches to tackle the missing data issue in KT field. Our work is one of the few studies to address such challenge.

To that end, we suppose a deep generative model such as Variational Autoencoders (VAE) [25] could effectively generate data for the missing values because of its superiority over other generative models (e.g., Generative Adversarial Networks) in time series data generation [26,27]. Furthermore, given the challenge arisen from the longitudinal KT data, we make two hypotheses: (i) a training style that can reflect the subject longitudinal nature could be more effective; (ii) the information from subject descriptors could potentially represent the latent space better and help improve the quality on the data generation. To validate hypothesis (i), we develop a subject-based training style where we split and impute data by student IDs to reflect the longitudinal nature of the subjects. The benefit of doing so is to maintain the complete sequence for each student whereas splitting by row number could separate the individual sequence and entail inefficient training. Thus, applying subject-based training on top of VAE framework could potentially address the challenge. To validate hypothesis (ii), we leverage a module from the existing Longitudinal VAE (LVAE) [28] framework called additive multi-output Gaussian Process (GP) prior that can extrapolate the correlation between time-invariant subject descriptors and the latent space to enhance the latent variable learning. Given the longitudinal nature of the LVAE framework, a subject-based training can be naturally applied to boost data generation quality. Furthermore, we build LSTM kernels to both VAE and LVAE frameworks because LSTMs are good at extrapolating the temporal relationship from multi-variate time series data [29,30]. With the generated data from the proposed frameworks, we will be able to impute them back for retraining and evaluate the effectiveness of the imputed data on boosting the original model performance. Besides, we are also interested to discover how robust our generated data can be on boosting the original model

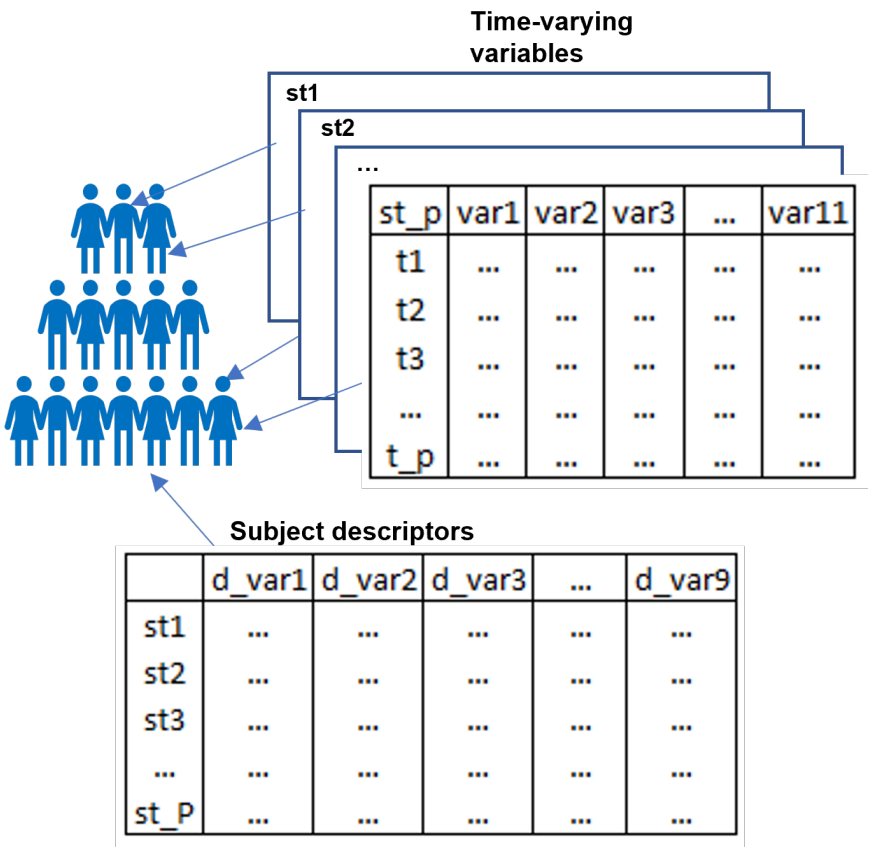


Figure 3.1: An Illustration of the Longitudinal (student) Data in KT Field. 'p': student p. 'P': total # of students.

performance, e.g., by only applying a fraction of the generated data.

Thus, this work attempts to make the following contributions:

- Overcoming the issue of the missing KT data, we conduct *subject-based* training on KT data via LSTM-VAE framework
- Leveraging the additive GP prior module from LVAE, we form a LSTM-LVAE framework to showcase the superiority of training additional *subject descriptors* for better latent space representation
- We demonstrate the robustness of only using a fraction of the generated data to boost the original model performance

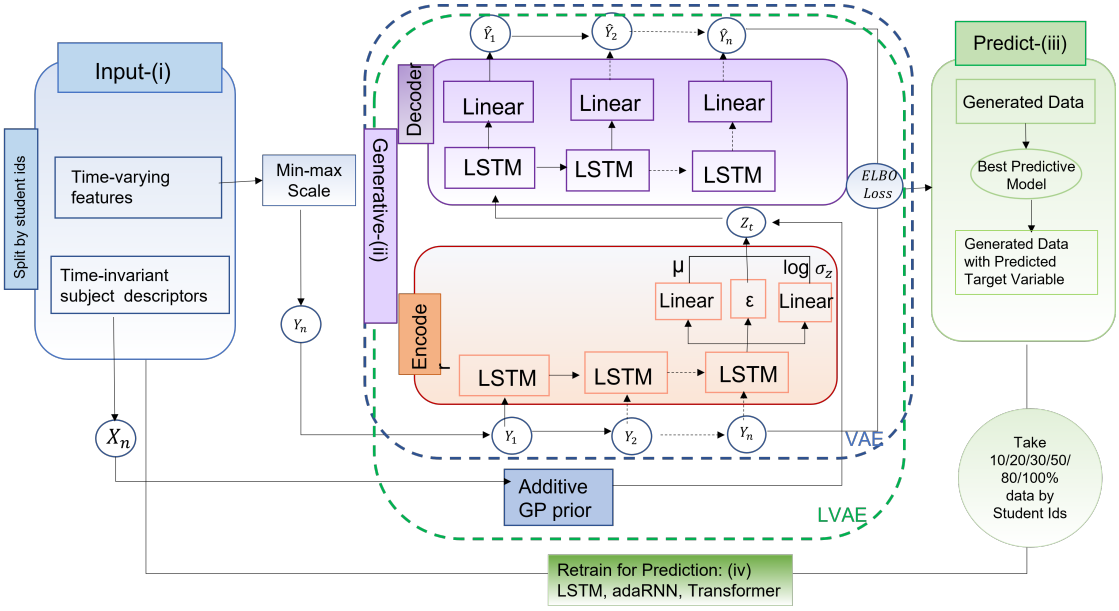


Figure 3.2: Overview of the methodology proposed in this work.

3.1 Method

We propose two deep generative frameworks: LSTM-VAE and LSTM-LVAE. The both frameworks use subject-based training. We explain the details as follows.

3.1.1 Problem setting

According to [28], let D be the dimensionality of the observed data, P be the number of unique students, n_p be the total number of longitudinal samples from student p, and $N = \sum_{p=1}^{P} n_p$ be the total number of samples. Therefore, the longitudinal samples for student p are denoted as $Y_p = [y_1^p, ..., y_{n_p}^p]^T$, where each sample $y_t^p \in \mathcal{Y}$ and $\mathcal{Y} = \mathbb{R}^D$. The subject descriptors for students are represented as $X_p = [x_1^p, ..., x_{n_p}^p]^T$, where $x_t^p \in \mathcal{X}$ and $\mathcal{X} = \mathbb{R}^Q$, Q be the number of descriptors. The latent space is then denoted as $\mathcal{Z} = \mathbb{R}^L$ and a latent embedding for all N samples as $Z = [z_1, ..., z_N]^T \in \mathbb{R}^{N \times L}$ with L being the number of latent dimensions. To generate data, a joint generative model is then parameterized by $w = \{\psi, \theta\}$ as $p_w(y, z) = p_\psi(y|z)p_\theta(z)$. Therefore, if the latent variable z is known, it will be easy to infer y and hence generate the desired data.

3.1.2 VAE and LVAE

To infer the latent variable z given y, the posterior distribution is $p_w(z|y) = p_\psi(y|z)p_\theta(z)/p_w(y)$ and is generally intractable due to the marginalization over the latent space $p_w(y) = \int p_\psi(y|z)p_\theta(z)dz$. Therefore, Variational Auto-Encoder [25] introduced the approximated version posterior, noted as $q_\phi(z|y)$ instead of the true posterior $p_w(z|y)$ and fit the approximate inference model by maximizing the Evidence Lower Bound (ELBO) of the marginal log-likelihood w.r.t. ϕ:

$$\log p_w(Y) \geq \mathcal{L}(\phi, \psi, \theta; Y)$$

$$\triangleq \mathbb{E}_{q_\phi}[\log p_\psi(Y|Z)] - D_{KL}(q_\phi(Z|Y)||p_\theta(Z)) \to \max_\phi,$$

where $\mathbb{E}_{q_\phi}[\log p_\psi(Y|Z)]$ is a reconstruction error, measuring the difference between the input and the encoded-decoded data. and D_{KL} denotes the Kullback-Leibler Divergence (KLD), measuring the divergence between $q_\phi(Z|Y)$ and $p_\theta(Z)$. In practice, we minimize the negative ELBO: $D_{KL}(q_\phi(Z|Y)||p_\theta(Z) - \mathbb{E}_{q_\phi}[\log p_\psi(Y|Z)]$, where all the parameters are learned simultaneously together: $\mathcal{L}(\phi, \psi, \theta; Y) \to \min_{\phi,\psi,\theta}$.

When facing the longitudinal data, [28] hypothesize z has relationship with both Y and X and formulate the generative model as

$$p_w(Y|X) = \int_Z p_\psi(Y|Z, X)p_\theta(Z|X)dZ$$

$$= \int_Z \prod p_\psi(y_n|z_n)p_\theta(Z|X)dZ,$$

where $p_\psi(y_n|z_n)$ is normally distributed probabilistic decoder and $p_\theta(Z|X)$ is defined by the multi-output additive GP prior that regulates the joint structure of Z with descriptors X. The Additive GP is a Gaussian process prior as $f(x) \sim GP(\mu(x), K(x, x'|\theta))$, where $\mu(x) \in \mathbb{R}_\mathbb{L}$ is the mean (assumed as 0) and $K(x, x'|\theta)$ is a matrix-valued positive definite cross-covariance function (CCF). Based on the practice of [31], LVAE constructs the additive GP components with squared exponential CFs (from continuous variables), categorical CFs (from categorical covariates),the interaction CFs (the product of the categorical and squared exponential CFs) and the product of the squared exponential CFs and the binary CFs. Finally,

the ELBO function changes to the following after factoring the descriptors X:

$$\log p_w(Y|X) \geq \mathcal{L}(\phi, \psi, \theta; Y, X)$$

$$\triangleq \mathbb{E}_{q_\phi}[\log p_\psi(Y|Z)] - D_{KL}(q_\phi(Z|Y)||p_\theta(Z|X) \rightarrow \max_\phi.$$

LVAE differentiates from VAE in that it hypothesizes there exists a relationship between $\mathcal{X}$ and the latent space $\mathcal{Z}$ and uses an additive multi-output Gaussian Prior to extract that relationship.

3.1.3 Generative Frameworks

Based on above solutions, two generative frameworks are developed (see in Figure 3.2). It has 4 phases: (i) input phase that pre-processes data; (ii) generative phase where data gets generated via the generative model framework; (iii) prediction phase where we predict target variable for the generated data; (iv) retraining phase, where we combine the original data and generated data to retrain for donwstream prediction task.

From the figure, after input phase (i), we see that the data gets separated into two sets: (a) time-varying data (noted as $Y = \{y_1, ..., y_n\}$); (b) time-invariant subject descriptors (noted as $X = \{x_1, ..., x_n\}$). The time-varying data y_n goes through a min-max scaler, a typical time series data normalization method [32], and enters the LSTM encoder to generate μ and $\log \sigma_z$ for the latent distribution Z_t. The time-invariant subject descriptors X_n on the other hand are only fed into the Additive GP prior module to train for the approximated GP prior with its output merging into the latent space Z_t. Next, the decoder samples on the latent distribution and reconstructs data $\hat{Y}_n$, namely encoded-decoded data, based on the latent features from Z_t. Here, we name the generative framework that only includes the encoder and decoder as LSTM-VAE and the framework that includes encoder, decoder and the additive GP prior module as LSTM-LVAE. We omit LSTM prefix for simplicity. After that, we compare Y_n to $\hat{Y}_n$ for evaluation via ELBO. VAE assigns the equal weight for both reconstruction and KLD errors whereas LVAE assigns a weight to KLD to regularize further. Once we have good generation quality, we generate data on the missing data which has all the subject descriptor information but missing on all the time-varying features.

Before entering phase (iii), we conduct initial prediction on the original data via models that work well with multi-variate time series data: LSTM, adaRNN and Transformer. Similar to LSTM, adaRNN (i.e., adaptive RNN) [33] is a recurrent neural network but based on the Gated Recurrent Unit that comprises two gates (i.e., reset gate and update gate). It usually trains faster than LSTM and easy to modify and works better if the sequence is not too long. Because some KT data could present non-sequential characteristics, we include the original Transformer model [34], whose attention mechanism and positional embedding are great for non-sequential data. To evaluate these models, we use Root Mean Square Error (RMSE) as our target variable is continuous (i.e., score rate, the possible score obtained per question divides the total scores obtained per student). After the initial round of prediction is performed, we conduct phase (iii) by selecting the best predicting model to predict the target variable for the generated data from phase (ii). In phase (iv), we impute the fraction of 10/20/30/50/80/100% of the generated data (with target variable) back to the original data and retrain for the downstream predictive task.

3.1.4 Subject-based Training

Besides the generative frameworks, this work also takes a new training strategy, that is, subject-based training. We refer subject-based training to a style where data are split and imputed back by student IDs instead of row number. We call the training using row-number splitting as non-subject based training. For example, in subject-based setting, 70% of student IDs are extracted as training data and 10% student IDs are extracted as validation data whereas in non-subject based setting, 70% of total rows are extracted as training data and 10% total rows are extracted as validation data) (see the illustration in Figure 3.3). We see the split points by IDs are not the same as splitting by row number. It indicates there is chance that the sequence of certain students will be cut into two pieces, leaving them into two different sets (e.g, val and test). If we split the data by student IDs, we can impute the generated data back to the original data via IDs and keep the learning sequence relevant and complete for each student. If we opt for row-number splitting, the student's original sequence will be interfered and not be trained appropriately.

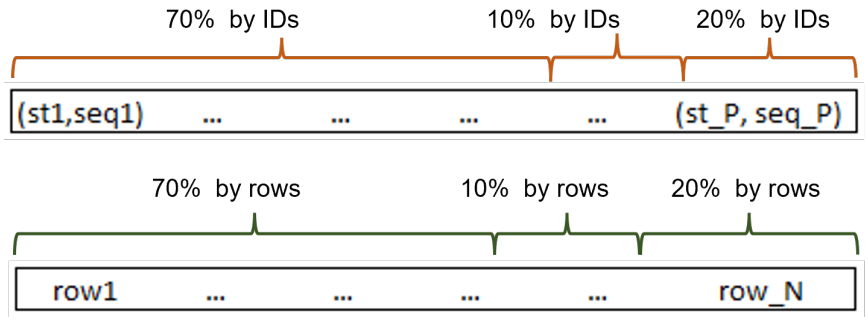


Figure 3.3: An Illustration of Split by IDs vs. Row Number

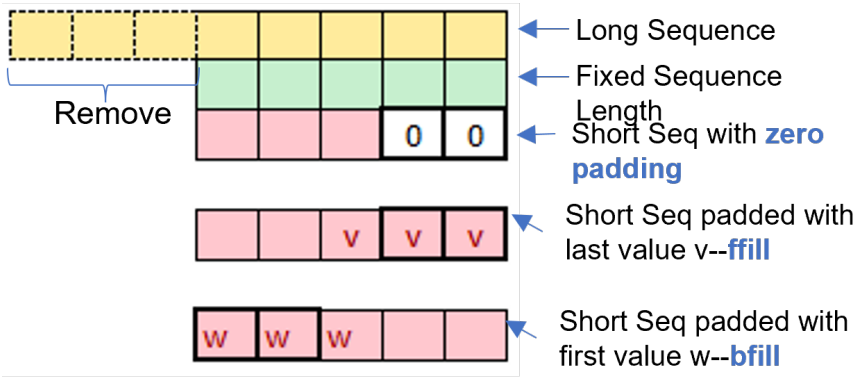


Figure 3.4: An illustration of KT data sequence aligning process and 3 Padding Strategy.

3.2 Experiments

In this section, we carry out two major experiments. The first experiment is to generate the data and impute back to the original data for retraining. It has three steps: (a) generate knowledge tracing data by utilizing VAE and LVAE; (b) predict the target variable for the generated data using the best model obtained from the KT prediction task; (c) merge the generated data with the original data to retrain for model performance. The second experiment is to validate the robustness of imputed data on boosting the original model performance. More specifically, we add a fraction of the generated data in the cadence of 10%, 20%, 30%, 50%, 80%, 100% during the retraining phase to examine the boosting effect.

3.2.1 Data sets

To achieve above, we need to apply our model onto the data sets that have subject descriptors so that we can use LSTM-LVAE model to generate missing data. Unfortunately, the public data sets (e.g., ASSISTment datasets, Junyi, STATICS, EdNet, etc) in KT field do not contain subject descriptor information such as the student's grade level, gifted or not. This is also why the renowned deep learning models such as DKT, DKVMN, NPA, SAINTS [9–11, 35] are not included in the chapter because most of these models are generated for single variable KT data or take data feature as hyper-parameters. Thus, we use the two private multivariate KT data sets from K12.com platform (an online K-Grade 12 education platform). They are : (1) Grade 10 geometry course (noted as Geom) quiz answering data set with average sequence length of 150 time steps; (2) Grade 11 algebra II (noted as Alg2) quiz answering data set with average sequence length of 150 time steps. Each data set contains 11 temporal features (i.e. sequence number, assessment duration, attempts per question, total attempts, question difficulty, item difficulty, standard difficulty, question reference, item reference, standard id, question type) from July 2017 to June 2019 and 7 subject descriptors that define the student profiles (i.e., school ID, special ED, student id, free reduced lunch, gifted_talented, grade level, score rate). The Geom data set contains 3,265 total students with 412,397 observed instances whereas the Alg2 data has 2,110 total students with 277,548 observed instances.

3.2.2 Identify Missing Values

In practice, it is hard to identify the missing steps each student has because their learning experience varies. Thus, we develop a regime where we first find all the quiz times of a school where the student is located and then fill up the missing times by comparing to the school's full quiz taking schedule. For example, if school A has 100 quiz times but student A only has 60 records, we fill out the remaining 40 quiz time steps based on the event time variable. This approach is a bit rigorous, assuming all the students are required to test for the same number of quizzes if they are in the same school and skipping any quiz is considered as a missing step. In reality, there might be scenarios where students are allowed to skip, which is complex to study and hence we use this approach as it is straightforward. With

that, we are able to retrieve the missing time steps before and after the current temporal steps for all the students. As the students are known, this missing data has all the subject descriptor information.

3.2.3 Data Processing

To conduct the training for generation, we split the data by 0.5/0.1/0.2/0.2 for train/val/test/generate and 0.7/0.1/0.2 for train/val/test during downstream prediction (see in Table 3.1). Note that the generation set with a ratio of 0.2 is used to evaluate the quality of generation whereas the generated set we use to impute back to the original data is generated from missing data. Based on the above missing data identification regime, we are able to identify 3,233 out of 3,265 total students who have a total of 799,408 missed instances from the Geom course and 2,057 out of 2,110 students who have a total of 516,884 missed instances from the Alg2 course (see in Table 3.1). Because all the ratios are applied to both subject and non-subject based training, the generated data from missing values will be imputed back to the original data via IDs in the subject-based training and via row number in the non-subject based training in the splits of train/val/test. Both training styles align data to a fixed sequence length which is due to the model input requirement of 3D dimensions (i.e., batch size * sequence length * number of dimensions). This also aligns with the typical data processing technique for KT model training [8,10,36]. If the actual student learning sequence is longer than the fixed sequence length, we cut the part where it exceeds. If the sequence is shorter than the fixed sequence, we pad it (see in Figure 3.4). We use three padding strategies to find an optimal model performance: (a) zero paddding; (b) ffill; (c) bfill. Ffill pads forward with the last value 'v' whereas bfill pads backward with the first value 'w' (see in Figure 3.4). Bfill in practice assumes that a student gets the same quiz result in his missed quiz as his first quiz result whereas ffill assumes that a student gets the same quiz result in his missed quiz as his last quiz result. Zero-padding just simply assumes that a student gets zero in his missing quiz.

3.2.4 Generation and Imputation

We train three generative models: VAE-NS (non-subject), VAE (subject-based) and LVAE (subject-based) to generate missing data. Since LVAE is only possible

Table 3.1: Data Statistics for Geom and Alg2 Data. * is Downstream Task Split

Split Part (Ratio)	Geometry (Geom)		Algebra II (Alg2)	
	# Student	# of Rows	# Student	# of Rows
Train (0.5)	1,633	215,632	1,055	137,409
Validate (0.1)	326	42,259	211	30,652
Test (0.2)	653	82,707	422	60,709
Generation (0.2)	653	71,799	422	48,778
Data Total	3,265	412,397	2,110	277,548
Train* (0.7)	2,286	287,431	1,477	186,187
Validate* (0.1)	326	42,259	211	30,652
Test* (0.2)	653	71,799	422	60,709
Data Total*	3,265	412,397	2,110	277,548
Missing Train (0.7)	2,256	559,586	1,440	361,819
Missing Validate (0.1)	322	79,941	206	51,688
Missing Test (0.2)	645	159,882	411	103,377
Missing Data Total	3,223	799,408	2,057	516,884

to train if we have descriptor information, which relies on student ID information, we do not apply non-subject training for LVAE. After data is generated for all the missing data, we impute back the generated data from VAE-NS by the row-number splits and impute the generated data from VAE and LVAE by ID splits (see in Figure 3.5). We do not only impute back the generated data to the train set because we believe the data augmentation on all the train, val and test sets will make the model performance harder to improve than we only augment the train set but leave the test set the same.

3.2.5 Downstream Prediction

There are two rounds of downstream predictions. The initial round is conducted on the original data using the three padding strategies to find the best performance model so that we can use it to predict the target variable for the generate data. The second round is a retraining round where we impute back the generated data using the best padding strategy. The second round has two parts: (i) we conduct the re-training on the combined data that contains all the generated data and the original data by IDs (for VAE, LVAE) and by row number (for VAE-NS); (ii) we conduct retraining on the combined data with a fraction (i.e., 10/20/30/50/80/100%) of

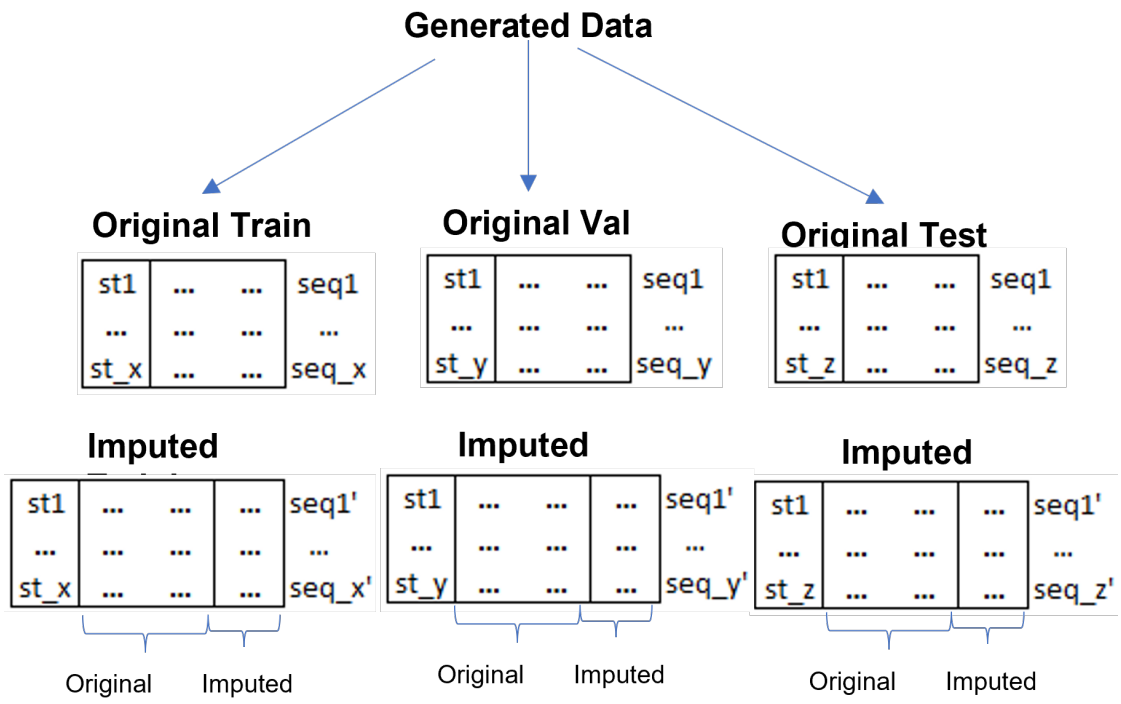


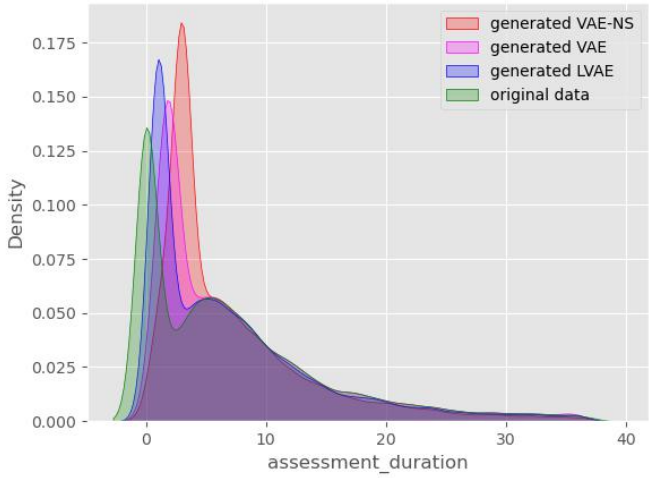
Figure 3.5: An Illustration of the Data Imputation Process.

the generated data and the original data only by IDs (for VAE, LVAE) because VAE-NS does not show salient improvement with the data it generates.

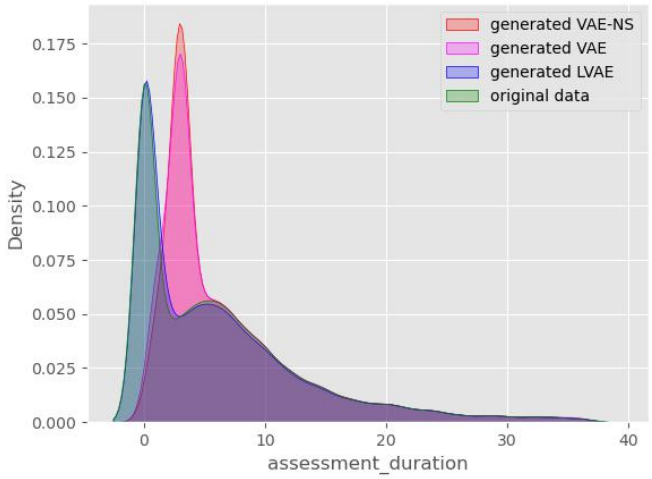
3.3 Results

3.3.1 Evaluating the Quality of the Generated Data

We exhaustively train three generative models (i.e., VAE-NS, VAE and LVAE) until its loss stops improving with different sets of hyper-parameter tuning to reach the best result. We observe that VAE-NS model is hard to converge and stops early with final loss of around 13.4349 for Alg2 data set and 0.6282 for geom data set. VAE is able to decrease loss to 0.2652 for Alg2 data set and 0.2661 for geom data set. LVAE however can decrease loss to 0.2291 for alg2 data and 0.1863 for geom data set with the latent dimension as 64 and hidden dimension as 128. Figure 3.6 selects the 'assessment_duration' feature to compare the data distribution between original data and generated data by VAE-NS, VAE and LVAE. We can tell that the Geom generated data for the feature 'assessment_duration' from VAE-NS sways



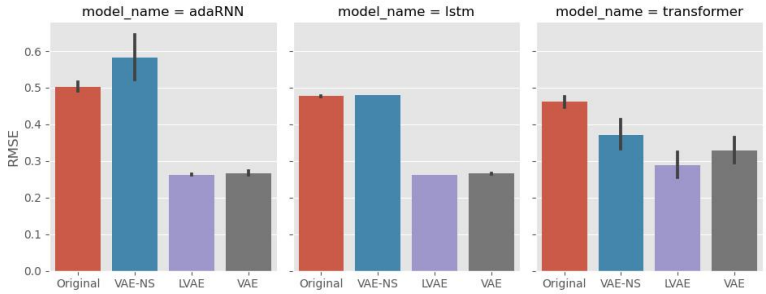
(a) Geom Data Set



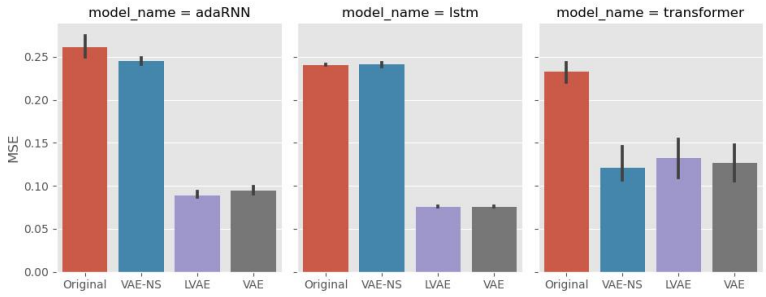
(b) Alg2 Data Set

Figure 3.6: 'Assessment Duration' Feature Distribution Comparison Between Original Data and Generated Data

the farthest from the original data whereas the generated data from VAE and LVAE are closer to the original data distribution with LVAE slightly better. The same case applies to the Alg2 data. The plot also shows that generally both VAE and LVAE can reconstruct data closely to the original data distribution, indicating that we are safe to use such generated data for downstream prediction tasks.



(a) Geom Data Set



(b) Alg2 Data Set

Figure 3.7: Average Retraining RMSE Using Generated Data From Different Models. The error bar shows the min. and max. of the 10 random seeds.

Table 3.2: Average RMSE by Padding Strategy, Models and Data sets. The boldface represents the best performance.

Avg. RMSE	Geometry (Geom)			Algebra II (Alg2)		
	adaRNN	LSTM	Transformer	adaRNN	LSTM	Transformer
Bfill	0.50734	0.47665	0.48613	0.52034	**0.48967**	0.49463
Ffill	0.51946	**0.47664**	0.49713	0.51895	0.48995	0.49632
Zero	**0.48160**	0.47702	**0.40208**	**0.48860**	0.49173	**0.45138**

3.3.2 Evaluating the Effectiveness of the Imputed Data

Before we impute the generated data, we conduct the first round of downstream prediction via three models (i.e., LSTM, adaRNN and Transformer) by three padding strategies (i.e., bfill, ffill and zero padding) to select the best model performance as the baseline original data model performance. We run 10 random seeds for each model, padding strategy and data set. Table 3.2 shows the detailed average RMSE for each model by padding schemes. We see that adaRNN and Transformer model obtain the best performances when padded with zero whereas LSTM model

Table 3.3: Average RMSE by Generative Models for Prediction Tasks. The bold face represents the best performance.

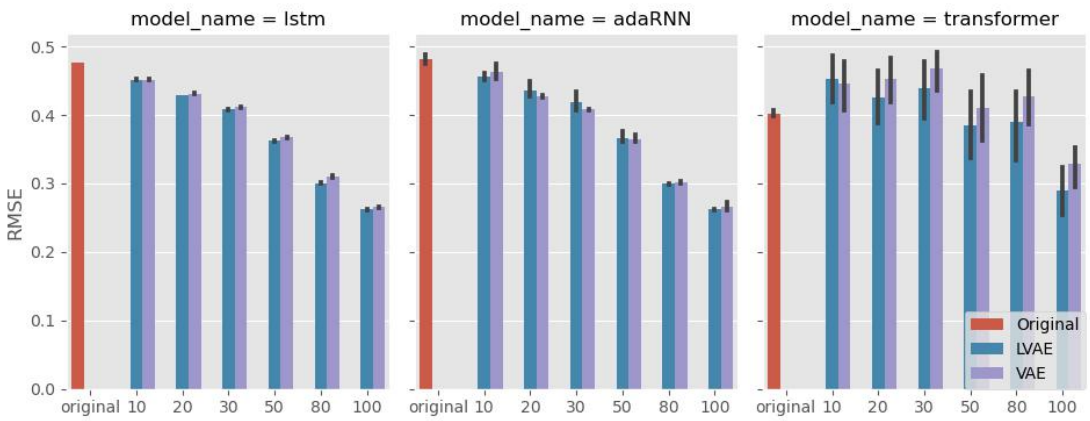
Avg. RMSE	Geometry (Geom)			Algebra II (Alg2)		
	adaRNN	LSTM	Transformer	adaRNN	LSTM	Transformer
Original	0.48160	0.47664	0.40208	0.48860	0.49173	0.45138
VAE-NS	0.58251	0.48030	0.37090	0.49388	0.48989	0.34071
VAE	0.26570	0.26559	0.32902	0.30304	**0.27293**	**0.35260**
LVAE	**0.26226**	**0.26185**	**0.28913**	**0.29470**	0.27326	0.35911

obtains its best performance via ffill padding for Geom data and bfill for Alg2 data. We then use the best performing model to predict target variable for the generated data and impute back the generated data (with the target variable) to the original data set for retraining.

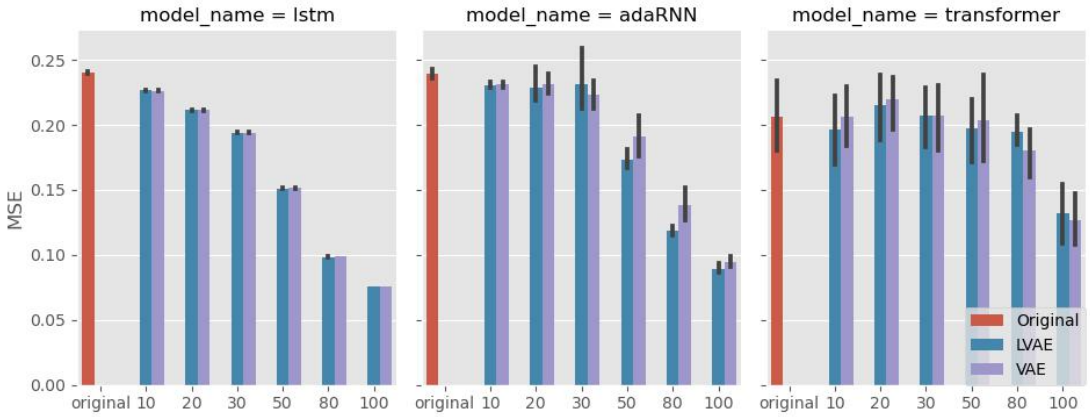
We observe the retrained model performance surpasses the original model performance by big margins (see in Table 3.3). From the table on column 1 under Geom data, we observe the retrained adaRNN model performance using the generated data from VAE is 0.26570, almost about 50% lower than the original model performance of 0.48160 in RMSE. Oppositely, the retrained model performance using the generated data from VAE-NS has RMSE of 0.58251, which is higher than the original model RMSE. This might indicate the generated data from non-subject based training perturbs the original data and creates negative gain. Further, we notice the model performance of using generated data from LVAE is even slightly better than VAE with a lower average RMSE of 0.26226. This phenomenon is present across the three models for Geom data. For Alg2 data, we also observe superior performance from both VAE and LVAE. However, the retrained model using generated data from VAE seems to perform slightly better than the one with LVAE generated data. Figure 3.7 visually presents the sharp drop of the average RMSE after imputing the generated data from both VAE and LVAE models.

3.3.3 Evaluating the Robustness of the Imputed Data

Once we learn that imputed data can boost original model performance to a significant extent, we further experiment to validate the robustness of the imputed data. More specifically, we impute the number of students in the fraction of 10%, 20%, 30%, 50%, 80%, 100% back to their original train/val/test sets. The choice of percentage increments in number of students are arbitrary but all the students



(a) Geom Data Set



(b) Alg2 Data Set

Figure 3.8: Average RMSE by % of Imputed Data vs. Original Data.

are linked back via their IDs to the original train/val/test sets. It is designed this way so that it is harder for the retrained model to outperform the original model as the number of students in the train/val/test set are still the same but with longer sequences. Figure 3.8 showcases the effectiveness of adding different fractions of students to boost the original model performance. For LSTM and adaRNN model, we observe that the model performance starts to boost after imputing only 10% of student IDs back. As the percentage gets higher, we see higher boosting. For Transformer model, it starts to boost after imputing 50% of student IDs back. This confirms a known fact that large models such as Transformer model needs more data to boost its performance. In general, imputing data based on the subjects

can boost the model to a great extent.

3.4 Summary

In conclusion, to augment missing data in KT field, we first identified missing values by school testing schedules and then we train two deep generative models (i.e., VAE and LVAE) to generate quality data in the subject-based setting for imputation. With the imputed data, we are able to boost the original model by almost 50% in average RMSE. In addition, we validate the robustness of the imputed data and observe that only 10% of students data are needed to boost the original model performance for small to medium models such as LSTM and adaRNN and 50% of students data are needed to boost large models such as Transformer. In future, we plan to test the effectiveness of training using the varying length, instead of fixed length, on the model performance.

Chapter 4
Generalize Between The Same Domain: A Comparison Approach

4.1 Introduction

Above chapters have explored multiple ML methods to tackle the data scarcity issue via data augmentation approach in the content of knowledge tracing, this chapter will switch the focal point to the knowledge generalization challenge mentioned in Chapter 1, specifically Scenario (a) (i.e., model generalization) in the same content of knowledge tracing. As we already know, a student's knowledge state could be determined by educational information such as skills (learning standards that a student has to master in one particular subject), item difficulty, cognitive information such as student ability, slip/guess factor, etc. Existing KT models such as BKT [37] is built based on the cognitive factors (i.e., slip/guess factors) and deep learning models such as DKT [7], DKVMN [9], NPA [8], SAKT [10] tried to learn educational information (e.g., skills, item difficulty, number of questions taken) as hyperparameters in their neural networks to increase model performance. However, an unseen data set might have different education information, which leads to mismatched model parameters. Hence, the model would fail to generalize. In the case of mismatched parameters, because many KT deep learning models learn exclusively for the specific data set, a covariance shift [38] often happens in the new data, where the marginal probability distributions of the covariates in the two

data sets are different but the conditional distributions are the same [33]. One generalizability study [39] found a SOTA model with 92% high accuracy could only generate 14% on a new data set. Thus, to have good model performance on the new data set, we need to retrain the model which leads to the repeated time and effort and could be very expensive in the real world production setting. To solve such problem, we first choose generic models that take educational information as no hyperparameters such as RNN-based structure (e.g., LSTM, AdaRNN [33]) and Transformer (adapted from the original version [34]) [1] for time series data prediction. Second, we adopt the inductive transfer learning (see in Chapter 1) to generalize knowledge within the same domain from the source model to the target data to save retraining time and effort. To this end, we employ two transfer learning methods: (i) a feature-based approach where we freeze n layers of a pre-trained source model and continuously train/fine-tune on the target task assuming the layers would retain common features between the source and target data; (ii) an instance-based approach using Maximum Mean Discrepancy (MMD) function to reduce the distance between the source and target data assuming there exists a subset of data that have similar distribution in both source and target data. Moreover, we are interested to investigate the efficacy of transfer learning if we increase the size of the source to train the source model. To that end, we augment data sets using the methodology introduced in Chapter 3. To compare the ability to transfer between domains and tasks, namely *transferability*, we set up further experiments to examine the saliency of transferibility between different models and different methods as a note for future source model and method selection for transfer learning. With that, we aim to answer the following research questions (RQs):

- RQ1: Can transfer learning effectively improve model generalization on multivariate KT data?
- RQ2: Does the size of source data matter if we want to boost transferability?
- RQ3: How does each model structure's transferability differ?
- RQ4: Do we see salient difference between the two transfer learning methods?

[1] https://github.com/maxjcohen/transformer

By answering the RQs, this chapter makes the following contributions:

- Solving the poor generalization issue, we conduct transfer learning to generalize knowledge from source model to predict target task via two methods
- We discover the effect of transfer learning is increased by training on the augmented source data
- We demonstrate that the transferability of Transformer model is more salient than AdaRNN and LSTM
- We showcase that freezing-layer approach is more effective and less expensive to generalize knowledge than the method where we train via a MMD loss function

4.2 Related Work

Transfer learning was first introduced to recognize and apply knowledge and skills learned in previous tasks to new tasks. Later researchers found the transferability of knowledge can be used to reduce the need and effort to recollect and retrain models and outperforms models without transfer learning mechanism by a large margin [4]. In transfer learning, the knowledge is obtained from a source training task in a source domain and is applied to a target task in the target domain. To achieve the effective knowledge transfer in various settings, techniques although vary mainly fall under four categories: (i) instance-transfer [40–46], (ii) feature-representation-transfer [47–52], (iii) parameter-transfer [53–55], (iv) relational-knowledge-transfer [56, 57] (see details in Chapter 1). Instance-based approach which this work adopts estimates the weights corresponding to each instance in the source domain and is typically proportional to the distance between source and target density distributions [58]. MMD as one of the distance measures was first proposed by Pan et al. with the assumption that there exists a subset of instances that have similar distributions in the source and target domains. Another method this work uses is feature-transfer. This method attempts to extract relevant features from the source data and apply that to the target data. The transferred knowledge is encoded into the shared features and can be transferred across tasks [4]. To discover such relevant features, our work attempts to freeze the

different layers of the pre-trained source model to retain the features from source data.

As far as transfer learning model framework is concerned, many of them originated from image processing or NLP field. While it is noticeable that the value of knowledge transfer is huge in those areas, we should not ignore its value in time series data where transfer learning can be used to extract useful information from the past time lags for future prediction. AdaRNN is such a model that is created to tackle the covariance shift between different periods of time and apply the useful learnings from the past segments to forecast for the future segments. The RNN-based model has a pre-training mechanism which features a temporal distribution matching to adaptively match the distributions between the RNN cells of two periods while capturing the temporal dependencies [33]. It has experimented on time series data such as human activity recognition, air quality prediction, household power consumption and financial analysis data, achieving 2.6% increase in classification tasks and 9.0% in regression tasks when comparing to SOTA methods. We think this would be a good model to adopt for our purpose. We also include Transformer model because it has been proved to have superior transfer learning ability in the NLP field. However, we would like to validate its transfer learning efficacy on the time series data. Moreover, LSTM as a plain RNN structure model is included as a baseline to adaRNN and transformer model.

4.3 Experiment

4.3.1 Data sets

To examine the multi-variate covariance shift between data sets, we use two K12.com multi-variate data sets: (i) Grade 10 geometry course quiz answering data set (noted as Geom data) with 98.71% of students who have missing data; (2) Grade 11 algebra II quiz answering data set (noted as Alg2 data) with 97.49% of students who have missing data. Each data set contains 8 temporal features (i.e., question type, sequence number, assessment duration, total attempts, attempts per question, question difficulty, item difficulty, standard difficulty) from Jan to June 2019 (see in Table 4.1). Because this work also compares the transfer learning effect change on a larger source data, we fill up the missing values by augmenting the

Table 4.1: Data Statistics for Geom and Alg2 Data For Non-augmented (noted as 'non-aug') and Augmented (noted as 'Aug') Data

Splits	Geometry (# of rows)		Algebra II (# of rows)	
	Non-aug	Aug	Non-aug	Aug
Train (0.7)	288,784	619,173	194,286	405,224
Validate (0.1)	41,255	88,453	27,755	57,889
Test (0.2)	82,510	176,907	55,510	115,778
Total	412,548	884,533	277,551	578,891

two data sets using the generative technology introduced in Chapter 3 with target variables predicted by the correspondent LSTM, adaRNN and Transformer models (see details in Chapter 3). Note that we do not take the augmented data generated by LVAE from Chapter 3 because we want to make our case reproducible without the need to apply a special algorithm or framework. After augmentation, the data size almost doubled (see in Table 4.1). These two particular data sets are chosen because they are from the same platform (i.e., K12.com), ensuring that the transfer learning is performed within the same domain and meaningful. To perform general transfer learning, we do not typically require the source and target need to be from the same platform/system. However, in the education domain, if the teaching platforms are different, the knowledge learned from one platform will not be applicable to the other. For example, many public data sets in KT field are from different platforms such as ASSISTments data (e.g., mainly K-12 data), Junyi (e.g., college students in Taiwan), STATICS (e.g., college students that learn statistics at Carnegie Mellon University). Thus, the learning we draw from STATICS data set cannot be easily applied to the K-12 learning in ASSISTments platform because many of the K-12 students do not even have statistics lessons or it is too long of learning gap between the college and grades under 12.

4.3.2 Training Setup

Training Methods: During training, we split the data into training (Ta)/validation (V)/test (Te) sets with a ratio of [0.7,0.1,0.2] for the source data s and target data t. To compare the generalization between the training with and without the transfer learning, we carefully design five training methods to predict the same target testing set Te_t: (1) baseline training, where we train on the source training data

Table 4.2: RMSE Comparison Between ML and DL Methods For Non-augmented Data Sets. The ML performances are best estimator results after randomized search CV (5). The DL performance is the average by the same hyperparameter tuning across 5 random seeds. Combine: train/val/test are a mix of source and target; Boldface represents the best performance.

Model Name	Geom → Alg2		Alg2 → Geom	
	Baseline	Combine	Baseline	Combine
Random Forest	0.7136	0.5851	0.5637	0.5893
AdaBoost	0.7012	0.6193	0.5899	0.6028
XGBoost	0.7070	0.6929	0.6926	0.6936
LSTM	0.4828	0.4824	0.4705	0.4700
AdaRNN	0.4854	0.4996	0.4722	0.4791
Transformer	**0.4065**	**0.3707**	**0.3809**	**0.3641**

Ta_s and validate on source validation set V_s to obtain a source model M and then test on target test set Te_t; (2) combined training, where we combine the training data of both source and target data (Ta_{s+t}) to train and combine the both validation sets (V_{s+t}) to validate and test on Te_t; (3) further-training, where we continuously train the model M but with the target train data Ta_t, validate on V_t and test on Te_t; (4) further-train with frozen layers, where we freeze layers of M and further train on Ta_t, validate on V_t and test on Te_t. To ensure the deep learning (DL) models we select have superior performance than traditional machine learning (ML) models, we conduct training on three traditional ML models whose performances can be comparable to DL models in various situations: Random Forest, AdaBoost and XGBoost (see details in Table 4.2). Because traditional ML are not convenient to do further-training, we only compared training method (1) and (2) between ML and DL models. We observe that the selected DL models out-perform the ML models (i.e., Random Forest, AdaBoost, XGBoost) to a great extent with Transformer has the best performing RMSE. Therefore, we opt to conduct further experiments on the DL models.

Given LSTM is a 2 layer model, we only freeze the first layer-feature layer. We freeze the first two layers (feature layers) out of AdaRNN's total four layers. Transformer has 4 layers with encoder, decoder as feature layers and 2 linear layers. We also freeze two feature layers for it; (5) transfer learning via MMD, where we train a MMD loss between Ta_s^l and Ta_t^l, l is the smaller length of the source and

Table 4.3: Data Assignment By Training Methods For Non-augmented Data. $\mathcal{G}$: Geom data; $\mathcal{A}$: Alg2 data; l: the smaller data length of the $\mathcal{G}$ and $\mathcal{A}$. FT: further-train; TR: transfer learning.

Train Method	$Geom\ (\mathcal{G}) \rightarrow Alg2\ (\mathcal{A})$			$Alg2\ (\mathcal{A}) \rightarrow Geom\ (\mathcal{G})$		
	Ta	V	Te	Ta	V	Te
Baseline	$\mathcal{G}$	$\mathcal{G}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{G}$
Combine	$\mathcal{G}+\mathcal{A}$	$\mathcal{G}+\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}+\mathcal{G}$	$\mathcal{A}+\mathcal{G}$	$\mathcal{G}$
Further-train (FT)	$\mathcal{G}+\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}+\mathcal{G}$	$\mathcal{G}$	$\mathcal{G}$
FT+Freeze Layers	$\mathcal{G}+\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}+\mathcal{G}$	$\mathcal{G}$	$\mathcal{G}$
TR via MMD	$\mathcal{G}^l+\mathcal{A}^l$	$\mathcal{G}+\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}^l+\mathcal{G}^l$	$\mathcal{A}^l+\mathcal{G}^l$	$\mathcal{G}$

target data, then we validate on V_{s+t} and test on Te_t (see details in Table 4.3). Method (1) is a classical generalization approach where you train a source model using source data and test on the target data. Method (2) can be considered retraining method where we merge the majority of target data (Ta_t) back to the source data and validate on both of the source and target data to finally test on the target unseen data Te_t. In practice, Method (2) is resource-intensive as we have to completely train the model from fresh, which requires increased computation power and time due to the increased data set size. This is also the method we should try to avoid in the practical setting. Method (3) is basically a continuously train method, where we continuously train the pre-trained model using the existing checkpoints for more epochs but on Ta_t, validate on V_t and test on Te_t. This method requires relatively less resource because the computation power and time will only be spent on the target data. Method (4) is a feature-based transfer learning where we extract the good features that can be shared between the source and target data from the source model and apply that to the target data. Time and computation power is the same as Method (3). For Method (5), we adopt the instance-based transfer learning where a MMD transfer loss is implemented between the source and target data for l rows and validate on both the source and target validation set but still test on Te_t. It is different from Method (3) where the complete data set of source and target will be used for training and there is no MMD loss included in (3).

Hyperparameter-tuning: The data is split by row number and organized into a fixed length in the pre-processing step (see in the non-subject-based process in

Table 4.4: Model Parameters by Model. We apply the same for the source models and transfer learning models. '-' indicates the model network does not use such parameter. 'Q/V/H/N' is specific to Transformer models and stand for the query (Q), value (V), number of heads (H) and number of encoder and decoder layers to stack (N).

Model Name	#In	#Out	# Layers	#Hidden	Q	V	H	N	#Attention
LSTM	8	1	2	128	-	-	-	-	-
adaRNN	8	1	2	64	-	-	-	-	-
Transformer	8	1	-	-	8	8	8	2	4

Chapter 3 Figure 3.3). We think the row-number splitting is sufficient because we do not need to do imputation which requires alignment with students to not interfere the sequence forming, as demonstrated in Chapter 3. As far as hyperparameter tuning for the training, we keep the training parameters unchanged for all the models with learning rate of 2e-4, 5 epochs, weight decay of 5e-4 at batch size of 32. Depending on different model architectures, we tune the model parameters differently. Table 4.4 lists all the customized hyperparameters we use. For the two RNN-based models which use hidden dimensions, adaRNN uses 64 hidden dimensions for each layer following [33] and LSTM uses 128 hidden dimensions for 2 layers following the LSTM hyperparameter specification in [7] and [8]. For Transformer model, we set up hyperparameter values for Q/V/H/N following the guidance from both original paper [34] and the adapted transformer for time series data [2]. In the scenario of comparing feature-based and instance-based method in Section 4.4.4, we adjust the hyperparameter tuning for Q/V/H/ to be 16 and attention size to be 8 with N=4.

Loss Function: To compare the generalization among different models, we use Root Mean Squared Error (RMSE) to evaluate the model performance which aligns with the typical KT model evaluation metric and also more robust compared to the L1 error. Hence, we use MSE (Mean Squared Error) error as loss function across models. During the training for transfer learning via MMD, we apply the loss function $\mathcal{L} = L_s + L_t + w * L_{tr}$, where L_s and L_t are the source and target RMSE error and L_{tr} is the MMD loss. MMD is a distance measure function to find the discrepancy between the source and target domain and can be represented as $dist(X\prime_{src}, X\prime_{tar}) = \left\| \frac{1}{n1} \sum_{i=1}^{n1} \phi(x_{src_i}) - \frac{1}{n2} \sum_{i=1}^{n2} \phi(x_{tar_i}) \right\|_{\mathcal{H}}$ where ϕ is a mapping

[2]https://github.com/maxjcohen/transformer

function. We use Adam optimizer to optimize the total loss $\mathcal{L}$ during training.

4.4 Results

4.4.1 Transfer Learning Improves Generalization

Table 4.5 presents all the model performances in RMSE error on Te_t by different training methods and model architectures. Δ is a relative change in % of RMSE between the baseline and other training methods. The lower the Δ, the better. More importantly, the negative Δ shows the drop in RMSE, indicating the improvement of generalization on Te_t. From the table, we can see the lowest Δ are located in the bottom two rows of the table, which represent the model performances of the two transfer learning methods (i.e., FT+Freeze Layers and TR via MMD) for both generalization directions. In addition, majority of them are negative, especially for 'FT+Freeze Layers' method, from -6.22% to -0.09% for each model. These negative Δ are also significant after tested for two sample t-test except for the Δ on the LSTM model when generalizing from Geom to Alg2 (insig.). This indicates that transfer learning improve model generalization is not due to chance. Therefore, we can answer RQ1 that transfer learning can effectively improve model generalization in general. Note that we exclude Δ of 'Combine' training method from comparison because it is a quite costy method and should be avoided in practice. Even when we compare to the 'Combine' method, transfer learning results are still superior across models with the only exception on the Transformer model Δ of 'Combine' method for Geom $\rightarrow$ Alg2 with -8.22% (Combine) vs. -6.2% (FT+Freeze Layer).

4.4.2 Augmented Source Data Increases Transfer Learning Effect

The above demonstrates that transfer learning is effective on generalizing knowledge between the similar size data sets (289k vs. 194k rows). To answer RQ2, we further conduct experiments to validate whether augmenting source data could increase the transfer learning effect. Therefore, we augment the Geom and Alg2 data via VAE model and predict the correspondent target variable using LSTM, AdaRNN and Transformer models which are introduced in Chapter 3. To ensure

Table 4.5: RMSE Comparison Among Models With and Without Transfer Learning For Non-augmented Data Sets. The performances are the average by the same hyperparameter tuning across 5 random seeds. Combine: train/val/test are a mix of source and target; TR: transfer learning; boldface indicates the biggest reduction in RMSE excluding 'combine' method; * indicates two-sample T-test significance.

Train Method	Geom → Alg2			Alg2 → Geom		
	LSTM	AdaRNN	Transformer	LSTM	AdaRNN	Transformer
Baseline	0.4828	0.4854	0.4065	0.4705	0.4722	0.3809
Combine	0.4824	0.4996	0.3707	0.4700	0.4791	0.3641
Combine Δ	-0.08%	+2.92%	-8.81%*	-0.11%*	+1.46%	-4.39%*
Further-train (FT)	0.4827	0.4824	0.4541	0.4700	0.4721	0.3664
FT Δ	-0.01%	-0.06%*	+11.71%	-0.11%*	-0.02%	-3.78%*
FT+Freeze Layers	0.4823	0.4818	0.3812	0.4701	0.4708	0.3599
FT+Freeze Layers Δ	**-0.09%**	**-0.75%***	**-6.22%***	-0.09%*	**-0.29%***	**-5.49%***
TR via MMD	0.4825	0.4962	0.3894	0.4699	0.4929	0.3834
TR via MMD Δ	-0.06%	+2.22%	-4.23%*	**-0.14%***	+4.38%	+0.67%

the generalization is from the same target data and make performances comparable, we only replace the source data with augmented data and keep the target data as non-augmented. Table 4.6 showcases the data assignment for train/val/test after augmenting the source data by each training method. Table 4.7 presents the Δ comparison by each model and training method. We also include the Δ obtained from the non-augmented data in Table 4.5 and mark it as †. By doing so, we can test the hypothesis that whether or not the augmented data has different transfer learning effect. We observe from the table that the lowest Δs (marked with boldface) still remain in the bottom two rows which are the model performances for transfer learning training methods. Except for the LSTM model when generalizing from Geom to Alg2 (i.e., -0.06% for FT+Freeze Layers and -0.02% for TR via MMD), the model performances from both the transfer learning methods generalize better than the baseline model with significance. Moreover, these significant drops have bigger magnitude than the ones from non-augmented data except for the 'TR via MMD' performance on LSTM model when generalizing from $Alg2_a$ to Geom (i.e., -0.06% vs. -0.14%). For example, for AdaRNN transferring from Geom to Alg2, the average RMSE error when generalizing from augmented data to target data drops by 2.54% comparing to 0.75% drop when training on non-augmented data. The superiority is seen in the transfer direction of $Alg2_a$ → Geom as well. Therefore, to answer RQ2, we observe that the source model trained with the augmented data gain better transfer learning effect except for LSTM model which has

Table 4.6: Data Assignment By Training Methods for Augmented Data. $\mathcal{G}$: Geom data; $\mathcal{A}$: Alg2 data; l: the smaller data length of the $\mathcal{G}$ and $\mathcal{A}$, a indicates augmented data

Train Method	$Geom_a$ ($\mathcal{G}$) $\rightarrow$ $Alg2$ ($\mathcal{A}$)			$Alg2_a$ ($\mathcal{A}_a$) $\rightarrow$ $Geom$ ($\mathcal{G}$)		
	Ta	V	Te	Ta	V	Te
Baseline	$\mathcal{G}_a$	$\mathcal{G}_a$	$\mathcal{A}$	$\mathcal{A}_a$	$\mathcal{A}_a$	$\mathcal{G}$
Combine	$\mathcal{G}_a + \mathcal{A}$	$\mathcal{G}_a + \mathcal{A}$	$\mathcal{A}$	$\mathcal{A}_a + \mathcal{G}$	$\mathcal{A}_a + \mathcal{G}$	$\mathcal{G}$
Further-train (FT)	$\mathcal{G}_a + \mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}_a + \mathcal{G}$	$\mathcal{G}$	$\mathcal{G}$
FT+Freeze Layers	$\mathcal{G}_a + \mathcal{A}$	$\mathcal{A}$	$\mathcal{A}$	$\mathcal{A}_a + \mathcal{G}$	$\mathcal{G}$	$\mathcal{G}$
TR via MMD	$\mathcal{G}_a^l + \mathcal{A}^l$	$\mathcal{G}_a + \mathcal{A}$	$\mathcal{A}$	$\mathcal{A}_a^l + \mathcal{G}^l$	$\mathcal{A}_a^l + \mathcal{G}^l$	$\mathcal{G}$

Table 4.7: Generalization Performance Comparison Among Models With and Without Transfer Learning For Augmented Data Sets. The performances are averaged across 5 random seeds using the same training parameter tuning. Combine: train/-val/test are a mix of source and target; TR: transfer learning; † is the Δ from Table 4.5; boldface is the lowest Δ for each model excluding 'combine' method and † Δ.

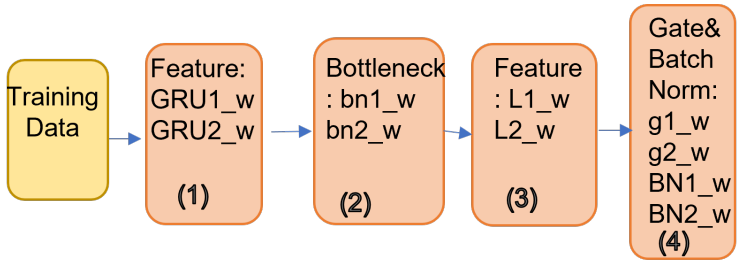
Train Method	$Geom_a \rightarrow$ Alg2			$Alg2_a \rightarrow$ Geom		
	LSTM	AdaRNN	Transformer	LSTM	AdaRNN	Transformer
Base	0.4827	0.4820	0.3902	0.4700	0.4766	0.3822
Combine	0.4824	0.4914	0.3542	0.4696	0.4701	0.3663
Combine†Δ	-0.08%	+2.92%	-8.81%*	-0.11%*	+1.46%	-4.39%*
CombineΔ	-0.06%	+1.96%	-9.22%*	-0.08%*	-1.37%*	-4.17%*
FT	0.4824	0.4823	0.3590	0.47009	0.4915	0.3704
FT†Δ	-0.01%	-0.61%*	11.71%	-0.11%*	-0.02%	-3.78%*
FT Δ	-0.06%	+0.07%	-8.00%*	+0.02%	+3.13%	-3.08%*
FT+Freeze Layers	0.4824	0.4697	0.3517	0.4701	0.4703	0.3634
FT+Freeze Layers†Δ	-0.09%	-0.75%*	-6.22%*	-0.09%*	-0.29%*	-5.49%*
FT+Freeze Layers Δ	**-0.06%**	**-2.54%***	**-9.87%***	+0.01%	**-1.33%***	**-4.90%***
TR via MMD	0.4826	0.4969	0.3857	0.4697	0.4940	0.4081
TR via MMD† Δ	-0.06%	+2.22%	-4.23%*	-0.14%*	+4.38%	+0.67%
TR via MMD Δ	-0.02%	+3.09%*	-1.16%	**-0.06%***	+3.64%	+6.77%

very minimal gains (i.e., $< 0.2\%$). It is worth noticing that the further-trained methods are not that salient for LSTM ($Alg2_a \rightarrow$ Geom) and AdaRNN models (both transfer directions) as they present positive changes (i.e., +0.02%, +0.07% and +3.13%) on the RMSE, indicating LSTM and AdaRNN's poor facilitation on the generalization from the augmented source data. The reason could be because the further-train method is essentially continuous training on the existing checkpoints of the source model. With such operation, no feature extraction are applied and it is expected we do not see valid improvement on the model generalization.

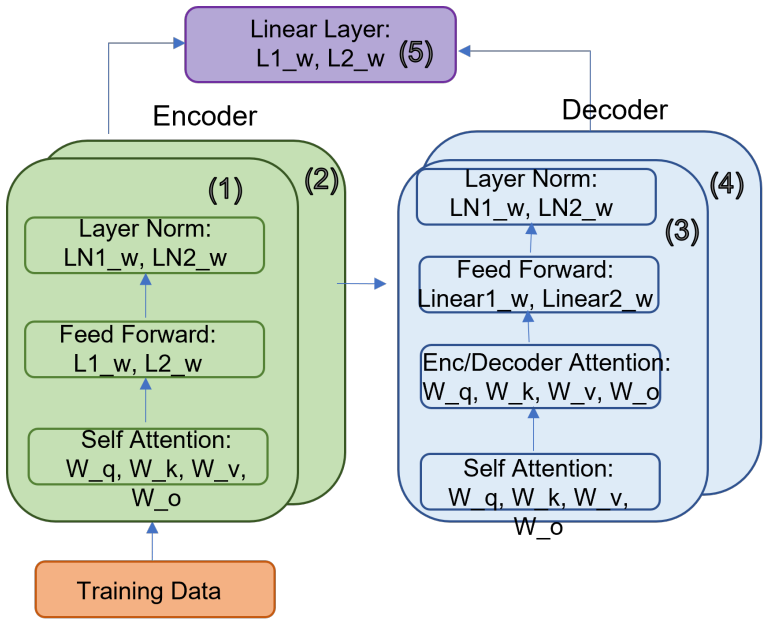
4.4.3 Transferability Saliency Comparison Between Models

From Section 4.4.2, we observe that Transformer models gain the highest improvement with 6.22% for Geom → Alg2 and 5.49% for Alg2 → Geom with significance. LSTM has the worst generalization with under 0.2% improvement whereas AdaRNN has improvement of 0.75% for Geom → Alg2 and 0.29% for Alg2 → Geom for the method 'FT+Freeze Layers'. It seems that the Transformer models from both generalization directions transfer knowledge significantly better than the LSTM, AdaRNN models (see row 4 from the bottom) with -9.87% (Transformer) vs. -2.54% (AdaRNN) vs. -0.06% (LSTM) for $Geom_a \rightarrow$ Alg2 and -4.90% (Transformer) vs. -1.33% (AdaRNN) vs. +0.01% (LSTM) for $Alg2_a \rightarrow$ Geom in the 'FT+Freeze Layers' category. That is about 3-4 times better, emphasizing the superior transferability from the Transformer models. On the other hand, with the least improvement and insignificance, LSTM seems to be the worst model to choose when it comes to transfer knowledge. Therefore, we will leave it out and compare AdaRNN model and Transformer model for their saliency in transfer learning. The AdaRNN structure contains 4 layers including one feature layer, one bottleneck layer, another feature layer and a batch normalization layer whereas the Transformer model is a modified version of the original version from [34], which contains one encoder, one decoder, two linear layers with encoder and decoder to be two-head multi-head attention modules (see in Figure 4.1). We then freeze the two feature extraction layers for AdaRNN (the feature and bottleneck layer) and freeze encoder and decoder (2 head attention feature extractor) layers for Transformer model. Although frozen the same number of layers (2 layers), AdaRNN's generalizability looks much weaker than the Transformer model's.

To find out how the difference arrives between the AdaRNN and Transformer models in terms of transferability, we freeze parameter groups within each model type to examine the inner workings behind the layer-freezing for the difference. In Figure 4.1, we split Transformer model into five weights groups: (1) the first head encoder; (2) the second head encoder; (3) the first head decoder; (4) the second head decoder; (5) a linear layer. (1)-(4) are the feature extractor groups inside the Transformer network. We split AdaRNN into four weights groups: (1) a feature layer with two GRU cells; (2) a bottleneck layer with two batch normalization weights; (3) a feature layer with two linear weights; (4) a gate and batch



(a) AdaRNN layers



(b) Transformer Layers

Figure 4.1: An Illustration on the Major Parts of AdaRNN and Transformer Neural Networks. (1)-(5) are the weights groups in each network.

normalization layer that contains gate weights and batch normalization weights (see in Figure 4.1). The (1)-(2) weights groups are the feature extractor inside the AdaRNN network. We then freeze only the weights and keep the bias trainable for each group to further train the existing source model (i.e., baseline model). Table 4.8 demonstrates the average RMSE acquired after we freeze each weights group for AdaRNN and Transformer models comparing to the results of freezing two layers for each model. From the table, we observe that AdaRNN stops updating (marked as $\rightarrow$) performance after weights group (3) which is right after feature and bottleneck layers whereas Transformer model performances keep changing (marked as $\uparrow$, $\downarrow$) at each weight group even after feature extractor (weights group (4)). We also

Table 4.8: Transfer Learning Freezing Layers vs. Freezing Parameter Comparison Between AdaRNN and Transformer Model For Non-augmented and Augmented Data. The performances are averaged across 5 random seeds. We freeze 1 layer for AdaRNN and 2 layers for Transformer. '†' indicates the result is from Table 4.5 whereas '‡' indicates the result is from Table 4.7. * indicates the two-sample T-test significance with the 'Freeze Layers' performance as base sample.

Geom → Alg2						
Model Name	Freeze Layers	Freeze Parameters				
		1	2	3	4	5
AdaRNN	0.4818†	0.4822	0.4813*↓	0.4813*→	0.4813*→	/
Transformer	0.3812†	0.4010↑	0.3512↓	0.3494↓	0.3535↑	0.3740*↑
AdaRNN$_a$	0.4697‡	0.4816*↑	0.4812*↓	0.4812*→	0.4812*→	/
Transformer$_a$	0.3517‡	0.3511↓	0.3511*→	0.3513↑	0.3519↑	0.3722*↑
Alg2 → Geom						
Model Name	Freeze Layers	Freeze Parameters				
		1	2	3	4	5
AdaRNN	0.4708†	0.4702↓	0.4704↑	0.4704→	0.4704→	/
Transformer	0.35994†	0.3589↓	0.3612↑	0.3591↓	0.3601↑	0.3660*↑
AdaRNN$_a$	0.4703‡	0.4923↑	0.4609*↓	0.4609*→	0.4609*→	/
Transformer$_a$	0.3634‡	0.3607*↓	0.3602*↓	0.3603*↑	0.3603*→	0.3732*↑

observe Transformer models have better initial performances than AdaRNN after freezing the first weight group, e.g., 0.4010 (Transformer) vs. 0.4822 (AdaRNN) for Geom → Alg2. This could be because that AdaRNN was designed for training MMD loss and its special module 'Temporal Distribution Matching' only works for time series without the longitudinal nature such as weather and utility time series data. Meanwhile, the superiority of Transformer models could be due to the excellent feature extraction from its encoder and decoder structures with multiple heads (also known as multi-head attention) that potentially learn different patterns by each head. Thus to answer RQ3, we conclude that Transformer has better transferability comparing to the other two model structures (i.e., AdaRNN and Transformer) due to its excellent feature extraction from the multi-head attention mechanism.

4.4.4 Transferability Saliency Comparison Between Methods

From Section 4.4.2, we discover that 'FT+Freeze Layers' seems to always perform better than 'TR via MMD' for the same training parameter tuning. In addition, all

the significant drop in RMSE are obtained from the 'FT+Freeze Layers' method except for one (i.e., LSTM Δ of -0.14% for Alg2 $\rightarrow$ Geom), indicating that the feature-based method via freezing layers may have superior performance over the instance based transfer learning method via MMD. The reason could be that freezing layers is able to lock down the common features between source and target data and gain increased performance when further training whereas the instance-based method relies on training the MMD loss function to minimize the distance between the source and target data, which could require much time and computational resource to reach its optimal performance. Note that we only train the method with 5 epochs for all the models, which might not be enough to reach the best performance for MMD method. To validate this, we increase training epochs and further tune hyperparamters for the 'TR via MMD' method. We observe that the RMSE is able to go down when the epochs are increased to 10 and the hidden dimensions are doubled to 256 except for Geom $\rightarrow$ Alg2 on the augmented data (see Δ in Table 4.9). This infers that the training cost of 'TR via MMD' method is much higher than 'FT+Freeze Layers', indicating 'FT+Freeze Layers' is computationally frugal and effective with less adjustment on hyperparameter tuning.

In addition, Section 4.4.1 and 4.4.2 demonstrate that the performances from 'TR via MMD' method are not significantly different from the baseline method whereas the performances from 'FT+Freeze Layers' are. Thus, we can conclude the answer to RQ4 that the feature-based transfer learning method featuring 'FT+Freeze Layers' has better transferability than the instance-based method featuring 'TR via MMD' method.

4.5 Limitations

Although this chapter offers an interesting angle to compare the transferability between different models and different methods, the experiment design are rather exploratory than empirically strict. Our claims are only induced from two multivariate data sets and limited to two models for Transformer model to compare. As future work, the transferability experiments should be extended to multiple other data sets and increasing the number of models to compare with for Transformer model is also desirable to solidify the results.

Table 4.9: Feature-based vs. Instance-based Transferability Comparison For Transformer Model. The performances are averaged across 5 random seeds. '†' indicates the result is from Table 4.5 whereas '‡' indicates the result is from Table 4.7. Δ is the relative % change in RMSE from 'TR via MMD' method in the same row.

Geom →Alg2						
Data Type	FT+FL	TR via MMD	Continue-Train on TR via MMD			
			Hid_Dim=256	Δ	Ep=10	Δ
Non-aug	$0.3812^{\dagger}$	$0.3894^{\dagger}$	0.4312	+10.74%	0.3760	-3.43%
Aug	$0.3517^{\ddagger}$	$0.3857^{\ddagger}$	0.4064	+4.92%	0.4004	+3.82%
Alg2→Geom						
Non-aug	$0.3599^{\dagger}$	$0.3834^{\dagger}$	0.3799	-0.91%	0.3773	-1.59%
Aug	$0.3634^{\ddagger}$	$0.4081^{\ddagger}$	0.3939	-3.47%	0.4045	-0.86%

4.6 Summary

To reduce the model generalization error, this chapter proposed transfer learning approach via feature-based (i.e., freeze layers) and instance-based methods (i.e., MMD training). We tested the transfer learning effectiveness on both non-augmented and augmented data as source data and demonstrated that transfer learning can successfully improve generalization by a margin of 5-10%. Furthermore, we showcase that knowledge can be more effectively transferred from augmented source data to small size data than between the similar size data sets. Moreover, we compared the transferability between three models and between two methods. We observe that Transformer is the most salient at transferring knowledge as a model structure and feature-based approach by freezing layers is empirically more effective and computationally frugal in generalizing knowledge from source to target task.

Chapter 5
Generalize Between Different Domains: A TAPT Approach

5.1 Introduction

In the math education community, teachers, Intelligent Tutoring Systems (ITSs) and Learning Management Systems (LMSs) have long focused on bringing learners to the target mastery over a set of skills, also known as **Knowledge Components (KCs)**. Common Core State Standards (CCSS)[1] is one of the most common categorizations of knowledge components skills in mathematics from kindergarten to high school in the United States with a full set of 385 KCs. For example, in the CCSS code *7.NS.A.1*, *7* stands for 7-th grade, *NS* stands for the topic *Number system*, *A.1* stands for the lesson number [59]. In the process of using KCs, the aforementioned stakeholders often encounter the challenges in three scenarios: (1) teachers know how to describe the areas where a student is unable to master but don't know what exact skill code that is (S_1), (2) ITSs need to tag instructional videos with KCs for better content management (S_2), and (3) LMSs need to know what KCs a problem is associated with in recommending instructional videos to aid problem solving (S_3).

The solutions to these scenarios typically framed the problem as the *multinominal classification*–i.e., given the input text, predicts one most relevant KC label out of many KCs: $I(nput) \mapsto text$ and $O(utput) \mapsto KC$. Prior research solutions included SVM-based [60], Non-negative Matrix Factorization (NMF) [61], Skip-gram

[1] www.corestandards.org

Table 5.1: Examples of three data types, all having the KC label “8.EE.A.1"

Data Type	Text
Description Text	Know and apply the properties of integer exponents to generate equivalent numerical expressions
Video Title	Apply properties of integer exponents to generate equivalent numerical expressions
Problem Text	Simplify the expression: (z2)2 *Put parentheses around the power if next to coefficient, for example: 3x2=3(x^2),x5=x^5

Representation [62], Neural Network [39] or even cognitively-based knowledge representation [63]. Existing solutions, however, used relatively small number of labels (e.g., 39 or 198) from CCSS with the input of problem text only (similar to Table 5.1-Row 3) [39, 60, 62]. In addition, they often rely on customized feature engineering that varies from data set to data set and needs to to rebuilt whenever a new testing data set distribution deviates from the original training data set.

Such finding gives rise to the need of a knowledge transfer model. It will conduct transductive transfer learning, which generalize the knowledge between different domains-i.e., from other domains to the education domain for text prediction task. Based on the huge success in the NLP field with high-performing pre-trained language models, we adopt the pretrained language model approach to carry out the transfer learning in the above setting. Toward such goal, in this chapter, we propose the following research questions:

RQ1: Can we build a transfer learning model that leverages recent success of pre-trained models in NLP field and prove its superior performance to the prior state-of-the-art (SOTA) models and even the original general language model (denoted as BASE BERT [64])?

RQ2: How does such model generalize across different skill-related text data? Would augmenting the training data help to improve the generalization further?

RQ3: How to better evaluate the miss-predictions which education experts deemed not incorrect from such model?

By answering the above research questions, this chapter makes the following contributions:

- Train a Task-adaptive Pretraining (TAPT) model from the BASE BERT on three types of skill-related text data and compare its performance to the prior models and BASE BERT. Our experiments demonstrate that the TAPT model outperforms all the prior SOTA methods and obtains 0.5-2.3% improvement on the BASE BERT.
- Improve the TAPT model's generalizability with a margin of 1-3% via augmenting the training data.
- Propose a new evaluation measure, *TEXSTR*, that enables 56-69% more KC labels to be correctly predicted than using the classical measure of *accuracy*.

5.2 Related Work

KC Models. Rose et al. [63] is one of the earliest work predicting knowledge components, which took a cognitively-based knowledge representation approach. The scale of KCs it examined was small with only 39 KCs. Later research extended the scale of KCs using a variety of techniques. For example, Desmariais [61] used non-negative matrix factorization to induce Q-matrix [65] from simulated data and obtained an accuracy of 75%. The approach did not hold when applying to real data and only got an accuracy of 35%. The two aforementioned studies shared the same drawback: not using the texts from the problems. Karlovcec et al. [60] used problem text data from the ASSISTments platform [1] and created a 106-KC model using 5-fold cross validation via ML approach SVM, achieving top 1 accuracy of 62.1% and top 5 accuracy of 84.2%. Pardos et al. [62] predicted for 198 labels and achieved 90% accuracy via Skip-gram word embeddings of problem id per user (no problem text used). However, Patikorn et al. [39] did a generalizability study of Pardos et al. [62]'s work and only achieved 13.67% accuracy on a new data set. They found that was because Pardos et al. [62]'s model was over-fitting due to memorizing the question templates and HTML formatting as opposed to encoding the real features of the data. Hence, Patikorn et al. [39] removed all the templates and HTML formatting and proposed a new model using Multi-Layer-Perceptron algorithm, which achieved 63.80% testing accuracy and 22.47% on a new data set. The model of Patikon et al. [39] became the highest performance for the type of problem text. The preceding research is only focused on problem related content

(ID or texts) whereas our work uses not only the problem text but also the KC descriptions and video title data covering a broad range of data.

Pre-Trained BERT Models. The state-of-the-art language model BERT (Bidirectional Encoder Representations From Transformer) [64] is a pre-trained language representation model that was trained on 16 GB of unlabeled texts including Books Corpus and Wikipedia with a total of 3.3 billion words and a vocabulary size of 30,522. Its advantage over other pre-trained language models such as ELMo [66] and ULMFiT [67] is its bidirectional structure by using the *masked language model* (MLM) pre-training objective. The MLM randomly masks 15% of the tokens from the input to predict the original vocabulary id of the masked word based on its context from both directions [64]. The pre-trained model then can be used to train from new data for tasks such as text classification, next sentence prediction.

Users can also further pre-train BERT model with their own data and then fine-tune. This combining process has become popular in the past two years as it can usually achieve better results than fine-tuning only strategy. Sun et al. [68] proposed a detailed process on how to further pre-train new texts and fine-tune for classification task, achieving a new record accuracy. Models such as FinBERT [69], ClinicalBERT [70], BioBERT [71], SCIBERT [72], and E-BERT [73] that were further pre-trained on huge domain corpora (e.g., billions of news articles, clinical texts or PMC Full-text and abstracts) were referred as *Domain-adaptive Pre-trained* (DAPT) BERT and models further pre-trained on task-specific data are referred as *Task-adaptive Pre-trained* (TAPT) BERT by Gururangan et al. [74] such as MelBERT [75] (Methaphor Detection BERT). Although DAPT models usually achieve better performance (1-8% higher), TAPT models could also demonstrate competitive or even higher performance (2% higher) according to Gururangan et al. [74]. In Liu et al. [69], FinBERT-task was 0.04% higher than domain FinBERT in accuracy. In addition, TAPT models require less time and resource to train. In light of this finding, we use the task-specific data to further pre-train the BERT model.

5.3 The Proposed Approach

To improve upon existing solutions to the problem of auto-labeling educational content, we propose to exploit recent advancements by BERT language models.

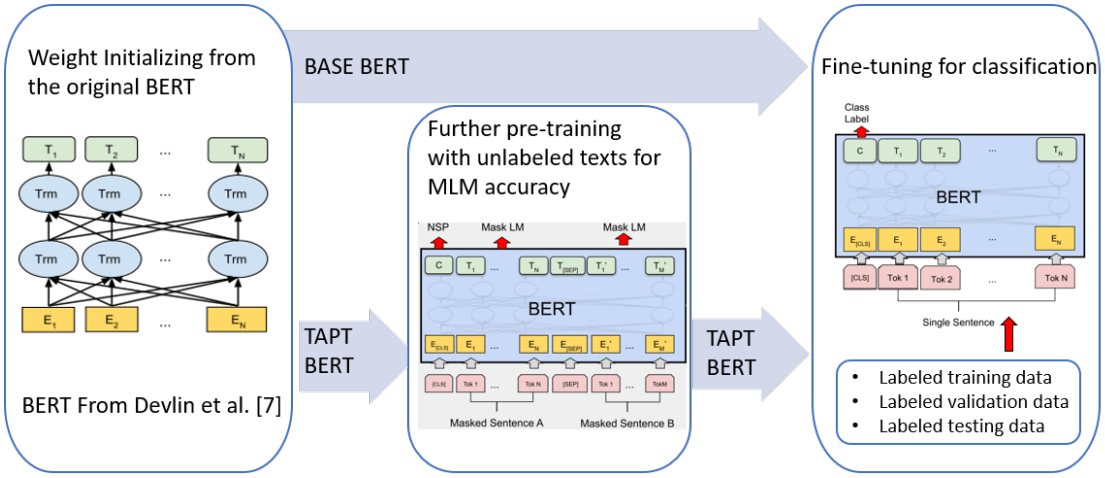


Figure 5.1: An illustration of training and fine-tuning process of BASE vs. TAPT BERT

Since BERT can encode both linguistic structures and semantic contexts in texts well, we hypothesize its effectiveness in solving the KC labeling problem. By effectively labeling the KCs, we expect to solve the challenges incurred from three scenarios in Section 5.1.

5.3.1 Task-Adpative Pre-Trained (TAPT) BERT

In particular, we propose to adopt the Task-adaptive Pre-trained (TAPT) BERT and fine-tune it for three types of data. The “pre-training" process is unsupervised such that unlabeled task-specific texts get trained for MLM objective whereas the “fine-tuning" process is supervised such that labeled task-specific texts get trained for classification (see Fig. 5.1). We call a BERT model that only has a fine-tuning process as BASE. For the TAPT model, we first initialize the weights from the original BERT (i.e., BERT-base-uncased model). Then, we further pre-train the weights using the unlabeled task-specific texts as well as the combined task texts (see details in Section 5.4.1) for MLM objective, a process of randomly masking off 15% of the tokens and predict their original vocabulary IDs. The pre-training performance is measured by the accuracy of MLM. Once the TAPT model is trained, we fine-tune the TAPT model with the task-specific labeled texts by splitting them into training, validation and testing data sets and feed them into the last softmax layer for classification. We measure the performance of fine-tuning via

the testing data accuracy. For BASE, we do not further train it after initializing the weights but directly fine-tune it with the task-specific data for classification (see Fig. 5.1). To show the effectiveness of the TAPT BERT approach, we compare it against six baselines including BASE BERT for three tasks:

- T_d: to predict K-12 KCs using data set D_d (description text) based on S_1
- T_t: to predict K-12 KCs using data set D_t (video title text) based on S_2
- T_p: to predict K-12 KCs using data set D_p (problem text) based on S_3

5.3.2 Evaluating KC Labeling Problem Better: *TEXSTR*

In the regular setting of multinomial classification to predict KC labels, the evaluation is done as binary–i.e., exact-match or non-match. For instance, if a method predicts a KC label to be *7.G.B.6*, but its ground truth is *7.G.A.5*, *7.G.B.6* is considered to be a non-match. However, the incorrectly predicted label of *7.G.B.6* could be closely related to *7.G.A.5* and thus still be useful to teachers or content organizers. For example, in Fig. 5.2, the input to the classification problem is a video title "Sal explains how to find the volume of a rectangular prism fish tank that has fractional side lengths." Its ground truth label is *7.G.B.6* (7-th grade geometry KC), described as "Solve real world problem involving ... volume ... composed of ... prisms." When one looks at three non-match labels, however, their descriptions do not seem to be so different (see in Fig. 5.2). That is, all of the three non-match labels (*6.G.A.2*, *5.MD.C.5*, and *5.MD.C.3*) mention "volume solving" through "fine/relate/recognize with operations and concepts," which is quite similar to the KC description of the ground truth. However, due to the nature of exact-match based evaluation, these three labels are considered wrong predictions. Further, domain experts explain that some skills are prerequisites to other skills, or that some problems have more than one applicable skills (thus multiple labels) and they could all be correct.

Therefore, we argue that using a strict exact-matching based method in evaluating the quality of the predicted KC labels might be insufficient in practical settings. We then propose a method that considers both semantic and structural similarities among KC labels and their descriptions to be an additional measure to evaluate the usability of the predicted labels.

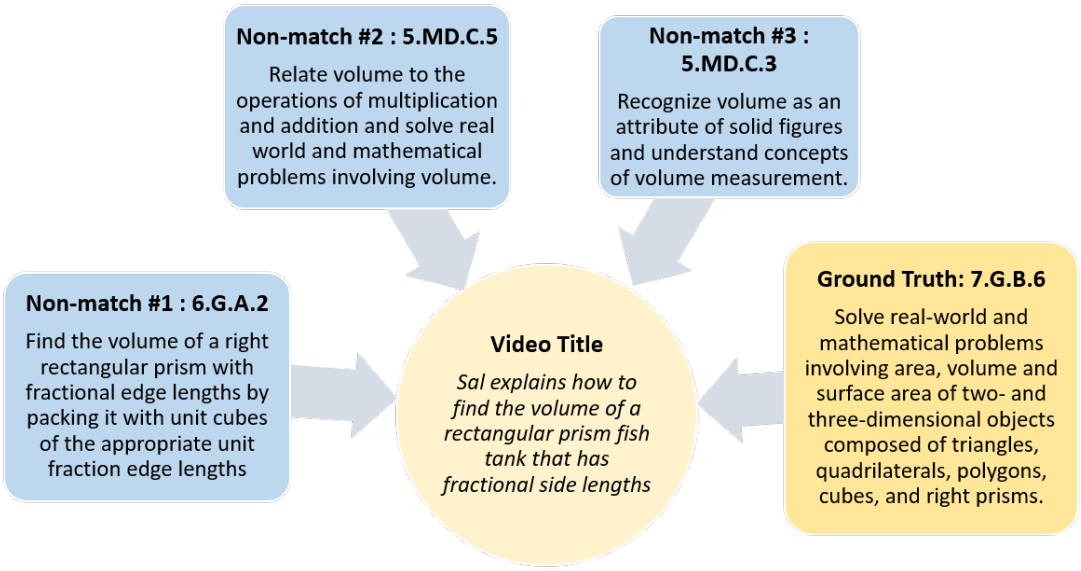


Figure 5.2: An illustration of multiple possibilities of a correct label for a given video title text

- Semantic Similarity (C_t): We adopt the Doc2Vec algorithm [76] to capture the similarity between KC labels. Doc2Vec, derived from word-vector algorithm, generates similarity scores between documents instead of words and is proved to have lower error rate (7.7-16%) than the word vector approach [76].
- Structural Similarity (C_s): We exploit prerequisite relationships among skills (KC labels) and capture such as edges and KC labels as nodes in a graph. The prerequisite relationships are extracted from a K-G8 math coherence map by Jason Zimba [77] and a high school (G9-G12) coherence map by UnboundEd Standard Institue [78]. Then, we adopt Node2Vec algorithm [79] that is efficient and flexible in exploring nodes similarity and achieved a new record performance in network classification problem [79].

In the end, we craft a new evaluation measure, named as *TEXSTR* (Λ), by combining both C_t and C_s as follows: $\Lambda = \alpha \cdot C_t + (1 - \alpha) \cdot C_s$, where α controls the weight between C_t and C_s as an oscillating parameter.

5.4 Empirical Validation

5.4.1 data sets and Evaluation Measure

Table 5.2 summarizes the details of the data sets for pre-training and fine-tuning processes. D_d contains 6,384 description texts (84,017 tokens) and 385 math KCs (an example shown in Fig. 5.1-a). Part of D_d are extracted from Common Core Standards website[2] and part are provided by k12.com[3], an education management organization that provides online education to American students from kindergarten to Grade 12. D_t contains 6,748 video title texts (62,135 tokens) and 272 math KCs (an example shown in Fig. 5.1-b) Part of D_t are extracted from *Youtube.com* (via youtube DataAPI[4]) and part are provided by k12.com. D_p contains 13,722 texts (589,549 tokens) and 213 math KCs provided by ASSISTments[5] (an example shown in Fig. 5.1-c). Further, D_{d+t}, D_{d+p}, D_{t+p}, and D_{all} are different combinations of the unlabeled texts from D_d, D_t, and D_p. They are only used in the TAPT pre-training process. We pre-process all aforementioned texts by removing all the templates and HTML markups to avoid over-fitting, suggested by the prior highest accuracy method [39]. In the TAPT pre-training process, 100% of the unlabeled texts from the aforementioned data sets are used for pre-training. In fine-tuning process for both TAPT and BASE BERT, only D_d, D_t, and D_p are used and 72% of their texts and labels are used for training, 8% are for validation and 20% are for testing (see in Table 5.2 Row 1-3 and Col. 6-8).

As an evaluation measure, following prior research [39,60–63] for direct comparison, we use Accuracy@k as (TP + TN)/(TP + TN + FP + FN), when a method predicts top-k KC labels. Further, we evaluate our method using the proposed *TEXSTR* measure.

5.4.2 Pre-training and Fine-tuning Details

To further pre-train, we follow the same pre-training process of original BERT with the same network architecture (12 layers, 768 hidden dimensions, 12 heads, 110M

[2]http://www.corestandards.org/math
[3]http://www.k12.com
[4]http://developers.google.com/youtube/v3
[5]http://www.assistments.org/

Table 5.2: A Summary Statistics of Data sets.

Name	# Labels	# Texts	# Tokens	Fine-tuning Partition		
				Training (72%)	Validation (8%)	Testing (20%)
D_d	385	6,384	84,017	4,596	511	1,277
D_t	272	6,748	62,135	4,858	540	1,350
D_p	213	13,722	589,549	9,879	1,098	2,745
D_{d+t}	/	13,132	146,152	/	/	/
D_{d+p}	/	20,106	673,566	/	/	/
D_{t+p}	/	20,470	651,684	/	/	/
D_{all}	/	26,854	735,701	/	/	/

parameters) but on our own unlabeled task-specific texts (see Col. 4 in Table 5.2). With an 8-core v3 TPU, we further train all our models with 100k steps, achieving MLM accuracy of above 97% that lasts about 1-4 hours. We experiment hyper-parameters such as learning rate (lr) $\in \{1e-5, 2e-5, 4e-5, 5e-5, 2e-4\}$, batch size (bs) $\in \{8, 16, 32\}$, and max-sequence length (max-seq-len) $\in \{128, 256, 512\}$. The highest MLM accuracy was achieved when lr $\leftarrow$ 2e-5, bs $\leftarrow$ 32, and max-seq-len $\leftarrow$ 128 (for D_d and D_t) and max-seq-len $\leftarrow$ 512 with the same lr and bs (for D_p, D_{d+p}, D_{t+p}, D_{all}). To fine-tune, we also follow the original BERT script by splitting D_d, D_t, D_p into 72% for training, 8% for validation and 20% for testing per task. We experiment ep $\in \{5, 10, 25\}$ due to the small size of the data size and retain the same hyper-parameter search for lr, bs, max-seq-len. We find that the best testing accuracy is obtained when ep $\leftarrow$ 25, lr $\leftarrow$ 2e-5, bs $\leftarrow$ 32, and max-seq-len $\leftarrow$ 128 for D_d, D_t whereas the best testing accuracy for D_p is obtained when ep $\leftarrow$ 25, lr $\leftarrow$ 2e-5, bs $\leftarrow$ 32, and max-seq-len $\leftarrow$ 512. We find that after ep $\leftarrow$ 25, it is difficult to gain significant increase on the testing accuracy. Hence, the optimal hyper-parameters while task-dependent seem to have very minimal change across tasks. This finding is consistent with SCIBERT reported [72].

5.4.3 Result #1: TAPT BERT vs. Other Approaches

Table 5.3 summarizes the experimental results of six baseline approaches and TAPT BERT for each task. For baseline methods, we group them into categories (see in Table 5.3) (1) classical ML, (2) prior work, and (3) BASE BERT. By including popular ML methods such as Random Forest and XGBoost, we aim to compare its performance to the one from prior ML work (SVM) proposed by Karlovec et al [60] in the literature review. As to comparing to the prior highest accuracy

Table 5.3: Accuracy comparison (best and 2nd best accuracy in blue bold and underlined, respectively, $BL\dagger$ for baseline best, and * for statistical significance with p-value < 0.001)

Approach Type	Algorithm	D_d		D_t		D_p	
		ACC@1	ACC@3	ACC@1	ACC@3	ACC@1	ACC@3
Classical ML	SVM [60]	44.87	70.40	48.15	70.30	78.07	87.69
	XGBoost	43.07	71.34	45.33	66.15	77.63	87.94
	Random Forest	49.26	78.78	49.33	74.37	78.03	88.23
Prior Work	Skip-Gram NN [62]	34.07	34.15	43.00	43.52	76.88	77.06
	Sklearn MLP [39]	50.53	74.41	48.22	57.95	80.70	81.13
BERT	BASE	48.30	76.40	50.99	76.55	81.73	90.99
	TAPT	**50.60**	**79.29**	**52.71**	**78.83**	**82.43**	**92.51**
Improvement	$\|TAPT - BL\dagger\|$	0.07	0.51	1.72	2.28	0.70	1.52
	$\|TAPT - BASE\|$	2.30*	0.51*	1.72*	2.28*	0.70*	1.52*

method [39], we applied the same 5-fold cross-validation on our own problem texts and obtain ACC@1 (Accuracy of the Top 1 Prediction) and ACC@3 (Accuracy of Top 3 Predictions). Overall, we see that TAPT models outperform all other methods at both ACC@1 and ACC@3 across three tasks. Note TAPT models here are simply trained on the unlabeled texts from D_d, D_t, and D_p. Compared to the best method in baseline, the TAPT model has an increase of 0.07%, 1.72%, 0.70% at ACC@1 and 0.51%, 2.28%, 1.52% at ACC@3 across the three tasks. Compared to the BASE BERT, the TAPT BERT shows an increase of 2.30%, 1.72%, 0.70% at ACC@1 and 0.51%, 2.28%, 1.52% at ACC@3 across the three tasks. ACC@1 and ACC@3 from both TAPT and BASE models are the average performance over five random seeds with significant difference (see last row in Table 5.3). BERT variants such as FinBERT [69], SCIBERT [72], BioBERT [71] and E-BERT [73] were able to achieve a 1-4% increase when further trained on much larger domain knowledge corpus (i.e., 2-14 billion tokens). Our corpus although comparatively small with D_d (84,017 tokens), D_t (62,135 tokens), and D_p (589,549 tokens) still result in a decent improvement of 0.51-2.30%.

5.4.4 Result #2: Augmented TAPT model and Its Generalizability

In addition to the simply trained TAPT models (referred as simple TAPT) in Table 5.3, we augment the pre-training data and form another four TAPTs ($TAPT_{d+t}$, $TAPT_{d+p}$, $TAPT_{t+p}$ and $TAPT_{all}$). We call them augmented TAPT. Table 5.4

Table 5.4: ACC@3: BASE vs. TAPT BERT. (best and 2nd best per row in bold and underlined, and subscripts indicate outperformance over BASE)

Data	BASE	Simple			Augmented			
		$TAPT_d$	$TAPT_t$	$TAPT_p$	$TAPT_{d+t}$	$TAPT_{d+p}$	$TAPT_{t+p}$	$TAPT_{all}$
D_d	76.40	$79.29_{2.89}$	$78.78_{2.38}$	$77.84_{1.44}$	$\underline{79.40}_{3.00}$	$\mathbf{79.56}_{3.16}$	$79.01_{2.61}$	$79.01_{2.61}$
D_t	76.55	$\underline{77.85}_{1.30}$	$\mathbf{78.83}_{2.28}$	$76.30_{-0.25}$	$77.56_{1.01}$	$77.56_{1.01}$	$77.70_{1.15}$	$77.78_{1.23}$
D_p	90.99	$91.22_{0.23}$	$91.44_{0.45}$	$\underline{92.51}_{1.52}$	$92.06_{1.07}$	$92.50_{1.51}$	$\mathbf{92.64}_{1.65}$	$92.35_{1.36}$

showcases the differences in ACC@3 between simple and augmented TAPT. For D_d, augmented $TAPT_{d+p}$ outperforms all simple TAPT models (ACC@3 = 79.56%) and augmented $TAPT_{d+t}$ achieves the second best ACC@3 (79.40%). For D_t, all the augmented TAPT models only outperform simple $TAPT_p$. For D_p, augmented $TAPT_{t+p}$ outperforms all simple TAPTs with ACC@3 of 92.64%. To sum up, augmenting the pre-training data for TAPT models seems to help increase the accuracy further.

Furthermore, we compare the generalizability of TAPT to BASE BERT over different data sets. We define the *generalizability* as task accuracy (specifically ACC@3) that a model can obtain when applied to a different data set. Both BASE and TAPT BERTs are pre-trained models and obtain task accuracy via fine-tuning on a different task data. The subscripts in Table 5.4 present the difference in ACC@3 between TAPT and BASE BERT, showcasing who has stronger generalizability ($-$ sign indicates weak generalizability). For D_d, all simple and augmented TAPT models generalize better than BASE, especially augmented TAPTs have an average of about 3% increase. For D_t, all TAPT models have better generalizability than BASE with over 1% average increase except for $TAPT_p$. For D_p, we also see all the TAPTs generalize better than BASE model with the augmented $TAPT_{t+p}$ having the best generalizability.

5.4.5 Result #3: TEXSTR Based Evaluation

Following the definition of *TEXSTR* (=Λ) in Chapter **??** Section 5.3.2, we vary the values of α by {0, 0.5, 1} and generate three variations of Λ for top-3 predictions. We then decide the percentage of miss-predictions to be reconsidered based on Λ value by three cut-off thresholds {0.5, 0.75, 0.9}. Before that, we make sure that the predicted labels are not subsequent to the ground truth, i.e., if the ground truth is *7.G.A.2*, a predicted label such as *8.G.A.3* shall not be reconsidered as

correct because it is the skill to be learned subsequently "after" *7.G.A.2*. In such a case, we exclude predicted labels that have subsequent relations to the ground truth and calculate Λ. Table 5.5 presents the percentage of miss-predictions after removing the subsequent-relation labels by three Λ thresholds when $\alpha \in \{0, 0.5, 1\}$. Across three values of α and data sets, note that 56-73% of miss-predictions could be reconsidered as correct if $\Lambda > 0.5$, 5-53% of them could be reconsidered if $\Lambda > 0.75$, and 0-32% could be reconsidered if $\Lambda > 0.9$. The wide percentage range for $\Lambda \in \{0.75, 0.9\}$ infers that higher thresholds of Λ are more sensitive to the change of α.

To further ensure the *TEXSTR* measure to be useful in practice, we conduct an empirical study where eight experienced K-12 math teachers rate each pair of top-3 KC labels and the corresponding text (e.g., description, video title, or problem text) on a scale of 1 to 5. The Fleiss' kappa value to assess the multi-rater agreement among eight teachers is 0.436, which is considered as moderate agreement by Landis et al. [80]. We ensure that none of top-3 miss-predicted KCs are subsequent to ground truths and have Λ score at least 0.5. Then, we quantify the *relevance* (Υ) score as either Λ score (when $\alpha = 0.5$) or teachers' rating of [1,5] range divided by 5 (to be on the same scale as *TEXSTR*'s [0,1]). Table 5.6 summarizes three varying relevance scores ($\Upsilon \in \{0.5, 0.75, 0.9\}$) on the pair of top-3 predictions and the texts. For Top-1 predictions, *TEXSTR* considers all of them to have $\Upsilon > 0.5$ (due to the pre-selection) and 37.93% of all have $\Upsilon > 0.75$ and 3.45% have $\Upsilon > 0.9$. Teachers, on the other hand, think that only 54.31% of the texts have $\Upsilon > 0.5$ ($\downarrow$ 45.69% from Λ) but 43.53% have $\Upsilon > 0.75$ ($\uparrow$ 5.6% from Λ) and 31.03% have $\Upsilon > 0.9$ ($\uparrow$ 27.58% from Λ). We also find a similar pattern for Top-2 and Top-3 predictions where teachers find 6.47-6.89% more cases than *TEXSTR* that have $\Upsilon > 0.75$ and 9.48-13.79% more cases than *TEXSTR* that have $\Upsilon > 0.9$. This indicates that *TEXSTR* is more conservative than teachers in judging the relevance of KC labels to texts when $\Upsilon \in \{0.75, 0.9\}$, suggesting *TEXSTR* is effective in reassessing miss-predictions and "recover" them as correct labels in practice.

Table 5.5: % of miss-predictions recovered by *TEXSTR* (Λ)

Data	# Miss-predictions	$\Lambda > 0.5$			$\Lambda > 0.75$			$\Lambda > 0.9$		
		$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$
D_d	248	70.16	68.95	72.98	52.82	24.19	8.87	32.26	2.42	0.81
D_t	240	58.33	55.83	57.5	37.92	17.08	6.67	17.08	0	1.25
D_p	166	60.84	56.63	58.43	38.55	16.27	5.42	18.67	1.2	1.2

Table 5.6: % of top-3 predictions by relevance (Υ) level when $\alpha = 0.5$

Υ	Top 1			Top 2			Top 3		
	Λ	Teachers	Δ	Λ	Teachers	Δ	Λ	Teachers	Δ
> 0.5	100	54.31	-45.69	100	40.95	-59.05	100	21.98	-78.02
> 0.75	37.93	43.53	5.60	20.69	27.16	6.47	6.9	13.79	6.89
> 0.9	3.45	31.03	27.58	0	13.79	13.79	0	9.48	9.48

5.5 Summary

The chapter classified 385 math knowledge components from kindergarten to 12th grade using three data sources (e.g., KC descriptions, video titles, and problem texts) via the *Task-adaptive Pre-trained* (TAPT) BERT transfer learning model. The TAPT model has achieved a new record by outperforming six baselines by up to 2% at ACC@1 and up to 2.3% at ACC@3. We also compared TAPT to BASE BERT and found the accuracy of the TAPT BERT increased by 0.5-2.3% with a significant p-value. Furthermore, the chapter discovered that the TAPT model trained on the augmented data by combining different task-specific texts had better ACC@3 than the TAPT model simply trained on the individual data sets. In general, the TAPT model has better generalizability than BASE BERT by up to 3% at ACC@3 across different tasks. Finally, the chapter proposed a new evaluation measure *TEXSTR* to reassess the predicted KCs by taking into account semantic and structural similarity. *TEXSTR* was able to reconsider 56-73% of miss-predictions as correct for practical use.

Chapter 6
Generalize Between Different Domains: A DAPT Approach

6.1 Introduction

The arrival of transformer-based language model, BERT [64], has revolutionized the NLP research and affected the ways transfer learning are carried out. One strength of BERT is its ability to adapt to a new task which we have proved effective in Chapter 5 and also by other works [68], [81], [75], [82], [74]. Beside adapting to new tasks, it is also flexbile to adapt to a new domain through pre-training to transfer the learning from general language field to a target field. By taking an advantage of this benefit researchers have adapted BERT into diverse domains such as FinBERT [69], ClinicalBERT [83], BioBERT [71], SCIBERT [72], E-BERT [73], LiBERT [84] with improved performances. In the domain of mathematics, as mathematical text often use domain or context specific words, together with math equations and symbols, we posit that mathematics-customized BERT would help researchers and practitioners sort out the meaning of ambiguous language better by using surrounding text to establish “math" context. Further, such an improved context-aware understanding of language could help develop and improve solutions for challenging NLP tasks in mathematics. Although we have developed TAPT models in Chapter 5 to gain high accuracy for the individual tasks, a domain-adapted BERT model is expected to handle multiple tasks with one model instead of training individual TAPT models from the individual task data. This is knowledge generalization applied in the same setting, namely, between different domains

but is an extension to Chapter 5.

Therefore, we restate the tasks and uncover what challenges and/or opportunity the existing methods leave us to. The popular AI tasks in the education domain are: (i) large-scale knowledge component (KC, a.k.a. skill) prediction (denoted as T_{kc}), (ii) open-ended question answer scoring (i.e., auto-grading) (denoted as T_{ag}), and (iii) knowledge tracing (KT) correctness prediction (denoted as T_{kt}).

The struggle with T_{kc} (e.g., predicting the right mathematical skill for a given text description) is partly attributed to its tediousness and labor-intensive work for teachers/tutors to label all knowledge components in texts where they need to organize mathematical problems, or descriptions of instructional videos, etc. The traditional way to address this challenge of T_{kc} is to use machine learning to classify them via feature extraction [39,60,62], which has produced decent results but cost-intensive because researchers need to develop individual sets of features for each model. For T_{ag}, although they are becoming less popular due to the difficulty of developing universal automated support in assessing the response quality, it is still important because open-ended questions are known to be able to provide critical evaluation in testing students true critical thinking and understanding. The existing methods tends to solve the automation task from seeking similarity via distance features customized towards individual training data [85–88].

Third, *Knowledge Tracing*, a very important task in the education domain, is defined as the task of tracing students' knowledge state, which represents their mastery of educational content based on their past learning activities. Predicting students' next question correctness as a KT task is, for instance, well studied [8,10,37,89,90] in the general ML and DL fields but these solutions tend to simply include question-answer time sequence pair as input. The current solutions are still not able to capture the complex nature of students' learning activities over extended periods of time. Last but not the least, all the solutions developed towards the aforementioned tasks live in its silo system. In other words, they are developed individually for each task and none of them can be applied over each other to get similar high performance due to the label space and conditional probability differences.

In light of the recent successes from transfer learning language models such as ELMo [66], ULMFiT [67] and BERT [64], the above challenge could be likely tackled by a transfer learning solution approached via the high-performing pre-

trained language model. Therefore, we propose the below research questions:

RQ1: Can we develop a BERT-like transfer learning model to predict multiple NLP tasks in math domain instead of a single task? Can it be more effective than the prior methods and BASE BERT ?

RQ2: What makes this BERT variant more effective than the prior methods and the BASE BERT?

RQ3: How to make this BERT variant more beneficial to the education community to help improve mathematics education further?

To answer the first research question, we realize that directly applying BERT to mathematical tasks has limitations. First, the original BERT (i.e., BASE BERT) was trained mainly on general domain texts (e.g., general news articles and Wikipedia pages). As such, it is difficult to estimate the performance of a model trained on these texts on tasks using data sets that contain mathematical text. Second, the marginal distributions of general corpora is quite different from mathematical corpora (e.g., mathematical equations and symbols), which can often be a problem for mathematical task related models. Therefore, we hypothesize that a special BERT model needs to be trained on mathematical domain corpora to be effective in mathematics-related tasks. Thus, we make the following contributions in this chapter:

1. We build MathBERT by pre-training the BASE BERT on mathematical domain texts ranging from pre-k to high-school to graduate level mathematical curriculum, books and paper abstracts and evaluate its performance by comparing to five baseline models including the prior best method and BASE BERT with a margin of [1%,22%] and [2%,8%] respectively.

2. We build and release a custom vocabulary `mathVocab` to examine its contribution to MathBERT's superior performance via comparing the performance of MathBERT pre-trained with `mathVocab` to MathBERT pre-trained with the original BASE BERT vocabulary.

3. We publicly release MathBERT as a community resource below, which enables the use cases where the two major learning management systems: ASSISTments and K12.com by Stride adopt MathBERT to improve the math education on their systems.

- `https://github.com/tbs17/MathBERT` for codes on how to further-train and fine-tune, and
- `https://huggingface.co/tbs17/MathBERT` for PyTorch version MathBERT and tokenizer.
- AWS S3 URLs [1] for Tensorflow version MathBERT and tokenizer.

6.2 Related Work

The state-of-the-art (SOTA) language model BERT (Bidirectional Encoder Representations From Transformer) [64] is a pre-trained language representation model that was trained on 16 GB of unlabeled texts, including Books Corpus and Wikipedia, with a total of 3.3 billion words and a vocabulary size of 30,522. Its advantage over other pre-trained language models such as ELMO [66] and ULMFiT [67] is its bidirectional structure by using the *masked language model* (MLM) pre-training objective [64]. The MLM randomly masks 15% of the tokens from the input to predict the original vocabulary id of the masked word based on its context from both directions [64]. The pre-trained model can be used directly to fine-tune on new data for NLP understanding and inference tasks or further pre-trained to get a new set of weights for transfer learning.

The further pre-training process has become popular in the past two years as it is able to achieve better results than the fine-tuning only strategy. According to Gururangan et al. [74], there are two styles of further pre-training on the BASE BERT [64]: (i) further pre-train the BASE BERT on a task-specific data set with tasks being text classification, question and answering inference, paraphrasing, etc. Gururangan et al. [74] call this kind of model a Task-adaptive Pre-trained (**TAPT**) Model. (ii) further pre-train the BASE BERT on a domain-specific data set with domains being finance, bio-science, clinical fields, etc. Gururangan et al. [74] call this kind of model a Domain-adaptive Pre-trained (**DAPT**) Model. Both TAPT and DAPT BERT models start the further pre-training process from the BASE BERT weights but pre-train on different types of corpora. TAPT BERT models pre-train on task-specific data, whereas DAPT BERT models pre-train on the

[1] http://tracy-nlp-models.s3.amazonaws.com/mathbert-basevocab-uncased/
http://tracy-nlp-models.s3.amazonaws.com/mathbert-mathvocab-uncased/

Table 6.1: Corpora Comparison for DAPT BERT Models

Domain	Name	# Tokens	Corpora
General NLP	Original BERT	3.3B	News article, Wikepedia
Bio Medicine	BioBERT	18B	PubMed, PMC articles
Clinical Medicine	ClinicalBERT	2M (notes)	Hospital Clinical Notes
Science	SciBERT	3.2B	Semantic Scholar Papers
Job	LiBERT	685M	LinkedIn search query profile, job posts
E-commerce	E-BERT	233M (reviews)	Amazon Data Set[2]
Finance	FinBERT	12.7B	Reuters News stories
Mathematics	**MathBERT** (This Work)	100M	Math curriculum and books, Math arXiv paper abstract

Table 6.2: Corpora Comparison for TAPT BERT Models. * indicates that the number is an estimation based on 150 tokens per sentence

Domain	Data set	# Tokens	Task
BioMed	ChemProt [74]	1.5M*	relation classification
	RCT [74]	12M*	abstract sent. roles
Comp. Sci.	ACL-ARC [74]	291,150*	citation intent
	SCIERC [74]	697,200*	relation classification
News	HyperPartisan [74]	96,750*	partisanship
	AgNews [68, 74]	5.6M	topic
Reviews	Yelp [68]	25M	review sentiment
	IMDB [68, 74]	14.6M	review sentiment
Linguistics	VUA-20 [75]	205,425	metophor detection
	VUA-Verb [75]	5,873	metophor detection
Mathematics	KC [81]	589,549	skill code detection

domain-specific data before they are fine-tuned for use in any downstream tasks (see the process illustrated in Figure 6.1).

The domain specific corpora that DAPT BERT models trained on are usually huge (e.g., billions of news articles, clinical texts or PMC full-text and abstracts), which help DAPT BERT models achieve SOTA performance in the corresponding domains. For example, FinBERT [69], ClinicalBERT [83], BioBERT [71], SCIBERT [72]. Other DAPT models such as E-BERT [73] and LiBERT [84] not only further pre-trained on the domain specific corpora but also modified the transformer architecture to achieve better performance for the domain related tasks. A

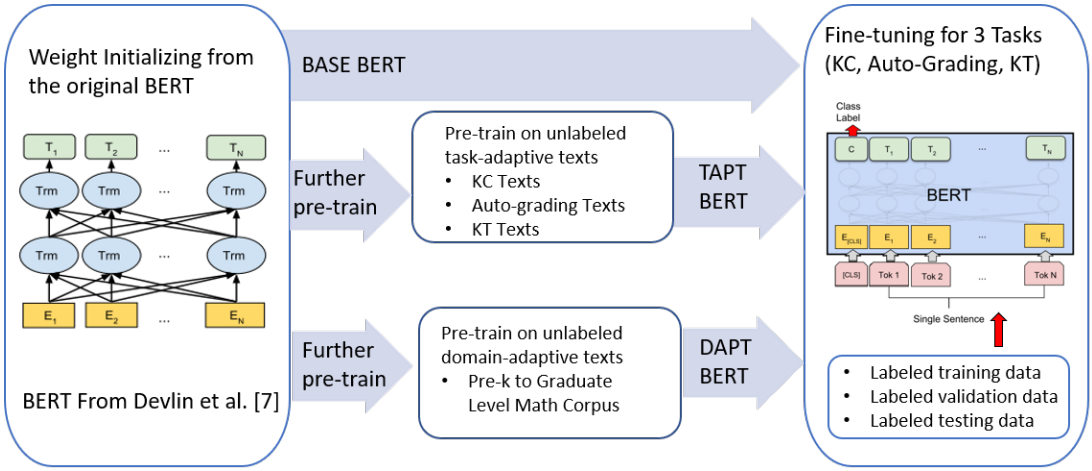


Figure 6.1: An illustration of training and fine-tuning process of BASE vs. TAPT vs. DAPT BERT models. The pre-training data are from this study. KC, Auto-grading, and KT Texts are task data for T_{kc}, T_{ag}, and T_{kc} respectively.

comparison between different domain-specific BERT models' corpora is shown in Table 6.1. From the table, we can see that BioBERT was pre-trained on the largest set of tokens (18B) whereas our MathBERT is pre-trained on the smallest set of tokens (100M). Although the scale of training data is much smaller than the BASE BERT, MathBERT is still more effective in evaluating mathematics related tasks.

There are also a few works that focus on TAPT models. Sun et al. [68] proposed a detailed process on how to further pre-train a TAPT BERT model and fine-tune it for three types of classification tasks (i.e., sentiment, question, and topic), achieving a new record accuracy. Shen et al. [81] pre-trained a TAPT BERT model to predict knowledge components and surpassed the BASE BERT accuracy by about 2%. MelBERT [75] further pre-trained the RoBERTa-base BERT on well-known public English data sets (e.g.,VUA-20, VUA-Verb) that have been released in metaphor detection tasks and obtained [0.6%, 3%] out-performance over the RoBERTa-base [82]. Gururangan et al. [74] pre-trained RoBERTa-base [82] on famous task data sets (e.g., Chemprot, RCT, ACL-ARC, SCIERC, Hyperpartisan, AgNews, and IMDB tasks) and obtained [0.5%, 4%] better performance than RoBERTa-base. Table 6.2 presents the training data size for the aforementioned TAPT Models, showcasing that TAPT models have much smaller training data size than the DAPT BERT models. In general, DAPT models usually achieve better performance (1-8% higher) than TAPT models [74]. Although DAPT BERT

models require more time and resource to train, they have wider applications than TAPT BERT models because they do not need to retrain in the case of different tasks, where TAPT BERT models tend to.

In light of the aforementioned success, we also build a DAPT model, MathBERT, that is further pre-trained from the BASE BERT model with a dedicated mathematical corpus. With the similar goal to our MathBERT, we note that the work by [91] was also independently announced about the same time (i.e., [91] was submitted to arXiv while our MathBERT was released to GitHub and Hugging Face, both in May 2021). Peng et al. [91] also built a pre-trained BERT from the mathematical formula data and applied it on three formula-related tasks (i.e., math info retrieval, formula topic classification, formula headline generation). However, as they claimed, their BERT is the first pre-trained model for mathematical formula understanding and was only trained on 8.7 million tokens of formula latex data with the 400 surrounding characters from arXiv papers (graduate-level). Our MathBERT is pre-trained on 100 million tokens of more general purpose mathematical corpora including curriculum, books, and arXiv paper abstracts, covering all the grade bands from pre-k to college graduate-level. Our training data not only include formulas and their contexts but also more general mathematical instructional texts from books, curriculum, MOOC courses, etc. We consider our work has a potential to be widely used for "general" mathematics-related tasks. For instance, MathBERT in Hugging Face has been downloaded more than 100K times since May 2021 with the peak of 31,370 in July, 2021. As [91] has not released their code and model artifacts, we could not compare our results directly to theirs. We welcome further comparison and analysis by releasing all our code and model artifacts at `https://github.com/tbs17/MathBERT`.

6.3 Building MathBERT

6.3.1 Math Corpora

MathBERT is pre-trained on mathematics related corpora that comprise mathematics curricula from pre-k to high school, mathematics textbooks written for high school and college students, mathematics course syllabi from Massive Online Open Courses (MOOC) as well as mathematics paper abstracts (see in Table 6.3). We

Table 6.3: Math Corpus Details. Note all the corpus is in mathematics domain

Source	Math Corpora	Tokens
arxiv.org	Paper abstract	64M
classcentral.com	College MOOC syllabus	111K
openculture.com	pre-k to College Textbook	11M
engageny.org	Pre-k to HS Curriculum	18M
illustrativemathematics.org	K-12 Curriculum	4M
utahmiddleschoolmath.org	G6-8 Curriculum	2M
ck12.org	K-12 Curriculum	910K

crawl these data from popular mathematics curriculum websites (illustrativemathematics.org, utahmiddleschoolmath.org, engageny.org), a free text book website (openculture.com), a MOOC platform (classcentral.com), and arXiv.org, with a total data size of around 3GB and 100 Million tokens. The mathematics corpora not only contain text but also mathematics symbols and equations. Among all these data, the text book data is in PDF format and we hence converted them into text format using the Python package `pdfminer`[3], which preserves the mathematics symbols and equations (see sample text in Figure 6.2).

6.3.2 Training Details and Outcome

To pre-train MathBERT efficiently, we adopt a similar data processing strategy to the ROBERTa model, which threaded all the sentences together and split them into a maximum length of 512-token sequence sections [82]. In other words, one sequence of data is longer than the original single sentence from the mathematics corpora. Inspired by SciBERT [72], we create a custom mathematical vocabulary (`mathVocab`) using Hugging Face `BertWordPieceTokenizer`[4] with a size of 30,522 from the BASE BERT. We select 50 words from the same rank tier of #2100 to #2150 and discover that `mathVocab` has more mathematical jargon than the original vocabulary (`origVocab`) from BERT [64] (see in Table 6.4).

We use 8-core TPU machine from Google Colab Pro to pre-train the BASE BERT on the mathematics corpora. The largest batch size (bs) we can fit into the TPU memory is 128 and the best training learning rate (lr) is $5e-5$ with

[3]https://pypi.org/project/pdfminer/

[4]https://huggingface.co/docs/tokenizers/python/latest/quicktour.html

Table 6.4: Vocabulary Comparison: origVocab vs. mathVocab. Tokens in blue are mathematics domain specific.

Vocab Type	50 Selected Tokens (from #2100-#2150)
origVocab	##y, later, ##t, city, under, around, did, such, being, used, state, people, part, know, against, your, many, second, university, both, national,##er, these, don, known, off, way, until, re, how, even, get, head, ..., didn, ##ly, team, american, because, de, ##l, born, united, film, since, still, long, work, south, us
mathVocab	cod, exist, ##olds, coun, ##lud, ##ments, squ, ##ings, known, ele, ##ks, fe, minutes, continu, ##line, addi, small, ##ology, triang, ##velop, ##etry, log, converg, asym, ##ero, norm, ##abl, ##ern, every, ##otic, ##istic, cir, ##gy, positive, hyper, dep, ##raw, ##ange, analy, equival, ##ynam, call, mon, numerical, fam, conject, large, ques, ##sible, surf

maximum sequence length (max-seq) of 512 for both MathBERT with `origVocab` and `mathVocab`. We measure the effectiveness of training via Mask Language Modeling (MLM) accuracy (ACC), where the model predicts the vocabulary ID of the masked words in a sentence [64]. For training steps, we find both versions of MathBERT reach their best result at 600K with MLM accuracy of above 99.8% after a training time of 5 days each. We release MathBERT model artifacts trained with `origVocab` and `mathVocab` in both Tensorflow and Pytorch versions (see in `https://github.com/tbs17/MathBERT`). Specifically, one can use AWS S3 bucket URLs[5] to download the Tensorflow version of model artifact. The Pytorch version can be downloaded from the Hugging Face Repo[6] or directly installed within the Hugging Face's framework under the name space "`tbs17`".

[5]http://tracy-nlp-models.s3.amazonaws.com/mathbert-basevocab-uncased
http://tracy-nlp-models.s3.amazonaws.com/mathbert-mathvocab-uncased

[6]`https://huggingface.co/tbs17/MathBERT`

6.4 Downstream Math NLP Tasks

6.4.1 Three Tasks

We use three mathematical tasks mentioned in Section 6.1 to demonstrate the usefulness of MathBERT. They can be formulated as follows:

- KC Prediction (T_{kc}): a single sentence *multinominal classification* problem (213 labels) with $Input(I) \mapsto text$ and $Output(O) \mapsto KC$ (i.e., one of 213 labels).
- Auto-grading (T_{ag}): a two-sentence *multinominal classification* problem (5 labels) with $I \mapsto Question\&Answer$ pair and $O \mapsto Score$.
- KT Correctness (T_{kt}): a two-sentence *binary classification* problem with $I \mapsto Question\&Answer$ pair and $O \mapsto Correctness$.

6.4.2 Task Data

The three task data sets are noted as D_{kc} for T_{kc}, D_{ag} for T_{ag}, and D_{kt} for T_{kt}, respectively. They are used not only to fine-tune for task classification but also for pre-training TAPT BERT models, which will serve as baseline models for MathBERT in Section 6.5. All of three data sets are provided from ASSISTments [1]. We use the same mathematical problem data set as in the best performing prior work [81] with 13,722 texts and 213 labels for KC prediction. The auto-grading task data is the same as in the best performing prior work [85] with 141,186 texts to predict scores 1 to 5. The KT data is the text version (269,230 texts and 2 labels) of the ASSISTments 2009 data[7], the numeric form of which was used by the best performing prior work [8].

Among the three data sets, D_{kc} has the smallest number of records (13,722 rows) but the most unique labels (213 labels), whereas D_{kt} has the largest number of records (269,230 rows) but the least unique labels (2 labels) (see in Table 6.5). These three data sets were chosen due to their accessibility and we don't expect our results would be significantly better or worse if we choose other data sets.

[7]https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data/skill-builder-data-2009-2010

Table 6.5: Task Data Details. KC: Knowledge Component, KT: Knowledge Tracing. All data from ASSISTments platform [1].

Task	#Labels	#Texts	#Fine-tune Split		
			Train (72%)	Validate (8%)	Test (20%)
D_{kc}	213	13,722	9,879	1,098	2,745
D_{ag}	5	141,186	101,653	11,295	28,238
D_{kt}	2	269,230	193,845	21,539	53,846

Table 6.6: Example texts of the three tasks with labels

Task Data	Label	Text
D_{kc}	8.EE.A.1	Simplify the expression: (z2)2 Put parentheses around the power if next to coefficient, for example: 3x2=3(x^2),x5=x^5
D_{ag}	5	Q: Explain your answer on the box below. A: because it is the same shape, just larger, making it similar
D_{kt}	1	Q: What is 2.6 + (-10.9)? A: -8.3

When fine-tuning, both the labels and texts are used (see Column 2 and 3) with split ratio of 72% training, 8% validating, and 20% testing. When pre-training for TAPT BERT models, only the unlabeled texts are used for further pre-training without splitting (see Column 3).

Table 6.6 provides examples from the three task data sets. In D_{kc}, the label '8.EE.A.1' represents a knowledge component (KC) code where '8' means Grade 8, 'EE' is the skill name called 'Expression and Equation', and 'A.1' is the lesson code. There are total of 213 KC codes in D_{kc} with each represented by a specific knowledge component. In D_{ag}, the label '5' is the grading score '5' for the answer in the text. There are total of 5 labels in D_{ag} with '5' being the highest and '1' being the lowest. In D_{kt}, the label '1' means 'correct' for the answer in the text. There are total 2 labels in D_{kt} with another label '0' meaning 'incorrect' for student answers.

Table 6.7: Training Steps and Accuracy: MathBERT vs. TAPT vs. MathBERT+TAPT

Model	Task	Steps	MLM ACC (%)	
			origVocab	mathVocab
MathBERT	/	600K	99.85	99.95
TAPT	T_{kc}	100K	100	/
	T_{ag}	100K	99.10	/
	T_{kt}	120K	99.04	/
MathBERT+TAPT	T_{kc}	100K	100	99.99
	T_{ag}	100K	99.95	99.96
	T_{kt}	100K	99.67	99.68

6.4.3 Task Training and Fine-tuning

We pre-train BASE BERT on the unlabeled texts of D_{kc}, D_{ag}, D_{kt} to build TAPT BERT models and compare their performance to MathBERT. The difference between TAPT and DAPT BERT training is illustrated in Figure 6.1 where the input corpora is different. DAPT BERT models have much larger corpora whereas TAPT BERT models are more specific to tasks. We pre-train three TAPT models with `origVocab` from the BASE BERT [64]. Among them, $TAPT_{kc}$ and $TAPT_{ag}$ reach the best results at 100K steps and $TAPT_{kt}$ reaches its best result at 120K steps with the MLM accuracy of above 99%. Each of the TAPT models takes approximately 1 day to train. In addition to creating TAPT models pre-trained from BASE BERT, we also pre-train TAPT models from the MathBERT weights, called MathBERT+TAPT. They reach the best results at steps of 100K for both `origVocab` and `mathVocab` with the MLM accuracy of above 99.6%. The MathBERT+TAPT models also take approximately 1 day each to pre-train. We try to keep the MLM accuracy of TAPT Models similar to MathBERT (see in Table 6.7). For fine-tuning, we apply D_{kc}, D_{ag}, D_{kt} onto BASE BERT, TAPT BERT, MathBERT, and MathBERT+TAPT models separately. We discover that hyper-parameter tuning has more to do with the task data instead of the model itself. In other words, the best hyper-parameter combinations are the same across MathBERT, TAPT, and MathBERT+TAPT but vary from task to task. Table 6.8 shows the optimal combinations of all the hyper-parameters for each task. This result is obtained after hyper-parameter search on lr $\in \{1e-5, 2e-5, 5e-5, 8e-5, 1e-4\}$, bs $\in \{8, 16, 32, 64, 128\}$, max-seq $\in \{128, 256, 512\}$, and ep $\in \{5, 10, 15, 25\}$.

Table 6.8: Optimal Hyper-parameter Combination for Task fine-tuning

Task	learning rate	batch size	max sequence length	epochs
T_{kc}	5e-5	64	512	25
T_{ag}	2e-5	64	512	5
T_{kt}	5e-5	128	512	5

Table 6.9: Performance Comparison: MathBERT vs. Baseline Methods across Five Random Seeds. Bold font indicates best performance and underlined values are the second best. * indicates statistical significance. Δ shows relative improvement (%) of MathBERT over baselines.

Method	Vocab	T_{kc} (%)		T_{ag} (%)	T_{kt} (%)	
		F1	ACC	AUC	AUC	ACC
Prior Best (p)	/	88.69 [81]	92.51 [81]	85.00 [85]	81.82 [8]	77.11 [8]
BASE-BERT (b)	orig	90.14	91.78	88.67	88.90	86.88
TAPT (t)	orig	91.77	92.96	90.34	95.88	93.49
MathBERT (m)	orig (o)	**92.67**	93.79	<u>90.57</u>	96.04	94.07
	math (c)	92.51	93.60	90.45	95.95	94.01
MathBERT+TAPT (mt)	orig (o)	92.54	<u>93.82</u>	**90.73**	<u>97.25</u>	<u>95.52</u>
	math (c)	<u>92.65</u>	**93.92**	90.46	**97.57**	**95.67**
Δ_{m-p}	orig	+4.49%	+1.38%	+6.55%	+17.38%	+21.99%
	math	+4.31%	+1.18%	+6.41%	+17.27%	+21.92%
Δ_{m-b}	orig	+2.81%*	+2.19%*	+2.14%*	+8.03%*	+8.28%*
	math	+2.63%*	+1.98%*	+2.01%*	+7.93%*	+8.21%*
Δ_{m-t}	orig	+0.98%*	+0.89%*	+0.25%*	+0.17%	+0.62%*
	math	+0.81%*	+0.69%*	+0.12%	+0.07%	+0.56%*
Δ_{m-mt}	orig	+0.14%	-0.03%	-0.18%	-1.26%*	-1.54%*
	math	-0.15%	-0.35%	-0.01%	-1.69%*	-1.77%*
$\Delta_{m^c-m^o}$	/	-0.17%	-0.20%	-0.13%	-0.09%	-0.06%
$\Delta_{mt^c-mt^o}$	/	+0.12%	+0.11%	-0.30%	+0.33%*	+0.16%

6.5 Evaluation of MathBERT

We denote MathBERT pre-trained with `origVocab` as MathBERT-orig and MathBERT pre-trained with `mathVocab` as MathBERT-custom. To evaluate their effectiveness across the tasks of T_{kc}, T_{ag} and T_{kt}, we fine-tune MathBERT on D_{kc}, D_{ag} and D_{kt} and compare the performance to the baseline models (see in Table 6.9). We group the baseline models into four categories: (1) Prior solutions with the best

known performance, [8, 81, 85], (2) BASE BERT without any further pre-training, (3) TAPT BERT models pre-trained on the task specific texts from BASE BERT weights, and (4) MathBERT+TAPT models pre-trained on the task-specific texts from MathBERT weights in both `origVocab` and `mathVocab` versions.

We use both F1 and ACC (i.e., Accuracy @Top 3) to measure T_{kc} prediction results because traditionally, KC problems have been evaluated using ACC [39, 60, 62,63]. We provide the additional measure (F1) to account for the imbalance in the KC labels in D_{kc}. In addition, we use Area-Under-the-Curve (AUC) to measure T_{ag} because AUC is the typical measure used for the auto-grading problem. Finally, both AUC and ACC are used to measure T_{kt} because historically both metrics were used for evaluation [7–10]. After obtaining the best hyper-parameter tuning for each task from Table 6.8, we run each model with five random seeds. We report the average value over five random seeds for each model and use t-tests to evaluate the significance of these results. A t-test is not applied to prior test results as we do not have the five random seeds results from the prior best method due to the lack of accessible codes.

In Table 6.8, we note that MathBERT-orig is about 1.38% to 22.01% better and MathBERT-custom is about 1.18% to 21.92% better than the best prior methods across all metrics and tasks. In addition, MathBERT-orig outperforms BASE BERT by about 2.14 % to 8.28%, all with statistical significance and MathBERT-custom outperforms it by about 1.98% to 8.21% across metrics and tasks, all with statistical significance. Both versions of MathBERT out-performs TAPT BERT models by [0.07%,0.98%] relatively with statistical significance for all tasks. We see both versions of MathBERT under-perform the MathBERT+TAPT models by 0.03 % to 1.77% across all the metrics except for F1 score on T_{kc} from MathBERT-orig. However, only the metrics for T_{kt} have obtained significance. This is expected as MathBERT+TAPT was further pre-trained by adapting it to the task-specific data on top of the MathBERT weights.

In addition, the best performance for each task is all from MathBERT related models. For example, for T_{kc}, the best F1 performance is from MathBERT-orig followed by the second best from MathBERT+TAPT-custom whereas the best and second-best ACC are from both of the MathBERT+TAPT versions (`origVocab` & `mathVocab`). For T_{ag}, we find the best AUC is from MathBERT+TAPT-orig followed by MathBERT-orig. For T_{kt}, the best and second best AUC and ACC

are from both versions of MathBERT+TAPT with MathBERT+TAPT-custom having higher performance.

6.6 Use Cases

In this section, we describe the ongoing activities to incorporate MathBERT into two popular learning platforms.

6.6.1 ASSISTments

ASSISTments is an online learning platform that focuses on K-12 mathematics education. Within ASSISTments, teachers assign course work and view reports on their students. The reports show statistics on the class's performance and the responses of each student. Within the reports, teachers see a timeline of how each student progressed through the assignment and can grade students' open ended responses as well as leaving comments. Figure 6.3 shows an example of an open ended response within a student's report, together with the score and comment left by the teacher.

These open ended responses provide the first opportunity to use MathBERT within ASSISTments. ASSISTments has recently begun using Sentence-BERT [92] to suggest grades to open response questions [86]. MathBERT provides a more domain-specific BERT model for this task with high AUC. The similar task in our experiment T_{ag} obtains 6.55% higher in AUC than the prior best work [85] which uses Sentence-BERT [86], and can replace the current Sentence-BERT implementation. MathBERT can not only provide teachers with suggested grades based on students' open ended responses, but also be used to suggest comments for teachers based on the content of the students' answers.

In addition to MathBERT's benefit to teachers using ASSISTments, MathBERT can also be used to enhance the student experience. As students complete problem sets in the ASSISTments Tutor, shown in Figure 6.4, they can be shown general educational material, such as YouTube videos, if they need additional guidance. MathBERT can be used to identify relevant content by predicting the skills required to solve the problem. As the fine-tuning results for T_{kc} using MathBERT-orig shows, the F1 score and ACC for the top 3 predictions are 92.67% and 93.79%

respectively. Relevant supplemental education material can then be selected and shown to the student. Identifying the skills required to solve a problem will also integrate well with ASSISTments' Automated Reassessment and Relearning System (ARRS) [93]. This service automatically creates follow-up assignments for students when they fail to learn the material they were assigned. The purpose of the follow-up assignments is to test students' knowledge with problems similar to the ones the students previously got wrong. Although MathBERT was tested on text prediction tasks such as T_{kc}, T_{ag} and T_{kt}, it is not limited to only text prediction problems and can be applied to determine textual similarity, similar to the Semantic Textual Similarity Benchmark (STS-B) task from General Language Understand Evaluation (GLUE)[8] which BASE BERT was evaluated on for its performance [64]. Therefore, we can use MathBERT to automatically evaluate problems for similarity, either by determining the skills required to solve the problems, or by directly comparing problem texts.

6.6.2 K12.com by Stride

Stride, Inc that manages the learning platform of K12.com, is a leading education management organization that provides online education to American students from kindergarten to Grade 12 as well as adults. K-G12 math teachers rely on the Stride system to give math lessons, assign practice, home work, or exams, and grade them to provide feedback to students. Teachers have long been challenged by the time and effort they spend to grade and give feedback on open-ended math questions where various answers could be right and it is difficult to scale feedback for immediacy and volume.

Therefore, Stride is considering an automatic scoring pipeline where they can train a model on their huge proprietary reservoir of open-ended responses and teacher feedback to automatically suggest scores and generate constructive feedback/comments for teachers to use. MathBERT could be a nice fit for this model and play two roles: (i) MathBERT fine-tunes on students' responses (input) with ground truth teacher scoring (label) to predict scores with high accuracy (as suggested by T_{ag}), and (ii) MathBERT fine-tunes on the different scores (input) associated with teacher feedback (label) to predict/generate teacher feedback for a

[8] https://gluebenchmark.com/

certain kind of score. For example, a student may only correctly answer part of the question and get a score of 3 out of 5, MathBERT can recommend a feedback such as 'You are very close! Can you tell us more?'. The prediction output from MathBERT can then be wrapped into a question-specific teaching assistant API that prompts in front of students to guide them to reach the full score and truly master the knowledge component (see the pipeline in Figure 6.5).

The pipeline will be split into three phases: (i) collect data (i.e. responses, score, and feedback); (ii) use MathBERT to fine-tune on the training data and predict scores and feedback, suggested to teachers via API. Teachers semi-auto grade and give feedback using MathBERT suggested score and feedback. The final grade and feedback given to the students will then be sent back to the model to further fine-tine, and (iii) improve the accuracy of the question-specific teaching assistant API for fully automatic-scoring where teachers will only play a role in monitoring, reviewing the scores, and providing feedback.

As a proof of concept, Figure 6.6 illustrates what MathBERT will output after fine-tuning on the open-ended responses, scores, and feedback after phase 1. The red words are the feedback that the question-specific API will generate to guide students to achieve a full score. The points (in the yellow box) will be predicted by MathBERT and automatically suggested to teachers.

6.7 Discussion and Limitation

Although we have verified that MathBERT is more effective than the BASE BERT for mathematics related tasks with a proportional improvement of [1.98%, 8.28%] with statistical significance, the effect from an in-domain vocabulary (`mathVocab`) is not what we expect. As we see from Table 6.9, MathBERT-custom has under-performed MathBERT-orig when directly fine-tuned on, but outperformed MathBERT-orig when further pre-trained on task specific data. However, t-tests show MathBERT-orig is not significantly better than MathBERT-custom and MathBERT+ TAPT-custom's out-performance over MathBERT+TAPT-orig is only statistically significant for T_{kc}.

Regarding the finding where we don't find significant help from `mathVocab` to boost performance further, SciBERT [72] explained that the out-performance over BASE BERT could be mainly from the domain corpus pre-training rather than the

in-domain vocabulary that got trained on. Therefore, our discovery is consistent with what SciBERT has found about the custom vocabulary effect. In addition, we note that MathBERT is not only applicable in text prediction tasks but also for other NLP understanding tasks such as paraphrasing, question and answering, and sentence entailment tasks. We evaluate MathBERT for T_{kc}, T_{ag}, and T_{kt} because these three tasks have been heavily studied and their test data are available to us.

In future, we plan to pre-train another MathBERT on “informal" mathematics-related texts as opposed to the formal mathematical content (e.g., math curriculum, book and paper) that the current MathBERT is pre-trained on. We could potentially use such an informal MathBERT to generate answers/conversations for mathematics tutoring chat bots in order to improve the interaction between teachers/tutors and students.

6.8 Summary

In this chapter, we built and introduced MathBERT-orig and MathBERT-custom to effectively transfer the learning from a general language model to the three mathematics-related target tasks. Users can use the code from github to access the model artifacts. We showed that MathBERT not only out-performed prior best methods by [1.18%, 22.01%], but also proportionally out-performed the BASE BERT by [1.98%, 8.28%] and TAPT BERT models by [0.25%, 0.98%] with statistical significance. Although MathBERT-custom was pre-trained with the mathematical vocabulary (`mathVocab`) to reflect the special nature of mathematical corpora, we didn't find the significant over-performance to MathBERT-orig. MathBERT currently is being adopted by two major learning management systems (i.e., ASSISTments and K12.com) to build automatic-scoring/commenting solutions to benefit teachers and students.

1.4 Continuous Functions

We define continuous functions and discuss a few of their basic properties. The class of continuous functions will play a central role later.

Definition 1.14. *Let f be a function and c a point in its domain. The function is said to be continuous at c if for all $\epsilon > 0$ there exists a $\delta > 0$, such that $|f(c) - f(x)| < \epsilon$ whenever x belongs to the domain of f and $|x - c| < \delta$. A function f is continuous if it is continuous at all points in its domain.*

(a) Content of a Math Book

SURFACE DEFECTS IN GAUGE THEORY AND KZ EQUATION

NIKITA NEKRASOV AND ALEXANDER TSYMBALIUK

ABSTRACT. We study the regular surface defect in the Ω-deformed four dimensional supersymmetric gauge theory with gauge group $SU(N)$ with $2N$ hypermultiplets in fundamental representation. We prove its vacuum expectation value obeys the Knizhnik-Zamolodchikov equation for the 4-point conformal block of the $\widehat{\mathfrak{sl}}_N$-current algebra, originally introduced in the context of two dimensional conformal field theory. The level and the vertex operators are determined by the parameters of the Ω-background and the masses of the hypermultiplets, the cross-ratio of the 4 points is determined by the complexified gauge coupling. We clarify that in a somewhat subtle way the branching rule is parametrized by the Coulomb moduli. This is an example of the BPS/CFT relation.

(b) Abstract of a Math arXiv Paper

6.RP.A.3c

Focus Standard:	6.RP.A.3	Use ratio and rate reasoning to solve real-world and mathematical problems, e.g., by reasoning about tables of equivalent ratios, tape diagrams, double number line diagrams, or equations. c. Find a percent of a quantity as a rate per **100** (e.g., **30%** of a quantity means **30/100** times the quantity); solve problems involving finding the whole, given a part and the percent.
Instructional Days:	6	
Lesson 24:	Percent and Rates per 100 (P)[1]	
Lesson 25:	A Fraction as a Percent (P)	
Lesson 26:	Percent of a Quantity (P)	
Lessons 27–29:	Solving Percent Problems (P, P, E)	

(c) Snippet of a Math Curriculum

Figure 6.2: Sample mathematical corpora text from math book, arXiv paper abstract, and curriculum

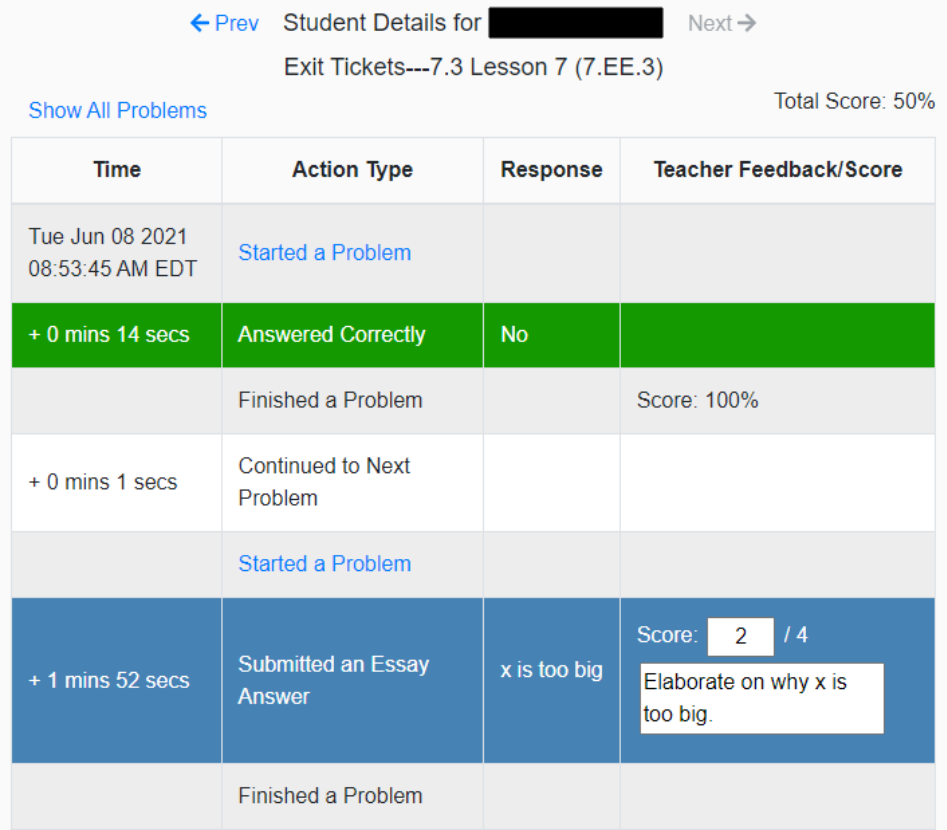


Figure 6.3: An open response in a student's report with the teacher's score and comment.

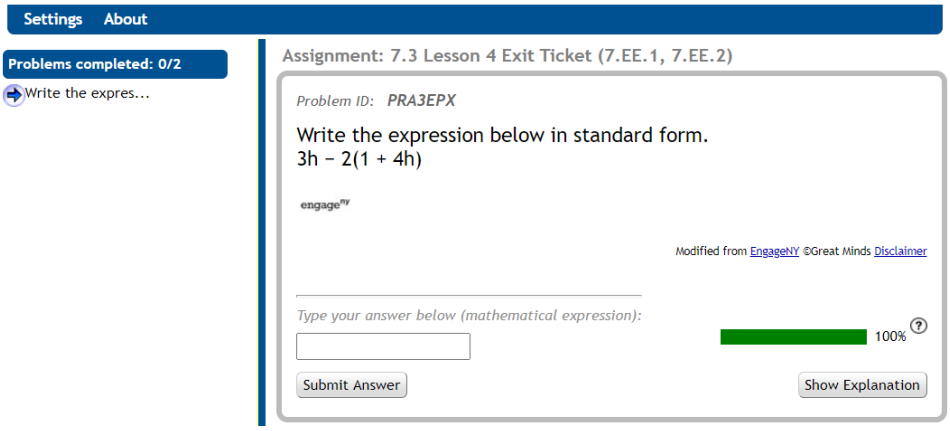


Figure 6.4: The ASSISTments Tutor, as seen by students when completing problem sets.

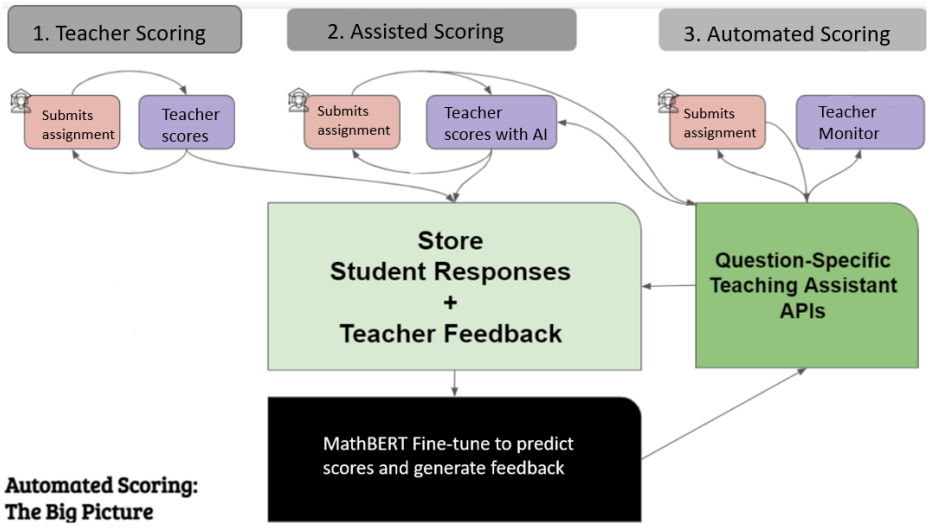


Figure 6.5: Stride auto-scoring pipeline using MathBERT

Middle School Math Unit Test:

(5 points)

2. Which sign should be written in the box: = or ≠ ? Show your work, and explain your reasoning.

$3(14 + 2 \square 8) \square 120 - 15 \square 2$

Model Answer

$3(14 + 2 \square 8)$
$3(14 + 16)$
$3(30)$
expression on left: 90

$120 - 15 \square 2$
$120 - 30$
expression on right: 90

Both sides of the equation simplify to 90, so the correct sign is =.

Award points for specific answers as shown below (for a total of 0–5 points).

Points	Concept Addressed	Feedback for Student Answers
2	Correctly simplifies the left side.	You have to follow the order of operations to simplify an expression. Go back and review the Expressions lesson to review the order.
2	Correctly simplifies the right side.	You have to follow the order of operations to simplify an expression. Go back and review the Expressions lesson to review the order.
1	Correctly concludes that the correct sign is =.	An equation is a sentence that indicates that two expressions are equal in value. Go back to the Equations lesson and review how to determine if two expressions form an equation.

Feedback for completely correct answer:

You correctly determined that the expressions should be joined by an *equal to* sign because the expressions have the same value.

Figure 6.6: Stride auto-scoring model output in the unit test

Chapter 7
Conclusion

To sum up the above chapters, we point out the two challenging issues with AI in education: (i) data scarcity; (ii) knowledge generalizability. To solve the first issue, this thesis proposed two data augmentation methods derived from statistics and generative model perspectives. To solve the second issue, we proposed transfer learning mechanism to effectively generalize knowledge from source model or source domain to a different task or data domain. Below we summarize the limitations, ethical consideration and application of our approaches.

7.1 Limitations

Data Scarcity: Despite that the two data augmentation methods introduced in this thesis are demonstrated effective on the education data, the methods attempted are rather limited and the vast majority of other data augmentation methods have not been validated. For example, the branch of meta learning for data augmentation comprises methods such as smart augmentation [5] which reduces over-fitting efficiently, autoaugment [94], a reinforcement based algorithm to auto search for an optimal policy for augmentation, as well as population based augmentation which trains a series of population of neural networks and optimize their output to find the optimal state quickly. In terms of computation cost, the first method introduced in Chapter 2 might be a bit expensive to operate as we loop through all the time lags (100-600). We suggest practitioners conduct autocorrelation analysis or use Genetic Algorithm [16] to find the best time lag instead if the data sequence is short such as below 50. Furthermore, when we generate data using VAE structures in the subject-based training style, we have not demonstrated

its efficacy on the public education-domain data sets and the private data sets we introduced are not open to the public. Hence, this might harm the reproducibility a bit.

Knowledge Generalization: This thesis categorized knowledge generalization into two scenarios with Scenario (a) as model generalization and Scenario (b) as knowledge transfer. To overcome the model generalization issue, this thesis conducted effective transfer learning and compared the saliency of the transferability between different model structures as well as transfer learning methods (i.e., feature- and instance-based). However, the thesis is a bit limited on types of model structures and not exhaustive on the transfer learning methods comparison due to the constraint of the data types and conventional models used in the education domain. Moreover, in spite of that feature-based transfer learning via freezing layers are demonstrated efficient and less computationally expensive than the instance-based approach featuring a MMD loss function, the thesis did not exclude the possibility of different findings if more time and resources are given to exhaust the hyperparameter tuning for the instance-based method. Regarding Scenario (b), although the proposed solutions (i.e., TAPT and DAPT) have high efficacy and are well applicable in the education domain, one weakness of the TAPT/DAPT approach is that the further training on mathematical symbols and equations is loose as we relied on the pdf extractors' ability to extract the signs and equation representation. If the pdf extractors cannot extract complex equations or decompose them into a latex syntax, they will not be included into our training corpus. In addition, the performance of MathBERT-custom model that are not demonstrated as significantly better than the MathBERT-base could be potentially improved by modifying the neural network structure to fully represent the custom dictionary of the math.

7.2 Ethical Consideration

Ethical considerations in research are a set of principles that include voluntary participation, informed consent, anonymity, confidentiality, potential for harm, and results communication [1]. In the case of this thesis, we will relate to data annoymity and confidentiality, potential for harm. For the numerical data in this thesis (see

[1] https://www.scribbr.com/methodology/research-ethics/

in Chapter 2, 3 and 4), the identity information of students from both the public data sets and private data sets have been removed and hence the confidentiality is maintained. The model artifacts generated from these chapters are not shared with the public but the model performances are shared only for demonstration purpose. Therefore, we do not foresee any potential harm to the public. As for the textual data we extracted online to train TAPT and DAPT models (see in Chapter 5 -6), they are open source curriculum and arxiv paper content data which do not contain any student identity or author information. In addition, given the size of the corpora, we did not make them available to download but disclosed the code on how to extract them. Therefore, we do not violate the confidentiality principle. As far as the model artifacts related to MathBERT from Chapter 6 that are shared on huggingface.co [2], we do not concern the issue of the potential harm either because the training corpora is not biased to any specific race, gender but plain mathmatical curriculum explanation and academic documents in the math domain. A suggested usage of our model is to predict on the education related textual data such as question contents instead of any personal or identity related textual data (see in our model disclaimer [3] on huggingface website).

7.3 Application and Impact

To apply the results of our solutions to combat data scarcity on longitudinal data sets, we recommend adopting the VAE and LVAE structures to generate data because they are demonstrated to be more efficient and effective in augmenting data for longitudinal data comparing to the time lag optimization approach, which is a bit computationally costy. Meanwhile, researchers and practitioners should keep mind of subject-based training by splitting and imputing data via subjects instead of row numbers when generating data for longitudinal time sequence data. When it comes to apply transfer learning solutions to improve model generalization (i.e., Scenario (a)), we recommend using the feature-sharing approach by freezing a number of layers of a source model on the target data instead of co-training on the source and target data for a transfer loss function (e.g., MMD function). The reason is that freezing layers on the previously trained model could more efficiently

[2]https://huggingface.co/tbs17/MathBERT
[3]https://huggingface.co/tbs17/MathBERT

transfer the numerical knowledge and only need to fine-tune on the target data. It is usually computationally frugal comparing to retraining the whole model on source and target data. For users who have limited access to the computational resources, this will become an easier and convenient solutions. Besides that, the specification of the hyperparameter tuning revealed in this thesis is for reference only and users are welcome to apply different turnings to fit their specific problems. However, it is worth pointing out that we apply the same training hyperparameters such as learning rate, weight decay, batch size and epochs to compare the efficiency between different models. We do not rule out the possibility that one could gain better results after extensively tuning a specific model and get a different results from ours. To conduct knowledge transfer (i.e., Scenario (b)) and apply MathBERT that we release on huggingface.co [4], researchers and practitioners are welcome to use it in their favorite deep learning framework as we have provided both pytorch and tensorflow versions. However, as we mentioned above, MathBERT is trained on open source mathematics curriculum and arxiv paper content, the appropriate usage is not to predict general language texts such as "Paris is the [MASK] of France" but rather a sentence like "students apply these new understandings as they reason about and perform decimal [MASK] through the hundredths place" for a fill-mask task. Furthermore, MathBERT is suitable to use for GLUE[5] like tasks such as paraphrasing, natural language inference including similarity, entailment, reference in the mathematics content. However, this thesis only provided testimony on tasks such as sequence classification, token classification and question answering (see in Chapter 6). We believe the generalization superiority from MathBERT can benefit the prediction tasks in the education domain to a decent extent, entailing less training time and efforts.

As for the social impact of the proposed technical contributions, this thesis rather provides a tool, a method that any researcher or industry practitioner could use to help achieve any specific project goal for social good. For example, using deep generative technology could generate data points for the missing values in the finance transaction data so that we can detect abnormal financial activities if the observed data points are not align with the generated data distribution; Using freeze-layer transfer learning can reduce model training time and avoid retraining,

[4]https://huggingface.co/tbs17/MathBERT
[5]https://gluebenchmark.com/tasks

which will yield quick turn-around on the newly ingested training data. Using task/domain adapted further training techniques, a light-weighted language model or chat-bot can be trained for special education community whose members are quite different from average people's learning and understanding capability which large language models such as GPT series or ChatGPT provide but are biased towards.

7.4 Summary

Through this thesis, we provided solutions to two major issues in the education AI domain: (i) data scarcity; (ii) knowledge generalization. Two approaches are suggested and evaluated to tackle each issue. For issue (i), the first solution is to select the best time lag when augmenting data. It is an optimized approach based on the time series lagging practice and is able to improve the SOTA model performance by 32% of AUC in classification task and 12% of RMSE in regression task. The second solution of (i) is to generate data while conducting subject-based training via VAE structures (i.e., VAE and LVAE) on longitudinal data. Our method allows the augmented data to boost original model by almost 50%. Our work also found only a fraction of the generated data (i.e., 10-50%) needed to surpass the SOTA model performance. For issue (ii), our solution is conducting transfer learning to generalize the knowledge from source data to target data for Scenario (a) (i.e., model generalization) and Scenario (b) (i.e., knowledge transfer). To this end, we proposed individual frameworks from simple to complex to adopt multiple methods of transfer learning to solve knowledge generalization in the education domain. For Scenario (a) (i.e., model generalization), we compared two methods (i.e., feature-based and instance based) and model structures (i.e., Transformer and AdaRNN) that conduct transfer learning. We discovered that the feature-sharing method via layer-freezing is less expensive but out-performs the instance-based method featuring MMD loss function up to 5 times (see in Chapter 4). In addition, we observe that Transformer model is 3-4 times more superior than AdaRNN and LSTM to generalize model performances when applying the same training hyperparameters. For Scenario (b) (i.e., knowledge transfer), we developed two approaches: (i) a Task-adpative Pretraining (TAPT) model (see details in Chapter 5); (ii) a Domain-adaptive Pretraining (DAPT) model MathBERT (see

details in Chapter 6) to further train on general language models to adapt to the education domain/tasks. The TAPT approach successfully transferred the semantic and contextual features from a BASE BERT to the target education task and outperformed the prior methods and BASE BERT with a margin up to 2%. The DAPT model MathBERT was trained on and adapted towards a huge mathematics corpus and is suitable to predict multiple target tasks in the education domain with one single model. It successfully outperforms the prior SOTA and BASE BERT by a margin of [1%, 22%] and [2%, 8%] respectively.

At the end, we conclude this thesis with limitations, ethical considerations and applications for each of our solutions. Although all the technical contributions this thesis proposed was applied in the education domain, their motivation and intuition could also be applicable in other fields such as bio-medicine, healthcare fields. Overall, we hope to leverage our proposed methods to effectively improve AI solutions in education domain and to truly benefit teachers and students for their teaching and learning.

Bibliography

[1] Heffernan, N. T. and C. L. Heffernan (2014) "The ASSISTments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching," *Int'l Journal of Artificial Intelligence in Education*, **24**(4), pp. 470–497.

[2] Aayushi Bansal, M., D. Rewa Sharma, and D. Mamta Kathuria (2022) "A Systematic Review on Data Scarcity Problem in Deep Learning: Solution and Applications," *ACM Computing Surveys*, **54**.
URL `https://doi.org/10.1145/3502287`

[3] Ding, J., X. Kang, V. N. Gudivada, X. Li, and X. Kang (2019) "A Case Study of the Augmentation and Evaluation of Training Data for Deep Learning," *ACM Journal of Data and Information Quality*, **11**(4).
URL `https://doi.org/10.1145/3317573`

[4] Pan, S. J. and Q. Yang (2010) "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, **22**(10), pp. 1345–1359.

[5] Lemley, J., S. Bazrafkan, and P. Corcoran (2017) "Smart Augmentation Learning an Optimal Data Augmentation Strategy," *IEEE Access*, pp. 5858–5869.

[6] Wang, J. and L. Perez (2017) "The Effectiveness of Data Augmentation in Image Classification using Deep Learning," .

[7] Piech, C., J. Spencer, J. Huang, S. Ganguli, M. Sahami, L. Guibas, J. Sohl-Dickstein, S. University, and K. Academy (2015) "Deep Knowledge Tracing," in *Advances in Neural Information Processing Systems*.

[8] Lee, Y., Y. Choi, J. Cho, A. R. Fabbri, H. Loh, C. Hwang, Y. Lee, S.-W. Kim, and D. Radev (2019) "Creating A Neural Pedagogical Agent by Jointly Learning to Review and Assess," in *arXiv preprint arXiv:1906.10910v2*.

[9] Zhang, J., X. Shi, I. King, and D.-Y. Yeung (2017) "Dynamic Key-Value Memory Networks for Knowledge Tracing," in *Int'l World Wide Web Conference Committee*.

[10] Pandey, S. and G. Karypis (2019) "A Self-Attentive model for Knowledge Tracing," in *Proc. Int'l Conf. on Educational Data Mining*.

[11] Shin, D., Y. Shim, H. Yu, S. Lee, B. Kim, and Y. Choi (2021) "SAINT+: Integrating Temporal Features for EdNet Correctness Prediction," in *Proc. Conf. Learning Analytics and Knowledge*.

[12] Wang, Y., K. Lin, Y. Qi, Q. Lian, S. Feng, Z. Wu, and G. Pan (2018) "Estimating brain connectivity with varying-length time lags using a recurrent neural network," *IEEE Transactions on Biomedical Engineering*, **65**(9), pp. 1953–1963.

[13] Lim, Y. B., I. Aliyu, and C. G. Lim (2019) "Air Pollution Matter Prediction Using Recurrent Neural Networks with Sequential Data," in *Proc. Int'l Conf. on Intelligent Systems, Metaheuristics & Swarm Intelligence*.

[14] Fung, P. L., M. A. Zaidan, O. Surakhi, S. Tarkoma, T. Petäjä, and T. Hussein (2021) "Data imputation in in situ-measured particle size distributions by means of neural networks," *Atmospheric Measurement Techniques*, **14**, pp. 5535–5554.

[15] Surakhi, O. M., M. A. Zaidan, S. Serhan, I. Salah, and T. Hussein (2020) "An optimal stacked ensemble deep learning model for predicting time-series data using a genetic algorithman application for aerosol particle number concentrations," *Computers*, **9**(4), pp. 1–26.

[16] Bouktif, S., A. Fiaz, A. Ouni, and M. A. Serhani (2018) "Optimal deep learning LSTM model for electric load forecasting using feature selection and genetic algorithm: Comparison with machine learning approaches," *Energies*, **11**(7).

[17] Ribeiro, G. H., P. S. De Neto, G. D. Cavalcanti, and I. R. Tsang (2011) "Lag selection for time series forecasting using Particle Swarm Optimization," *Proc. of the Int'l Joint Conference on Neural Networks*, pp. 2437–2444.

[18] Surakhi, O., M. A. Zaidan, P. L. Fung, N. H. Motlagh, S. Serhan, M. Alkhanafseh, R. M. Ghoniem, and T. Hussein (2021) "Time-lag selection for time-series forecasting using neural network and heuristic algorithm," *Electronics (Switzerland)*, **10**(20), pp. 1–22.

[19] Choi, Y., Y. Lee, J. Cho, J. Baek, B. Kim, Y. Cha, D. Shin, C. Bae, and J. Heo (2020) "Towards an Appropriate Query, Key, and Value Computation for Knowledge Tracing," in *Proc. Conf. Learning at Scale*.

[20] Choi, Y., Y. Lee, D. Shin, J. Cho, S. Park, S. Lee, J. Baek, C. Bae, B. Kim, and J. Heo (2020) "EdNet: A Large-Scale Hierarchical Dataset in Education," in *Proc. Int'l Conf. Artificial Intelligence on Education*.

[21] Wang, Z., A. Lamb, E. Saveliev, P. Cameron, Y. Zaykov, J. M. Hernández-Lobato, R. E. Turner, R. G. Baraniuk, C. Barton, S. P. Jones, S. Woodhead, and C. Zhang (2020) "Diagnostic Questions:The NeurIPS 2020 Education Challenge," in *Proc. Conf NeurIPS Competition Track*, pp. 1–27.

[22] Swamy, V., A. Guo, S. Lau, W. Wu, M. Wu, Z. Pardos, and D. Culler (2018) *Deep knowledge tracing for free-form student code progression*, vol. 10948 LNAI, Springer International Publishing.
URL http://dx.doi.org/10.1007/978-3-319-93846-2_65

[23] Dai, M., J.-L. Hung, X. Du, H. Tang, and H. Li (2021) "Knowledge Tracing: A view of Available Technologies," *Journal of Educational Technology Development and Exchange*, **14**(2).

[24] Roderick J. A. Little, D. B. R. (2002) *Statistical Analysis with Missing Data*.

[25] Kingma, D. P. and M. Welling (2019) *An introduction to variational autoencoders*, vol. 12.

[26] Le, T., S. Wang, and D. Lee (2020) "GRACE: Generating Concise and Informative Contrastive Sample to Explain Neural Network Model's Prediction," in *KDD*.
URL https://doi.org/10.1145/3394486.

[27] Fährmann, D., N. Damer, F. Kirchbuchner, A. Kuijper, M. Choras, and M. Pawlicki (2022) "Lightweight Long Short-Term Memory Variational Auto-Encoder for Multivariate Time Series Anomaly Detection in Industrial Control Systems," *Sensors*.
URL https://doi.org/10.3390/s22082886

[28] Ramchandran, S., G. Tikhonov, K. Kujanpää, M. Koskinen, and H. Lähdesmäki (2021) "Longitudinal Variational Autoencoder," in *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 130.
URL https://github.com/

[29] Pearlmutter, B. A. (1989) "Learning state space trajectories in recurrent neural networks," in *International JointConference on Neural Network*, pp. 365–372.

[30] Giles, C. L., G. M. Kuhn, and R. J. Williams (1994) "Neural Networks: Theory and Applications," in *IEEE Transactions on Neural Networks*, vol. 45, IEEE, pp. 89–90.

[31] Cheng, L., S. Ramchandran, T. Vatanen, N. Lietzén, R. Lahesmaa, A. Vehtari, and H. Lähdesmäki (2019) "An additive Gaussian process regression model for interpretable non-parametric analysis of longitudinal data," *Nature Communications*.
URL `https://doi.org/10.1038/s41467-019-09785-8`

[32] Yu, M., F. Xu, W. Hu, J. Sun, and G. Cervone (2021) "Using Long Short-Term Memory (LSTM) and Internet of Things (IoT) for localized surface temperature forecasting in an urban environment," .

[33] Du, Y., J. Wang, W. Feng, S. Pan, T. Qin, R. Xu, and C. Wang (2021) "AdaRNN: Adaptive Learning and Forecasting for Time Series," in *Proc. Int' Conf. Information and Knowledge Management*.

[34] Vaswani, A., G. Brain, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, . Kaiser, and I. Polosukhin (2017) "Attention Is All You Need," in *Proc.Conf. Neural Information Processing Systems*.

[35] Minn, S., Y. Yu, M. C. Desmarais, F. Zhu, and A. Vie (2018) "Deep Knowledge Tracing and Dynamic Student Classification for Knowledge Tracing," in *IEEE International Conference on Data Mining*.
URL `https://github.com/simon-tan/DKT-DSC.git`

[36] Pardos, Z. A. and N. T. Heffernan (2011) "KT-IDEM: Introducing Item Difficulty to the Knowledge Tracing Model," in *International Conference on User Modeling, Adaption and Personalization*.

[37] Corbetr, A. T. and J. R. Anderson (1995) "Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge," *User Modeling and User-Adapted Interaction*, **4**, pp. 253–278.

[38] Shimodaira, H. (2000) "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of Statistical Planning and Inference*, **90**(2), pp. 227–244.

[39] Patikorn, T., D. Deisadze, L. Grande, Z. Yu, and N. Heffernan (2019) "Generalizability of methods for imputing mathematical skills needed

to solve problems from texts," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, **11625 LNAI**, pp. 396–405.

[40] Dai, W., G.-R. Xue, Q. Yang, and Y. Yu (2007) "Transferring Naive Bayes Classifiers for Text Classification," in *Proc. Assoc. for the Advancement of Artificial Intelligence Conf. Artificial Intelligence*.

[41] Jiang, J. and C. Zhai (2007) "Instance Weighting for Domain Adaptation in NLP," in *Proc. Ann. Meeting of the Assoc. Computational Linguistics,*, Association for Computational Linguistics, pp. 264–271.

[42] Liao, X., Y. Xue, and L. Carin (2005) "Logistic Regression with an Auxiliary Data Source," in *Proc. Intl Conf. Machine Learning*.

[43] Huang, J., A. J. Smola, A. Gretton, K. M. Borgwardt, and B. Schölkopf (2007) "Correcting Sample Selection Bias by Unlabeled Data," in *Proc. Conf. Neural Information Processing Systems*.

[44] Bickel, S., M. Brückner, and T. Scheffer (2007) "Discriminative Learning for Differing Training and Test Distributions," in *Proc. Int'l Conf. Machine Learning*.

[45] Sugiyama, M., S. Nakajima, H. Kashima, P. Von B, and M. Kawanabe (2008) "Direct Importance Estimation with Model Selection and Its Application to Covariate Shift Adaptation," in *Proc. Conf. Neural Information Processing Systems,*.

[46] Fan, W., I. Davidson, B. Zadrozny, and P. S. Yu (2005) "An Improved Categorization of Classifier's Sensitivity on Sample Selection Bias," in *IEEE Int'l Conf. Data Mining*.

[47] Raina, R., A. Battle, H. Lee, B. Packer, and A. Y. Ng (2007) "Self-taught Learning: Transfer Learning from Unlabeled Data," in *Proc. Intl Conf. Machine Learning*.

[48] Dai, W., G.-R. Xue, Q. Yang, and Y. Yu (2007) "Co-clustering based Classification for Out-of-domain Documents," in *Proc. ACM SIKDD Int'l Conf. Knowledge Discovery and Data Mining*.

[49] Blitzer, J., R. Mcdonald, and F. Pereira (2006) "Domain Adaptation with Structural Correspondence Learning," in *Proc. Conf. Empirical Methods in Natural Language*, pp. 120–128.

[50] Argyriou, A., T. Evgeniou, and M. Pontil (2007) "Multi-Task Feature Learning," in *Proc. Conf. Neural Information Processing Systems*.

[51] Argyriou, A., C. A. Micchelli, M. Pontil, and Y. Ying (2008) "A Spectral Regularization Framework for Multi-Task Structure Learning," in *Proc. Conf. Neural Information Processing Systems*.

[52] Wang, C. and S. Mahadevan (2008) "Manifold Alignment using Procrustes Analysis," in *Proc. Intl Conf. Machine Learning*.

[53] Bonilla, E. V., M. A. Chai, and C. K. I. Williams (2008) "Multi-task Gaussian Process Prediction," in *Proc. Conf. Neural Information Processing Systems*.

[54] Schwaighofer, A., V. Tresp, and K. Yu (2005) "Learning Gaussian Process Kernels via Hierarchical Bayes," in *Proc. 17th Ann. Conf. Neural Information Processing Systems*.

[55] Gao, J., W. Fan, J. Jiang, and J. Han (2008) "Knowledge Transfer via Multiple Model Local Structure Mapping," in *Proc. SIGKDD Intl Conf. Knowledge Discovery and Data Mining*.

[56] Mihalkova, L. and R. J. Mooney (2008) "Transfer Learning by Mapping with Minimal Target Data," in *Proc. Assoc. for the Advancement of Artificial Intelligence Workshop Transfer Learning for Complex Tasks*.

[57] Davis, J. and P. Domingos (2008) "Deep Transfer via Second-Order Markov Logic," in *Proc. Assoc. for the Advancement of Artificial Intelligence Workshop Transfer Learning for Complex Tasks*.

[58] Uguroglu, S. and J. Carbonell (2011) "Feature selection for transfer learning," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, **6913 LNAI**(PART 3), pp. 430–442.

[59] corestandards.org *Coding the Common Core State Standards (CCSS)*, *Tech. rep.*
URL `http://www.corestandards.org/wp-content/uploads/Math_Standards.pdf`

[60] Karlovčec, M., M. Córdova-Sánchez, and Z. A. Pardos (2012) "Knowledge component suggestion for untagged content in an intelligent tutoring system," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, **7315 LNCS**, pp. 195–200.

[61] Desmarais, M. C. (2012) "Mapping Question Items to Skills with Non-negative Matrix Factorization," *ACM SIGKDD Explorations Newsletter*, **13**(2).

[62] Pardos, Z. A. (2017) "Imputing KCs with Representations of Problem Content and Context," in *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, pp. 148–155.

[63] Rosé, C., P. Donmez, G. Gweon, A. Knight, B. Junker, W. Cohen, K. Koedinger, and N. Heffernan (2005) "Automatic and Semi-Automatic Skill Coding With a View Towards Supporting On-Line Assessment," in *Proceedings of the conference on Artificial Intelligence in Education*, pp. 571–578.

[64] Devlin, J., M. W. Chang, K. Lee, and K. Toutanova (2019) "BERT: Pre-training of deep bidirectional transformers for language understanding," in *The Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, pp. 4171–4186.

[65] Birenbaum, M., A. E. Kelly, and K. K. Tatsuoka (1993) "Diagnosing Knowledge States in Algebra Using the Rule-Space Model," *Journal for Research in Mathematics Education*, **24**(5), pp. 442–459.

[66] Peters, M. E., M. Neumann, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer (2018) "Deep contextualized word representations," in *Proceedings of NAACL-HLT*, pp. 2227–2237.

[67] Howard, J. and S. Ruder (2018) "Universal Language Model Fine-tuning for Text Classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pp. 328–339.

[68] Sun, C., X. Qiu, Y. Xu, and X. Huang (2019) "How to Fine-Tune BERT for Text Classification?" *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, **11856 LNAI**(2), pp. 194–206.

[69] Liu, Z., D. Huang, K. Huang, Z. Li, and J. Zhao (2020) "FinBERT: A Pre-trained Financial Language Representation Model for Financial Text Mining," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence Special Track on AI in FinTech*.

[70] Alsentzer, E., J. R. Murphy, W. Boag, W.-H. Weng, D. Jin, T. Naumann, and M. B. A. Mcdermott (2019) "Publicly Available Clinical BERT Embeddings," in *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pp. 72–78.

[71] Lee, J., W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang (2020) "Data and text mining BioBERT: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, p. 12341240.

[72] Beltagy, I., K. Lo, and A. Cohan (2019) "SCIBERT: A pretrained language model for scientific text," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*, pp. 3615–3620.

[73] Zhang, D., Z. Yuan, Y. Liu, Z. Fu, F. Zhuang, P. Wang, H. Chen, and H. Xiong (2020) "E-BERT: A Phrase and Product Knowledge Enhanced Language Model for E-commerce," in *arXiv preprint arXiv:2009.02835v2*.

[74] Gururangan, S., A. Marasoviťcmarasoviťc, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, N. A. Smith, and Allen (2020) "Don't Stop Pretraining: Adapt Language Models to Domains and Tasks," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

[75] Choi, M., S. Lee, E. Choi, H. Park, J. Lee, D. Lee, and J. Lee (2021) "MelBERT : Metaphor Detection via Contextualized Late Interaction using Metaphorical Identification Theories," in *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

[76] Le, Q. and T. Mikolov (2014) "Distributed Representations of Sentences and Documents," in *Proceedings of the 31st International Conference on International Conference on Machine Learning*, pp. II–1188II–1196.

[77] Zimba, J. (2012) *A Graph of the Content Standards*, *Tech. rep.*
URL `https://achievethecore.org/page/844/a-graph-of-the-content-standards`

[78] UnboundEd (2017) *A Coherence Map for the High School Standards*, *Tech. rep.*
URL `https://www.unbounded.org/other/8610`

[79] Grover, A. and J. Leskovec (2016) "Node2vec: Scalable feature learning for networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 13-17, pp. 855–864.

[80] Landis, J. R. and G. G. Koch (1977) "The Measurement of Observer Agreement for Categorical Data," *Biometrics*, **33**(1), p. 159.

[81] Shen, J. T., M. Yamashita, E. Prihar, N. Heffernan, X. Wu, S. Mcgrew, and D. Lee (2021) "Classifying Math Knowledge Components via Task-Adaptive Pre-Trained BERT," in *Proceedings of the Conference on Artificial Intelligence in Education*.

[82] Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, and P. G. Allen (2019) "RoBERTa: A Robustly Optimized BERT Pretraining Approach," in *arXiv preprint arXiv:1907.11692v1*.

[83] Huang, K. and J. Altosaar "ClinicalBert: Modeling Clinical Notes and Predicting Hospital Readmission," in *arXiv preprint arXiv:1904.05342v2*.

[84] Guo, W., X. Liu, S. Wang, H. Gao, A. Sankar, Z. Yang, Q. Guo, L. Zhang, B. Long, B.-C. Chen, and D. Agarwal (2020) "DeText: A Deep Text Ranking Framework with BERT," in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*.

[85] Erickson, J. A., A. F. Botelho, S. Mcateer, A. Varatharaj, and N. T. Heffernan (2020) "The Automated Grading of Student Open Responses in Mathematics ACM Reference Format," in *Proceedings of the 10th Learning Analytics and Knowledge Conference*.

[86] Baral, S., A. F. Botelho, J. A. Erickson, and N. T. Heffernan (2021) "Improving Automated Scoring of Student Open Responses in Mathematics," in *Educational Data Mining*.

[87] Uzen, N. S., A. N. Gorban, J. Levesley, and E. M. Mirkes (2019) "AUTOMATIC SHORT ANSWER GRADING AND FEEDBACK USING TEXT MINING METHODS," *Procedia Computer Science*.

[88] Lan, A. S., D. Vats, A. E. Waters, and R. G. Baraniuk (2015) "Mathematical Language Processing: Automatic Grading and Feedback for Open Response Mathematical Questions," in *Proceedings of the Second ACM Conference on Learning @Scale*, pp. 167–176.
URL `http://dx.doi.org/10.1145/2724660.2724664`

[89] Thai-Nghe, N., L. Drumond, A. Krohn-Grimberghe, and L. Schmidt-Thieme (2010) "Recommender system for predicting student performance," *Procedia Computer Science*, **1**(2), pp. 2811–2819.

[90] Liu, Q., Z. Huang, Y. Yin, E. Chen, H. Xiong, Y. Su, and G. Hu (2019) "EKT: Exercise-aware Knowledge Tracing for Student Performance Prediction," in *IEEE Transactions on Knowledge and Data Engineering*.

[91] Peng, S., K. Yuan, L. Gao, and Z. Tang "MathBERT: A Pre-Trained Model for Mathematical Formula Understanding," in *arXiv preprint arXiv:2105.00377v1*.

[92] Reimers, N. and I. Gurevych (2019) "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 3982–3992.

[93] Wang, Y. and N. T. Heffernan (2014) "The effect of automatic reassessment and relearning on assessing student long-term knowledge in mathematics," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, **8474 LNCS**, pp. 490–495.

[94] Cubuk, E. D., B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le (2019) "Autoaugment: Learning augmentation strategies from data," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, **2019-June**(Section 3), pp. 113–123.

Vita

Jia Tracy Shen

Work and Education

- **2004-2008** B.A. in English at Zhejiang University, Hangzhou, China
- **2008-2010** Management Trainee at Baxter International, Shanghai,China
- **2010-2012** M.A. in International Affairs at Penn State University, State College, PA
- **2013-2014** Research Associate at GFK North America , New York, NY
- **2014-2018** Business Intelligence Analyst at AccuWeather,Inc, State Colelge, PA
- **2018-2019** Data Scientist at AccuWeather, Inc, State College, PA
- **2019-Now** Data Scientist at K12.com, Remote
- **2018-2023** Ph.D in Information Sciences and Technology at Penn State University, State College, PA

Field of Study

- Machine Learning (Deep Learning) & AI with a domain focus in Education
- Develop Large Language Models to conduct Natural Language Understanding
- Develop Deep Generative Models VAE-basedto generate time series data to boost performances of deep times series models such as LSTM, GRU, Transformer
- Conduct Transfer learning to boost Knowledge Generalization

Publication

- **Shen, Jia Tracy**, et al. (2023). Imputing Knowledge Tracing Data with Subject-Based Training via LSTM Variational Autoencoders Frameworks. AI4ED Workshop of Association for the Advancement of Artificial Intelligence (**AAAI 2023 W**).

- **Shen, Jia Tracy**, et al. MathBERT: A Pre-trained Language Model for General NLP Tasks in Mathematics Education. Proceedings of the Conference on Neural Information Processing Systems (**NeurIPS 2021 W**)
- **Shen, Jia Tracy**, et al. Classifying Math Knowledge Components via Task-Adaptive Pre-Trained BERT. Proceedings of the Conference on Artificial Intelligence in Education (**AIED2021**)