Empirical Evaluation of Deep Learning Models for Knowledge Tracing: Of Hyperparameters and Metrics on Performance and Replicability

Sami Sarsa
Aalto University
sami.sarsa@aalto.fi

Juho Leinonen
Aalto University
juho.2.leinonen@aalto.fi

Arto Hellas
Aalto University
arto.hellas@aalto.fi

New knowledge tracing models are continuously being proposed, even at a pace where state-of-the-art models cannot be compared with each other at the time of publication. This leads to a situation where ranking models is hard, and the underlying reasons of the models' performance – be it architectural choices, hyperparameter tuning, performance metrics, or data – is often underexplored. In this work, we review and evaluate a body of deep learning knowledge tracing (DLKT) models with openly available and widely-used data sets, and with a novel data set of students learning to program. The evaluated knowledge tracing models include Vanilla-DKT, two Long Short-Term Memory Deep Knowledge Tracing (LSTM-DKT) variants, two Dynamic Key-Value Memory Network (DKVMN) variants, and Self-Attentive Knowledge Tracing (SAKT). As baselines, we evaluate simple non-learning models, logistic regression and Bayesian Knowledge Tracing (BKT). To evaluate how different aspects of DLKT models influence model performance, we test input and output layer variations found in the compared models that are independent of the main architectures. We study maximum attempt count options, including filtering out long attempt sequences, that have been implicitly and explicitly used in prior studies. We contrast the observed performance variations against variations from non-model properties such as randomness and hardware. Performance of models is assessed using multiple metrics, whereby we also contrast the impact of the choice of metric on model performance. The key contributions of this work are the following: Evidence that DLKT models generally outperform more traditional models, but not necessarily by much and not always; Evidence that even simple baselines with little to no predictive value may outperform DLKT models, especially in terms of accuracy – highlighting importance of selecting proper baselines for comparison; Disambiguation of properties that lead to better performance in DLKT models including metric choice, input and output layer variations, common hyperparameters, random seeding and hardware; Discussion of issues in replicability when evaluating DLKT models, including discrepancies in prior reported results and methodology. Model implementations, evaluation code, and data are published as a part of this work.

Keywords: knowledge tracing, deep learning, memory networks, attention-based models, hyperparameter optimization, evaluation metrics, replicability

1. INTRODUCTION

Knowledge tracing (KT) is a student modeling task where knowledge or skill is estimated based on a trace of interactions with learning activities, such as course exercises. While new knowledge tracing models and modifications of existing ones are presented regularly (Yudelson et al., 2013; Piech et al., 2015; Zhang et al., 2017; Yeung and Yeung, 2018; Abdelrahman and Wang, 2019; Nakagawa et al., 2019; Pu et al., 2020), it is not always clear what the actual factors that contribute to the performance of the proposed models are (Wilson et al., 2016; Lipton and Steinhardt, 2019). To what extent do the results depend on the proposed model algorithms and the used data, or are optimizations the actual key to the results?

When working with deep learning models – machine learning models that use multiple processing layers for transforming an input to an output – the resulting models are often not transparent and hence difficult to interpret. Substantial performance differences can be observed already due to the choice of hyperparameters that are used to train the models (Bouthillier et al., 2021). Even the used hardware and software can influence the outcomes; for example, the documentation of PyTorch – an open source machine learning library – warns that "*Completely reproducible results are not guaranteed across PyTorch releases, individual commits, or different platforms. Furthermore, results may not be reproducible between CPU and GPU executions, even when using identical seeds*"[1]. Deeper insight into the factors that contribute to differences in performance between models can be gained through reproduction, replication and ablation. Since the terms reproduction and replication are often used interchangeably, in this work by *reproducibility* we denote using the same data and methodology – including code – by different researchers, and by *replicability* we denote using the same methodology but re-implementing the work and with different data and researchers, as is done in previous work (Patil et al., 2016; Stevens, 2017). Although the need for replication and reproduction has been highlighted within the educational data mining domain (Ihantola et al., 2015; Gardner et al., 2019), such studies are still scarce.

In our work, we replicate earlier studies, reimplementing models ourselves following descriptions available in the original articles. We review and evaluate a body of KT model algorithms with a focus on KT algorithms that utilize Deep Learning Knowledge Tracing (hereafter DLKT). As baselines, we use both non-learning models and Bayesian Knowledge Tracing and a recent Logistic Regression -based model. We triangulate factors that contribute to the performance of the models, including metric choice, input and output variations in model structure, non-model properties (e.g. maximum input sequence lengths, random seeding and hardware) and commonly used hyperparameters. The evaluations are conducted with seven datasets – six open ones and a novel dataset made openly available as a part of this study – and seven metrics, with the purpose of identifying and discussing performance differences between the models, the datasets, and the metrics. Our research questions for this study are as follows.

RQ1 How do DLKT models compare to naive baselines and non deep-learning KT models?

RQ2 How do DLKT models perform on the same and different datasets as originally evaluated with?

RQ3 What is the impact of variations in architecture and hyperparameters on DLKT models' performance?

[1] https://pytorch.org/docs/stable/notes/randomness.html, accessed 2020-12-01

2. BACKGROUND

2.1. KNOWLEDGE TRACING

Knowledge tracing (Corbett and Anderson, 1994) is an approach for student modeling in which students' actions within a learning environment are used for estimating their current knowledge with regard to each individual knowledge component[2]. The estimates of current knowledge are typically probabilistic, where mastery is assumed when the models posit at least a 95% probability that the student has mastered the knowledge component.

In practice, most if not all of the knowledge tracing algorithms are used to create regression models, which are used for estimating relationships between a dependent variable (in our context, often knowledge) and one or more independent variables (in our context, often variables describing students' behavior or outcomes from the learning environment). One particular family of regression models that has been used in educational data mining is logistic regression models, which are often used for estimating the probability of a given class (e.g. has mastered the topic / has not mastered the topic) based on a set of independent variables.

Knowledge tracing is often used in intelligent tutoring systems (ITS) that provide personalized tutoring to students (Anderson et al., 1985; Nwana, 1990; VanLehn, 2011; Ma et al., 2014). ITSs have been shown to even outperform teacher-led large-group instruction, computer-based instruction, and the use of textbooks or workbooks (Ma et al., 2014). Recent research has suggested that intelligent tutoring systems can be nearly as effective as human tutors (VanLehn, 2011; Ma et al., 2014).

Here, we revisit models for knowledge tracing. The topics include probabilistic graphical models, factor analysis models, and deep learning models.

2.1.1. Probabilistic Graphical Models

Probabilistic graphical models are probabilistic models that use graphs for presenting the conditional dependencies between the studied variables. The most prominent probabilistic graphical model for knowledge tracing is Bayesian Knowledge Tracing (Corbett and Anderson, 1994) (BKT[3]). BKT assumes that knowledge is binary, i.e. that a student has either mastered or not mastered the topic, and furthermore, that once a topic has been mastered, it cannot be unlearned or forgotten. Interactions with the task environment are related to individual knowledge components, and the interactions may produce an output, which is observable evidence that indicates whether the student's action was correct or incorrect.

Whenever the system produces an output based on a student's actions, which represents either correct or incorrect knowledge, the student's knowledge of the knowledge component is updated. In BKT, the update is based on four parameters (described more formally in 3.3), which define the probability that the student had already learned the knowledge component (prior learning), the probability that the student's action led to the student learning the knowledge component (transition), the probability that the student's interaction with the task environment produced a correct output by accident (guess), and the probability that the student's interaction

[2]While the original article (Corbett and Anderson, 1994) uses the term *rule* for describing the concepts and objectives that the student is expected to learn, we refer here to the term as *knowledge component* as it is more commonly used in contemporary research on the topic.

[3]The original article uses the term Knowledge Tracing for the approach. Over time, the approach has been relabeled as Bayesian Knowledge Tracing by the scientific community.

with the task environment produced an incorrect output by accident (slip). In the original article, data from all students were used to estimate optimal values for the four parameters.

A range of BKT variations has been published since the original article. For example, researchers have added information on whether and how the student was helped during the process (Chang et al., 2006; Lin and Chi, 2016), adjusted the probability of guessing or slipping using additional factors (Baker et al., 2008), adjusted the prior learning and transition parameters based on data from each individual student instead of all students (Yudelson et al., 2013), and included information on the task difficulty to the process (Pardos and Heffernan, 2011).

In addition to the BKT variants, there exists other approaches based on probabilistic graphical model for KT. For example, Partially Observable Markov Decision Processes (Rafferty et al., 2011) have been used to model student behavior. Similarly, methods such as (Dynamic) Bayesian Networks (Pearl, 1985; Pearl, 1988; Ghahramani, 1997) have been applied, e.g., for estimating student's knowledge (Rowe and Lester, 2010), for extending BKT to allow multiple knowledge components (Pardos et al., 2008), for modeling dependencies between knowledge components (Käser et al., 2014), and for adjusting the BKT prior learning parameter based on additional factors (Pardos and Heffernan, 2010).

2.1.2. Factor Analysis Based Models

Factor analysis is a method for reducing the amount of observed variables into a potentially lower number of unobserved variables called factors. The methods briefly described next, Item Response Theory, Learning Factors analysis, and Performance Factors Analysis all use existing data to estimate one or more latent (unobserved) variables.

Item Response Theory (IRT) (Hambleton and Swaminathan, 1985), from the field of psychometrics, is used for assessing student's ability and item difficulty based on students' responses to, e.g., questionnaire items. As IRT models can be used to produce estimates of item difficulty and student ability, they have been used for estimating the probability with which students will answer questions correctly (Johns et al., 2006). However, the basic IRT model does not provide means for dynamically updating students' knowledge, and thus, the model has been incorporated with knowledge tracing, resulting in models that outperform BKT (Khajah et al., 2014; Khajah et al., 2014; González-Brenes et al., 2014). In addition, recent work on logistic regression models incorporating more aspects from students' behavior has been shown to also outperform DLKT models (Gervet et al., 2020).

Other variants purposed for the knowledge tracing task are Learning Factors Analysis (Cen et al., 2006) (LFA) and Performance Factors Analysis (Pavlik et al., 2009) (PFA). LFA is comprised of three parts, which are (1) a parameter that quantifies student's ability, (2) difficulty of the knowledge components defined through a single parameter for each knowledge component, and (3) learning rate or benefit of frequency of prior practice for each knowledge component. Similar to the basic IRT model, LFA has downsides in how it can be used for adaptive environments as it does not account for the correct or incorrect outputs from the task environment, i.e. correct and incorrect responses by the student. PFA accounts for this by reconfiguring the model by making it more sensitive to student performance, and providing better means to adapt to the student's performance. Literature has provided evidence of PFA outperforming both LFA and BKT (Pavlik et al., 2009), although there also exists evidence on BKT performing on par with PFA (Gong et al., 2010). In addition, it has been observed that, in particular tasks, PFA performs almost on par with a DLKT model (Xiong et al., 2016), which we discuss next.

2.1.3. Deep Learning Models for Knowledge Tracing

Deep learning models are a set of machine learning models that are based on (artificial) neural networks. Neural networks are networks of "neurons" organized in layers that are used as information processing systems for e.g. classification tasks. Neurons in the neural networks are mathematical functions that take a fixed sized set, i.e. a vector, of inputs, have a weight (learnable parameter) for each input, and produce an output as a weighted sum of the inputs. The output is then processed using an activation function (e.g. a sigmoid function).

The first layer of a neural network is its input, i.e. the data that is fed to the network. A single layer may contain an arbitrary amount of neurons and a neural network can consist of one or more layers of neurons without an upper bound. Complex neural networks, i.e., ones that are deeply layered, are often called deep learning models or deep networks. In contrast, neural networks with a relatively simple layer structure with few layers are sometimes referred to as shallow networks (Bianchini and Scarselli, 2014). The most common type of layer in a neural network, one that is used commonly in the models explored in this paper, is the fully connected layer (FCL), wherein each neuron (and weight) of the layer is connected to each of its inputs.

Training a neural network consists of providing training data to the network and adjusting the weights of the neurons so that the overall network learns to produce a desired output. During learning, the performance of the model is estimated using an error function that compares the output of the model with the expected output, i.e. true outcome. Neural networks often require extensive amounts of training data in order to produce generalizable results. This is especially true when the training data is complex, such as when it contains sequential relations, e.g. temporality or word order. Thus, adjustments to neural networks have been made to better conform to certain kinds of data relations. The Recurrent Neural Network (RNN) (Hochreiter and Schmidhuber, 1997a) is a result of one such adjustment that has been widely adopted. In an RNN, connections between neurons are constructed as a directed graph that can represent a temporal sequence, and where the output of each neuron can be fed back to the network (similar to recursion). RNNs have been shown to work well in various domains, including natural language processing (Mikolov et al., 2011), stock market prediction (Kamijo and Tanigawa, 1990; Saad et al., 1998), and protein sequence prediction (Saha and Raghava, 2006)

RNNs were first used for the Knowledge Tracing task in a study by Piech et al. (Piech et al., 2015). They showed that "Deep Knowledge Tracing" (DKT) outperformed BKT, despite modeling all the skills jointly instead of building a separate model for each skill. Since then, DKT has been shown to perform well in a range of studies (Wilson et al., 2016; Lin and Chi, 2017; Mao et al., 2018; Montero et al., 2018; Ding and Larson, 2019). This has also led to a range of DLKT-based approaches that have been proposed and evaluated for the Knowledge Tracing task. Their contributions include both proposing new model algorithms by introducing techniques from the broader machine learning domain and also how to leverage different sorts of input into the models. For example, Zhang et al. (Zhang et al., 2017) introduce DKVMN, a memory network that utilizes next skill input when predicting next attempt correctness, that is not present in the original DKT model, and Abderahman et al. (Abdelrahman and Wang, 2019) further propose use of memory network architecture in tandem with LSTMs in their SKVMN model. As a more input oriented example, The models by Su et al. (Su et al., 2018) (EERNN) and Liu et al. (Liu et al., 2019) (EKT) utilize textual content of exercises. Pandey et al. (Pandey and Karypis, 2019) started a line of research on self-attentive models known as transformers for knowledge tracing with their SAKT model. Further attention-based models include RKT (Pandey and Srivastava,

2020), context-aware AKT (Ghosh et al., 2020) that fuses DLKT with IRT, SAINT (Choi et al., 2020) with an encoder-decoder architecture, time-aware encoder-decoder by Pu et al. (Pu et al., 2020), and a time-aware version of SAINT called SAINT+ (Shin et al., 2021). Another line of development has concentrated on graph neural networks with models such as GKT (Nakagawa et al., 2019) that models learning of skills as a network and JKT (Song et al., 2021) which takes a step further by modeling both skill and exercise relations. Many of these models have been introduced since this study was started, which highlights the rapid development and popularity of the field, but also the need for high quality methodology and replication work for fair comparisons and disambiguation of causes for performance differences.

2.2. Replicating Work from Others

Discussions on the importance of replicating work from others have stemmed from observations that published results do not always hold up under scrutiny (Ioannidis, 2005b; Moonesinghe et al., 2007). Researchers from various disciplines have voiced out their concerns about the state of replicability in their fields, which include, for example, biology (Begley and Ellis, 2012), computing education (Ahadi et al., 2016), educational data mining (Gardner et al., 2019), health care (Ioannidis, 2005a), political science (Golden, 1995) and computational science (Peng, 2011). This concern was highlighted also in a multi-disciplinary survey of over 1500 researchers, where about three quarters of the respondents considered that they could only trust *at least half* of the papers in their field (Baker, 2016). Whether published results hold may depend also on the field of science; for example, one study from psychology replicated 100 experimental and correlational studies from three journals and found that only about one-third to one-half of the original findings could be replicated (Open Science Collaboration, 2015), raising discussion about how to conduct replication studies (Gilbert et al., 2016; Anderson et al., 2016).

Despite the need for replication studies, studies that attempt to replicate earlier findings remain relatively rare (Muma, 1993; Makel et al., 2012). This rarity has been linked with valuing innovation and original research (Mackey, 2012; Asendorpf et al., 2013; Ahadi et al., 2016) and the concern that replications may not be publishable (Fanelli, 2011; Spellman, 2012; Ahadi et al., 2016). Both of these issues may steer researchers away from conducting and publishing replication studies. In the survey of over 1500 researchers, where relative lack of trust towards earlier results was highlighted as one of the issues, over half of the surveyed researchers had been unable to replicate an experiment from others (Baker, 2016), suggesting that there is a dire need of sufficient details for replication as well as a need to maintain quality. Indeed, calls for more detailed reporting of experimental designs, requiring publication of data and research materials, and rewarding replication attempts have been made (Asendorpf et al., 2013; Ioannidis et al., 2014).

With replication studies, one could – for example – identify issues with how methodological details are presented and consequently also highlight the importance of replication to the community (Ihantola et al., 2015; Gardner et al., 2019). By reimplementing algorithms as a part of the replication studies, one may also reveal bugs in existing research code, that could lead researchers into faulty conclusions (Bhandari Neupane et al., 2019). Although DLKT models often outperform older models, the relative performance of evaluated models may be influenced, among other factors, by the context, the task, the data, the hyperparameter tuning and the chosen metrics (Xiong et al., 2016; Lalwani and Agrawal, 2017; Mao et al., 2018; Ding and Larson, 2019). The relative impact of these factors could be revealed through replication studies.

2.3. Effects of Metrics on Reported Performance

Metrics for evaluating model performance have been extensively studied and developed, and different metrics are used to evaluate different aspects of model performance. Relying on any single metric alone is prone to provide misleading information on model performance (National Research Council and Climate Research Committee, 2005). Plenty of discussion has concentrated on the goodness of accuracy and (ROC-)AUC (Receiving Operator Characteristic Area Under Curve) as metrics, where some scrutinize the former (Ling et al., 2003; Halimu et al., 2019) as misleading and others scrutinize the latter (Jeni et al., 2013; Muschelli, 2020) as potentially masking bad performance. Both of these are popular metrics in the Knowledge Tracing field as well as other domains (Pahikkala et al., 2009; Huang et al., 2019).

Within the field of educational data mining, some studies have been conducted on the effect of metric choice on model performance, although on different metrics and not so much on accuracy and AUC. As an example, Pelanek at al. (Pelánek, 2015) compare MAE (Mean Average Error, RMSE (Root Mean Squared Error), LL (Log-likelihood aka binary cross-entropy) and AUC on BKT, PFA and Elo rating system adaptation (Pelánek, 2014) models. A rather recent study by Effenberger et al. (Effenberger and Pelánek, 2020) compares the highly similar metrics MAE and RMSE for multiple models for student modeling, including the mentioned Elo adaption, an IRT model and Random Forest, and show that metric choice affects model performance and can even affect model ranking. They also highlight other issues in methodological choices such as filtering choices of data in preprocessing.

2.4. Recent Comparisons of Deep Learning Models for Knowledge Tracing

There are signs that the popularity of replication studies in educational data mining is on the rise. This has been especially visible within the Educational Data Mining community, where recent calls for papers at e.g. the Educational Data Mining conference have included reproducibility of research as a topic of interest[4].

A recent article by Gervet et al. (Gervet et al., 2020) compared different models that have been used for KT. Their evaluated models included (1) different DLKT models, i.e. DKT (Piech et al., 2015), SAKT (Pandey and Karypis, 2019) and a feedforward network; (2) regression models, i.e. IRT, PFA (Pavlik et al., 2009), DAS3H (Choffin et al., 2019) and a logistic regression model (LR); and (3) a variation of BKT called BKT+ (Khajah et al., 2016) that adds individualization, forgetting and discovery of knowledge components. Additionally, they conducted ablation studies to examine which features of different models could explain differences in their performance. The metrics that were used to compare the models were AUC score and RMSE, and the study used nine different datasets. The results showed that DKT and logistic regression performed the best, with DKT being the best model in five datasets and LR being the best in four. SAKT underperformed DKT in all of the datasets, which is contrary to prior results by Pandey and Karypis (Pandey and Karypis, 2019), where SAKT performed significantly better compared to DKT. The results by Gervet et al. also suggest that dataset size affects model performance: LR worked better for smaller datasets and DKT performed better for larger ones. Additionally, DKT seemed to reach peak performance faster when compared to LR, for which

[4]See e.g. https://educationaldatamining.org/edm2021/call-for-papers/, accessed 2020-10-15

the performance continues to improve over a longer time.

In a similar study, Pandey et al. (Pandey et al., 2021) compared DLKT models. They included DKT (Piech et al., 2015), DKVMN (Zhang et al., 2017), SAKT (Pandey and Karypis, 2019) and RKT (Pandey and Srivastava, 2020) in their comparison. The models were compared using the EdNet dataset (Choi et al., 2020) by comparing accuracies and AUCs. The results of Pandey et al. suggest that the self-attention based models (RKT and SAKT) perform better compared to DKT and DKVMN.

Similar to Pandey et al. (Pandey et al., 2021), Mandalapu et al. (Mandalapu et al., 2021) used the EdNet dataset (Choi et al., 2020) to compare different knowledge tracing models. They used two versions of the EdNet dataset: the full data with 600,000 students and a pruned dataset with 50,000 students. They evaluated a baseline model (prediction based on probability of correctness in training set), a logistic regression model, DKT (Piech et al., 2015), and SAKT (Pandey and Karypis, 2019). The models were compared with the AUC metric. Based on the results of Mandalapu et al. (Mandalapu et al., 2021), the logistic regression model very slightly outperformed DKT and SAKT (0.77 AUC for LR vs 0.76 AUC for DKT and SAKT) in the smaller 50,000 student dataset. In the full dataset, SAKT outperformed DKT (0.76 vs 0.72 AUC), while the logistic regression model could not be trained with the full dataset. Contrary to Gervet et al. (Gervet et al., 2020), Mandalapu et al. (Mandalapu et al., 2021) found that DKT's performance was worse with a larger dataset. Mandalapu et al. hypothesized that SAKT benefits from a larger dataset size. Interestingly, out of the two models present in both Pandey et al.'s and Mandalapu et al.'s studies – SAKT and DKT –, SAKT performed approximately similarly in both Mandalapu et al.'s and Pandey et al.'s studies achieving around 0.76 AUC, while there was a difference in the performance of DKT, for which Mandalapu et al. got an AUC score of 0.72 and Pandey et al. an AUC score of around 0.742. Since the model and the data are the same, the difference of around 0.022 AUC is likely due to differences in hyperparameter tuning or the used hardware; in the present study, we also explore the effects of hyperparameter tuning and hardware.

What is evident from all the comparisons is that the results of comparing different models are affected by many factors. Interestingly, the authors of prior comparison studies mostly relied on AUC scores to evaluate the models: AUC is the only performance metric present in all three of (Gervet et al., 2020), (Pandey et al., 2021), and (Mandalapu et al., 2021), despite the possibility of models performing differently with different metrics (Caruana and Niculescu-Mizil, 2004; National Research Council and Climate Research Committee, 2005; Gunawardana and Shani, 2009; Sanyal et al., 2020). In addition, the differences between the best few models in the three comparison studies are not particularly large: the AUC scores between the best two models, for example, are typically within 0.02 AUC of each other. Further, when comparing the performance of models, relying on a single metric can be misleading (Caruana and Niculescu-Mizil, 2004; National Research Council and Climate Research Committee, 2005) and it is important to understand what the metric is measuring.

Previous research comparing multiple models have rarely studied the impact of architectural variations with a positive exception of Gervet et al. (Gervet et al., 2020). When new models are introduced, it is also unclear whether the models that are used for comparing the new model's performance are tuned with similar rigor as the proposed new model. From the replication perspective, seeking to understand the underlying reasons for the performance increases is rare. One example where this is conducted well is the study by Khajah et al. (Khajah et al., 2016) who explored introducing new parameters to BKT and observed performance increases similar

to DLKT while keeping the model more explainable.

3. Evaluated Models

3.1. Overview

We inspect three naive models, two commonly used baseline models, and three deep learning knowledge tracing (DLKT) models. We also include three variants of the DLKT models that we report as separate models; one of these is a novel variant created for the purposes of this study. The models and their respective abbreviations are summarized in Table 1.

The models included in this study were chosen based on recentness and gained traction at the beginning of this work in late 2019, where each selected model reported state-of-the-art results and possibly also conflicting results for the previous state-of-the-art; the included DLKT models have also all been used in more recent works that build on existing knowledge tracing models (Liu et al., 2019; Trifa et al., 2019; Oya and Morishima, 2021). The naive models were selected to provide a more comprehensive analysis of how different results can be achieved even with simpler methods as well as to provide validity of the machine learning models' usefulness, while BKT and a logistic regression model were chosen as baselines to include non-DLKT models for comparison. We note that modern variants of BKT may provide significantly better results than the baseline we use[5], which includes an individualized BKT (Yudelson et al., 2013) variant.

For our logistic regression baseline we use the Best-LR[6] (GLR) model that builds on PFA, IRT and DAS3H, and was the best performing logistic regression model in the recent study by Gervet et al. (Gervet et al., 2020).

All the evaluated models, outlined in Table 1, accept inputs as sequences of exercise attempts per student and output the probability of next attempt correctness for each attempt in the input. An attempt consists of a skill id s_t and correctness c_t at time step t, where time is an increasing integer sequence. The common variables used in the subsequent descriptions of the models are summarized in Table 2.

Next, we explore the architectures of the studied models with some mathematical detail to portray a concrete picture of the differences between the models. We mainly follow the mathematical notation from the original articles. Some differences to the notation are introduced for instance to maintain consistency of variable definitions across the model descriptions in this work; the model descriptions also include a less mathematical overview of how the models work as well as illustrative figures of the DLKT model architectures. The used notation assumes basic understanding of linear algebra and neural networks.

3.2. Naive Baselines

3.2.1. Mean

Mean is our simplest model. It is also the only one of the naive models that is computed using training data. The mean model takes the mean of the correctness values in training data and uses

[5] https://github.com/myudelson/hmm-scalable, accessed 2020-03-01

[6] https://github.com/theophilee/learner-performance-prediction, accessed 2021-05-27

Table 1: Summary of evaluated models.

	Model	Shorthand	Details
Naive	Mean prediction	Mean	A simple statistic
	Next as previous	NaP	A simple baseline model
	Next as previous N's mean	NaPNM	A slightly less simple baseline model
Non-DLKT	Bayesian Knowledge Tracing	BKT	Commonly used as a baseline, predecessor to DLKT models (Yudelson et al., 2013)
	Gervet et al. Logistic Regression	GLR	Logistic regression model with best input feature combination in (Gervet et al., 2020)
Deep Learning Knowledge Tracing (DLKT)	Long Short Term Memory (Recurrent Neural Network) Deep Knowledge Tracing	LSTM-DKT	First DLKT model (Piech et al., 2015)
	Vanilla (Recurrent Neural Network) Deep Knowledge Tracing	Vanilla-DKT	First DLKT model along with LSTM-DKT (Piech et al., 2015)
	Dynamic Key-Value Memory Network (MXNet implementation)	DKVMN	First DLKT model with separate next attempt skills as input (Zhang et al., 2017)
	Dynamic Key-Value Memory Network (as depicted in its respective paper)	DKVMN-Paper	Same as above
	Self Attentive Neural Network	SAKT	A DLKT model based on Transformer neural network (Pandey and Karypis, 2019)
	LSTM-DKT with next skill input	LSTM-DKT-S+	LSTM-DKT variation with added skill input as in DKVMN and SAKT, presented in this work

the computed mean as prediction output. To keep the model as simple as possible, the mean is computed over all of the training data values, i.e. not per student or skill.

For metrics that operate on binary prediction values (correct or incorrect) instead of percentages, the mean predictor model is equivalent to Majority Vote aka ZeroR (Zero Rate) classifier that predicts all values as the most common label in given data. This happens because the mean prediction is rounded in order to compute the metric score. For metrics that are computed from continuous values (e.g. RMSE), the models are not equivalent.

While the mean model can be considered to have no predictive power, for that reason specifically, it is good for evaluating whether other models are useful in practice or not (Devasena et al., 2011).

3.2.2. Next as Previous

Another naive model we use is predicting the correctness of a student's answer at time $t+1$ to be the same as the student's previous answer correctness c_t. Formally, $\text{NaP}(t+1) = c_t$. Similarly to mean prediction, NaP has minimal predictive power and thus serves as an additional validity check for more sophisticated models.

Table 2: Common variables used in the descriptions of the evaluated models.

Variable	Description
$T \in \mathbb{N}$	Number of attempts in an attempt sequence
$t \in \{1..T\}$	Time step of an attempt sequence
$S \in \mathbb{N}$	Number of distinct skills
$\boldsymbol{s} \in \{1..S\}^T$	Set of skill identifiers in an attempt sequence
$\boldsymbol{c} \in \{0,1\}^T$	Set of correctness values in an attempt sequence
$s_t \in \{1..S\}$	Skill identifier at time step (or attempt number) t
$c_t \in \{0,1\}$	Attempt correctness at time step t (1 indicates correct and 0 incorrect)
$y_t \in [0,1]$	Model output, i.e. prediction of attempt correctness produced at time step t (y_t is an estimate of c_{t+1} produced by the model)
$\sigma(x)$	Standard logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$
$\tanh(x)$	Hyperbolic tangent $\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$
$\boldsymbol{x} \odot \boldsymbol{y}$	Element-wise product between $\boldsymbol{x}$ and $\boldsymbol{y}$.

3.2.3. Next as Previous N's Mean

For our final naive baseline, we extend the Next as Previous model to compute the probability of a student answering an exercise correct as the mean of N (or t if $t < N$) previous correctnesses so that NaPNM$(t+1) =$ mean$(c_t, .., c_{\max(t-N,1)})$. We report results for NaP3M and NaP9M.

3.3. Bayesian Knowledge Tracing

Bayesian Knowledge Tracing (BKT) (Corbett and Anderson, 1994) models student knowledge as a latent, i.e. hidden, variable. It is a special case of the Hidden Markov Model, which is a statistical model containing unobservable variable states where the probability of a state is dependent only of the immediately previous state and no other states. In BKT, student knowledge of a skill s is represented as a binary variable m_s indicating skill mastery (either *true* or *false*).

The latent student state is determined by four parameters.

The first is prior learning $P(L_0)$, which is the probability that a student has learned a skill before attempting to apply it. Transition $P(T)$ is the probability of transition from *not mastered* to *mastered* state for a student's knowledge of a skill after an attempt to apply it. Guess $P(G)$ is the probability of applying a skill correctly by coincidence. Slip $P(C)$ is the probability of applying a skill incorrectly by coincidence. BKT models each skill separately and thus sets these parameters individually for each skill. The inner workings of the BKT model is explained in more detail in e.g. (Yudelson et al., 2013).

3.4. Gervet et al. Best-LR

The Best-LR (GLR) model is a logistic regression model introduced in a recent Deep Learning for Knowledge Tracing study (Gervet et al., 2020); the model was the best performing logistic regression model within the study.

The model leverages knowledge tracing input features that are used in previous logistic regression models for knowledge tracing with an additional feature of total success and error counts for previous student attempts. The authors describe the model as "DAS3H (Choffin et al., 2019) without time-window features but augmented with total count features, or equivalently PFA (Pavlik et al., 2009) with rescaled count features, augmented with total counts features and IRT student ability and question difficulty parameters". Mathematical details of the GLR model can be found in the original work.

3.5. Deep Learning Knowledge Tracing

3.5.1. RNN-DKT

The first DLKT model, DKT (Deep Knowledge Tracing) (Piech et al., 2015), is an RNN, which is a neural network that uses an internal memory, often referred to as kernel, to process sequential inputs such as student attempts at an exercise. An RNN for DLKT receives the encoded sequences of previous attempts and skills as input and produces a probability of next attempt correctness as output. The probability of next attempt correctness is output for each part of the input sequence.

The original DKT model was tested with both a "vanilla" recurrent kernel, i.e., a simple fully connected layer (FCL) with a hyperbolic tangent ($\tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$) activation function, and a more complex Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997a) recurrent kernel. The main advantage of LSTM over vanilla RNN is that LSTM is specifically designed to not lose information across the recurrence over time for long input sequences by using a gate structure to control computation flow over the recurrent network. In our experiments, we refer to the DKT models that incorporate an RNN architecture, i.e. RNN-DKTs, by their kernels, namely Vanilla-DKT and LSTM-DKT.

Mathematically speaking, an RNN kernel is an update equation f for its hidden state $\boldsymbol{h}$ at timestep t that takes the current input $\boldsymbol{x}_t$ and previous hidden state as parameters, i.e., $\boldsymbol{h}_t = f(\boldsymbol{x}_t, \boldsymbol{h}_{t-1})$. In our context, $\boldsymbol{x}_t$ is an embedded vector representation of skill s_t and correctness c_t for timestep t.

For our vanilla RNN, we used the SimpleRNN[7] Keras implementation which consists of multiplying inputs and weights, summing the result to weighted previous outputs, and applying an activation function to the resulting sum. We use tanh as the activation function and thus, our vanilla RNN kernel is $\boldsymbol{h}_t = \tanh(\boldsymbol{h}_{t-1}\boldsymbol{W}^h + \boldsymbol{x}_t\boldsymbol{W}^x + \boldsymbol{b})$, where $\boldsymbol{W}^x$ and $\boldsymbol{W}^h$ are trainable matrices of input and hidden state weights and $\boldsymbol{b}$ is a bias vector. The RNN kernel output is equal to $\boldsymbol{h}_t$ for the vanilla RNN.

For the sake of conciseness, we hereafter use σ to denote the standard logistic function $\mathrm{f}(x) = \frac{1}{1+e^{-x}}$ that converts real values into probabilities, i.e. values between 0 and 1. Also, we use $\odot$ to denote element-wise product. For the LSTM kernel, which has multiple variations (Hochreiter and Schmidhuber, 1997a; Gers and Schmidhuber, 2001; Graves, 2013), we use the Keras kernel[8] which is based on the Hochreiter and Schmidhuber (Hochreiter and Schmidhuber, 1997b) variant. The chosen LSTM kernel is a complex layer that comprises an input gate, a forget gate, an output gate and a memory cell. The variables $\boldsymbol{W}^g$ and $\boldsymbol{U}^g$ in the following equations represent trainable input and hidden state weight matrices that are specific for computing

[7] https://keras.io/api/layers/recurrent_layers/simple_rnn/, accessed 2020-01-15

[8] https://keras.io/api/layers/recurrent_layers/lstm/, accessed 2020-01-15

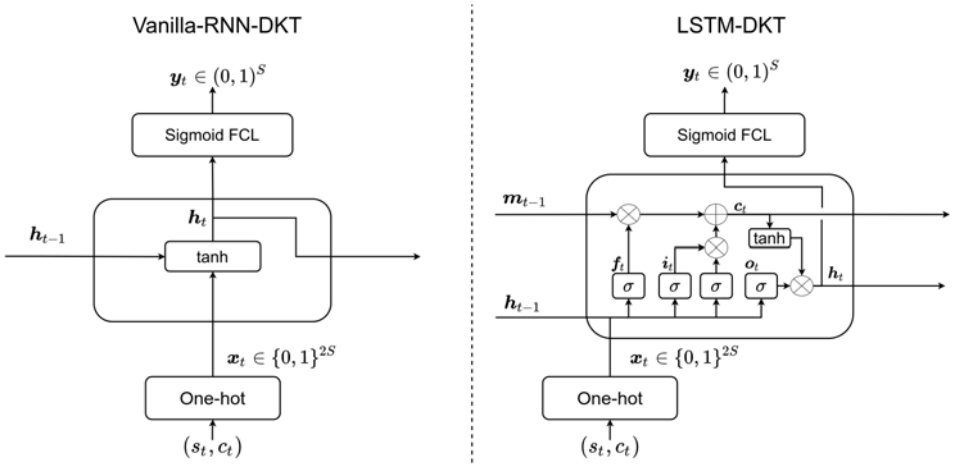

Figure 1: Architectures of Vanilla-DKT and LSTM-DKT models.

g, where g represents which part of the kernel the weights relate to. The LSTM kernel parts are defined as follows:

input gate: $\boldsymbol{i}_t = \sigma(\boldsymbol{x}_t \boldsymbol{W}^i + \boldsymbol{h}_{t-1} \boldsymbol{U}^i + \boldsymbol{b}^i)$
forget gate: $\boldsymbol{f}_t = \sigma(\boldsymbol{x}_t \boldsymbol{W}^f + \boldsymbol{h}_{t-1} \boldsymbol{U}^f + \boldsymbol{b}^f)$
output gate: $\boldsymbol{o}_t = \tanh(\boldsymbol{x}_t \boldsymbol{W}^o + \boldsymbol{h}_{t-1} \boldsymbol{U}^o + \boldsymbol{b}^o)$
memory cell: $\boldsymbol{m}_t = \boldsymbol{f}_t \odot \boldsymbol{m}_{t-1} + \boldsymbol{i}_t \odot \sigma(\boldsymbol{x} \boldsymbol{W}^m + \boldsymbol{h}_{t-1} \boldsymbol{U}^m + \boldsymbol{b}^m)$
output vector: $\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{m}_t)$

The original RNN-DKT models consist of the recurrent kernel and an output layer that produces the outputs $\boldsymbol{y}_t = \sigma(\boldsymbol{o}_t \boldsymbol{W}^y + \boldsymbol{b}^y)$ of the neural net. The RNN-DKT architectures for the two kernels (Vanilla-RNN and LSTM) are depicted in Figure 1.

The RNN-DKT models predict the probabilities of a student having mastered skills given the student's exercise attempt sequence. For each attempt at time t, the attempt sequence contains the resulting correctness of the attempt $c_t \in \{0, 1\}$, where $t \in \{1..T\}$, and a skill id $s_t \in \{1..S\}$, where S is the number of distinct skill tags. Given that T denotes the number of exercise attempts for a student, the neural network takes as input a sequence of student's exercise attempt tuples (s_t, c_t) that are combined into integers $2s_t + c_t$ and then converted into embedded one-hot vectors $\boldsymbol{x}_t \in \mathbb{R}^{2S}$. The $T \times 2S$ sized sequence is then fed to a standard LSTM recurrent neural network with T outputs $\boldsymbol{y}_t \in (0, 1)^S$, one per each time step t. In the resulting output vector, each vector dimension represents the probability of a student knowing the corresponding skill. The model is trained by considering only the skill tag at time $t + 1$ with a binary cross entropy loss $L(\boldsymbol{y}^T \cdot \delta(s_{t+1}), c_{t+1})$ per student attempt where L denotes the binary cross entropy function and δ denotes one-hotting. Note that T in the loss function denotes transpose and not the number of attempts. Overall, this means that the model can improve itself for the upcoming skill based on previous skills for each time step. A successful modification (Yeung and Yeung, 2018), coined DKT+, has been created to improve the current timestep's skill output in addition to the next skill output. This modification has been left out from the current study however, since the differences in model performance are not reportedly significant.

3.5.2. DKVMN

Dynamic Key-Value Memory Network (DKVMN) (Zhang et al., 2017) is an RNN similar to the previously discussed DLKT models LSTM-DKT and Vanilla-DKT, however, its architecture differs significantly from these two models. DKVMN is built to leverage the additional information of next attempt skill tag as an input in addition to the current skill tag and the current attempt correctness. The architecture is heavily inspired by RNNs that use a read-write memory recurrent kernel that incorporates an attention mechanism (Graves et al., 2014). Namely, these RNNs are the Memory Network (Weston et al., 2015; Sukhbaatar et al., 2015) and the Key-Value Memory Network (Miller et al., 2016) models, which were originally designed to solve natural language processing problems. An additional difference is that the DKVMN, as presented originally, replaces the one-hot encoding of inputs used in LSTM-DKT with embedding matrices (Gal and Ghahramani, 2016) to transform the inputs into dense vector representations.

In brief, the DKVMN consists of a recurrent kernel with read and write processes where read denotes the process of generating values for output of the kernel, and the write process involves updating the state of the recurrent kernel. The key inputs are used to compute attention weights via a key memory layer, whereas value inputs are used in the write process. The computed attention weights are applied to both read and write processes of the recurrent kernel. The outputs of the read process of the kernel are then fed to a feed-forward network that produces the model outputs, i.e., the probabilities of exercise correctness.

The DKVMN model involves two embedding layers to process the skill and correctness inputs into fixed-size dense vectors. A key embedding layer for the next skill id in the input sequence, and a value embedding layer for concatenated current skill and correctness. We denote the ith row of a weight matrix $\boldsymbol{A}$ as $\boldsymbol{A}(i)$, key embedding size as K and value embedding size as V. Using a key embedding matrix $\boldsymbol{E}^k \in \mathbb{R}^{S \times K}$, key embeddings $\boldsymbol{k}_t$ are obtained from the matrix row that corresponds to the next skill in sequence, i.e. $\boldsymbol{k}_t = \boldsymbol{E}^k(s_{t+1})$. Similarly, value embeddings $\boldsymbol{v}_t$ are obtained from a value embedding matrix $\boldsymbol{E}^v \in \mathbb{R}^{2S \times V}$ so that $\boldsymbol{v}_t = \boldsymbol{E}^v(2s_t + c_t)$.

We describe the DKVMN recurrent kernel using two distinct steps. The first step is the read process, where an input of a single timestep is read and an output is produced. The second step is the write process, in which the DKVMN recurrent kernel (value memory) is updated for processing the next timestep. DKVMN has two separate memory layers, one for values and one for keys, both with the same memory size M. The key memory $\boldsymbol{M}^k \in \mathbb{R}^{K \times M}$, which is non-recurrent, is used to compute attention weights $\boldsymbol{w}_t$ by multiplying each key memory row with the key embedding so that $\boldsymbol{w}_t = \big(\text{softmax}(\boldsymbol{k}_t \boldsymbol{M}^k(1)), .., \text{softmax}(\boldsymbol{k}_t \boldsymbol{M}^k(M))\big) \in \mathbb{R}^M$, where the softmax function is $\text{softmax}(\boldsymbol{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$ for all values $i \in \{1..N\}$ in vector $\boldsymbol{x}$.

A read value vector $\boldsymbol{r}_t$ is computed using the attention weights and current value memory $\boldsymbol{M}_t^v \in \mathbb{R}^{M \times V}$ as $\sum_{i=1}^{M} w_t(i) \boldsymbol{M}^v(i) \in \mathbb{R}^V$, i.e. the read value is a weighted sum of value memory rows, where key memory provides the weights. Note that for the first input, the read value is not dependent on correctnesses, but only the skill id s_{t+1} and the initial weights of the value memory. The read value is then concatenated with the key embeddings and the concatenated vector is fed to a dense, i.e., fully connected layer (FCL) of size S' with weights $\boldsymbol{W}^s \in \mathbb{R}^{K+V \times S'}$, which is called the summary layer as it is meant to summarize student knowledge state at time t. Then, the summary layer output, i.e., the summary vector $\boldsymbol{s}_t = \tanh(\text{concat}(\boldsymbol{k}_t, \boldsymbol{r}_t)\boldsymbol{W}^s + \boldsymbol{b}^s) \in (-1, 1)^{K+V}$, is fed to a binary, i.e. one-neuron, sigmoid activated output layer with weights $\boldsymbol{W}^y \in \mathbb{R}^{S' \times 1}$ resulting in a probability prediction $y_t = \sigma(\boldsymbol{s}_t \boldsymbol{W}^y + \boldsymbol{b}^y) \in (0, 1)$ for exercise

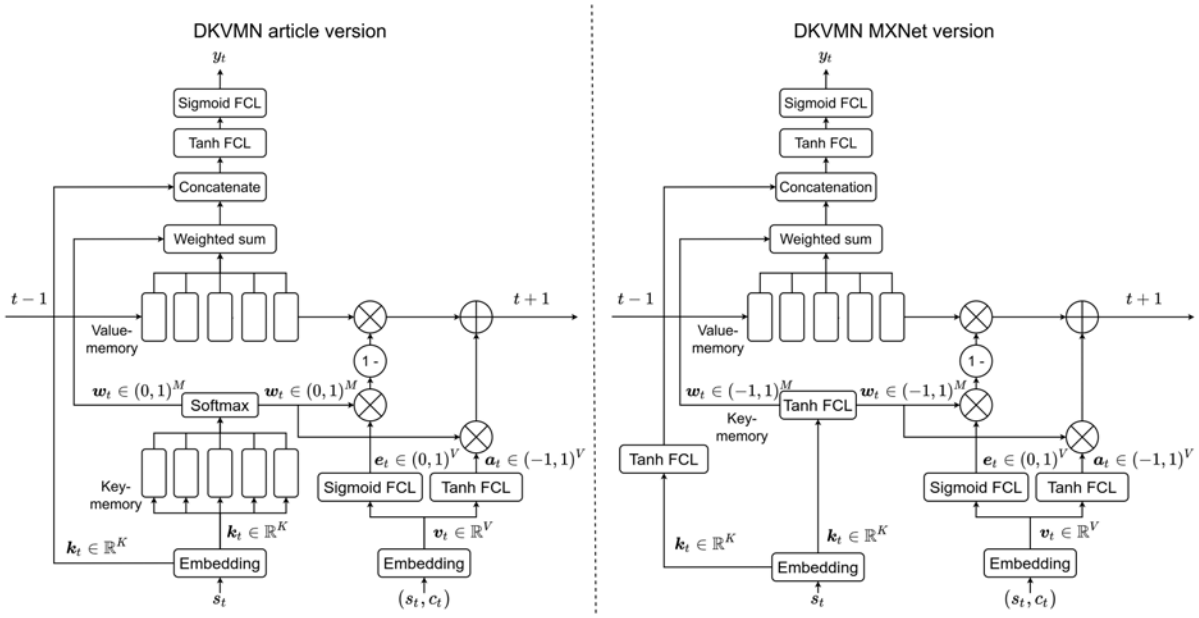

Figure 2: DKVMN architectures of the article (Zhang et al., 2017) and the MXNet implementation based on https://github.com/jennyzhang0215/DKVMN.

correctness at time $t+1$.

The write process of DKVMN involves value memory, attention weights, an addition layer and an erase layer. The addition layer is a FCL with weights $\boldsymbol{W}^a \in \mathbb{R}^{V \times V}$ with tanh activation. The erase layer is a FCL with sigmoid activation and weights $\boldsymbol{W}^e \in \mathbb{R}^{V \times V}$. The addition and erase layers both take value embeddings v_t as input and respectively produce an addition vector $\boldsymbol{a}_t = \tanh(\boldsymbol{v}_t \boldsymbol{W}^a + \boldsymbol{b}^a) \in (-1,1)^V$ and an erase vector $\boldsymbol{e}_t = \sigma(\boldsymbol{v}_t \boldsymbol{W}^e + \boldsymbol{b}^e) \in (0,1)^V$ as output. Each value memory row is then updated by multiplying the value memory row with complemented attention weighted erase vector values e_t summed with the addition vector a_t, i.e. the updated value memory $\boldsymbol{M}^v_{t+1}(i) = \boldsymbol{a}_t + \boldsymbol{M}^v_t(i)(1 - \boldsymbol{w}_t(i)\boldsymbol{e}_t)$.

During reimplementation and verification of DKVMN, we noticed that the MXNet implementation found on the author's GitHub page[9] included a fully connected layer (FCL) with tanh activation function for generating key vectors from key embeddings, which is not mentioned in the original article nor in the GitHub documentation. Notably, we find that the MXNet implementation replaces the key memory layer described in the article with a regular dense layer with softmax activation. The differences between the two models (the version presented in the article and the version in the implementation) is shown in Figure 2 that represents the model architectures of the article version and MXNet version. Both the article version and the MXNet version have been reimplemented for the present evaluation.

3.5.3. Self-Attentive Knowledge Tracing

Self-Attentive Knowledge Tracing (SAKT) (Pandey and Karypis, 2019) is an attentive recurrent neural network that is based on the self-attentive neural network Transformer (Vaswani et al.,

[9] https://github.com/jennyzhang0215/DKVMN, accessed 2020-01-01

2017). Similar to DKVMN, SAKT uses embedding layers to create dense vector representations of encoded skill-correctness inputs and next skill key inputs. The key embeddings and value embeddings in SAKT are created as described in the read process of DKVMN with the addition of positional encoding to value embeddings. The positions are added to value embeddings to add a notion of order in time, because unlike RNNs, attention networks do not operate sequentially. SAKT leverages a self-attention layer to process sequential input without explicit ordering that effectively reveals distance in time.

To be more precise, SAKT incorporates a self-attention layer called multi-head attention that computes attention for each of its inputs simultaneously. For preventing the use of future value inputs, SAKT uses a mask on the multi-head attention that constrains the attention computation to only the preceding inputs. The self-attention layer outputs a vector of real values that is then passed to a feed-forward network that outputs the attempt correctness predictions of the SAKT model. As opposed to attention in DKVMN, while DKVMN leverages attention for its read values per each timestep, SAKT applies attention jointly on the whole key and value sequences that precede each attempt.

More formally, as in DKVMN, the inputs for SAKT are key embeddings $\boldsymbol{k}_t$ and value embeddings $\boldsymbol{v}_t$ for each timestep t. To add information of timesteps in SAKT, position embeddings $\boldsymbol{p}_t = \boldsymbol{W}^p(t)$ are added to value embeddings to create positioned value embeddings $\boldsymbol{v}_t^p = \boldsymbol{p}_t + \boldsymbol{v}_t$. These embeddings are then passed on to the multi-head attention mechanism.

Multi-head attention first projects its inputs, key and positioned value embeddings, and projects them with linear dense layers $\boldsymbol{W}^k \in \mathbb{R}^{K \times A}$ and $\boldsymbol{W}^v \in \mathbb{R}^{V \times A}$ to query vectors $\boldsymbol{q}_t^a = \boldsymbol{k}_t \boldsymbol{W}^k$, key vectors $\boldsymbol{k}_t^a = \boldsymbol{v}_t^p \boldsymbol{W}^v$ and value vectors $\boldsymbol{v}^a = \boldsymbol{v}_t^p \boldsymbol{W}^v$, each of size A. Note that the key embeddings serve as query inputs in multi-head attention terminology, and that the positioned value embeddings serve as both key and value inputs for the multi-head attention. The projections are fed into a dot-product attention layer (Luong et al., 2015). $\text{Attention}(\boldsymbol{q}_t^a, \boldsymbol{k}_t^a, \boldsymbol{v}_t^a) = \text{softmax}(\frac{\boldsymbol{q}_t^a \cdot \boldsymbol{k}_t^a}{\sqrt{A}}) \cdot \boldsymbol{v}^a$, whose outputs are then concatenated and projected once more with a linear fully connected layer.

SAKT processes the outputs of the multi-head attention with a feed-forward network that consists of three fully connected layers, where the first layer is ReLU activated and the second is linear. The final layer is a one neuron sigmoid activated output layer for producing exercise correctness probabilities. The model architecture and the attention mechanism are illustrated in Figure 3.

3.6. Model Variations

Next, we describe and summarize the model variations that we include to the present study. First, the RNN-DKT contains two kernel versions, a vanilla kernel and an LSTM kernel. The kernel is the main component of RNN-DKT, and therefore we consider the two RNN-DKT versions, Vanilla-DKT and LSTM-DKT, as separate models. Second, the DKVMN model has a model variant that modifies its recurrent behaviour. The inclusion of the two variations are due to the observed differences between the original DKVMN article and the authors' MXNet implementation as discussed at the end of Section 3.5.2. Third, we also introduce our own variation to LSTM-DKT that is based on the differences of DKVMN and SAKT when compared to the original LSTM-DKT. Aside from the major architectural differences of the models, we seek to quantify the effect of giving the LSTM-DKT model information about the next attempt skill input, as is done in DKVMN and SAKT. The LSTM-DKT-S+ that includes key vector

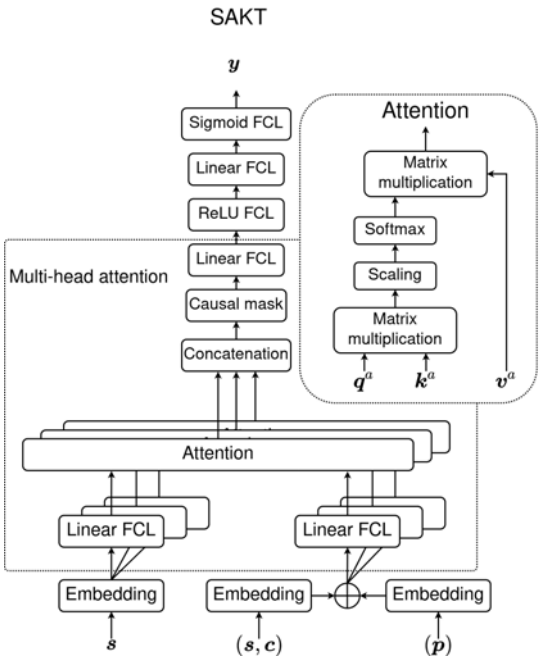

Figure 3: Self-Attentive Knowledge Tracing (SAKT) architecture. In SAKT, value embeddings are used for both value inputs $\boldsymbol{v}^a$ and key inputs $\boldsymbol{k}^a$ in the attention layer, i.e. $\boldsymbol{k}^a = \boldsymbol{v}^a$.

inputs at time step $t + 1$ in the same fashion as in DKVMN and SAKT is shown next to the original LSTM-DKT model in Figure 4.

In addition, we examine the impact of input and output processing variations for each model and model variation, i.e. one-hot encoding versus using an embedding layer, and the skills-to-scalar output in SAKT and DKVMN versus the output-per-skill layer in the original RNN-DKT models. Whereas one-hot encoding produces a vector with a size relative to the number of skills in a dataset which risks the vectors becoming impractically large, embedding layer allows one to keep the embedding size smaller by learning how to represent categorical inputs as arbitrary sized real-valued vectors. Piech et al. (Piech et al., 2015) acknowledged this problem in one-hot encoding already in their introduction of the RNN-DKT models. They proposed using random low-dimensional Gaussian vector encodings to tackle the problem, which corresponds to using an embedding layer minus the learning. For the output layer variations there are two distinct differences: The most obvious difference is that skills-to-scalar introduces an additional layer with the activation function tanh to the recurrent outputs. The second, more subtle difference, is in how the learned skills are represented and learned in the output layer(s). When training the output-per-skill variant, the skill weights are trained separately and only the output weights that correspond to the next attempt skill is trained out of all of the skill weights. Conversely, for the skills-to-scalar variant, all of the skill weights (input weights of the skill summary layer) are trained for each output regardless of skill. The final scalar output corresponds to the probability of the student getting the next attempt correct regardless of the skill unless the model is provided with next skill information as input.

These variations are illustrated for the LSTM-DKT model in Figure 5. In general, they allow us to gain insight into how much of the models' performance differences can be attributed to minor model modifications that are interchangeable between the models as they affect only

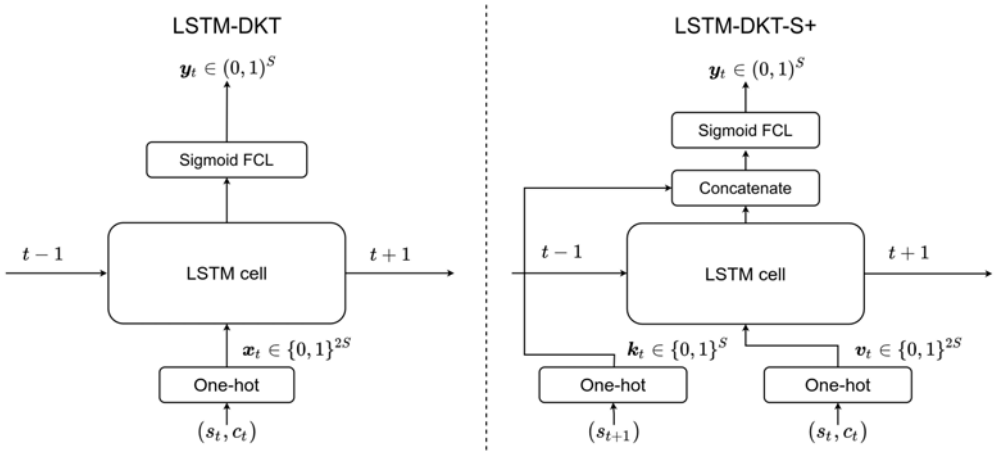

Figure 4: Architectures of LSTM-DKT, described in 3.5.1, and LSTM-DKT-S+ that includes information about the future skill in the input.

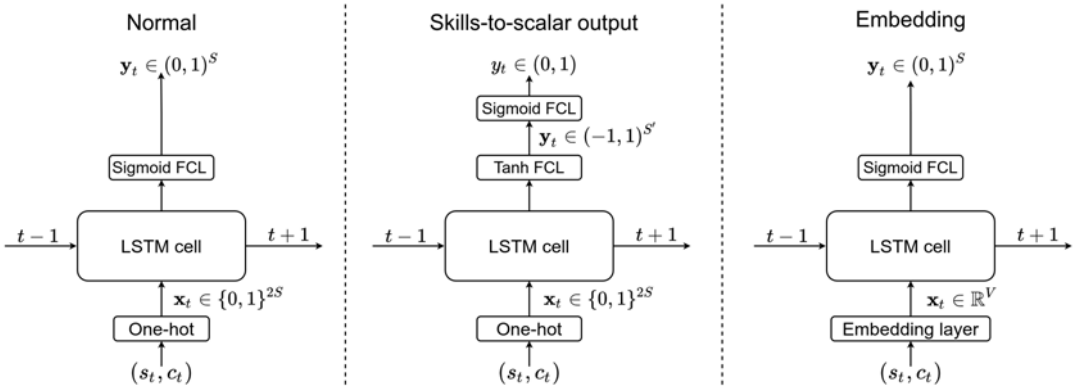

Figure 5: Output layer and input encoding variations shown for the LSTM-DKT model. In the present study, the same variations have been applied to the other models as well.

pre- and post-sequential processing.

The model variations in input and output processing are minor architectural changes and they can be easily implemented as hyperparameters. Thus, we consider the input and output model variations as hyperparameters among other tunable model variables that may or may not affect model performance. The hyperparameter optimizations are outlined in more detail in the next Section.

4. METHODOLOGY

4.1. DATASETS

For the purposes of our study, we use seven datasets (summarized in Table 3), which are as follows: 1) ASSISTments 2009 updated[10], 2) ASSISTments 2015[11], 3) ASSISTments 2017 Data Mining Competition[12], 4) Statics 2011[13], 5) Synthetic-5 (k=2) and 6) Synthetic-5 (k=5) [14], and 7) a new dataset called IntroProg[15].

The first six of the datasets have been previously used in knowledge tracing evaluations, while the seventh one is a new dataset created for the purposes of this study. The ASSISTments 2009 updated, 2015, and 2017 come from the ASSISTments system (Heffernan et al., 2006; Heffernan and Heffernan, 2014; Feng et al., 2009). The 2009 updated dataset is an updated version of the original 2009 dataset, accounting for DKT-specific evaluation related issues in the original dataset (Xiong et al., 2016). The Statics 2011 dataset comes from an engineering statics course (Steif and Bier, 2014), and the Synthetic-5 datasets are simulated datasets which were originally used to test the LSTM-DKT model (Piech et al., 2015). The synthetic datasets have been built using IRT to generate student responses for $k \in 1..5$ hidden exercise concepts for a sequence of 50 exercises per student – for the present study, we include data for $k = 2$ and $k = 5$ as these are pre-generated and publicly available. The $k = 5$ version has also been used in later works.

The seventh dataset, IntroProg, contains information from a total of 3273 students. The data comes from a 2 ECTS[16] introductory programming course organized by a Nordic research-oriented university. In the course, students learn principles of procedural programming (i.e. input/output, variables, conditionals, loops) and learn to work with basic data structures (lists, maps). In total, the course has 61 programming exercises. The course is offered as an online textbook with an integrated programming environment and an automated assessment system. Whenever a student submits an exercise to the automated assessment system, the system records the submission as well as information on the correctness of the attempt. Each entry in the data includes a student id, an exercise id, and correctness of the attempt. An attempt is classified as correct when all instructor-written tests pass for the submitted exercise. There is no limit for the number of submissions to exercises. Only the best submission is considered when grading the course, which means that the student can continue working on an exercise even after a failed submission.

For data preprocessing, we follow the methodology of (Zhang et al., 2017). If the data

[10]Retrieved from https://github.com/jennyzhang0215/DKVMN/tree/master/data/assist2009_updated, accessed 2020-01-15

[11]Retrieved from https://sites.google.com/site/assistmentsdata/home/2015-assistments-skill-builder-data, accessed 2020-01-15

[12]Retrieved from https://sites.google.com/view/assistmentsdatamining/dataset, accessed 2020-01-15

[13]Retrieved from https://github.com/jennyzhang0215/DKVMN/tree/master/data/STATICS, accessed 2020-01-15

[14]Retrieved from https://github.com/chrispiech/DeepKnowledgeTracing/tree/master/data/synthetic, accessed 2020-01-15

[15]Available with the source code and results at https://zenodo.org/record/5808919

[16]ECTS stands for European Credit Transfer System. One ECTS is approximately 25 to 30 hours of work, although this naturally varies between students.

does not include skill identifiers, exercise identifiers are used instead as skill identifiers. As an exception, for the Statics dataset, we use exercise identifiers instead of skill identifiers. Data rows that do not contain a skill identifier, a user identifier, or correctness are discarded from the analyses. That is, every attempt must contain a skill identifier (or an exercise identifier for datasets where such is used as skill identifier), a user identifier and a correctness value. Similarly, for datasets where the correctness of an attempt was not a binary variable (ASSISTments 2015), we only used the attempts that were correct or incorrect. Finally, if a student has at most one attempt, data from the student is excluded as such data is impractical for training or evaluating the models.

Table 3: Summary of the datasets used in our evaluation. "Max Att." stands for maximum attempts in the data for a single student, and "E. ids" stands for the total number of exercise identifiers in the data.

Dataset	Max Att.	Students	Attempts	Correct (%)	E. ids	Skill ids
ASSISTments 2009 (u)	1261	4151	326k	214k (66%)	110	
ASSISTments 2015	632	19917	683k	514k (72%)	100	-
ASSISTments 2017	3057	1709	943k	351k (37%)	3162	102
Statics 2011	1181	333	189k	145k (77%)	1223	98
Synthetic-5 K=2	50	4000	200k	137k (69%)	50	-
Synthetic-5 K=5	50	4000	200k	122k (61%)	50	-
IntroProg	1857	3273	172k	84k (49%)	64	-

4.2. Approach

4.2.1. Metrics

Results are reported using seven metrics: Accuracy (ACC), Area Under the Curve (AUC), Precision, Recall, F1 score, Matthews Correlation Coefficient (MCC), and Root-Mean-Square Error (RMSE). Multiple metrics are included, since e.g. (Caruana and Niculescu-Mizil, 2004) and (National Research Council and Climate Research Committee, 2005) suggest that models can perform differently depending on the metric and that using a single metric can lead to misguided judgement of model performance. Accuracy is included as an intuitive metric that tells the overall correct prediction percentage. The AUC, which is more precisely the Area Under the Curve of Receiving Operator Characteristic (AUROC, or ROC-AUC), is a commonly used metric in DLKT models. It is "equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance." (Fawcett, 2006). Precision gives insight on how many of the predicted positive values were actually positive, whereas Recall tells how many of all positive instances were correctly identified as positive. F1 score is the harmonic mean of Precision and Recall and is an often used metric although susceptible to imbalances in data (Chicco and Jurman, 2020). Matthews correlation coefficient (MCC) provides high scores only if predictions are accurate regardless of the rate of negative and positive elements in a dataset. RMSE is a commonly used metric, which unlike the other metrics, is computed from raw prediction values. These metrics are summarized in Table 4, where confusion matrix terms true positives (TP), false positives (FP), true negatives (TN) and false negative (FN), are used for defining some of the metrics.

Table 4: Definitions for the used metrics. True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) come from confusion matrix terminology.

Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1	$2 * \frac{\text{Precision}*\text{Recall}}{\text{Precision}+\text{Recall}}$
AUC	Area under a plot line with recall on y-axis and false positive rate ($\frac{FP}{FP+TN}$) on x-axis for decision thresholds from 0 to 1
MCC	$\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$
RMSE	$\sqrt{\frac{\sum_{i=1}^{N}(c_i - y_i)^2}{N}}$

For confusion matrix based metrics, we consider the attempt correctness c_t value 1 as a positive label and 0 as a negative label. To determine the polarity of predicted values, i.e. whether a value is positive or negative (0 or 1), we use 0.5 as the decision threshold. This means that a prediction y_t is considered positive if it is equal or greater to 0.5. RMSE is an exception in the used metrics as it does not use confusion matrix categories but is computed directly from model prediction values. AUC is also distinct from most other included metrics as it is independent of a decision boundary due to it being computed from a plot over all decision thresholds (from 0 to 1). Out of the seven metrics, AUC, and MCC take class imbalance into account, which can be beneficial on interpreting results from skewed datasets, although it can mask poor performance by weighing underrepresented labels too heavily (Lobo et al., 2008; Jeni et al., 2013).

Our simplest baseline model, Mean (majority vote for binary metrics) is by definition unable to produce meaningful AUC, MCC, Recall, Precision (when majority of labels are negative), and by extension F1 scores. For AUC, this happens for the Mean since for each decision boundary, every predicted value is on the same side of the decision boundary, wherefore Recall equals false positive rate and thus AUC is a constant 0.5. MCC cannot be computed due to division by zero, which also applies for Precision and Recall when the majority of labels are negative. When the majority of labels are positive, Precision is meaningful but Recall will always be one.

4.2.2. Naive and Non-DLKT Model Training

The naive models included in the study are outlined in Section 3.2. The naive models do not require training apart from using training set to determine the mean correctness for the *Mean* model. In addition, we use two non-DLKT baseline models BKT and GLR that are statistical classification models and contain parameters that need to be trained, i.e. learned. These are presented in Sections 3.3 and 3.4.

For BKT, we use the publicly available `hmm-scalable`[17] implementation by Yudelson, which includes individualized BKT (Yudelson et al., 2013). We evaluate the available solvers

[17] https://github.com/IEDMS/standard-bkt,https://github.com/myudelson/hmm-scalable, accessed 2020-03-01

for the BKT implementation and pick the best results per dataset according to the RMSE metric. We choose the best solver based on RMSE instead of AUC since by inspecting the results we find that there is minimal difference in AUC scores compared to choosing the best model by AUC. However, especially for the ASSISTments 2015, the best model by AUC is significantly worse on most other metrics. Also, previous studies have shown that AUC is a less viable metric for BKT than RMSE (Dhanani et al., 2014; Pelánek, 2015). For other hyperparameters we use the default settings. For GLR, we use the publicly available implementation[18] by Gervet et al. discussed in (Gervet et al., 2020). When training the model, we use the default L-BFGS solver provided in the implementation.

We use 5-fold cross validation to evaluate the models, i.e. the data is divided into five parts, and the models are then trained and evaluated five times using one part as the test set and the other parts as the training set. Each part is used once in test, and four times in training. The reported results are averaged over the five training and testing iterations. Dividing the data into the parts is performed by student, meaning that data from each student is present only in one part at a time and not divided over the parts.

In order to have the GLR model implementation comply with our methodology, we modified the data preprocessing and added 5-fold cross validation.

4.2.3. DLKT Model Training and Hyperparameter Optimization

All DLKT models are optimized using binary cross-entropy (log loss): $-(c_i \log(y_i) + (1 - c_i) \log(1 - y_i))$, which is an often-used loss function for training neural networks. The machine learning models are evaluated using 5-fold cross validation. The models are given one hundred training iterations with early stopping, where early stopping is performed if ten consecutive iterations do not improve the performance of the model, measured with binary cross-entropy[19]. A validation set which consists of 10% of the training data is used to determine the early stopping. Thus, the training data for each fold comprises 70% of the data while the remaining 20% is used for evaluation.

Grid search is used to find optimal hyperparameters for each model, summarized in Table 5. The best hyperparameters are selected based on averaged AUC score in the 5-fold cross-validation. The hyperparameters regarding model layers include recurrent layer size (memory size for DKVMN), key embedding size, value embedding size and skill summary layer size. Additionally, the number of attention heads is tuned for SAKT as it uses multi-head attention. The use of layer sizes depends on the model variation as some models and/or their variations either do not have a corresponding layer or the size is a set constant. For one-hot input model variations, the embedding sizes are not applicable as one-hot size is a constant determined by the count of distinct skill identifiers in a given dataset. Likewise, output-per-skill output layer size is determined by the same distinct skill count to provide an output for each skill as the name suggests. For model variations with such an output layer, a skill summary layer is not used. The DKT models apart from LSTM-DKT-S+ do not have separate key inputs, and thus key embedding size does not affect it.

Besides layer-specific hyperparameters, we tune the models for learning rate and max at-

[18] https://github.com/theophilee/learner-performance-prediction, accessed 2021-05-27

[19] Typically, a "main" evaluation metric is selected for early stopping. Here, we deem the training metric more suitable as we aim to measure the effect of evaluation metric choice.

Table 5: Overview of hyperparameters used for training DLKT models. Although there are a maximum of two options per hyperparameter, the variation count is not a power of two since some options cancel each other out. Also, not all hyperparameters affect every model, e.g. SAKT is the only model that uses attention heads.

Hyperparameter	Options
Recurrent layer size	{50, 100}
Key embedding size	{20, 50}
Value embedding size	{20, 50}
Skill summary layer size	{50, 100}
Input variation	{One-hot, Embedding}
Output variation	{Output-per-skill, Skill summary + Scalar}
Learning rate	{0.01, 0.001}
Dropout	{0.2}
Attention heads	{1, 5}
Batch size	{32}
Random Seed	{13, 42}
Optimizer	{Nadam}
Variations	72-240 (depending on model)

tempt count. Although the latter is not necessary, it is used both in DKVMN and SAKT due to implementation restrictions and possibly for faster training times. While the DKVMN article does not mention max attempt count, the MXNet implementation by the authors uses 200 as the max attempt count. SAKT authors mention using different max attempt counts for different datasets, where the values for max attempt count range from 50 to 500. Both DKVMN and SAKT implement max attempt count by splitting exceeding attempts as a new attempt sequence recursively until no attempt sequence exceeds the max attempt count, effectively creating new students into the dataset. We note that our implementations published as a part of this article do not include this restriction and can be trained with arbitrarily large attempt sequences, although the training time and space requirement increases significantly without max attempt counts. For the grid search, we use the max attempt count of 200 both for reproducibility purposes and to reduce model training time.

We also include random number generator seed as a hyperparameter to provide a baseline for evaluating the effect of hyperparameter tuning. The differences in the results of models whose hyperparameters differ only regarding the random seed can be safely assumed to be caused by randomness. This gives insight into what extent randomness may affect the 5-fold cross validation results.

All DLKT models were trained using input batches of size of 32, 20% dropout and the Nadam gradient descent optimizer (Dozat, 2016). While all of these may affect the results, we decided to not tune them systematically for each model to keep the hyperparameter option space reasonably sized. In addition to comparing the performance of the models where the best hyperparameters are selected based on AUC, we explore selecting best hyperparameters with other metrics. Target metric used for model training, for instance, can influence selected features and model performance on other metrics (Sanyal et al., 2020). We analyze the effect of choosing a metric in grid search by comparing the perceived performance loss in other metrics. By loss,

we mean the difference between the performance of the model with "best" hyperparameters selected based on a given metric, and the best result achieved when selecting models with some other metric. As an example, we compare the performance of a model that was selected based on AUC with models selected by other metrics to determine how the performance over all metrics differ between these models.

We acknowledge that we only included two options for each tuned hyperparameter. This, along with the untuned options, leaves room for speculation whether more broad tuning would provide significantly different results. All in all, considering all datasets and model variations, the number of trained and evaluated DLKT models totals 5,880 cross-validated models.

4.2.4. Maximum Attempt Cut

As briefly mentioned, DKVMN and SAKT, which leverage maximum attempt counts due to the restrictions in their original implementations, split input sequences according to the maximum attempt count hyperparameter. This means that if the maximum attempt count is 100 and a user has 950 attempts, the attempts would be split into 9 sets of 100 attempts and one with 50 attempts. Effectively, this multiplies the number of data points that the models are trained on, since the models consider the different sets from the same student as completely new students. Instead of having fewer students with longer attempt sequences, the models have more students with some having unnatural starting points for their attempt sequences. As it is poorly explained why this would be the preferred case, we test to what extent cutting off attempts that exceed the maximum attempt count affect the results. We use the same 200 as maximum attempt count splitting for maximum attempt count cutting. Further, we test the models for no maximum attempt count. The effect of splitting or cutting is evaluated only for the best hyperparameter options determined by the grid search using the hyperparameter options in Table 5.

4.2.5. Hardware and Software

The model training in this work was performed using computer resources within the Aalto University School of Science "Science-IT" project. All DLKT model training was conducted using CPUs unless otherwise noted. The models were implemented in Python using TensorFlow version 2.1.0.

Further, to evaluate the effect of hardware, the models were re-trained and evaluated using GPUs. GPU evaluation was only conducted for the hyperparameter values deemed best by grid search using CPUs. Due to implementation restrictions, as well as to compare machine learning framework versions, the TensorFlow version was updated to 2.6.2 for GPU computation. Thus, we also retrained the models with CPU on the new TensorFlow 2.6.2 version for the CPU versus GPU comparison and present CPU results for both TensorFlow 2.1.0 and 2.6.2, while the GPU results are presented only for TensorFlow 2.6.2.

5. RESULTS

In the following result tables, the best metric values for a given model have been selected by picking the model with the highest AUC score. Thus, the actual highest scores for other metrics may be higher than in the presented tables because the best model as determined by AUC might

not yield the best results for all metrics. Section 5.2.5 presents this aspect in greater detail. Full evaluation results are available in the online repository alongside the source code[20].

5.1. Models and Performance

Here, we show results related to RQ1 (How do DLKT models compare to naive baselines and non deep-learning KT models?) and RQ2 (How do DLKT models perform on the same and different datasets as originally evaluated with?). We compare the baseline model results and DLKT model results on the evaluated datasets both regarding AUC, our main comparison metric, and other evaluation metrics separately. We also study how our results compare to results obtained in previous work.

5.1.1. Baseline Models and DLKT Models

Here, we mostly focus on the performance of the models using AUC as the metric. Other metrics are considered in more detail in subsequent sections. These results are outlined in Appendix A.

The DLKT models consistently produce better predictions than the simple baselines and BKT. The GLR model reaches performance that is above that of the other baselines, including BKT, consistently. For the IntroProg dataset (Table 12 in Appendix A), GLR is the best performing model, and its performance is also on par with the performance of most DLKT models on some metrics on the ASSISTments 2015 dataset (Table 10 in Appendix A) and the Statics dataset (Table 13 in Appendix A), while falling behind DLKT models in other datasets.

None of the baselines apart from GLR reach AUC scores that are able to compete with the scores of any of the evaluated DLKT models. With BKT, which is the second-best performing baseline after GLR, the difference in AUC is smallest in the ASSISTments 2015, IntroProg and Statics (Tables 10, 12 and 13 in Appendix A) where BKT falls behind the bottom DLKT model AUC score by 2% points, 3.2% points, and 3.4% points respectively. In these three datasets, the difference between BKT and the bottom DLKT model is greater than the difference between the top DLKT model[21] and the bottom DLKT model[22]; 1.1% points, 0.6% points and 1.7% points for ASSISTments 2015, IntroProg and Statics respectively. Notably however, the best performing model for IntroProg is not a DLKT model but GLR: the difference in AUC score between GLR and the top-performing DLKT models LSTM-DKT and DKVMN is 1.6% points.

When considering BKT, it is in some cases unable to beat the naive baselines, in addition to its poorer performance compared to the DLKT models or GLR. This is evidenced in the Synthetic-K2 dataset (Table 14 in Appendix A), where BKT achieves a lower AUC score (0.635) than the NaP3M (0.654) or the NaP9M (0.709).

Overall, the DLKT models surpass GLR and other baselines as DLKT models outperform the best-performing baseline GLR on 3 out of 5 real datasets and on both synthetic datasets. A clear difference between model performances of the two best DLKT models (DKVMN and LSTM-DKT) versus baselines can be seen for each metric in Figure 6, which shows scores averaged over datasets.

[20] https://zenodo.org/record/5808919

[21] LSTM-DKT for ASSISTments 2015, DKVMN and LSTM-DKT for IntroProg, and DKVMN for Statics

[22] SAKT for ASSISTments 2015, Vanilla-DKT for IntroProg, and SAKT for Statics

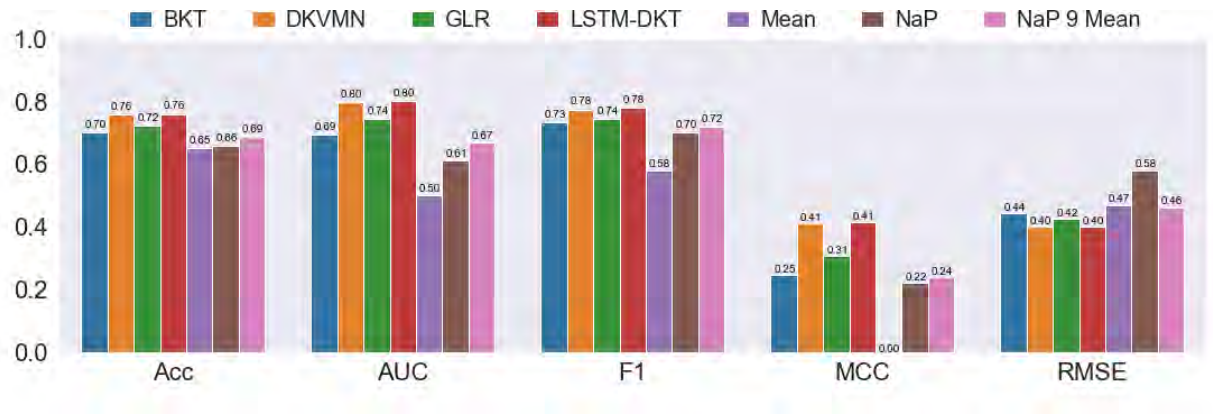

Figure 6: Result comparison of the two best performing deep learning models over baselines. Scores are averaged over all datasets.

5.1.2. Model Performance on Different Metrics

In the previous Section 5.1, we focused on the performance of the models when using AUC as the metric. Next, we look deeper into the differences in model performance when considering other metrics. These metrics are accuracy, precision, recall, F1, MCC, RMSE, which are presented in Section 4.2.1.

As noted previously, the DLKT models overall outperform GLR as well as other baselines when using AUC. The same observation can be made when studying the average performance of the models over all datasets, presented in Figure 6 (the figure includes the two best performing DLKT models and the baselines). When contrasted with AUC results, we however observe more variation between the performances of the DLKT models and the baselines. When looking at the results in Tables 12 and 13 in Appendix A for different metrics, GLR ranks among the best on IntroProg and Statics datasets. In IntroProg, GLR receives the best scores for Accuracy, AUC and MCC (0.4% points, 1.6% points, and 3.1% points respectively). In Statics, GLR receives scores similar to most DLKT models. In the other datasets, DLKT models perform better on all metrics, excluding precision and recall.

When ranking the models based on their performance in the different metrics, we observe a degree of variation between the metrics and the datasets. For example, for the ASSISTments 2015 dataset (Table 10 in Appendix A), the simple Mean baseline has F1-score (0.849) which is equal to the F1-Score of GLR, topping all DLKT models. Similarly, for the same dataset, the Mean baseline's accuracy (0.738) is close to the accuracy of the complex models, falling short 1.3% points from the best performing model (LSTM-DKT). When studying the average performance of the GLR and the best two DLKT models over all datasets, shown in Figure 6, we see that the differences are somewhat small when measured with RMSE, moderate for Accuracy and F1, and rather clear for AUC and MCC.

Considering only the DLKT models, the inter-model performance differs depending on the observed metric, and the differences partially depend on the dataset. For example, for the ASSISTments 2009 updated dataset (Table 9 in Appendix A), the accuracy, F1-Score and RMSE between the DLKT models, excluding SAKT, is at most 1% point, while AUC shows 1.6% point difference between the best and worst DLKT models.

There are some cases where the ranking of the models depends on the chosen metric, although the differences are often small. As an example, for the IntroProg dataset (Table 12 in Appendix A) when comparing SAKT and GLR, SAKT slightly outperforms in F1-Score (0.759 vs 0.755), while GLR outperforms in e.g. AUC (0.843 vs 0.825). Similar observations can be made also for the ASSISTments 2015 dataset (Table 10 in Appendix A), where SAKT outperforms GLR in AUC (0.714 vs 0.702) and MCC (0.230 vs 0.204), while GLR slightly outperforms SAKT in F1-Score (0.849 vs 0.846) and Accuracy (0.750 vs 0.748).

As for the top performing model all in all, there is no absolute victor, since no model dominates all datasets and every metric. Although the differences are not major, we find that LSTM-DKT (both original and S+ versions) and DKVMN (MXNet version, i.e. not DKVMN-Paper) perform consistently as the best or near the best over the evaluated models across the varying metrics. For AUC, this can be seen in Figure 7. Also, the top-performing DLKT models receive on average better scores for all metrics compared to the baseline GLR or other baselines as can be seen in Figure 6.

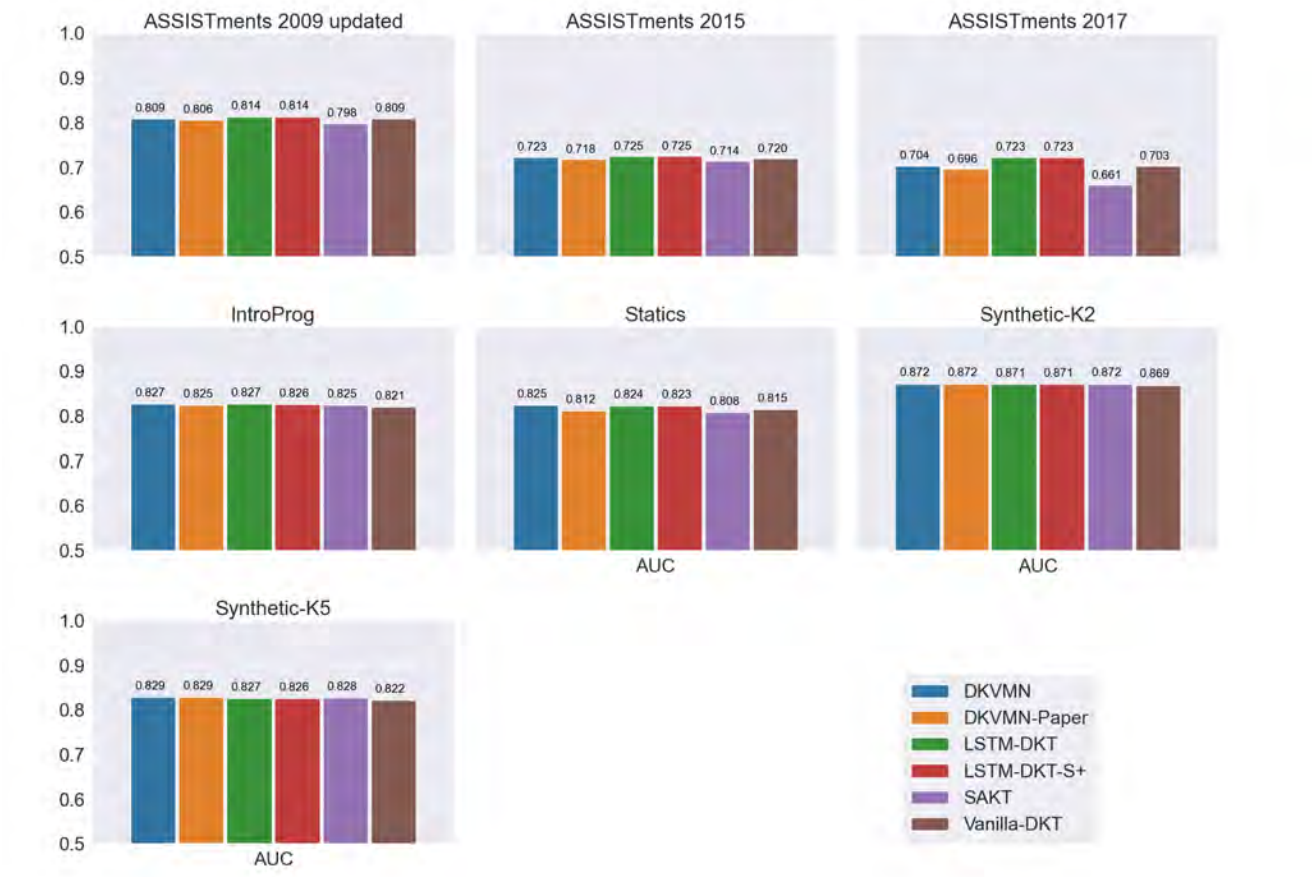

Figure 7: Best AUC scores of each DLKT model per dataset.

On closer inspection into the DLKT models, we observe that LSTM-DKT and LSTM-DKT-S+ hold the top AUC score on ASSISTments 2009, ASSISTments 2015 and ASSISTments 2017, while DKVMN holds the top AUC score on Statics, Synthetic-K2 (shared with SAKT)

and Synthetic-K5. Even though LSTM-DKT models are ranked first on more datasets, the only dataset where the performance difference can be considered other than marginal is ASSISTments 2017 where the difference in AUC score is 2.1% points. For all other datasets, the difference between the LSTM-DKT models and DKVMN is at most 0.5% points.

Additionally, when considering the LSTM-DKT and LSTM-DKT-S+, we observe that in most cases their performance is almost the same if not the same across all datasets and metrics. The largest differences can be observed in ASSISTments 2017 in F1-Score, where the difference is 0.4% points in favor of LSTM-DKT-S+, in IntroProg in MCC, where the difference is 0.4% points in favor of LSTM-DKT, and in Synthetic-K2 in MCC, where the difference is 0.4% points in favor of LSTM-DKT-S+.

All of the above model differences are derived from comparing models that have been hyperparameter tuned for the AUC score.

5.1.3. Model Performance and Previously Published Results

Here, we outline our results in light of previously published results. Tables 6 (ASSISTments 2009 / ASSISTments 2009 updated) and 7 (Statics 2011) summarize observed model performances in this and prior work: (Piech et al., 2015) (DKT), (Zhang et al., 2017) (DKVMN), (Pandey and Karypis, 2019) (SAKT), (Gervet et al., 2020) (GLR). The tables include AUC scores and use two datasets that are common between the articles. In addition, results from models evaluated in this study are included from (Yeung and Yeung, 2018) (DKT+) into the tables, although our present evaluation does not include LSTM-DKT+.

The results presented in this article agree with the previously reported results to some extent. When considering the standard deviation of the model-specific results between the articles, we note that there are considerable differences. For example, for SAKT, the standard deviation is 4.61% points in the ASSISTments 2009 updated dataset, and 2.47% points in the Statics 2011 dataset. Similarly, for LSTM-DKT, the standard deviation is 2.69% points in the ASSISTments 2009 updated dataset, and 1.03% points in the Statics 2011 dataset. For DKVMN, the standard deviations are 0.4% points and 0.75% points for the ASSISTments 2009 updated dataset and the Statics 2011 dataset, respectively. We also observe a minor difference (0.4% points) in Statics 2011 dataset for GLR but a large difference (4.3% points) in ASSISTments 2009 updated dataset.

In addition to the results shown in Tables 6 (ASSISTments 2009 / ASSISTments 2009 updated) and 7, we briefly discuss the results from other datasets (all results are available in the online repository).

When considering DKVMN results, we observe that the original article shows that DKVMN outperforms LSTM-DKT, mostly by a few AUC percentage points, on all four datasets that the models were evaluated on. In contrast, in our study when including the hyperparameter optimizations and model variations (shown in Appendix B in Tables 16, 17, 18, 19, 20, 21, and 22), DKVMN falls behind LSTM-DKT on the ASSISTments 2009 updated, ASSISTments 2015, and ASSISTments 2017 datasets. Conversely, on Synthetic-5 DKVMN is the slightly better performing model while in the original study DKVMN falls short of LSTM-DKT by 2.4% points.

For SAKT, we observe worse performance than reported in the original article, similar to the Gervet et al (Gervet et al., 2020) study. While the SAKT article reports considerable improvements on both LSTM-DKT and DKVMN on Statics, Assistments 2009, 2015 and 2017 datasets,

Table 6: AUC score matrix for models trained on the ASSISTments2009 updated dataset reported on previous articles and this article, labeled as *this*.

Article	LSTM-DKT	LSTM-DKT+	DKVMN	SAKT	BKT	GLR
DKT*	86	-	-	-	67	-
DKT+	82.212	82.227	-	-	-	-
DKVMN	80.53	-	81.57	-	-	-
SAKT	82.0	82.2	81.6	84.8	-	-
GLR	75.7	-	-	75.6	-	77.2
this	81.4	-	80.9	79.8	71.0	72.9
avg (sd)	81.3 (3.33)	82.2 (0.02)	81.4 (0.40)	80.1 (4.61)	69.0 (2.83)	75.1 (3.04)

A row contains an article identifier (specified in Table 1) and the best AUC scores for models reported in that article. Note that the high variance is partly explained by differences in the data used in training the models in the different studies. Most notably, the scores in the DKT article, noted with an asterisk, stems from using an older version of the data. Note also that our result for DKVMN in this table differs from our best results, as the results for the original output layer is used here for replication purposes.

Table 7: AUC score matrix models trained on the Statics 2011 dataset reported on deep learning model papers and this article, labeled as *this*.

Article	LSTM-DKT	LSTM-DKT+	DKVMN	SAKT	BKT	GLR
DKT+	81.59	83.49	-	-	-	-
DKVMN	80.20	-	82.84	-	-	-
SAKT	81.5	83.5	81.4	85.3	-	-
GLR	81.5	-	-	81.3	-	81.9
this	82.4	-	82.5	80.8	77.4	81.5
avg (sd)	81.7 (1.03)	83.5 (0.01)	82.25 (0.75)	82.5 (2.47)	77.4 (-)	81.7 (0.28)

A row contains an article name and best AUC scores for models reported in that article.

in our results, SAKT has worse performance than either DKVMN or LSTM-DKT on all but the IntroProg (not present in original) and the Synthetic datasets (only K-5 in original), where the differences are small to negligible.

We acknowledge that the tested hyperparameter combinations, although at times not reported in the articles, might differ between the studies, which can explain some of the observed larger differences. Some differences can also be attributed to variations in used data. It seems that the data preprocessing differs between the studies as the final used dataset sizes differ at times, as seen in Table 8. For example, while both we and Gervet et al. (Gervet et al., 2020) used the ASSISTments 2009 updated dataset, the data in the repository for the GLR article was considerably smaller (14% fewer lines) than the dataset at our disposal. In addition, we also observe that the DKT+ article has a typo in the Statics dataset size (189,927), where the correct dataset size is 189,297. The same typo appears also in the SAKT article.

The differences between interaction counts 708,631 and 683,801 in ASSISTments 2015 in DKT+ and SAKT when compared to DKVMN is due to the preprocessing step explained in the DKVMN article (Zhang et al., 2017), where correctness values other than 0 or 1 are excluded. The data in our work has slightly fewer interactions than that of DKVMN due to our exclusion of students with less than two attempts.

Table 8: Number of interactions used in different datasets in our article result comparisons as reported in the articles.

	ASSISTments 2009 updated	ASSISTments 2015	Statics
DKT+	328,291	708,631	189,927
DKVMN	325,637	683,801	189,297
SAKT	328k	708,631	189,927
GLR	278,868	658,887	189,297
this	325,515	683,331	189,297

Note that when downloading the dataset from the ASSISTments site[23], the only dataset that is openly accessible is the corrected and collapsed skill builder dataset. In the collapsed dataset, the number of interactions is 346,860, which does not directly match any of the datasets used in the studies outlined in Table 8. This could be due to small differences in preprocessing between the studies. Furthermore, we acknowledge that some of the publicly available datasets may have changed over time – as an example, the ASSISTments site and the available datasets have been updated even during the present work[24].

5.2. Optimal Hyperparameters and Model Variations

Here, we show results related to RQ3 (What is the impact of variations in architecture and hyperparameters on DLKT models' performance?). We inspect the impact of hyperparameter tuning on model performance, including analyzing the effect of the input and output model variations, and the effect of the metric choice for selecting the best hyperparameters. The evaluated hyperparameter combinations are outlined in Table 5 in Section 4.2.3. To discern the magnitude of the effect of the chosen hyperparameters, we also outline results related to random seed selection and used hardware (CPU vs GPU). We also show the impact of the metric choice used for selecting the best hyperparameters out of the grid searched variations.

5.2.1. Optimal Hyperparameters

The hyperparameters that yielded the best performance as measured by the AUC score are outlined in tables in Appendix B, one table per dataset. In general, the optimal hyperparameters are model- and dataset-specific. When considering layer sizes, the recurrent layer size is more often smaller (50) than larger (100) for all models apart from Vanilla-DKT, key and value layer sizes do not show a pattern, and summary layer size appears more often smaller (50) for DKVMN and DKVMN-Paper and larger (100) for other models when it is used.

As for the learning rates, the models most often perform better with 0.001 as the learning rate when compared to the other option 0.01. The only exception is DKVMN-Paper, which always performs better with 0.01 except for the two synthetic datasets. Also, in the ASSISTments 2017 dataset all models but SAKT and Vanilla-DKT perform better with 0.01 learning rate. The two models, SAKT and Vanilla-DKT are also the only ones that perform best with learning rate

[23] https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data/skill-builder-data-2009-2010, accessed 2021-04-01

[24] e.g. comparing the present – 2021/12/15 – version of the ASSISTments site with https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data/skill-builder-data-2009-2010, accessed 2021-04-01

0.001 on all the evaluated datasets. The number of attention-heads for SAKT also differs across datasets with 1 as the optimal for the datasets IntroProg and Statics and 5 for other datasets.

5.2.2. Input and Output Model Variations

In order to triangulate the differences in the evaluated model architectures, we compared input and output layer variations found between the model architectures that were compatible for all of the models. We compared the output layer variations output-per-skill against skill summary layer and a scalar output layer (skills-to-scalar output layers), and the input layer variations one-hot input against embedding layer.

When comparing output-per-skill and skills-to-scalar output, for LSTM-DKT, Vanilla-DKT and LSTM-DKT-S+, using an output-per-skill layer provided either as good or better results as opposed to using skills-to-scalar output layers. For SAKT, the differences are minimal for the best results, but with output-per-skill layer the SAKT model appears more robust to other hyperparameters as the worst scores are much higher than with skills-to-scalar output. On the other hand, DKVMN shows an opposite effect compared to LSTM-DKT, Vanilla-DKT and LSTM-DKT-S+ with sometimes clearly better performance for skills-to-scalar output layers. We highlight a few of these findings in Tables 23 and 24 included in Appendix B. Notably, in the ASSISTments 2017 dataset (Table 23 in Appendix B), DKVMN performs clearly worse when using output-per-skill than when using skills-to-scalar output layers (difference of 1.9% points), while the difference for DKVMN-Paper is 3.8% points. The DKT models on the other hand perform better with output-per-skill layer (LSTM-DKT 1.2% points, LSTM-DKT-S+ 0.5% points and Vanilla-DKT 2.0% points) For ASSISTments 2015 (Table 24 in Appendix B) DKVMN differences are much smaller and DKVMN performs slightly better with output-per-skill layer unlike in ASSISTments 2017. The results for LSTM-DKT, Vanilla-DKT and LSTM-DKT-S+ show a similar pattern as in ASSISTments 2017. When considering the performance of the evaluated models overall, skills-to-scalar output results have more variance than the output-per-skill results, as depicted by the standard deviation of the results in Tables 23 and 24 in Appendix B. This holds especially for SAKT.

When considering one-hot input and embedding layer, the differences in performance again depend on the model and the dataset, although the differences are often slight. There are a few exceptions, however, as shown in Tables 25 and 26 in Appendix B. In the Statics dataset (Table 25 in Appendix B), only SAKT performs worse when using one-hot encoding (0.8% point difference), while DKVMN-Paper performs marginally better with one-hot encoding (0.7% points). On the other hand, in the ASSISTments 2009 updated dataset (Table 26 in Appendix B), DKVMN-Paper and SAKT perform significantly worse when using one-hot encoding (3.3% points and 4.6% points, respectively), and Vanilla-DKT performs slightly worse (0.3% points). For the other models, the differences are non-existent. This illustrates that for some models, the effect of the input encoding is influenced by the data.

However, with one-hot inputs, the models often achieve higher minimum scores compared to using embedding layers. In the ASSISTments 2009 updated dataset, we see that LSTM-DKT and LSTM-DKT-S+ achieve much higher minimum scores with one-hot inputs. And similarly, Vanilla-DKT and SAKT have higher minimum scores for IntroProg dataset. This suggests that while using embedding layers appears to be the go-to choice when seeking maximal performance, using one-hot inputs can be a safer choice as they seem to be more robust regarding bad choice of hyperparameters.

5.2.3. Maximum Attempt Count

We considered the maximum attempt count used in original DKVMN and SAKT studies in three ways. The first approach was to split the input data into multiple students if a student's attempt count exceeded the maximum attempt count, the second approach was to discard the excessive attempts, and the third approach was to not use a maximum attempt count. We test the first two approaches with maximum attempt count 200 and 500. This analysis was conducted only for the best hyperparameters selected using grid-search as outlined in 4.2.3. The comparison results are shown in Table 28 in Appendix C.

When averaging over all the datasets, shown in Table 29 in Appendix C, the results suggest that – on average – most models benefit from using no maximum attempt count. SAKT and DKVMN-Paper are the exceptions that are appear unhindered by the use of maximum attempt count. The average differences between using and not using a maximum attempt count are subtle for all the DLKT models, however.

Upon inspection of the results for individual datasets, we find different patterns. In the ASSISTments 2015 dataset and in the Synthetic datasets (both K2 and K5), the differences are negligible to non-existent. In the case of the Synthetic datasets, this is explainable with the data having fewer attempts than the used maximum attempt counts. In other datasets, some noticeable differences can be observed. As an example, the largest difference in the ASSISTments 2009 dataset for DKVMN-Paper is 1.8% points in AUC in favor of using maximum attempt count (both cut 500 and split 200 perform similarly). For the ASSISTments 2017 dataset, the largest difference can be observed for LSTM-DKT, where there is a 2.7% point difference in AUC in favor of not using a maximum attempt count (when compared to cut 200). However, the drop in AUC from no maximum attempt count to the best split result (200) and best cut result (500) is 1.0% and 1.1% points respectively. Similarly, for Vanilla-DKT in the Statics dataset, there is a 3.1% point difference in AUC in favor of not using a maximum attempt count (when compared to cut 500). For no maximum attempt count versus best split (200), the difference is much smaller (0.7% points) but quite large (2.4% points) versus the best cut (200).

Notable differences are present also in other metrics, for instance 3.9% point difference in MCC for both DKVMN and LSMT-DKT in IntroProg (no maximum attempt count vs split 200). Furthermore, all the metrics do not show a certain approach consistently as the best. As an example, when considering the Statics dataset, the best performing option for DKVMN in terms of AUC or MCC is not using a maximum attempt count, while the best performing option for DKVMN in terms of Accuracy, F1-Score or RMSE is using cut 200. Averaged over all datasets, the metrics seem to tell a similar story as AUC. Although, there are slight differences, e.g. DKVMN is best without maximum attempt count according to all the metrics apart from F1-Score (precision and recall excluded).

5.2.4. Random Seed, Hardware and Software Version

As the random seed and the used hardware can also influence the performance of the models, we included two random seeds and two hardware types into the evaluation. As per software version matching challenges for GPU computation within the computing resources at our disposal (and also to examine potential effect of changing the machine learning framework version), we re-sorted to using TensorFlow 2.6.2 for the GPU calculations (instead of TensorFlow 2.1.0 that we used for CPU computation for the other analyses). To take this into account, when comparing the GPU and CPU, we evaluated the models on CPU with both TensorFlow 2.1.0 and 2.6.2. This

evaluation is conducted as a control to which we can compare the results from tuning the other hyperparameters that are more directly related to the models themselves.

Regarding random seed, the effects from changing the random seed are negligible (up to 0.1% points change) when looking at the best models. However, more considerable differences between the random seeds can be observed for the results with suboptimal hyperparameters. For instance, for SAKT with the ASSISTments 2015 dataset (Table 27 in Appendix B), the difference for the worst models is 1.8% points. In effect, this indicates that the effect of the random seed is larger for non-optimal hyperparameters, while the better models are more robust to the effect of the random seed. Moreover, when considering the standard deviation of the models' performance, there are noticeable differences between the models. As an example, the standard deviation of the models' performance when using different random seeds is noticeably higher for SAKT than for other models.

To consider the effect of hardware, we retrained the models on GPU with their optimal hyperparameters according to the grid search results for the AUC metric obtained on CPU trained models. The GPU versus CPU results are shown in Table 30 which can be found in the Appendix D. As noted previously, the GPU results were obtained using TensorFlow 2.6.2 as opposed to 2.1.0, which was used for the grid search. Due to this, in this evaluation, we also retrained the models with the best hyperparameters on CPU using TensorFlow 2.6.2.

Mostly, the differences between the two computation unit types are again subtle, ranging from 0.0 to 0.6% points (e.g. Statics dataset LSTM-DKT-S+ or SAKT in MCC score), similar to the random seed comparison. The differences are more often close to zero than above 0.2% points, especially for accuracy and RMSE. In general, we find slightly more differences between the TensorFlow versions on CPU than between CPU and GPU. The largest overall difference we find is 1.7% points (CPU-tf2.1.0 vs CPU-tf2.6.2) in F1 score in the ASSISTments 2017 dataset for Vanilla DKT. With the same TensorFlow versions the difference is only 0.1% points, when comparing CPU and GPU. For the same dataset and model, the AUC and MCC differences, 0.6% points and 0.9% points, are also notable.

5.2.5. Metrics and Determining the Best Model Hyperparameters

To consider the impact of tuning models for other metrics than the commonly used AUC score, we analyzed the model evaluation results for all the hyperparameter variations for each model and dataset. We conducted this to gain insight on how the choice of the metric that is used to pick the best model out of the trained models affects the performance measured by other metrics. These results are summarized in Figure 8, which shows the mean and max losses over our evaluated datasets and models for a given metric when some other metric is used to pick the best trained model among models with different hyperparameter options. For this metric comparison, we included the results of the maximum attempt count analysis, which is discussed in the next section. In the metric comparison, in addition to the evaluation metrics, we also include log loss that was used in model training.

Overall, picking a model based on the performance measured using a specific metric most likely does not mean that the model is the best when considering the other metrics. On average, the mean losses are small but noticeable, although in some cases the mean losses can be measured in multiple percentage points. For example, when picking a model based on F1-score and then looking at the MCC score, the mean loss is 1.5% points. For the other options, the mean losses are under 1% point, and mostly under 0.5% points. When considering the max loss,

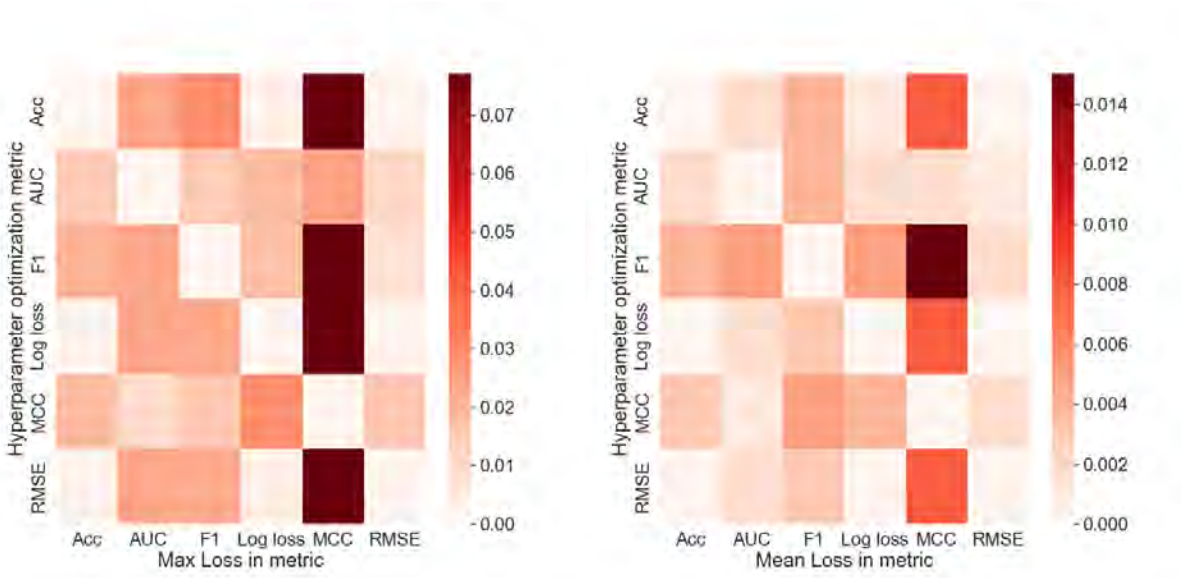

Figure 8: Mean and maximum differences in metric scores compared to optimal over datasets and DLKT models for using different metrics to select the "best" hyperparameters. The loss indicates how much lower scores for other metrics can be expected when choosing a metric for hyperparameter optimization.

i.e. highest loss when optimizing a specific metric and considering another metric within all the models and the datasets, the differences can be large. For example, when picking a model based on Accuracy, F1-Score, RMSE, or Log loss, and then looking at the MCC, the maximum loss is 7.7% points. When picking a model based on Accuracy and then looking at the F1-Score, the maximum loss is 3.0% points (similar to picking MCC and looking at Log loss). In most cases, the maximum loss is under 2.5% points.

When focusing on AUC, which is often used as the main comparison metric in knowledge tracing studies, also consequently in our replication study, we observe that if AUC is used for hyperparameter optimization, we sacrifice up to 2.5% points in MCC and up to 1.5% points in Accuracy and F1-Score. In other words, using another hyperparameter set for the same model and dataset would achieve 2.5% points higher MCC than the hyperparameter set used to obtain the best AUC score.

To summarize, according to our results, there does not appear to be a metric that would be optimal for optimizing all metrics, as all metrics when used for optimization risk losses for at least one other metric. The optimization metric comparison does however suggest that F1-score conveys the most risk, as evidenced by the mean losses which are the highest out of the studied metrics. On the other hand, looking at the max losses, all of the metrics convey a risk.

6. Discussion

6.1. Evaluation Results - No Silver Bullet

Our first two research questions relate to (RQ1) baseline and DLKT model performance, and (RQ2) to what extent this is data dependent. Overall, when comparing the model performance,

no single model consistently outperformed all the other models in all the metrics. The best performance is observed for the DLKT models, especially LSTM-DKT and DKVMN. The GLR also performed on par with the DLKT models in some datasets, while falling behind in others. The performance of BKT is typically better than the naive baseline models (mean and variants of next as previous), but worse than DLKT models and GLR. We acknowledge, however, that our BKT implementation is not heavily tuned. In addition, we point out that the naive baselines included in this study performed relatively well in one of the datasets, which showcases the benefit of including simple baselines when reporting results of more complex models. In ASSISTments 2015, the F1-score of Mean baseline model is the same as for the best performing model which is GLR. This indicates that in such cases, the usefulness of the best models can be questionable even if they outperform other complex models when they can not significantly outperform extremely simple statistical models. The observed high performance of the naive baselines in ASSISTments 2015 is possibly due to high skew in combination with relatively low number of attempts per student.

When considering the evolution of the knowledge tracing field, our evaluation and recent other evaluations of DLKT models (Gervet et al., 2020; Mandalapu et al., 2021; Pandey et al., 2021) show that the introduction of DLKT to the field has clearly advanced the performance of knowledge tracing. The major contributing factor seems to be the use of deep learning methodologies in general, and recent work that has focused on further exploration of deep learning for knowledge tracing. Thus, the move from simpler models to deep learning models has shown robust and verified improvements in knowledge tracing performance. Multiple more recent approaches appear promising (Nakagawa et al., 2019; Liu et al., 2019; Ghosh et al., 2020; Choi et al., 2020; Cheng et al., 2022; Pandey and Srivastava, 2020; Oya and Morishima, 2021; Shin et al., 2021; Song et al., 2021), and many claim significant performance improvements, but their results still require verification via replication studies.

Many of these newer models include slightly different inputs, such as skill and previous correctness as separate inputs (Choi et al., 2020), additional time related inputs (Shin et al., 2021) or leveraging both exercise and skill labels (Song et al., 2021), giving rise to the question of whether and how much the older architectures would also benefit from such input additions. We agree with this direction of exploring new inputs since DLKT models are powerful models that are likely to benefit from such additional information. Also, similar input modification can be seen in the move from DKT (Piech et al., 2015) to models that incorporate next skill id as additional input as in DKVMN (Zhang et al., 2017) and SAKT (Pandey and Karypis, 2019), although this addition does not appear to provide performance boost as evidenced by our experiments.

As a minor note, in our experiments, SAKT did not live up to the expectations laid out in the original article where the model was introduced. This is in line with prior results by Gervet et al. (Gervet et al., 2020) where SAKT also underperformed compared to the original article (Pandey and Karypis, 2019). However, newer studies that continued on the path started by SAKT of using self-attentive models have shown promising results: for example, RKT (Pandey and Srivastava, 2020), AKT (Ghosh et al., 2020) and SAINT+ (Shin et al., 2021).

Creating new models by introducing new types of inputs is a direction that has been witnessed also earlier in the KT field. As an example, Performance Factors Analysis (Pavlik et al., 2009) is an improvement over Learning Factors Analysis (Cen et al., 2006) that includes student ability as a separate input, achieving better performance. Similarly, there is a wide variety of ways in which BKT models can be improved by adding new inputs (Khajah et al., 2016), e.g.,

adding item difficulty as an input (Pardos and Heffernan, 2011). Effectively, as suggested by Khajah et al (Khajah et al., 2016), deep learning models can leverage regularities in the data that prior models cannot without adding such capability through explicitly added input features. As the deep learning models themselves are capable of performing feature engineering (LeCun et al., 2015), adding new inputs can provide them an easier starting point for this process.

On the other hand, one of the challenges of the introduction of DLKT models is the decrease in interpretability. As deep learning models are complex layered structures – effectively partially black boxes – the intricacies of the models are not easy to understand (Molnar, 2020). During training, connections between neurons in the layers are re-weighted and re-evaluated to optimize the output. This effectively means that there is a vast amount of trainable parameters, i.e., weights. For example, most of the models evaluated in this study have tens of thousands of weights that are updated during training. This leads to difficulty in, for instance, interpreting the effect of individual input features on the model outputs. Possible remedies for this include feature visualization, concept detection and finding influential instances (Molnar, 2020), although these often require forming hypotheses of what might work and what might not work and they remain heuristics for feature importance.

In contrast, when considering BKT or GLR, interpreting the results is more straightforward. For example, manual fine tuning of BKT through introduction of parameters can lead to performance comparable with DLKT models (Khajah et al., 2016), in addition to the state of being able to understand how the parameters are used. Similarly, when using GLR, ranking the importance of the features is straightforward. This has implications as understanding the models helps us to understand why some models perform better than others, as well as to understand – for example – contextual factors that contribute to the performance.

Due to this, DLKT models can lead to a disconnect between the use of learning theories and knowledge tracing. One could even see a link between using deep learning for knowledge tracing and using machine learning for natural language processing, where one of the famous quotes is "Every time I fire a linguist, the performance of the speech recognizer goes up" (Frederick Jelinek in the 1980s).

Key takeaways:

- No single model consistently outperforms all other models in all metrics. In our evaluation results, DKVMN and LSTM-DKT are most often ranked at the top and have the best performance on average.
- DLKT models in general come with an increase in model performance, at the cost of model interpretability. New models seem to emphasize leveraging inputs differently or adding additional inputs.
- Naive baseline models that are easy to implement and interpret – that is, models with little to no predictive power – can help understand the relative performance of more complex models.
- A prominent area for improving the performance of DLKT models is introducing different approaches to processing input and providing new input features.

6.2. Data - No One Model to Rule Them All

Related to our second research question (RQ2), focused on model performance on original evaluation data and new evaluation data, some models appear to be more capable of benefiting from big data than others. The dataset and its underlying distributions have a clear impact on model

performance. Data size is certainly a factor in DLKT model generalizability across datasets due to deep learning models' symbiotic relation with big data that follows from the models' tendency to overfit. In our results, we noticed no clear pattern with data size affecting model performance, and no model outperformed all the other models in all datasets. GLR and BKT fare relatively well compared to DLKT models in the ASSISTments 2015 dataset with the most students (19917) and the second most attempts (683k) but poorly in the ASSISTments 2017 dataset with the most attempts (943k). Also the SAKT model performs on par with the other DLKT models only in the IntroProg dataset which contains the fewest attempts (172k) within our datasets.

Other studies have come to different conclusions, which highlights that the impact of the data is still an open question. For example, Mandalapu et al. (Mandalapu et al., 2021) note that SAKT performs better than LSTM-DKT in larger datasets. In the same study though, a non-deep-learning logistic regression model beat both SAKT and LSTM-DKT even when supposedly deep learning excels on large data. Apart from data quantity, the underlying distribution of the data has a great influence on the performance of the models. Gervet et al. (Gervet et al., 2020) suggest that the number of learners per learning item or knowledge component (KC) is more important than the total number of interactions, and that some models benefit more from large amount of training data than others.

Similar to other studies, we also used synthetic (Synthetic-K5 and Synthetic-K2) datasets in our study. One interesting phenomenon we noticed (see Figure 7) is that the AUCs for the synthetic datasets seem to be more stable between different deep knowledge tracing models, i.e. the differences in AUC and other metrics are smaller, but still large compared to logistic regression and other baselines.

Our present work followed the methodology most commonly used in knowledge tracing studies where data from all students is used for training the models (accounting for train/test validation splits etc.). Some prior work has however suggested that one way to improve model performance would be to use a subset of the available data (Faraway and Augustin, 2018) – for example only a part of the students – to train the models, since some students seem to produce higher quality data than others (Alexandron et al., 2019; Yudelson et al., 2014), which can lead to model performance improvement over all students (Yudelson et al., 2014). This topic should also be further explored.

On a more general level, evaluations of knowledge tracing models typically rely on individual datasets. There are options in the machine learning domain that could potentially be used to better benefit from existing data; as an example, one could benefit from the use of domain adaptation and transfer learning techniques that have been successfully used in, for example, natural language processing (Zhuang et al., 2020). Some work already exists in adapting these methodologies for knowledge tracing (Cheng et al., 2022). However, we envision pre-trained knowledge tracing models similar to GPT-3 (Brown et al., 2020) that could then be fine-tuned with context-specific data.

Key takeaways:

- No single model outperformed all other models on all datasets.
- Even the best DLKT models did not always yield superior performance compared to other models in all datasets. The best model and also model type (DLKT vs non-DLKT) is dependent on data.
- When considering the adoption of knowledge tracing, to ensure the best fit for a specific

context, evaluate multiple models in that context instead of choosing the most recent state-of-the-art model.

6.3. Metrics in Reporting and Training - Not Just AUC

As we are measuring model performance (RQ1-RQ3), we should focus not only on the raw numbers but also on how these numbers are computed. A multitude of metrics for model evaluation exist and relying on a single metric can easily lead to misguided judgement of performance (National Research Council and Climate Research Committee, 2005). The usefulness of one metric over another depends on the task at hand, and as Gunawardana et al. (Gunawardana and Shani, 2009) state, "The decision on the proper evaluation metric is often critical, as each metric may favor a different algorithm". For instance, in identifying at-risk students, recall can be considered preferable over precision, since it can be argued that finding struggling students is preferable over finding non-struggling students. Misidentifying a well-performing student as a struggling student is not as costly as vice-versa, as the downside is that a well-performing student might be offered additional support that they do not need. Thus, in this case, tuning precision and recall in favor of recall might be more favorable than tuning precision and recall in favor of precision or with equal weights.

In knowledge tracing, we are not predicting dropping a course but rather single attempts at exercises, which makes the weighing of false positives and false negatives less clear. Although, without delving deeper into the matter, when considering the case of early intervention to help struggling students, similarly to retention prediction, it might be better to intervene more often than not. But the interventions still need to happen accurately enough to keep them valuable in the minds of learners.

DLKT studies often compare the performance of models using the AUC (ROC-AUC) metric which is also popular in other domains (e.g. medicine (Kim et al., 2017; Huang et al., 2019) and natural language processing (Pahikkala et al., 2009)). How AUC compares to accuracy, another popular metric, has drawn a lot of attention. Some formal and quantitative studies have shown AUC to be consistent with and more discriminative than accuracy (Ling et al., 2003; Huang and Ling, 2005; Halimu et al., 2019), and thus it has been claimed as superior. This, however, does not show that accuracy is worse than AUC, merely that it is less likely than AUC to show differences between the models. There is no guarantee that better AUC translates to better model (Jeni et al., 2013; Ozenne et al., 2015; Dhanani et al., 2014).

An often heard critique of accuracy (and also F1 metric), as opposed to e.g. AUC or MCC, is that high accuracy and F1 score can be a product of skewed data where positive labels outnumber negative labels. In such a case, the model may only predict positive labels well (Chicco and Jurman, 2020) and receive seemingly good metric scores, while still performing poorly with the minority class. AUC solves this issue by effectively accounting for skew in data. However, AUC with skewed data has also been argued to be a poor combination, since, in AUC, the minority class has the same impact on the metric score as the majority class (Ferri et al., 2009). Indeed, there is evidence that AUC may mask poor model performance (Jeni et al., 2013; Ozenne et al., 2015) and that AUC is less suitable than RMSE, for instance for BKT model evaluation (Dhanani et al., 2014).

One remedy to problems in AUC could be the visualization of the whole curve instead of reporting merely the area under it. However, drawing conclusions from the visualized curve also becomes more difficult at the same time, since comparing curves is not as straightforward

as numbers.

We argue that if we provide model accuracy and F1 scores alongside mean prediction with equally high scores, it easily breaks the illusion that a model with high accuracy is good. Accuracy and F1 scores can be good metrics with little risk of misinterpretation even on highly skewed datasets when mean or majority vote baseline is presented as comparison.

This leads to a dilemma in choosing the metrics to report and to determine model rankings. As an example of the metric choice dilemma, in our results for the ASSISTments 2015 dataset, DLKT models hold the best AUC and MCC scores by a significant margin. On the other hand, both the GLR and BKT models achieve almost the same performance on most other metrics. Also, the simple mean baseline is not far from the DLKT models in terms of accuracy, and the mean baseline holds the best F1-score tied with GLR. The DLKT models outperform GLR and BKT only on MCC and AUC which are the metrics often presented as alternatives to the "misleading" accuracy and other metrics that can be influenced by data skew.

In a related study, Effenberger et al. (Effenberger and Pelánek, 2020) evaluated the metrics MAE (mean absolute error) and RMSE (root mean squared error) in detail for student modeling. Similarly to our case, they reported cases where the choice of metric affected model ranking and also drew attention to the possibility of picking a "suitable" metric for a newly proposed model to make it appear better in comparison to previous models. They also showed that besides metric choice, the computation methodology (RMSE over whole data vs average over RMSEs per student data) of the metric may also affect model ranking.

Choosing a metric is not solely a problem in reporting, but also in hyperparameter tuning. When considering which metric to use when selecting the best hyperparameters for models via grid search, we observed that different hyperparameters may be chosen as the best when the metric used to choose them is changed. Consequently, the model with best hyperparameters according to one metric may not be optimal when considering other metrics. This problem is also noted by Sanyal et al. (Sanyal et al., 2020), who inspected how optimizing feature selection on different metrics influenced model performance and observed that metric selection can lead to large performance differences between datasets and selected features.

This further highlights the importance of choosing and understanding metrics for model comparison.

With scant rigorously studied information on evaluation metrics for knowledge tracing, and especially for DLKT, based on previous studies and our results, we suggest providing multiple metrics for evaluation. At least one unaffected by skew (e.g. AUC, MCC) and one affected by skew (e.g. Accuracy, F1-score) as proposed by (Jeni et al., 2013). In addition a generic error metric, such as RMSE, should be included for reliability, which is also suggested in (Liu et al., 2011). Providing Area Under Precision-Recall Curve (AUC-PC), which unlike AUC, is affected by skew but similarly to AUC is not affected by a decision threshold, could be a potential main metric to replace AUC since it has been shown to mask poor performance less than AUC (Ozenne et al., 2015; Saito and Rehmsmeier, 2015).

Key takeaways:

- Even though AUC is one of the most widely used metrics for evaluating model performance, with skewed data, it can mask poor model performance.
- Relying on a single metric in model evaluations can lead to misinformed decisions on model quality. Metrics that account and do not account for data imbalance should be used in model evaluations.

- The model that receives the best scores on a certain metric does not necessarily achieve the best results for other metrics. Thus, the metric that is used to determine the best model or best model hyperparameters is important to choose well, as well as to report for transparency.

6.4. Hyperparameters and Architecture Variations - Mind Your Randomness

Our third research question (RQ3) looked further into the variations in model architecture and hyperparameters, and examined how these affect model performance. Overall, hyperparameter tuning had a significant impact on model performance. This is to be expected as hyperparameter tuning heavily affects both overall model complexity (e.g. layer sizes affect the number of trainable parameters in a model) and model training itself (e.g. learning rate). Even though we explored a relatively small hyperparameter space (2 options per hyperparameter, as shown in Table 5), the differences between the worst and best hyperparameter combinations can be over 20% points (see e.g. Max-Min for SAKT in Table 25). The relative impact of hyperparameter tuning on the model performance depended on the model, as some models were less susceptible to their hyperparameters (see e.g. Tables 23-25). For instance, DKVMN is relatively stable over the hyperparameter combinations, while e.g., Vanilla-DKT is much less so.

In our exploration of the effect of different input (one-hotting vs. use of embedding layer) and output (skill summary layer and skill-to-scalar output vs. output per skill) variations for DLKT models, we identify two main findings. First, for many datasets and models there are mostly no differences in performance when using one-hot encoding when compared to using embedding layers. However, there are some exceptions where using embedding layers considerably outperforms one-hotting up to 4.6% point difference (for SAKT in the ASSISTments 2009 updated dataset in Table 26). This leaves little reason to one-hot inputs as opposed to using embedding layers, especially as one-hotting can cause memory problems when training models if the one-hot encoding become large due to high number of skills in data. On the other hand, while embedding layers appear to be the go-to choice when seeking maximal performance, using one-hot inputs may be a safer choice as they seem to be more robust regarding bad choice of hyperparameters in some datasets. Second, when considering output variations, using the more recent output version (skill layer and skills-to-scalar output layers) showed more variance in model performance when compared to the output per skill layer.

We also considered the impact of a maximum attempt count that has been incorporated in both DKVMN and SAKT (although the DKVMN article does not mention this). Both DKVMN and SAKT split student attempt sequence by a maximum attempt count, but neither article discussed this to an extent or analyzed the effect of the approach. To better understand how using a maximum attempt count affects KT model performance, we analyzed the effect of using maximum attempt count to split the student assignment sequences into smaller chunks thus artificially increasing the number of total students. This analysis was conducted by re-training our models using the best hyperparameters according to our grid search tuning with an additional hyperparameter: maximum attempt count filter method (none vs cut vs split). None indicates that no maximum attempt count was applied, cut indicates that the attempts after the maximum attempt count were discarded, and split effectively divides sequences longer than the maximum attempt count into multiple sequences that have at most maximum attempt count as their length. When testing the effect of the maximum attempt count, we used 200 and 500 as the choices for

maximum attempt count.

Overall, as shown in Appendix C, we found significant differences (e.g. up to 3.1% points AUC) in model performances depending on the filter method and some differences between the maximum attempt count 200 and 500, although not in all models in all datasets. For instance, in the Statics dataset, no model is unaffected by the additional hyperparameter tuning. The Synthetic datasets were not influenced by the filter as the attempt sequences are shorter than 200. These results showcase that maximum attempt count should not be overlooked when tuning hyperparameters and comparing knowledge tracing models. This is further emphasized by SAKT and Vanilla-DKT appearing to benefit from applying the maximum attempt count (split or cut) in comparison to other models. The other models perform better with no maximum attempt count while SAKT and Vanilla-DKT are on average much less affected. In other words, tuning maximum attempt count can bias evaluation in favor of certain models and this is something that should be accounted for and discussed. We do not, however, suggest that using maximum attempt count should be completely discouraged as it does speed up model training and reduce model space requirements, although there appears to be little other benefit according to our evaluation.

To quantify the effect of variation in performance due to properties not related to models, we also explored the effect of random seed and hardware (CPU vs GPU). First, we found that the values used as random seeds have very slight effects on model performance (see Table 27), although the effects are considerable on poor choice of hyperparameters. This indicates that the danger of randomness affecting model performance is present but mitigated by a large hyperparameter space for tuning. Second, we found mostly little to no differences in model performance depending on the hardware that was used to run the model (CPU or GPU). Some differences are larger, however. The biggest is 0.6% points difference for MCC and F1-Score, while the biggest difference in AUC is merely 0.3% points. On the other hand, we observed a tad more and larger differences between TensorFlow versions, with the largest being 1.7% points in F1-Score and the largest AUC score difference is 0.6% points. Thus, although unlikely based on our results, it is possible to have large performance difference when tuning hyperparameters on one machine learning framework version and then using the hyperparameters on another version another. Consequently, one should not take granted that the tuned hyperparameters work the same in another setting.

Even though the performance differences between the best models are usually negligible (e.g. 0.1% points AUC) when considering one of these non-model properties, the combination of the effects of hardware, seed, and hyperparameter tuning could easily change the ranking of the best performing models. This suggests that minor improvements in performance could be due to random chance, and thus to regard some model as the new "state-of-the-art", the improvement in e.g. AUC scores should be considerable and consistent across multiple datasets. Based on our results, we would be careful to consider even a 1% point increase in performance as a true improvement in terms of model architecture unless such an improvement was repeated in multiple studies and contexts.

Key takeaways:

- Hyperparameter tuning has a significant impact on model performance. Our results indicate that optimal hyperparameters are model- and dataset-specific.
- Input and output variations, which we used as hyperparameters but which could be pre- and post-processing steps, also impact model performance.

- Maximum attempt count filtering strategy influences model performance, and its effect depends on used model and dataset.
- Randomness (e.g. random seed, hardware) and machine learning framework version can affect model performance, although the observed impact in our evaluations is mostly very slight.

6.5. Replication Process and Findings - the Devil is in the Details

One of the key aspects of science is providing sufficient details of the used methodology that allows tracing the steps of the researchers to conduct similar research and to improve on it. Replication studies can be conducted in different ways (Ihantola et al., 2015; Patil et al., 2016), where one is seeking to reproduce earlier findings with the same data and the same methodology. In such a case, the objective is to examine the methodology to determine whether the steps are explained clearly enough and to explore whether there are considerations that were omitted from the original study. Earlier studies have suggested, however, that keeping the same data and methodology, but changing the researchers, can already lead to challenges with replication (Ihantola et al., 2015). Another approach to replication would be seeking to replicate the effect found in the original study with new data and possibly new or improved methodologies. In this case, the objective could be to study whether the data has an impact on the effect, and whether the effect generalizes beyond the original context.

Indeed, much of the presented work can be viewed as replication; reimplementing and comparing models from earlier work (RQ1), evaluating them on new data (RQ2) and analyzing how specific factors relate to performance (RQ3). To summarize our findings from the replication process, we found multiple issues, which are as follows. First, we observed differences between a model description in a published article and an associated code repository. Second, we observed methodological differences between a published article and an associated code repository. Third, we identified differences in data set sizes between articles, even though the data sets have been labeled the same and thus also likely understood to be the same. Finally, fourth, in some cases, we were unable to reach similar performance as reported in the original articles.

There are naturally a multitude of explanations for our findings. As an example, for the first and the second case, it is possible that the code that authors have added to their repository is an earlier version of their work, and does not represent the version reported in the article. This situation can be problematic however, as others may directly rely on the available implementations, instead of reimplementing the work based on the article. For the third case, it is possible that there are differences in data preprocessing steps that are not sometimes fully explained in the articles. We did, however, also observe a case where it is likely that data sizes were originally mistyped in an article and then copy-pasted to another article by different authors. For the fourth case, it is possible that some methodological steps have been omitted in the article, which could lead to better results. In this case, however, others have also struggled to replicate the earlier performance.

These findings follow the trend visible also in other fields, where researchers have identified problems with replicating results from prior published studies (Baker, 2016; Open Science Collaboration, 2015; Ioannidis, 2005b; Moonesinghe et al., 2007).

There are a lot of positive signals as well. As an example, authors often had placed their research code and used data in repositories for inspection, and there are multiple open datasets that can be used to evaluate model performance. Some authors also reported the hyperparameter

variations and preprocessing steps they had used when training the models, and the articles often included evaluations of proposed models against other recently proposed models over evaluating only against simpler baselines that are easy to outperform. One commendable work in this line of work is centralized algorithm and data repositories such as DataShop (Koedinger et al., 2010), which help reproduction, although considering replication, using the same data and methodology can disallow further examination of implementation details, which may lead to overlooking issues in implementation.

Our replication results show that old models can easily surpass newer ones and vice versa given the right hyperparameters. When introducing a new model that outperforms others, it is important to report the used hyperparameter options for both the new and the old models, and to verify that the better results are not due to more rigorous hyperparameter tuning for the introduced model. Similarly, data preprocessing decisions such as splitting student data based on maximum attempt count or omitting data by minimum attempt count can impact model performance, and even non-model specific properties such as hardware has an influence over the results.

Examples of the difficulty in replicating prior work can be easily found by looking at the reported performance of models. Some papers agree on the results of some models on certain datasets and disagree on the results of others. The degree of variance in observed performances is also relatively high, although the biggest differences could be explainable by the use of different versions of data.

In order to replicate machine learning work well, as much information as possible about the original work is beneficial, which is why we would like to emphasize the importance of source code accessibility. When the source code for a work is easy to find and analyze, many aspects of a machine learning model can be viewed that are often missing from a scientific paper. Whether missing pieces of implementation details are due to lack of rigor, estimated importance or a limitation of the publishing platform (e.g. paper length), they might include key components that explain why one model is better than the other.

This raises an interesting point about replicating machine learning work in general. Different types of expertise are needed to effectively work in the ML domain. On one hand, one needs to have sufficient mathematical skills to understand mathematical notations which are the most common way to communicate new ML models in academic publications. On the other hand, also good skills are required in programming and the frameworks being used in order to understand the source code of the models when those models are available openly. In most works, the mathematical notation of the model present in the publication hopefully matches the source code of the model. When this is not the case, it is not evident whether the researchers replicating the work should follow the mathematical notation (which likely includes the novel aspects of the model) or the source code (which likely was used to compute the results presented in the article). It is a good question whether conference organizers and journal editors should require reviewers to also review the source code and evaluate whether it matches the mathematical notation in the publication. We acknowledge that this can be a highly time-consuming process, however.

Key takeaways:

- In our study, we found multiple discrepancies in algorithm and data descriptions, which have the potential to influence study outcomes.
- Publishing source code and data is a good practice that should be continued. Preferably, links to code and data should be included in articles for ease of access and transparency. Further, we recommend linking to a specific version of code and data to verify that the

linked code is indeed the definitive version used in the study and to allow for further improvements while keeping the connection to the original article.

- When publishing studies, include methodological details that allow replication, including data preprocessing steps, evaluated hyperparameter options, and other specifics of model training.
- When presenting a new state-of-the-art model, other evaluated models should be evaluated with the same rigor (e.g. data preprocessing, hyperparameter tuning) as the proposed model to clarify that improvements stem from the proposed model algorithm and not from other factors.

6.6. Limitations of Work

Here, we summarize the key limitations of this work. First, we acknowledge that there are a wide variety of knowledge tracing models, including newer ones, that were not included in the empirical evaluation. Multiple of these are briefly discussed in section 2.1.3. Limiting the number of evaluated models was a deliberate choice, as we meticulously reimplemented the algorithms as well as compared and contrasted the available implementations with the details outlined in the respective articles.

Second, while we implemented most of the models compared in this study, for the baseline Bayesian Knowledge Tracing we used Yudelson's implementation that is available on GitHub[25]. Similarly, for the logistic regression, we used the Best-LR (GLR) model with slight modifications by Gervet et al. (Gervet et al., 2020). Their model, too, is available on GitHub[26]. Thus, our results related to BKT and GLR are not full replications (where the model would be reimplemented from scratch). We decided not to reimplement the models as our focus in this work is on scrutinizing DLKT models, furthermore BKT and GLR were only used as baselines to which we can compare the DLKT models.

Third, we have used seven datasets in this study, six publicly available ones and one novel dataset (IntroProg) which is published alongside this work. Thus, our results are only applicable to these datasets, and do not necessarily extend to other datasets. As an example, many recent studies have included the EdNet dataset (Choi et al., 2020) in their evaluations, including the studies by Mandalapu et al. (Mandalapu et al., 2021) and Pandey et al. (Pandey et al., 2021). When compared to the datasets used in this study, the EdNet dataset is larger and thus may allow the models to utilize information in a way that is not possible in the present datasets.

Fourth, we acknowledge that our explored hyperparameter space is not very extensive, and it is possible that further tuning could have realized better gains. All in all, in our study, there are between 72 and 240 hyperparameter variations per model, and in total, we evaluated 5,880 cross-validated DLKT models. With the resources available to us, significantly extending the explored hyperparameter space would have led to far longer model training time.

Fifth, although we explored a range of hyperparameters, we did not look into item-aware input and output, which also could influence the results. For an exploration of this aspect of knowledge tracing models, see e.g. studies by Gervet et al. (Gervet et al., 2020) and Vie and Kashima (Vie and Kashima, 2019).

Finally, we acknowledge that when selecting best hyperparameter values for models, we

[25] https://github.com/myudelson/hmm-scalable, accessed 2020-03-01

[26] https://github.com/theophilee/learner-performance-prediction, accessed 2021-05-27

selected them based on AUC, similarly as is done in e.g. (Zhang et al., 2017). As discussed in Section 5.2.5, using another metric to select the best models would have led to slightly different results. We decided against extensive exploration of selection metric when reporting results, as it would have led to more complex reporting (effectively multiplying the reported results by the number of metrics), and as AUC is commonly used as a principal metric in the knowledge tracing community.

7. Conclusion

In this article, we reviewed models for knowledge tracing, evaluating the performance of eleven models. We evaluated deep learning knowledge tracing (DLKT) models (Vanilla-DKT, LSTM-DKT, LSTM-DKT-S+, DKVMN, DKVMN-Paper, SAKT) and baseline models (Mean, Next as Previous, Next as Previous N's Mean, BKT, GLR). Out of these, LSTM-DKT-S+ is our own variant of LSTM-DKT, which takes next skills into account as inputs. For the DKVMN, the two versions are presented as the version in the article and the version in the repository differed from each other. See Table 1 for details of the models. All models were evaluated against seven datasets using seven metrics. For deep learning knowledge tracing models, we evaluated the impact of input and output variations (one-hot encoding versus embedding layer; output-per-skill layer versus skill layer and skills-to-scalar output layers), and also maximum attempt count handling on model performance. The effect of non-model properties such as hardware and random seed were examined as a baseline to which model improvements could be compared to. The deep learning knowledge tracing models were reimplemented for this study.

The motivation of our study was five-fold. First, to re-evaluate the proposed models of earlier studies by reimplementing them and comparing them to each other and to simple baselines. Second, to highlight how the choice of metric used in reporting affect perceived model performance and hyperparameter tuning. Third, to show how hyperparameters and variations in model input and output architecture can have an effect on models' performance. Fourth, to explore model performance across different contexts (datasets). Fifth, to emphasize the importance of replication studies and give pointers on how to make such studies easier, as well as to give pointers to help make results of future KT studies more robust. We also publish our implementations, datasets and evaluation code for use in future research.

To summarize, our research questions and their answers are as follows:

RQ1 How do DLKT models compare to naive baselines and non deep-learning KT models? **Answer:** The evaluated DLKT models generally outperform the baseline models. DLKT models in general outperform the non-DLKT baselines BKT and the logistic regression model GLR. GLR does, however, achieve performance on par with some of the DLKT models on two datasets and is the best performing model on one dataset. BKT is on par with the DLKT models and GLR on one dataset, in all metrics but AUC and MCC, but BKT fares worse on other datasets. Naive baselines mostly perform poorly when compared to the more complex models, but on one dataset, the accuracy of the best DLKT models is not much better than the Mean model, and the Mean model also achieves better F1-Score than the DLKT models. This highlights both the performance of the DLKT models as well as the importance of including naive baseline models into KT model comparisons to verify usefulness of models.

RQ2 How do DLKT models perform on the same and different datasets as originally evaluated with? **Answer:** In our evaluations, LSTM-DKT, LSTM-DKT-S+ and DKVMN had the best performance on average out of the evaluated models, but the differences in performance of

any of the DLKT models were not great. We found that the relative performance of the models depended on the context, i.e. the dataset. We did not find differences in performance between LSTM-DKT and our variant with additional next skill input LSTM-DKT-S+, but we found that the DKVMN version implemented based on the DKVMN authors' repository performed on average slightly better than the DKVMN-Paper version, which is the version introduced in the article. We found considerable variance between previously reported results for our evaluated models, especially for SAKT, which is in line with other prior comparison studies. When comparing the results over different metrics, we observed that the ranking of the models can differ depending on the inspected metric, which highlights the importance of reporting multiple metrics for model evaluation. We also analyzed the effect of using different metrics for hyperparameter tuning and found the same pattern there; the model which receives the best score on one metric does not always receive the best score on another metric.

RQ3 What is the impact of variations in architecture and hyperparameters on DLKT models' performance? **Answer:** Overall, the impact of variations in hyperparameters contributes significantly to model performance. The extent to which model performance depends on hyperparameters could be seen as a quality factor of the models, where more robust models are less dependent on extensive hyperparameter tuning. Explored variations in model architecture, one-hot encoding vs embedding layer for input, and output per skill layer vs skill layer and skills-to-scalar output layers, showed some variation in performance, up to 4.6% point AUC. Non-model properties, i.e. hardware, machine learning framework version, and random seed, had mostly negligible impact on model performance. Although, we found some cases where machine learning framework version and the hardware led to over 0.5% point difference in results in various metrics. Maximum attempt count filtering (no filtering, cut, split – split has been used implicitly and explicitly in some previous work) had also a noticeable impact in model performance, and should be taken into account and reported if used when training models.

As a part of the work, we reimplemented the deep learning knowledge tracing algorithms following the details presented in the respective articles. During the implementation, we observed inconsistencies between algorithm descriptions and implementations, as well as inconsistencies between the reported dataset sizes. As an example, the architecture of the DKVMN differed between the article and the implementation, and the ASSISTments 2015 dataset had nearly 10% difference in size between the articles likely due to varying preprocessing steps.

Furthermore, we highlight the need to identify the sources for empirical gains, which has been pointed out to be a concern within the machine learning discipline (Lipton and Steinhardt, 2019): in our evaluations, we observed that some of the previous findings reported in the literature may have more to do with hyperparameter tuning than proposed neural network structures.

We call out others to also perform similar studies where KT models are evaluated through replication and reimplementation of the models, where the preprocessing of the data and any possible hyperparameter tuning is explicitly stated and performed with the same level of rigor for all compared models.

ACKNOWLEDGEMENTS

We acknowledge the computational resources provided by the Aalto Science-IT project. We are grateful for the grant by the Media Industry Research Foundation of Finland which partially funded this work. We thank the reviewers for their valuable comments that helped improve this manuscript.

REFERENCES

ABDELRAHMAN, G. AND WANG, Q. 2019. Knowledge tracing with sequential key-value memory networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Y. Maarek, J.-Y. Nie, and F. Scholer, Eds. ACM Press, New York, NY, USA, 175–184.

AHADI, A., HELLAS, A., IHANTOLA, P., KORHONEN, A., AND PETERSEN, A. 2016. Replication in computing education research: Researcher attitudes and experiences. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, J. Sheard and C. S. Montero, Eds. Koli Calling '16. Association for Computing Machinery, New York, NY, USA, 2–11.

ALEXANDRON, G., YOO, L. Y., RUIPÉREZ-VALIENTE, J. A., LEE, S., AND PRITCHARD, D. E. 2019. Are MOOC learning analytics results trustworthy? with fake learners, they might not be! *International Journal of Artificial Intelligence in Education 29,* 4, 484–506.

ANDERSON, C. J., ŠTĚPÁN BAHNÍK, BARNETT-COWAN, M., BOSCO, F. A., CHANDLER, J., CHARTIER, C. R., CHEUNG, F., CHRISTOPHERSON, C. D., CORDES, A., CREMATA, E. J., PENNA, N. D., ESTEL, V., FEDOR, A., FITNEVA, S. A., FRANK, M. C., GRANGE, J. A., HARTSHORNE, J. K., HASSELMAN, F., HENNINGER, F., VAN DER HULST, M., JONAS, K. J., LAI, C. K., LEVITAN, C. A., MILLER, J. K., MOORE, K. S., MEIXNER, J. M., MUNAFÒ, M. R., NEIJENHUIJS, K. I., NILSONNE, G., NOSEK, B. A., PLESSOW, F., PRENOVEAU, J. M., RICKER, A. A., SCHMIDT, K., SPIES, J. R., STIEGER, S., STROHMINGER, N., SULLIVAN, G. B., VAN AERT, R. C. M., VAN ASSEN, M. A. L. M., VANPAEMEL, W., VIANELLO, M., VORACEK, M., AND ZUNI, K. 2016. Response to comment on "estimating the reproducibility of psychological science". *Science 351,* 6277, 1037–1037.

ANDERSON, J. R., BOYLE, C. F., AND REISER, B. J. 1985. Intelligent tutoring systems. *Science 228,* 4698, 456–462.

ASENDORPF, J. B., CONNER, M., DE FRUYT, F., DE HOUWER, J., DENISSEN, J. J. A., FIEDLER, K., FIEDLER, S., FUNDER, D. C., KLIEGL, R., NOSEK, B. A., PERUGINI, M., ROBERTS, B. W., SCHMITT, M., VAN AKEN, M. A. G., WEBER, H., AND WICHERTS, J. M. 2013. Recommendations for increasing replicability in psychology. *European Journal of Personality 27,* 2, 108–119.

BAKER, M. 2016. 1,500 scientists lift the lid on reproducibility. *Nature 533,* 7604 (May), 452–454.

BAKER, R. S. J. D., CORBETT, A. T., AND ALEVEN, V. 2008. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *International Conference on Intelligent Tutoring Systems*, B. P. Woolf, E. Aïmeur, R. Nkambou, and S. Lajoie, Eds. Springer, 406–415.

BEGLEY, C. G. AND ELLIS, L. M. 2012. Drug development: Raise standards for preclinical cancer research. *Nature 483,* 7391, 531–533.

BHANDARI NEUPANE, J., NEUPANE, R. P., LUO, Y., YOSHIDA, W. Y., SUN, R., AND WILLIAMS, P. G. 2019. Characterization of leptazolines a–d, polar oxazolines from the cyanobacterium leptolyngbya sp., reveals a glitch with the "willoughby–hoye" scripts for calculating nmr chemical shifts. *Organic Letters 21,* 20, 8449–8453.

BIANCHINI, M. AND SCARSELLI, F. 2014. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems 25,* 8, 1553–1565.

BOUTHILLIER, X., DELAUNAY, P., BRONZI, M., TROFIMOV, A., NICHYPORUK, B., SZETO, J., MOHAMMADI SEPAHVAND, N., RAFF, E., MADAN, K., VOLETI, V., EBRAHIMI KAHOU, S., MICHALSKI, V., ARBEL, T., PAL, C., VAROQUAUX, G., AND VINCENT, P. 2021. Accounting

for variance in machine learning benchmarks. In *Proceedings of Machine Learning and Systems*, A. Smola, A. Dimakis, and I. Stoica, Eds. Vol. 3. 747–769.

BROWN, T., MANN, B., RYDER, N., SUBBIAH, M., KAPLAN, J. D., DHARIWAL, P., NEELAKANTAN, A., SHYAM, P., SASTRY, G., ASKELL, A., AGARWAL, S., HERBERT-VOSS, A., KRUEGER, G., HENIGHAN, T., CHILD, R., RAMESH, A., ZIEGLER, D., WU, J., WINTER, C., HESSE, C., CHEN, M., SIGLER, E., LITWIN, M., GRAY, S., CHESS, B., CLARK, J., BERNER, C., MCCANDLISH, S., RADFORD, A., SUTSKEVER, I., AND AMODEI, D. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds. Curran Associates, Inc., 1877–1901.

CARUANA, R. AND NICULESCU-MIZIL, A. 2004. Data mining in metric space: An empirical analysis of supervised learning performance criteria. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, J. Gehrke and W. DuMouchel, Eds. KDD '04. Association for Computing Machinery, New York, NY, USA, 69–78.

CEN, H., KOEDINGER, K., AND JUNKER, B. 2006. Learning factors analysis–a general method for cognitive model evaluation and improvement. In *Proceedings of 8th International Conference on Intelligent Tutoring Systems*, M. Ikeda, K. D. Ashley, and T.-W. Chan, Eds. Springer, 164–175.

CHANG, K.-M., BECK, J. E., MOSTOW, J., AND CORBETT, A. 2006. Does help help? a Bayes net approach to modeling tutor interventions. In *AAAI Workshop on Educational Data Mining*. AAAI, 41–46.

CHENG, S., LIU, Q., CHEN, E., ZHANG, K., HUANG, Z., YIN, Y., HUANG, X., AND SU, Y. 2022. Adaptkt: A domain adaptable method for knowledge tracing. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, L. Akoglu, X. L. Dong, and J. Tang, Eds. WSDM '22. Association for Computing Machinery, New York, NY, USA, 123–131.

CHICCO, D. AND JURMAN, G. 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics 21,* 1, 1–13.

CHOFFIN, B., POPINEAU, F., BOURDA, Y., AND VIE, J.-J. 2019. Das3h: Modeling student learning and forgetting for optimally scheduling distributed practice of skills. In *Proceedings of The 12th International Conference on Educational Data Mining*, C. F. Lynch, A. Merceron, M. Desmarais, and R. Nkambou, Eds. International Educational Data Mining Society, 29–38.

CHOI, Y., LEE, Y., CHO, J., BAEK, J., KIM, B., CHA, Y., SHIN, D., BAE, C., AND HEO, J. 2020. Towards an appropriate query, key, and value computation for knowledge tracing. In *Proceedings of the 7th ACM Conference on Learning at Scale*, R. Kizilcec and S. Singer, Eds. Association for Computing Machinery, 341–344.

CHOI, Y., LEE, Y., SHIN, D., CHO, J., PARK, S., LEE, S., BAEK, J., BAE, C., KIM, B., AND HEO, J. 2020. EdNet: A large-scale hierarchical dataset in education. In *International Conference on Artificial Intelligence in Education*, I. I. Bittencourt, M. Cukurova, K. Muldner, R. Luckin, and E. Millán, Eds. Springer, 69–73.

CORBETT, A. T. AND ANDERSON, J. R. 1994. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction 4,* 4, 253–278.

DEVASENA, C. L., SUMATHI, T., GOMATHI, V., AND HEMALATHA, M. 2011. Effectiveness evaluation of rule based classifiers for the classification of iris data set. *Bonfring International Journal of Man Machine Interface 1,* Inaugural Special Issue, 05–09.

DHANANI, A., LEE, S. Y., PHOTHILIMTHANA, P. M., AND PARDOS, Z. 2014. A comparison of error metrics for learning model parameters in bayesian knowledge tracing. In *Workshop Approaching Twenty Years of Knowledge Tracing (BKT20y)*, S. G. Santos and O. C. Santos, Eds. CEUR-WS, 153–155.

DING, X. AND LARSON, E. C. 2019. Why deep knowledge tracing has less depth than anticipated. In *Proceedings of the 12th International Conference on Educational Data Mining*, C. F. Lynch, A. Merceron, M. Desmarais, and R. Nkambou, Eds. International Educational Data Mining Society, 282–287.

DOZAT, T. 2016. Incorporating nesterov momentum into adam. *http://cs229.stanford.edu/proj2015/054_report.pdf*.

EFFENBERGER, T. AND PELÁNEK, R. 2020. Impact of methodological choices on the evaluation of student models. In *International Conference on Artificial Intelligence in Education*, I. I. Bittencourt, M. Cukurova, K. Muldner, R. Luckin, and E. Millán, Eds. Springer, 153–164.

FANELLI, D. 2011. Negative results are disappearing from most disciplines and countries. *Scientometrics 90,* 3, 891–904.

FARAWAY, J. J. AND AUGUSTIN, N. H. 2018. When small data beats big data. *Statistics & Probability Letters 136*, 142–145.

FAWCETT, T. 2006. An introduction to ROC analysis. *Pattern Recognition Letters 27,* 8, 861–874.

FENG, M., HEFFERNAN, N., AND KOEDINGER, K. 2009. Addressing the assessment challenge with an online system that tutors as it assesses. *User Modeling and User-Adapted Interaction 19,* 3, 243–266.

FERRI, C., HERNÁNDEZ-ORALLO, J., AND MODROIU, R. 2009. An experimental comparison of performance measures for classification. *Pattern Recognition Letters 30,* 1, 27–38.

GAL, Y. AND GHAHRAMANI, Z. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Curran Associates Inc., Red Hook, NY, USA, 1027–1035.

GARDNER, J., YANG, Y., BAKER, R. S., AND BROOKS, C. 2019. Modeling and experimental design for mooc dropout prediction: A replication perspective. In *Proceedings of The 12th International Conference on Educational Data Mining*, C. F. Lynch, A. Merceron, M. Desmarais, and R. Nkambou, Eds. International Educational Data Mining Society.

GERS, F. A. AND SCHMIDHUBER, E. 2001. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks 12,* 6, 1333–1340.

GERVET, T., KOEDINGER, K., SCHNEIDER, J., AND MITCHELL, T. 2020. When is deep learning the best approach to knowledge tracing? *Journal of Educational Data Mining 12,* 3, 31–54.

GHAHRAMANI, Z. 1997. Learning dynamic bayesian networks. In *International School on Neural Networks, Initiated by IIASS and EMFCSC*. Springer, 168–197.

GHOSH, A., HEFFERNAN, N., AND LAN, A. S. 2020. Context-aware attentive knowledge tracing. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, M. Shah and S. Rajan, Eds. Association for Computing Machinery, 2330–2339.

GILBERT, D. T., KING, G., PETTIGREW, S., AND WILSON, T. 2016. Comment on "estimating the reproducibility of psychological science". *Science 351,* 6277, 1037.

GOLDEN, M. A. 1995. Replication and non-quantitative research. *PS: Political Science & Politics 28,* 03, 481–483.

GONG, Y., BECK, J. E., AND HEFFERNAN, N. T. 2010. Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. In *International Conference on Intelligent Tutoring Systems*, V. Aleven, J. Kay, and J. Mostow, Eds. Springer, 35–44.

GONZÁLEZ-BRENES, J., HUANG, Y., AND BRUSILOVSKY, P. 2014. General features in knowledge tracing to model multiple subskills, temporal item response theory, and expert knowledge. In *Proceed-*

ings of the 7th International Conference on Educational Data Mining, J. C. Stamper, Z. A. Pardos, M. Mavrikis, and B. M. McLaren, Eds. International Educational Data Mining Society, 84–91.

GRAVES, A. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

GRAVES, A., WAYNE, G., AND DANIHELKA, I. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

GUNAWARDANA, A. AND SHANI, G. 2009. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research 10,* 12, 2935–2962.

HALIMU, C., KASEM, A., AND NEWAZ, S. S. 2019. Empirical comparison of area under ROC curve (AUC) and mathew correlation coefficient (mcc) for evaluating machine learning algorithms on imbalanced datasets for binary classification. In *Proceedings of the 3rd International Conference on Machine Learning and Soft Computing*, T. Le-Tien, P. Terenziani, and D. Minh-Son, Eds. Association for Computing Machinery, 1–6.

HAMBLETON, R. K. AND SWAMINATHAN, H. 1985. *Item response theory: Principles and applications*. Springer.

HEFFERNAN, N. T. AND HEFFERNAN, C. L. 2014. The assistments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education 24,* 4, 470–497.

HEFFERNAN, N. T., TURNER, T. E., LOURENCO, A. L., MACASEK, M. A., NUZZO-JONES, G., AND KOEDINGER, K. R. 2006. The assistment builder: Towards an analysis of cost effectiveness of its creation. In *Proceedings of the 19th International Florida Artificial Intelligence Research Society Conference Conference*, G. C. J. Sutcliffe and R. G. Goebel, Eds. AAAI, 515–520.

HOCHREITER, S. AND SCHMIDHUBER, J. 1997a. Long short-term memory. *Neural Computation 9,* 8, 1735–1780.

HOCHREITER, S. AND SCHMIDHUBER, J. 1997b. LSTM can solve hard long time lag problems. In *Advances in Neural Information Processing Systems 9*, M. Mozer, M. Jordan, and T. Petsche, Eds. MIT Press, 473–479.

HUANG, J. AND LING, C. X. 2005. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on knowledge and Data Engineering 17,* 3, 299–310.

HUANG, L., SHEA, A. L., QIAN, H., MASURKAR, A., DENG, H., AND LIU, D. 2019. Patient clustering improves efficiency of federated machine learning to predict mortality and hospital stay time using distributed electronic medical records. *Journal of Biomedical Informatics 99*, 103291.

IHANTOLA, P., VIHAVAINEN, A., AHADI, A., BUTLER, M., BÖRSTLER, J., EDWARDS, S. H., ISOHANNI, E., KORHONEN, A., PETERSEN, A., RIVERS, K., RUBIO, M. A., SHEARD, J., SKUPAS, B., SPACCO, J., SZABO, C., AND TOLL, D. 2015. Educational data mining and learning analytics in programming: Literature review and case studies. In *Proceedings of the 2015 ITiCSE on Working Group Reports*, N. Ragonis and P. Kinnunen, Eds. Association for Computing Machinery, New York, NY, USA, 41–63.

IOANNIDIS, J. P. 2005a. Contradicted and initially stronger effects in highly cited clinical research. *Jama 294,* 2, 218–228.

IOANNIDIS, J. P. 2005b. Why most published research findings are false. *PLoS Medicine 2,* 8, e124.

IOANNIDIS, J. P., MUNAFO, M. R., FUSAR-POLI, P., NOSEK, B. A., AND DAVID, S. P. 2014. Publication and other reporting biases in cognitive sciences: detection, prevalence, and prevention. *Trends in Cognitive Sciences 18,* 5, 235–241.

JENI, L. A., COHN, J. F., AND DE LA TORRE, F. 2013. Facing imbalanced data–recommendations for the use of performance metrics. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, A. Niholt, M. Pantic, and S. D'Mello, Eds. IEEE, 245–251.

JOHNS, J., MAHADEVAN, S., AND WOOLF, B. 2006. Estimating student proficiency using an item response theory model. In *International Conference on Intelligent Tutoring Systems*, M. Ikeda, K. D. Ashley, and T.-W. Chan, Eds. Springer, 473–480.

KAMIJO, K. AND TANIGAWA, T. 1990. Stock price pattern recognition-a recurrent neural network approach. In *Proceedings of the International Joint Conference on Neural Networks*, W. Freeman and B. Kosko, Eds. Vol. 1. IEEE, 215–221.

KÄSER, T., KLINGLER, S., SCHWING, A. G., AND GROSS, M. 2014. Beyond knowledge tracing: Modeling skill topologies with bayesian networks. In *International Conference on Intelligent Tutoring Systems*, S. Trausan-Matu, K. E. Boyer, M. Crosby, and K. Panourgia, Eds. Springer, 188–198.

KHAJAH, M., LINDSEY, R. V., AND MOZER, M. C. 2016. How deep is knowledge tracing? In *Proceedings of the 9th International Conference on Educational Data Mining*, T. Barnes, M. Chi, and M. Feng, Eds. International Educational Data Mining Society, 94–101.

KHAJAH, M., WING, R., LINDSEY, R., AND MOZER, M. 2014. Integrating latent-factor and knowledge-tracing models to predict individual differences in learning. In *Proceedings of the 7th International Conference on Educational Data Mining*, J. C. Stamper, Z. A. Pardos, M. Mavrikis, and B. M. McLaren, Eds. International Educational Data Mining Society, 99–106.

KHAJAH, M. M., HUANG, Y., GONZÁLEZ-BRENES, J. P., MOZER, M. C., AND BRUSILOVSKY, P. 2014. Integrating knowledge tracing and item response theory: A tale of two frameworks. In *CEUR Workshop Proceedings*, I. Cantador, M. Chi, R. Farzan, and R. Jäschke, Eds. Vol. 1181. CEUR-WS, 7–15.

KIM, S. J., CHO, K. J., AND OH, S. 2017. Development of machine learning models for diagnosis of glaucoma. *PLoS One 12,* 5, e0177726.

KOEDINGER, K. R., BAKER, R. S., CUNNINGHAM, K., SKOGSHOLM, A., LEBER, B., AND STAMPER, J. 2010. A data repository for the EDM community: The PSLC DataShop. *Handbook of Educational Data Mining 43*, 43–56.

LALWANI, A. AND AGRAWAL, S. 2017. Few hundred parameters outperform few hundred thousand. In *Proceedings of the 10th International Conference on Educational Data Mining*, X. Hu, T. Barnes, A. Hershkovitz, and L. Paquette, Eds. Vol. 17. International Educational Data Mining Society, 448–453.

LECUN, Y., BENGIO, Y., AND HINTON, G. 2015. Deep learning. *nature 521,* 7553, 436–444.

LIN, C. AND CHI, M. 2016. Intervention-BKT: incorporating instructional interventions into bayesian knowledge tracing. In *International Conference on Intelligent Tutoring Systems*, A. Micarelli, J. Stamper, and K. Panourgia, Eds. Springer, 208–218.

LIN, C. AND CHI, M. 2017. A comparisons of BKT, RNN and LSTM for learning gain prediction. In *International Conference on Artificial Intelligence in Education*, E. André, R. Baker, X. Hu, M. M. T. Rodrigo, and B. du Boulay, Eds. Springer, 536–539.

LING, C. X., HUANG, J., AND ZHANG, H. 2003. AUC: a statistically consistent and more discriminating measure than accuracy. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*. Vol. 3. Morgan Kaufmann, 519–524.

LIPTON, Z. C. AND STEINHARDT, J. 2019. Troubling trends in machine learning scholarship: Some ml papers suffer from flaws that could mislead the public and stymie future research. *Queue 17,* 1 (feb), 45–77.

LIU, C., WHITE, M., AND NEWELL, G. 2011. Measuring and comparing the accuracy of species distribution models with presence–absence data. *Ecography 34,* 2, 232–243.

LIU, Q., HUANG, Z., YIN, Y., CHEN, E., XIONG, H., SU, Y., AND HU, G. 2019. EKT: Exercise-aware knowledge tracing for student performance prediction. *IEEE Transactions on Knowledge and Data Engineering 33,* 1, 100–115.

LIU, Q., TONG, S., LIU, C., ZHAO, H., CHEN, E., MA, H., AND WANG, S. 2019. Exploiting cognitive structure for adaptive learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Y. Li, R. Rosales, E. Terzi, and G. Karypis, Eds. Association for Computing Machinery, 627–635.

LOBO, J. M., JIMÉNEZ-VALVERDE, A., AND REAL, R. 2008. AUC: a misleading measure of the performance of predictive distribution models. *Global Ecology and Biogeography 17,* 2, 145–151.

LUONG, T., PHAM, H., AND MANNING, C. D. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, L. Màrquez, C. Callison-Burch, and J. Su, Eds. Association for Computational Linguistics, Lisbon, Portugal, 1412–1421.

MA, W., ADESOPE, O. O., NESBIT, J. C., AND LIU, Q. 2014. Intelligent tutoring systems and learning outcomes: A meta-analysis. *Journal of Educational Psychology 106,* 4, 901.

MACKEY, A. 2012. Why (or why not), when and how to replicate research. *Replication research in applied linguistics 2146*, 21–46.

MAKEL, M. C., PLUCKER, J. A., AND HEGARTY, B. 2012. Replications in psychology research how often do they really occur? *Perspectives on Psychological Science 7,* 6, 537–542.

MANDALAPU, V., GONG, J., AND CHEN, L. 2021. Do we need to go deep? knowledge tracing with big data. In *AAAI-2021 Workshop on AI Education: "Imagining Post-COVID Education with AI"*, N. Heffernan and P. Kim, Eds. AAAI. Available at https://arxiv.org/abs/2101.08349.

MAO, Y., LIN, C., AND CHI, M. 2018. Deep learning vs. bayesian knowledge tracing: Student models for interventions. *Journal of Educational Data Mining 10,* 2, 28–54.

MIKOLOV, T., KOMBRINK, S., BURGET, L., ČERNOCKÝ, J., AND KHUDANPUR, S. 2011. Extensions of recurrent neural network language model. In *Proceedings of the 2011 IEEE International Conference on Acoustics, Speech and Signal Processing*. 5528–5531.

MILLER, A., FISCH, A., DODGE, J., KARIMI, A.-H., BORDES, A., AND WESTON, J. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, J. Su, K. Duh, and X. Carreras, Eds. Association for Computational Linguistics, Austin, Texas, 1400–1409.

MOLNAR, C. 2020. *Interpretable machine learning*. Leanpub.

MONTERO, S., ARORA, A., KELLY, S., MILNE, B., AND MOZER, M. 2018. Does deep knowledge tracing model interactions among skills? In *Proceedings of the 11th International Conference on Educational Data Mining*, K. E. Boyer and M. Yudelson, Eds. International Educational Data Mining Society, 462–466.

MOONESINGHE, R., KHOURY, M. J., AND JANSSENS, A. C. J. 2007. Most published research findings are false – but a little replication goes a long way. *PLoS Medicine 4,* 2, e28.

MUMA, J. R. 1993. The need for replication. *Journal of Speech, Language, and Hearing Research 36,* 5, 927–930.

MUSCHELLI, J. 2020. Roc and auc with a binary predictor: a potentially misleading metric. *Journal of Classification 37,* 3, 696–708.

NAKAGAWA, H., IWASAWA, Y., AND MATSUO, Y. 2019. Graph-based knowledge tracing: Modeling student proficiency using graph neural network. In *IEEE/WIC/ACM International Conference on Web Intelligence*, P. Barnaghi, G. Gottlob, Y. Manolopoulos, T. Tzouramanis, and A. Vakali, Eds. WI '19. Association for Computing Machinery, 156–163.

NATIONAL RESEARCH COUNCIL AND CLIMATE RESEARCH COMMITTEE. 2005. Chapter 3: Principles for developing metrics. In *Thinking Strategically: The Appropriate Use of Metrics for the Climate Change Science Program*. The National Academies Press, 47–54.

NWANA, H. S. 1990. Intelligent tutoring systems: an overview. *Artificial Intelligence Review 4,* 4, 251–277.

OPEN SCIENCE COLLABORATION. 2015. Estimating the reproducibility of psychological science. *Science 349,* 6251, aac4716.

OYA, T. AND MORISHIMA, S. 2021. LSTM-SAKT: LSTM-encoded SAKT-like transformer for knowledge tracing, 2nd place solution for riiid! answer correctness prediction. *arXiv preprint arXiv:2102.00845*.

OZENNE, B., SUBTIL, F., AND MAUCORT-BOULCH, D. 2015. The precision–recall curve overcame the optimism of the receiver operating characteristic curve in rare diseases. *Journal of Clinical Epidemiology 68,* 8, 855–859.

PAHIKKALA, T., PYYSALO, S., BOBERG, J., JÄRVINEN, J., AND SALAKOSKI, T. 2009. Matrix representations, linear transformations, and kernels for disambiguation in natural language. *Machine Learning 74,* 2, 133–158.

PANDEY, S. AND KARYPIS, G. 2019. A self-attentive model for knowledge tracing. In *Proceedings of The 12th International Conference on Educational Data Mining*, C. F. Lynch, A. Merceron, M. Desmarais, and R. Nkambou, Eds. International Educational Data Mining Society, 384–389.

PANDEY, S., KARYPIS, G., AND SRIVASTAVA, J. 2021. An empirical comparison of deep learning models for knowledge tracing on large-scale dataset. *arXiv preprint arXiv:2101.06373*.

PANDEY, S. AND SRIVASTAVA, J. 2020. RKT: Relation-aware self-attention for knowledge tracing. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, C. Hauff, E. Curry, and P. C. Mauroux, Eds. Association for Computing Machinery, 1205–1214.

PARDOS, Z., HEFFERNAN, N., RUIZ, C., AND BECK, J. 2008. The composition effect: Conjuntive or compensatory? an analysis of multi-skill math questions in its. In *The 1st International Conference on Educational Data Mining*, R. Baker, T. Barnes, and J. Beck, Eds. International Educational Data Mining Society, 147–156.

PARDOS, Z. A. AND HEFFERNAN, N. T. 2010. Modeling individualization in a bayesian networks implementation of knowledge tracing. In *International Conference on User Modeling, Adaptation, and Personalization*, P. Bra, A. Kobsa, and D. Chin, Eds. Springer, 255–266.

PARDOS, Z. A. AND HEFFERNAN, N. T. 2011. KT-IDEM: Introducing item difficulty to the knowledge tracing model. In *International Conference on User Modeling, Adaptation, and Personalization*, J. A. Konstan, R. Conejo, J. L. Marzo, and N. Oliver, Eds. Springer, 243–254.

PATIL, P., PENG, R. D., AND LEEK, J. T. 2016. A statistical definition for reproducibility and replicability. *BioRxiv*, 066803.

PAVLIK, P. I., CEN, H., AND KOEDINGER, K. R. 2009. Performance factors analysis –a new alternative to knowledge tracing. In *Proceedings of the 2009 Conference on Artificial Intelligence in Education: Building Learning Systems That Care: From Knowledge Representation to Affective Modelling*, V. Dimitrova, R. Mizoguchi, B. du Boulay, and A. Graesser, Eds. IOS Press, NLD, 531–538.

PEARL, J. 1985. Bayesian networks: a model of self-activated memory for evidential reasoning. In *Proceedings of the 7th Conference of the Cognitive Science Society*, R. H. J. Granger and K. Eiselt, Eds. Cognitive Science Society, 15–17.

PEARL, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

PELÁNEK, R. 2014. Application of time decay functions and the elo system in student modeling. In *Proceedings of the 7th International Conference on Educational Data Mining*, J. C. Stamper, Z. A. Pardos, M. Mavrikis, and B. M. McLaren, Eds. International Educational Data Mining Society, 21–27.

PELÁNEK, R. 2015. Metrics for evaluation of student models. *Journal of Educational Data Mining 7,* 2, 1–19.

PENG, R. D. 2011. Reproducible research in computational science. *Science 334,* 6060, 1226–1227.

PIECH, C., BASSEN, J., HUANG, J., GANGULI, S., SAHAMI, M., GUIBAS, L. J., AND SOHL-DICKSTEIN, J. 2015. Deep knowledge tracing. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 505–513.

PU, S., YUDELSON, M., OU, L., AND HUANG, Y. 2020. Deep knowledge tracing with transformers. In *International Conference on Artificial Intelligence in Education*, I. I. Bittencourt, M. Cukurova, K. Muldner, R. Luckin, and E. Millán, Eds. Springer, 252–256.

RAFFERTY, A. N., BRUNSKILL, E., GRIFFITHS, T. L., AND SHAFTO, P. 2011. Faster teaching by pomdp planning. In *International Conference on Artificial Intelligence in Education*, G. Biswas, S. Bull, J. Kay, and A. Mitrovic, Eds. Springer, 280–287.

ROWE, J. P. AND LESTER, J. C. 2010. Modeling user knowledge with dynamic bayesian networks in interactive narrative environments. In *Proceedings of the 6th Artificial Intelligence and Interactive Digital Entertainment Conference*, G. M. Youngblood and V. Bulitko, Eds. AAAI, 57–62.

SAAD, E. W., PROKHOROV, D. V., AND WUNSCH, D. C. 1998. Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. *IEEE Transactions on Neural Networks 9,* 6, 1456–1470.

SAHA, S. AND RAGHAVA, G. P. S. 2006. Prediction of continuous b-cell epitopes in an antigen using recurrent neural network. *Proteins: Structure, Function, and Bioinformatics 65,* 1, 40–48.

SAITO, T. AND REHMSMEIER, M. 2015. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS One 10,* 3, e0118432.

SANYAL, D., BOSCH, N., AND PAQUETTE, L. 2020. Feature selection metrics: Similarities, differences, and characteristics of the selected models. In *Proceedings of the 13th International Conference on Educational Data Mining*, A. N. Rafferty, J. Whitehill, C. Romero, and V. Cavalli-Sforza, Eds. International Educational Data Mining Society, 212–223.

SHIN, D., SHIM, Y., YU, H., LEE, S., KIM, B., AND CHOI, Y. 2021. Saint+: Integrating temporal features for ednet correctness prediction. In *Proceedings of the 11th International Learning Analytics and Knowledge Conference*, M. Scheffel, N. Dowell, S. Joksimovic, and G. Siemens, Eds. LAK21. Association for Computing Machinery, New York, NY, USA, 490–496.

SONG, X., LI, J., TANG, Y., ZHAO, T., CHEN, Y., AND GUAN, Z. 2021. JKT: A joint graph convolutional network based deep knowledge tracing. *Information Sciences 580*, 510–523.

SPELLMAN, B. A. 2012. Introduction to the special section data, data, everywhere... especially in my file drawer. *Perspectives on Psychological Science 7,* 1, 58–59.

STEIF, P. AND BIER, N. 2014. OLI engineering statics-fall 2011, February 2014. https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=507.

STEVENS, J. R. 2017. Replicability and reproducibility in comparative psychology. *Frontiers in Psychology 8*, 862.

SU, Y., LIU, Q., LIU, Q., HUANG, Z., YIN, Y., CHEN, E., DING, C., WEI, S., AND HU, G. 2018. Exercise-enhanced sequential modeling for student performance prediction. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, S. A. McIlraith and K. Q. Weinberger, Eds. Number 1. AAAI, 2435–2443.

SUKHBAATAR, S., SZLAM, A., WESTON, J., AND FERGUS, R. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2440–2448.

TRIFA, A., HEDHILI, A., AND CHAARI, W. L. 2019. Knowledge tracing with an intelligent agent, in an e-learning platform. *Education and Information Technologies 24,* 1, 711–741.

VANLEHN, K. 2011. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist 46,* 4, 197–221.

VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L., AND POLOSUKHIN, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

VIE, J.-J. AND KASHIMA, H. 2019. Knowledge tracing machines: Factorization machines for knowledge tracing. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, P. V. Hentenryck and Z.-H. Zhou, Eds. Number 1. AAAI, 750–757.

WESTON, J., CHOPRA, S., AND BORDES, A. 2015. Memory networks. In *3rd International Conference on Learning Representations*, Y. Bengio and Y. LeCun, Eds. Available at https://arxiv.org/abs/1410.3916.

WILSON, K. H., XIONG, X., KHAJAH, M., LINDSEY, R. V., ZHAO, S., KARKLIN, Y., VAN INWEGEN, E. G., HAN, B., EKANADHAM, C., BECK, J. E., HEFFERNAN, N., AND MOZER, M. C. 2016. Estimating student proficiency: Deep learning is not the panacea. In *Neural Information Processing Systems, Workshop on Machine Learning for Education*, R. G. Baraniuk, J. Ngiam, C. Studer, P. Grimaldi, and A. S. Lan, Eds.

XIONG, X., ZHAO, S., VAN INWEGEN, E. G., AND BECK, J. E. 2016. Going deeper with deep knowledge tracing. In *Proceedings of the 9th International Conference on Educational Data Mining*, T. Barnes, M. Chi, and M. Feng, Eds. International Educational Data Mining Society, 545–550.

YEUNG, C.-K. AND YEUNG, D.-Y. 2018. Addressing two problems in deep knowledge tracing via prediction-consistent regularization. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, S. Klemmer and K. Koedinger, Eds. Association for Computing Machinery, 1–10.

YUDELSON, M., FANCSALI, S., RITTER, S., BERMAN, S., NIXON, T., AND JOSHI, A. 2014. Better data beats big data. In *Proceedings of the 7th International Conference on Educational Data Mining*, J. C. Stamper, Z. A. Pardos, M. Mavrikis, and B. M. McLaren, Eds. International Educational Data Mining Society, 205–209.

YUDELSON, M. V., KOEDINGER, K. R., AND GORDON, G. J. 2013. Individualized bayesian knowledge tracing models. In *International Conference on Artificial Intelligence in Education*, H. C. Lane, K. Yacef, J. Mostow, and P. Pavlik, Eds. Springer, 171–180.

ZHANG, J., SHI, X., KING, I., AND YEUNG, D.-Y. 2017. Dynamic key-value memory networks for knowledge tracing. In *Proceedings of the 26th International Conference on World Wide Web*,

E. Agichtein and E. Gabrilovich, Eds. WWW '17. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 765–774.

ZHUANG, F., QI, Z., DUAN, K., XI, D., ZHU, Y., ZHU, H., XIONG, H., AND HE, Q. 2020. A comprehensive survey on transfer learning. *Proceedings of the IEEE 109,* 1, 43–76.

A. MODEL COMPARISON RESULTS

Table 9: Results for ASSISTments 2009 updated dataset.

Model	Acc	AUC	Precision	Recall	F1	MCC	RMSE
Vanilla-DKT	.757±.009	.809±.010	.766±.021	.892±.011	.824±.014	.448±.020	.403±.006
LSTM-DKT	**.761±.009**	**.814±.011**	.767±.017	.898±.018	**.827±.016**	**.455±.021**	**.400±.006**
LSTM-DKT-S+	**.761±.011**	**.814±.010**	.765±.018	.901±.018	**.827±.016**	**.455±.021**	**.400±.006**
DKVMN	.758±.011	.809±.010	.765±.020	.895±.014	.825±.016	.450±.018	.403±.006
DKVMN-Paper	.755±.011	.806±.010	.760±.023	.902±.014	.825±.015	.443±.019	.405±.006
SAKT	.752±.015	.798±.008	.759±.017	.895±.031	.821±.021	.434±.015	.408±.008
GLR	.711±.029	.729±.032	.715±.033	.894±.045	.794±.038	.324±.074	.438±.014
BKT	.699±.010	.710±.029	.716±.033	.857±.015	.780±.024	.313±.039	.448±.004
Mean	.633±.058	.500±.000	.633±.058	**1.000±.000**	.774±.043	.000±.000	.484±.019
NaP	.713±.025	.686±.025	**.768±.038**	.776±.035	.772±.036	.373±.051	.535±.023
NaP 3 Mean	.681±.022	.695±.034	.733±.033	.771±.039	.751±.036	.288±.061	.485±.016
NaP 5 Mean	.697±.026	.698±.034	.736±.033	.803±.050	.768±.041	.315±.057	.470±.015
NaP 9 Mean	.694±.028	.694±.035	.727±.032	.816±.058	.769±.044	.301±.061	.463±.015

Table 10: Results for ASSISTments 2015 dataset.

Model	Acc	AUC	Precision	Recall	F1	MCC	RMSE
Vanilla-DKT	.749±.028	.720±.021	.767±.027	.943±.011	.846±.020	.230±.015	.414±.019
LSTM-DKT	**.751±.027**	**.725±.020**	.769±.025	.943±.013	.847±.021	**.239±.013**	.413±.019
LSTM-DKT-S+	**.751±.027**	**.725±.020**	.769±.026	.943±.013	.847±.020	.238±.015	**.412±.019**
DKVMN	.750±.028	.723±.020	.769±.026	.941±.013	.846±.021	**.239±.015**	.413±.019
DKVMN-Paper	.750±.027	.718±.022	.769±.028	.943±.008	.847±.020	.237±.017	.414±.019
SAKT	.748±.028	.714±.023	.767±.029	.942±.007	.846±.020	.230±.021	.415±.019
GLR	.750±.031	.702±.027	.763±.031	.959±.010	**.849±.022**	.204±.017	.416±.021
BKT	.747±.026	.694±.020	.761±.026	.955±.009	.847±.018	.209±.012	.421±.015
Mean	.738±.033	.500±.000	.738±.033	**1.000±.000**	**.849±.022**	.000±.000	.440±.017
NaP	.690±.036	.594±.013	**.786±.032**	.792±.034	.789±.033	.190±.026	.556±.032
NaP 3 Mean	.701±.036	.602±.024	.778±.033	.830±.029	.803±.031	.176±.027	.484±.026
NaP 5 Mean	.704±.035	.624±.024	.771±.033	.850±.025	.808±.029	.158±.025	.463±.023
NaP 9 Mean	.712±.033	.624±.023	.769±.032	.868±.022	.816±.027	.162±.022	.454±.021

Table 11: Results for ASSISTments 2017 dataset.

Model	Acc	AUC	Precision	Recall	F1	MCC	RMSE
Vanilla-DKT	.681±.015	.703±.009	.615±.028	.374±.049	.464±.044	.271±.020	.453±.008
LSTM-DKT	**.692±.017**	**.723±.010**	**.630±.019**	.412±.039	.498±.035	**.302±.016**	**.446±.009**
LSTM-DKT-S+	**.692±.016**	**.723±.010**	.626±.019	.420±.040	**.502±.035**	**.302±.014**	**.446±.009**
DKVMN	.680±.011	.704±.009	.611±.037	.377±.044	.466±.045	.270±.026	.452±.006
DKVMN-Paper	.678±.012	.696±.009	.617±.031	.342±.071	.438±.068	.257±.032	.454±.006
SAKT	.672±.014	.661±.022	.617±.026	.298±.082	.397±.084	.235±.036	.462±.004
GLR	.659±.004	.648±.005	.602±.008	.254±.007	.357±.009	.205±.007	.467±.002
BKT	.645±.016	.623±.002	.565±.020	.279±.018	.373±.013	.183±.009	.475±.006
Mean	.627±.005	.500±.000	.000±.000	.000±.000	.000±.000	.000±.000	.484±.001
NaP	.591±.001	.562±.002	.451±.007	**.450±.007**	.450±.007	.124±.004	.640±.001
NaP 3 Mean	.630±.005	.573±.002	.505±.004	.414±.007	.455±.005	.182±.005	.534±.002
NaP 5 Mean	.637±.004	.600±.002	.518±.005	.387±.008	.443±.006	.187±.005	.503±.002
NaP 9 Mean	.641±.004	.608±.003	.528±.005	.356±.008	.425±.007	.186±.006	.488±.001

Table 12: Results for IntroProg dataset.

Model	Acc	AUC	Precision	Recall	F1	MCC	RMSE
Vanilla-DKT	.752±.028	.821±.012	.756±.036	.746±.048	.751±.042	.483±.029	.410±.016
LSTM-DKT	.757±.027	.827±.013	.754±.032	.762±.056	.758±.044	.491±.030	.406±.016
LSTM-DKT-S+	.755±.027	.826±.014	.752±.030	.761±.060	.756±.045	.487±.029	.406±.017
DKVMN	.756±.028	.827±.015	**.757±.030**	.755±.055	.756±.043	.490±.031	.406±.018
DKVMN-Paper	.754±.027	.825±.013	.752±.033	.759±.058	.755±.045	.486±.029	.407±.016
SAKT	.754±.029	.825±.015	.743±.023	**.776±.068**	**.759±.045**	.482±.034	.407±.020
GLR	**.761±.007**	**.843±.008**	.754±.002	.757±.008	.755±.004	**.522±.014**	**.402±.005**
BKT	.713±.012	.789±.012	.738±.087	.684±.011	.708±.041	.423±.012	.436±.004
Mean	.513±.013	.500±.000	.000±.000	.000±.000	.000±.000	.000±.000	.500±.000
NaP	.716±.006	.716±.006	.708±.005	.710±.005	.709±.005	.431±.012	.533±.006
NaP 3 Mean	.708±.008	.767±.009	.696±.005	.713±.006	.704±.006	.416±.017	.466±.006
NaP 5 Mean	.707±.008	.772±.010	.693±.006	.717±.006	.704±.006	.414±.017	.456±.006
NaP 9 Mean	.705±.009	.774±.012	.687±.006	.723±.006	.704±.006	.410±.018	.450±.007

Table 13: Results for Statics dataset.

Model	Acc	AUC	Precision	Recall	F1	MCC	RMSE
Vanilla-DKT	.804±.017	.815±.014	.828±.017	.938±.013	.880±.014	.372±.021	.369±.015
LSTM-DKT	**.807±.016**	.824±.013	.832±.014	.937±.017	**.881±.014**	.383±.030	.365±.013
LSTM-DKT-S+	.806±.015	.823±.013	.832±.014	.936±.018	**.881±.013**	.383±.034	.365±.013
DKVMN	**.807±.017**	**.825±.012**	**.835±.017**	.932±.011	**.881±.014**	**.393±.023**	**.364±.014**
DKVMN-Paper	.803±.016	.812±.013	.831±.015	.933±.013	.879±.013	.374±.028	.369±.013
SAKT	.801±.018	.808±.017	.827±.019	.936±.012	.878±.014	.363±.027	.371±.015
GLR	.805±.006	.815±.004	.826±.005	.943±.006	**.881±.005**	.378±.007	.369±.004
BKT	.786±.022	.774±.028	.803±.028	.953±.012	.872±.013	.305±.070	.393±.013
Mean	.765±.007	.500±.000	.765±.007	**1.000±.000**	.867±.005	.000±.000	.424±.004
NaP	.705±.004	.589±.005	.807±.004	.808±.004	.808±.004	.179±.010	.543±.004
NaP 3 Mean	.729±.005	.640±.007	.801±.004	.859±.005	.829±.004	.180±.015	.453±.003
NaP 5 Mean	.740±.004	.652±.008	.797±.003	.886±.005	.839±.004	.176±.018	.435±.003
NaP 9 Mean	.751±.005	.661±.009	.791±.004	.916±.007	.849±.005	.170±.019	.422±.003

Table 14: Results for Synthetic-K2 dataset.

Model	Acc	AUC	Precision	Recall	F1	MCC	RMSE
Vanilla-DKT	.804±.003	.869±.003	.851±.004	.867±.007	.858±.003	.542±.007	.367±.002
LSTM-DKT	.805±.003	.871±.002	.848±.006	.872±.004	**.860±.002**	.541±.008	.366±.002
LSTM-DKT-S+	**.806±.003**	.871±.002	.851±.005	.868±.003	.859±.002	.545±.006	.366±.002
DKVMN	**.806±.003**	**.872±.002**	.852±.003	.867±.003	**.860±.002**	.546±.005	**.365±.002**
DKVMN-Paper	**.806±.002**	**.872±.002**	.853±.003	.865±.002	.859±.002	.547±.005	**.365±.002**
SAKT	**.806±.002**	**.872±.002**	**.855±.006**	.862±.005	.859±.001	**.548±.006**	**.365±.002**
GLR	.729±.004	.786±.002	.750±.003	.906±.005	.821±.003	.308±.013	.418±.002
BKT	.692±.004	.635±.005	.700±.005	.961±.000	.810±.003	.132±.002	.455±.002
Mean	.685±.005	.500±.000	.685±.005	**1.000±.000**	.813±.003	.000±.000	.465±.002
NaP	.644±.001	.585±.003	.738±.002	.744±.002	.741±.002	.171±.006	.597±.001
NaP 3 Mean	.680±.003	.654±.004	.753±.002	.794±.004	.773±.003	.235±.008	.489±.002
NaP 5 Mean	.688±.003	.681±.004	.754±.002	.808±.004	.780±.003	.247±.010	.465±.001
NaP 9 Mean	.707±.003	.709±.005	.764±.003	.827±.006	.794±.004	.289±.008	.448±.001

Table 15: Results for Synthetic-K5 dataset.

Model	Acc	AUC	Precision	Recall	F1	MCC	RMSE
Vanilla-DKT	.750±.002	.822±.002	.805±.003	.778±.004	.791±.003	.481±.004	.409±.001
LSTM-DKT	.752±.003	.827±.002	.807±.004	.779±.003	**.793±.003**	.485±.005	.406±.002
LSTM-DKT-S+	.751±.002	.826±.002	.805±.003	.779±.008	.792±.004	.483±.004	.406±.001
DKVMN	**.754±.003**	**.829±.002**	**.812±.003**	.775±.005	**.793±.003**	**.491±.006**	**.404±.002**
DKVMN-Paper	.753±.003	**.829±.003**	**.812±.003**	.773±.007	.792±.004	.490±.006	.405±.002
SAKT	.753±.003	.828±.002	.809±.008	.777±.004	**.793±.002**	.488±.008	.405±.002
GLR	.649±.004	.683±.006	.666±.004	.848±.005	.746±.002	.220±.011	.465±.001
BKT	.633±.002	.633±.002	.641±.004	.899±.010	.748±.002	.164±.006	.476±.000
Mean	.608±.003	.500±.000	.608±.003	**1.000±.000**	.756±.002	.000±.000	.488±.001
NaP	.565±.004	.543±.003	.641±.004	.647±.004	.644±.004	.086±.007	.659±.003
NaP 3 Mean	.579±.003	.560±.004	.645±.004	.682±.006	.663±.005	.103±.005	.551±.002
NaP 5 Mean	.596±.004	.581±.005	.655±.004	.710±.006	.681±.005	.133±.009	.522±.002
NaP 9 Mean	.610±.005	.605±.006	.664±.004	.725±.007	.693±.005	.163±.010	.504±.002

B. BEST HYPERPARAMETERS

The layer sizes in the tables are as follows: recurrent layer size (attention layer size for SAKT), key-embedding layer size, value-embedding layer size and summary layer size. Hyperparameters that are not used for a model are denoted by the dash symbol "-". As an example, if a model has no key-embeddings or a summary layer, recurrent layer size of 100 and value-embedding layer size of 20, the layer sizes are shown as 100,-,20,-.

Table 16: Best hyperparameters for DLKT models in ASSISTments 2009 updated dataset.

Model	DKVMN	DKVMN-Paper	LSTM-DKT	LSTM-DKT-S+	SAKT	Vanilla-DKT
AUC	0.809	0.806	0.814	0.814	0.798	0.809
Seed	13	13	42	42	13	13
Init lr	0.001	0.01	0.001	0.001	0.001	0.001
Layer sizes	50,20,50,-,	50,20,20,100	50,-,20,-,	50,-,20,-,	50,50,50,50	100,-,50,-,
N heads	-	-	-	-	5	-
One-hot input	True	False	False	False	False	False
Output per skill	True	False	True	True	False	True

Table 17: Best hyperparameters for DLKT models in ASSISTments 2015 dataset.

Model	DKVMN	DKVMN-Paper	LSTM-DKT	LSTM-DKT-S+	SAKT	Vanilla-DKT
AUC	0.723	0.718	0.725	0.725	0.714	0.72
Seed	13	42	13	13	13	42
Init lr	0.001	0.01	0.001	0.001	0.001	0.001
Layer sizes	50,20,20,-,	50,50,50,100	100,-,50,-,	100,-,50,-,	100,-,-,50	50,-,50,-,
N heads	-	-	-	-	5	-
One-hot input	False	False	False	False	True	False
Output per skill	True	False	True	True	False	True

Table 18: Best hyperparameters for DLKT models in ASSISTments 2017 dataset.

Model	DKVMN	DKVMN-Paper	LSTM-DKT	LSTM-DKT-S+	SAKT	Vanilla-DKT
AUC	0.704	0.696	0.723	0.723	0.661	0.703
Seed	13	42	42	42	42	42
Init lr	0.01	0.01	0.01	0.01	0.001	0.001
Layer sizes	100,20,50,100	50,50,50,100	100,-,20,-,	100,-,20,-,	50,20,20,-,	100,-,50,-,
N heads	-	-	-	-	5	-
One-hot input	True	False	False	False	False	False
Output per skill	False	False	True	True	True	True

Table 19: Best hyperparameters for DLKT models in IntroProg dataset.

Model	DKVMN	DKVMN-Paper	LSTM-DKT	LSTM-DKT-S+	SAKT	Vanilla-DKT
AUC	0.827	0.825	0.827	0.826	0.825	0.821
Seed	13	13	42	42	13	42
Init lr	0.001	0.01	0.001	0.01	0.001	0.001
Layer sizes	50,20,20,-,	50,50,20,50	50,-,50,-,	50,-,20,-,	50,-,-,100	50,-,20,-,
N heads	-	-	-	-	1	-
One-hot input	False	False	False	False	True	False
Output per skill	True	False	True	True	False	True

Table 20: Best hyperparameters for DLKT models in Statics dataset.

Model	DKVMN	DKVMN-Paper	LSTM-DKT	LSTM-DKT-S+	SAKT	Vanilla-DKT
AUC	0.825	0.812	0.824	0.823	0.808	0.815
Seed	13	13	13	13	42	42
Init lr	0.01	0.01	0.001	0.001	0.001	0.001
Layer sizes	50,20,20,-,	50,50,50,-,	50,-,-,-,	50,-,-,-,	50,-,-,-,	100,-,50,-,
N heads	-	-	-	-	1	-
One-hot input	True	True	True	True	True	False
Output per skill	True	True	True	True	True	True

Table 21: Best hyperparameters for DLKT models in Synthetic-K2 dataset.

Model	DKVMN	DKVMN-Paper	LSTM-DKT	LSTM-DKT-S+	SAKT	Vanilla-DKT
AUC	0.872	0.872	0.871	0.871	0.872	0.869
Seed	42	42	42	42	42	42
Init lr	0.001	0.001	0.001	0.001	0.001	0.001
Layer sizes	50,50,20,50	50,50,20,50	50,-,-,100	50,-,20,100	100,20,20,100	100,-,-,-,
N heads	-	-	-	-	5	-
One-hot input	True	False	True	False	False	True
Output per skill	False	False	False	False	False	True

Table 22: Best hyperparameters for DLKT models in Synthetic-K5 dataset.

Model	DKVMN	DKVMN-Paper	LSTM-DKT	LSTM-DKT-S+	SAKT	Vanilla-DKT
AUC	0.829	0.829	0.827	0.826	0.828	0.822
Seed	42	13	42	13	42	13
Init lr	0.001	0.001	0.001	0.001	0.001	0.001
Layer sizes	50,50,20,50	50,50,50,50	50,-,-,-,	50,-,-,-,	100,20,20,100	100,-,20,-,
N heads	-	-	-	-	5	-
One-hot input	True	False	True	True	False	False
Output per skill	False	False	True	True	False	True

Table 23: Output-per-skill effect on AUC in ASSISTments 2017 dataset.

Model	Output-per-skill	Max	Max-Min	Min	Sd
DKVMN	False	0.704	0.038	0.666	0.012
	True	0.685	0.011	0.674	0.002
DKVMN-Paper	False	0.696	0.058	0.638	0.019
	True	0.658	0.012	0.646	0.002
LSTM-DKT	False	0.711	0.048	0.663	0.010
	True	0.723	0.010	0.713	0.002
LSTM-DKT-S+	False	0.718	0.032	0.686	0.005
	True	0.723	0.007	0.716	0.002
SAKT	False	0.659	0.138	0.521	0.051
	True	0.661	0.038	0.623	0.005
Vanilla-DKT	False	0.683	0.126	0.557	0.036
	True	0.703	0.090	0.613	0.029

Table 24: Output-per-skill effect on AUC in ASSISTments 2015 dataset.

Model	Output-per-skill	Max	Max-Min	Min	Sd
DKVMN	False	0.720	0.006	0.714	0.001
	True	0.723	0.009	0.714	0.003
DKVMN-Paper	False	0.718	0.032	0.686	0.010
	True	0.716	0.029	0.687	0.006
LSTM-DKT	False	0.703	0.020	0.683	0.003
	True	0.725	0.005	0.720	0.002
LSTM-DKT-S+	False	0.720	0.016	0.704	0.003
	True	0.725	0.005	0.720	0.002
SAKT	False	0.714	0.194	0.520	0.062
	True	0.713	0.060	0.653	0.013
Vanilla-DKT	False	0.699	0.076	0.623	0.021
	True	0.720	0.056	0.664	0.019

Table 25: One-hot-input effect on AUC in IntroProg dataset.

Model	One-hot-input	Max	Max-Min	Min	Sd
DKVMN	False	0.827	0.003	0.824	0.001
	True	0.827	0.003	0.824	0.001
DKVMN-Paper	False	0.825	0.024	0.801	0.005
	True	0.818	0.012	0.806	0.003
LSTM-DKT	False	0.827	0.007	0.820	0.002
	True	0.826	0.007	0.819	0.003
LSTM-DKT-S+	False	0.826	0.002	0.824	0.001
	True	0.826	0.003	0.823	0.001
SAKT	False	0.817	0.265	0.552	0.055
	True	0.825	0.219	0.606	0.065
Vanilla-DKT	False	0.821	0.102	0.719	0.017
	True	0.821	0.088	0.733	0.018

Table 26: One-hot input effect on AUC in ASSISTments 2009 updated dataset.

Model	One-hot-input	Max	Max-Min	Min	Sd
DKVMN	False	0.809	0.010	0.799	0.003
	True	0.809	0.010	0.799	0.004
DKVMN-Paper	False	0.806	0.094	0.712	0.035
	True	0.773	0.062	0.711	0.017
LSTM-DKT	False	0.814	0.077	0.737	0.013
	True	0.814	0.026	0.788	0.011
LSTM-DKT-S+	False	0.814	0.072	0.742	0.010
	True	0.814	0.014	0.800	0.005
SAKT	False	0.798	0.253	0.545	0.071
	True	0.752	0.207	0.545	0.074
Vanilla-DKT	False	0.809	0.098	0.711	0.022
	True	0.806	0.101	0.705	0.023

Table 27: Random seed effect on AUC in ASSISTments 2015 dataset.

Model	Random seed	Max	Max-Min	Min	Sd
DKVMN	13	0.723	0.009	0.714	0.002
	42	0.723	0.009	0.714	0.002
DKVMN-Paper	13	0.718	0.031	0.687	0.009
	42	0.718	0.032	0.686	0.009
LSTM-DKT	13	0.725	0.042	0.683	0.012
	42	0.724	0.027	0.697	0.011
LSTM-DKT-S+	13	0.725	0.021	0.704	0.004
	42	0.725	0.018	0.707	0.003
SAKT	13	0.714	0.176	0.538	0.049
	42	0.713	0.193	0.520	0.062
Vanilla-DKT	13	0.719	0.096	0.623	0.024
	42	0.720	0.091	0.629	0.022

C. MAX ATTEMPT FILTER CUT VERSUS SPLIT

Table 28: Student attempt split effect.

Dataset	Model	Max attempt filter	Acc	AUC	Prec	Recall	F1	MCC	RMSE
ASSISTments 2009 updated	DKVMN	- & -	.759	.812	.763	.890	.822	.454	.401
		Cut & 200	.753	.809	.758	.880	.814	.452	.406
		Cut & 500	.758	.811	.762	.889	.821	.452	.402
		Split & 200	.758	.809	.765	.895	.825	.450	.403
		Split & 500	.758	.811	.762	.891	.821	.452	.403
	DKVMN-Paper	- & -	.745	.788	.750	.889	.814	.413	.411
		Cut & 200	.749	.802	.750	.886	.812	.441	.409
		Cut & 500	.756	.806	.759	.891	.820	.446	.404
		Split & 200	.755	.806	.760	.902	.825	.443	.405
		Split & 500	.754	.804	.759	.888	.818	.441	.406
	LSTM-DKT	- & -	.763	.817	.766	.892	.824	.463	.398
		Cut & 200	.756	.813	.761	.878	.815	.458	.404
		Cut & 500	.763	.816	.768	.888	.823	.463	.399
		Split & 200	.761	.814	.767	.898	.827	.455	.400
		Split & 500	.759	.815	.764	.889	.822	.455	.401
	LSTM-DKT-S+	- & -	.762	.817	.765	.893	.824	.462	.398
		Cut & 200	.756	.813	.761	.878	.815	.457	.404
		Cut & 500	.762	.815	.765	.892	.824	.460	.399
		Split & 200	.761	.814	.765	.901	.827	.455	.400
		Split & 500	.759	.815	.764	.888	.821	.455	.401
	SAKT	- & -	.743	.787	.756	.865	.807	.416	.415
		Cut & 200	.744	.795	.748	.879	.808	.430	.412
		Cut & 500	.746	.790	.756	.871	.809	.424	.413
		Split & 200	.752	.798	.759	.895	.821	.434	.408
		Split & 500	.746	.792	.754	.880	.812	.423	.413
	Vanilla-DKT	- & -	.758	.810	.761	.893	.822	.451	.403
		Cut & 200	.752	.808	.758	.876	.813	.449	.407
		Cut & 500	.757	.808	.759	.893	.821	.449	.404
		Split & 200	.757	.809	.766	.892	.824	.448	.403
		Split & 500	.753	.806	.761	.882	.817	.441	.406
ASSISTments 2015	DKVMN	- & -	.755	.725	.772	.945	.850	.240	.410
		Cut & 200	.754	.725	.772	.945	.850	.237	.411
		Cut & 500	.755	.725	.772	.945	.850	.240	.410
		Split & 200	.750	.723	.769	.941	.846	.239	.413
		Split & 500	.754	.725	.772	.945	.849	.238	.411
	DKVMN-Paper	- & -	.753	.719	.771	.946	.849	.233	.412
		Cut & 200	.753	.718	.772	.943	.849	.236	.413
		Cut & 500	.753	.719	.773	.941	.848	.238	.413
		Split & 200	.750	.718	.769	.943	.847	.237	.414
		Split & 500	.753	.719	.772	.943	.849	.238	.413
	LSTM-DKT	- & -	.755	.727	.772	.945	.850	.241	.410
		Cut & 200	.755	.727	.772	.946	.850	.240	.410
		Cut & 500	.754	.727	.772	.945	.850	.241	.410
		Split & 200	.751	.725	.769	.943	.847	.239	.413
		Split & 500	.754	.727	.772	.944	.850	.241	.410
	LSTM-DKT-S+	- & -	.754	.727	.772	.946	.850	.239	.410
		Cut & 200	.754	.727	.772	.946	.850	.238	.410
		Cut & 500	.754	.727	.772	.945	.850	.240	.410

Table 28: Student attempt split effect.

Dataset	Model	Max attempt filter	Acc	AUC	Prec	Recall	F1	MCC	RMSE
ASSISTments 2015	LSTM-DKT-S+	Split & 200	.751	.725	.769	.943	.847	.238	.412
		Split & 500	.755	.726	.772	.945	.850	.241	.410
	SAKT	- & -	.752	.715	.769	.948	.849	.225	.413
		Cut & 200	.753	.715	.769	.949	.849	.227	.413
		Cut & 500	.752	.715	.769	.948	.849	.227	.413
		Split & 200	.748	.714	.767	.942	.846	.230	.415
		Split & 500	.751	.714	.772	.939	.847	.234	.414
	Vanilla-DKT	- & -	.752	.722	.770	.946	.849	.229	.412
		Cut & 200	.753	.721	.769	.948	.849	.227	.412
		Cut & 500	.752	.721	.770	.947	.849	.228	.412
		Split & 200	.749	.720	.767	.943	.846	.230	.414
		Split & 500	.753	.721	.769	.949	.849	.228	.412
ASSISTments 2017	DKVMN	- & -	.684	.712	.626	.380	.472	.283	.450
		Cut & 200	.666	.690	.612	.391	.477	.263	.460
		Cut & 500	.676	.708	.627	.416	.500	.287	.455
		Split & 200	.680	.704	.611	.377	.466	.270	.452
		Split & 500	.680	.703	.615	.384	.471	.272	.453
	DKVMN-Paper	- & -	.681	.701	.630	.348	.448	.271	.453
		Cut & 200	.667	.685	.638	.341	.444	.261	.461
		Cut & 500	.673	.699	.642	.367	.467	.276	.457
		Split & 200	.678	.696	.617	.342	.438	.257	.454
		Split & 500	.676	.693	.624	.335	.433	.256	.456
	LSTM-DKT	- & -	.697	.734	.631	.450	.525	.321	.443
		Cut & 200	.676	.707	.620	.435	.511	.289	.455
		Cut & 500	.685	.724	.632	.461	.533	.312	.450
		Split & 200	.692	.723	.630	.412	.498	.302	.446
		Split & 500	.690	.722	.625	.433	.510	.303	.447
	LSTM-DKT-S+	- & -	.697	.734	.630	.452	.526	.321	.444
		Cut & 200	.676	.708	.622	.433	.511	.291	.455
		Cut & 500	.686	.726	.633	.462	.534	.315	.449
		Split & 200	.692	.723	.626	.420	.502	.302	.446
		Split & 500	.690	.721	.626	.427	.507	.302	.447
	SAKT	- & -	.660	.650	.600	.261	.363	.207	.466
		Cut & 200	.652	.659	.604	.312	.411	.221	.468
		Cut & 500	.657	.664	.612	.327	.426	.234	.467
		Split & 200	.672	.661	.617	.298	.397	.235	.462
		Split & 500	.665	.651	.609	.287	.383	.219	.465
	Vanilla-DKT	- & -	.679	.698	.627	.342	.443	.266	.454
		Cut & 200	.667	.687	.625	.365	.460	.261	.461
		Cut & 500	.670	.692	.632	.367	.464	.269	.459
		Split & 200	.681	.703	.615	.374	.464	.271	.453
		Split & 500	.675	.691	.613	.350	.443	.254	.456
IntroProg	DKVMN	- & -	.765	.847	.761	.754	.757	.529	.399
		Cut & 200	.753	.833	.763	.763	.763	.504	.408
		Cut & 500	.761	.843	.761	.756	.758	.522	.402
		Split & 200	.756	.827	.757	.755	.756	.490	.406
		Split & 500	.763	.842	.760	.755	.757	.521	.401
	DKVMN-Paper	- & -	.764	.845	.762	.749	.756	.527	.400
		Cut & 200	.752	.832	.763	.759	.761	.502	.409
		Cut & 500	.760	.842	.761	.752	.757	.520	.403
		Split & 200	.754	.825	.752	.759	.755	.486	.407

Table 28: Student attempt split effect.

Dataset	Model	Max attempt filter	Acc	AUC	Prec	Recall	F1	MCC	RMSE
IntroProg	DKVMN-Paper	Split & 500	.761	.841	.762	.746	.754	.518	.402
	LSTM-DKT	- & -	.765	.846	.761	.755	.758	.530	.400
		Cut & 200	.753	.833	.763	.764	.763	.504	.408
		Cut & 500	.761	.843	.760	.757	.758	.522	.402
		Split & 200	.757	.827	.754	.762	.758	.491	.406
		Split & 500	.763	.842	.760	.756	.758	.521	.401
	LSTM-DKT-S+	- & -	.764	.845	.760	.753	.756	.527	.400
		Cut & 200	.751	.832	.759	.767	.763	.501	.409
		Cut & 500	.759	.842	.758	.756	.757	.519	.403
		Split & 200	.755	.826	.752	.761	.756	.487	.406
		Split & 500	.761	.841	.757	.754	.756	.517	.402
	SAKT	- & -	.761	.843	.748	.768	.757	.521	.403
		Cut & 200	.749	.830	.751	.778	.764	.497	.411
		Cut & 500	.758	.839	.750	.767	.759	.516	.405
		Split & 200	.754	.825	.743	.776	.759	.482	.407
		Split & 500	.760	.839	.750	.767	.758	.515	.404
	Vanilla-DKT	- & -	.760	.840	.759	.744	.751	.519	.404
		Cut & 200	.748	.827	.760	.756	.758	.495	.412
		Cut & 500	.756	.836	.758	.746	.752	.511	.406
		Split & 200	.752	.821	.756	.746	.751	.483	.410
		Split & 500	.758	.835	.759	.742	.750	.511	.405
Statics	DKVMN	- & -	.813	.836	.841	.931	.884	.422	.360
		Cut & 200	.828	.830	.852	.950	.898	.379	.348
		Cut & 500	.805	.823	.832	.933	.880	.394	.367
		Split & 200	.807	.825	.835	.932	.881	.393	.364
		Split & 500	.807	.828	.837	.927	.880	.407	.365
	DKVMN-Paper	- & -	.807	.823	.832	.938	.881	.393	.366
		Cut & 200	.826	.827	.849	.951	.897	.368	.350
		Cut & 500	.801	.812	.825	.938	.878	.369	.371
		Split & 200	.803	.812	.831	.933	.879	.374	.369
		Split & 500	.801	.813	.831	.928	.877	.384	.372
	LSTM-DKT	- & -	.805	.819	.824	.948	.881	.375	.367
		Cut & 200	.822	.812	.840	.959	.896	.334	.355
		Cut & 500	.798	.803	.817	.948	.878	.351	.374
		Split & 200	.807	.824	.832	.937	.881	.383	.365
		Split & 500	.799	.811	.822	.942	.877	.364	.373
	LSTM-DKT-S+	- & -	.802	.807	.821	.947	.880	.362	.372
		Cut & 200	.819	.804	.837	.961	.894	.315	.358
		Cut & 500	.798	.806	.818	.946	.878	.353	.374
		Split & 200	.806	.823	.832	.936	.881	.383	.365
		Split & 500	.799	.813	.822	.941	.877	.364	.372
	SAKT	- & -	.802	.811	.826	.940	.879	.371	.370
		Cut & 200	.828	.835	.853	.947	.898	.382	.348
		Cut & 500	.798	.808	.824	.937	.876	.362	.373
		Split & 200	.801	.808	.827	.936	.878	.363	.371
		Split & 500	.800	.810	.829	.930	.877	.378	.372
	Vanilla-DKT	- & -	.805	.822	.829	.938	.880	.383	.367
		Cut & 200	.817	.798	.835	.960	.893	.306	.360
		Cut & 500	.794	.791	.815	.945	.875	.335	.379
		Split & 200	.804	.815	.828	.938	.880	.372	.369
		Split & 500	.800	.811	.824	.938	.877	.371	.373

Table 28: Student attempt split effect.

Dataset	Model	Max attempt filter	Acc	AUC	Prec	Recall	F1	MCC	RMSE
Synthetic-K2	DKVMN	- & -	.806	.872	.852	.867	.860	.546	.365
		Cut & 200	.806	.872	.852	.867	.860	.546	.365
		Cut & 500	.806	.872	.852	.867	.860	.546	.365
		Split & 200	.806	.872	.852	.867	.860	.546	.365
		Split & 500	.806	.872	.852	.867	.860	.546	.365
	DKVMN-Paper	- & -	.806	.872	.853	.865	.859	.547	.365
		Cut & 200	.806	.872	.853	.865	.859	.547	.365
		Cut & 500	.806	.872	.853	.865	.859	.547	.365
		Split & 200	.806	.872	.853	.865	.859	.547	.365
		Split & 500	.806	.872	.853	.865	.859	.547	.365
	LSTM-DKT	- & -	.805	.871	.848	.872	.860	.541	.366
		Cut & 200	.805	.871	.848	.872	.860	.541	.366
		Cut & 500	.805	.871	.848	.872	.860	.541	.366
		Split & 200	.805	.871	.848	.872	.860	.541	.366
		Split & 500	.805	.871	.848	.872	.860	.541	.366
	LSTM-DKT-S+	- & -	.806	.871	.851	.868	.859	.545	.366
		Cut & 200	.806	.871	.851	.868	.859	.545	.366
		Cut & 500	.806	.871	.851	.868	.859	.545	.366
		Split & 200	.806	.871	.851	.868	.859	.545	.366
		Split & 500	.806	.871	.851	.868	.859	.545	.366
	SAKT	- & -	.806	.872	.855	.862	.859	.548	.365
		Cut & 200	.806	.872	.855	.862	.859	.548	.365
		Cut & 500	.806	.872	.855	.862	.859	.548	.365
		Split & 200	.806	.872	.855	.862	.859	.548	.365
		Split & 500	.806	.872	.855	.862	.859	.548	.365
	Vanilla-DKT	- & -	.805	.869	.850	.867	.859	.542	.367
		Cut & 200	.805	.869	.850	.867	.859	.542	.367
		Cut & 500	.805	.869	.850	.867	.859	.542	.367
		Split & 200	.804	.869	.851	.867	.858	.542	.367
		Split & 500	.805	.869	.850	.867	.859	.542	.367
Synthetic-K5	DKVMN	- & -	.754	.829	.812	.775	.793	.491	.404
		Cut & 200	.754	.829	.812	.775	.793	.491	.404
		Cut & 500	.754	.829	.812	.775	.793	.491	.404
		Split & 200	.754	.829	.812	.775	.793	.491	.404
		Split & 500	.754	.829	.812	.775	.793	.491	.404
	DKVMN-Paper	- & -	.753	.829	.812	.773	.792	.490	.405
		Cut & 200	.753	.829	.812	.773	.792	.490	.405
		Cut & 500	.753	.829	.812	.773	.792	.490	.405
		Split & 200	.753	.829	.812	.773	.792	.490	.405
		Split & 500	.753	.829	.812	.773	.792	.490	.405
	LSTM-DKT	- & -	.752	.827	.807	.779	.793	.485	.406
		Cut & 200	.752	.827	.807	.779	.793	.485	.406
		Cut & 500	.752	.827	.807	.779	.793	.485	.406
		Split & 200	.752	.827	.807	.779	.793	.485	.406
		Split & 500	.752	.827	.807	.779	.793	.485	.406
	LSTM-DKT-S+	- & -	.751	.826	.805	.779	.792	.483	.406
		Cut & 200	.751	.826	.805	.779	.792	.483	.406
		Cut & 500	.751	.826	.805	.779	.792	.483	.406
		Split & 200	.751	.826	.805	.779	.792	.483	.406
		Split & 500	.751	.826	.805	.779	.792	.483	.406
	SAKT	- & -	.753	.828	.808	.780	.794	.488	.405

Table 28: Student attempt split effect.

Dataset	Model	Max attempt filter	Acc	AUC	Prec	Recall	F1	MCC	RMSE
Synthetic-K5	SAKT	Cut & 200	.753	.828	.808	.780	.794	.488	.405
		Cut & 500	.753	.828	.808	.780	.794	.488	.405
		Split & 200	.753	.828	.809	.777	.793	.488	.405
		Split & 500	.753	.828	.808	.780	.794	.488	.405
	Vanilla-DKT	- & -	.749	.822	.803	.777	.790	.478	.409
		Cut & 200	.749	.822	.803	.777	.790	.478	.409
		Cut & 500	.749	.822	.803	.777	.790	.478	.409
		Split & 200	.750	.822	.805	.778	.791	.481	.409
		Split & 500	.749	.822	.803	.777	.790	.478	.409

Table 29: Student attempt split effect averaged over datasets.

Model	Max attempt filter	Acc	AUC	Prec	Recall	F1	MCC	RMSE
DKVMN	- & -	.762	.805	.775	.792	.777	.424	.398
	Cut & 200	.759	.798	.774	.796	.779	.410	.400
	Cut & 500	.759	.802	.774	.797	.780	.419	.401
	Split & 200	.759	.798	.772	.792	.775	.411	.401
	Split & 500	.760	.801	.773	.792	.776	.418	.400
DKVMN-Paper	- & -	.758	.797	.773	.787	.771	.411	.402
	Cut & 200	.758	.795	.777	.788	.773	.406	.402
	Cut & 500	.757	.797	.775	.790	.774	.412	.403
	Split & 200	.757	.794	.771	.788	.771	.405	.403
	Split & 500	.758	.796	.773	.783	.769	.411	.403
LSTM-DKT	- & -	.763	.806	.773	.806	.784	.422	.399
	Cut & 200	.760	.799	.773	.805	.784	.407	.401
	Cut & 500	.760	.802	.772	.807	.785	.416	.401
	Split & 200	.761	.802	.772	.800	.781	.414	.400
	Split & 500	.760	.802	.771	.802	.781	.416	.401
LSTM-DKT-S+	- & -	.762	.804	.772	.805	.784	.420	.399
	Cut & 200	.759	.797	.772	.805	.783	.404	.401
	Cut & 500	.759	.802	.772	.807	.785	.416	.401
	Split & 200	.760	.801	.771	.801	.781	.413	.400
	Split & 500	.760	.802	.771	.800	.780	.415	.401
SAKT	- & -	.754	.787	.766	.775	.758	.397	.405
	Cut & 200	.755	.791	.770	.787	.769	.399	.403
	Cut & 500	.753	.788	.768	.785	.767	.400	.406
	Split & 200	.755	.787	.768	.784	.765	.397	.405
	Split & 500	.754	.787	.768	.778	.761	.401	.405
Vanilla-DKT	- & -	.758	.798	.771	.787	.771	.410	.402
	Cut & 200	.756	.790	.771	.793	.775	.394	.404
	Cut & 500	.755	.791	.770	.792	.773	.402	.405
	Split & 200	.757	.794	.770	.791	.773	.404	.404
	Split & 500	.756	.794	.768	.786	.769	.404	.404

D. HARDWARE COMPARISON CPU VERSUS GPU

Both the CPU and GPU results here were obtained with a later TensorFlow version than the hyperparameter tuning results, wherefore the CPU results may not exactly match the results in other result tables. Also the CPU hardware may differ from the other results as the models were trained and evaluated in a computation cluster with multiple different CPU nodes with varying CPU types.

Table 30: Hardware result comparison.

Dataset	Model	Hardware	Acc	AUC	Prec	Recall	F1	MCC	RMSE
ASSISTments 2009 updated	DKVMN	CPU-tf2.1.0	.758	.809	.765	.895	.825	.450	.403
		CPU-tf2.6.2	.757	.808	.765	.894	.825	.448	.404
		GPU	.757	.808	.765	.894	.825	.448	.404
	DKVMN-Paper	CPU-tf2.1.0	.755	.806	.760	.902	.825	.443	.405
		CPU-tf2.6.2	.755	.806	.763	.893	.823	.442	.405
		GPU	.754	.805	.762	.894	.823	.440	.405
	LSTM-DKT	CPU-tf2.1.0	.761	.814	.767	.898	.827	.455	.400
		CPU-tf2.6.2	.759	.811	.765	.898	.826	.451	.402
		GPU	.759	.811	.765	.897	.826	.451	.402
	LSTM-DKT-S+	CPU-tf2.1.0	.761	.814	.765	.901	.827	.455	.400
		CPU-tf2.6.2	.759	.810	.765	.898	.826	.451	.402
		GPU	.759	.810	.765	.899	.826	.451	.402
	SAKT	CPU-tf2.1.0	.752	.798	.759	.895	.821	.434	.408
		CPU-tf2.6.2	.752	.799	.755	.904	.823	.433	.408
		GPU	.752	.799	.756	.902	.823	.434	.408
	Vanilla-DKT	CPU-tf2.1.0	.757	.809	.766	.892	.824	.448	.403
		CPU-tf2.6.2	.758	.810	.768	.889	.824	.450	.403
		GPU	.758	.811	.766	.894	.825	.449	.402
ASSISTments 2015	DKVMN	CPU-tf2.1.0	.750	.723	.769	.941	.846	.239	.413
		CPU-tf2.6.2	.750	.723	.769	.941	.846	.237	.413
		GPU	.750	.722	.769	.941	.846	.239	.413
	DKVMN-Paper	CPU-tf2.1.0	.750	.718	.769	.943	.847	.237	.414
		CPU-tf2.6.2	.750	.722	.769	.941	.846	.238	.413
		GPU	.750	.721	.769	.942	.846	.235	.414
	LSTM-DKT	CPU-tf2.1.0	.751	.725	.769	.943	.847	.239	.413
		CPU-tf2.6.2	.750	.724	.769	.943	.847	.237	.413
		GPU	.750	.724	.769	.943	.847	.238	.413
	LSTM-DKT-S+	CPU-tf2.1.0	.751	.725	.769	.943	.847	.238	.412
		CPU-tf2.6.2	.751	.724	.768	.944	.847	.236	.413
		GPU	.750	.723	.770	.941	.846	.239	.413
	SAKT	CPU-tf2.1.0	.748	.714	.767	.942	.846	.230	.415
		CPU-tf2.6.2	.749	.714	.767	.944	.846	.230	.415
		GPU	.749	.714	.767	.943	.846	.230	.416
	Vanilla-DKT	CPU-tf2.1.0	.749	.720	.767	.943	.846	.230	.414
		CPU-tf2.6.2	.749	.721	.770	.938	.845	.236	.414
		GPU	.749	.721	.769	.939	.846	.236	.414
ASSISTments 2017	DKVMN	CPU-tf2.1.0	.680	.704	.611	.377	.466	.270	.452
		CPU-tf2.6.2	.680	.702	.614	.367	.459	.268	.453
		GPU	.680	.702	.612	.372	.462	.268	.453
	DKVMN-Paper	CPU-tf2.1.0	.678	.696	.617	.342	.438	.257	.454
		CPU-tf2.6.2	.678	.695	.616	.344	.438	.257	.454

Table 30: Hardware result comparison.

Dataset	Model	Hardware	Acc	AUC	Prec	Recall	F1	MCC	RMSE
		GPU	.678	.695	.616	.343	.438	.256	.454
ASSISTments 2017	LSTM-DKT	CPU-tf2.1.0	.692	.723	.630	.412	.498	.302	.446
		CPU-tf2.6.2	.689	.719	.620	.418	.499	.297	.448
		GPU	.689	.719	.619	.417	.497	.295	.448
	LSTM-DKT-S+	CPU-tf2.1.0	.692	.723	.626	.420	.502	.302	.446
		CPU-tf2.6.2	.690	.719	.619	.422	.502	.298	.448
		GPU	.689	.719	.621	.414	.496	.296	.448
	SAKT	CPU-tf2.1.0	.672	.661	.617	.298	.397	.235	.462
		CPU-tf2.6.2	.669	.655	.609	.292	.391	.227	.463
		GPU	.669	.656	.610	.294	.393	.229	.463
	Vanilla-DKT	CPU-tf2.1.0	.681	.703	.615	.374	.464	.271	.453
		CPU-tf2.6.2	.683	.708	.612	.398	.481	.279	.451
		GPU	.683	.709	.613	.397	.480	.280	.451
IntroProg	DKVMN	CPU-tf2.1.0	.756	.827	.757	.755	.756	.490	.406
		CPU-tf2.6.2	.756	.826	.757	.753	.755	.490	.406
		GPU	.756	.826	.757	.753	.755	.490	.406
	DKVMN-Paper	CPU-tf2.1.0	.754	.825	.752	.759	.755	.486	.407
		CPU-tf2.6.2	.755	.824	.757	.753	.754	.487	.407
		GPU	.755	.824	.756	.755	.755	.487	.407
	LSTM-DKT	CPU-tf2.1.0	.757	.827	.754	.762	.758	.491	.406
		CPU-tf2.6.2	.755	.824	.751	.763	.757	.488	.407
		GPU	.755	.824	.752	.761	.756	.488	.407
	LSTM-DKT-S+	CPU-tf2.1.0	.755	.826	.752	.761	.756	.487	.406
		CPU-tf2.6.2	.753	.823	.753	.752	.752	.484	.408
		GPU	.754	.824	.753	.755	.754	.485	.408
	SAKT	CPU-tf2.1.0	.754	.825	.743	.776	.759	.482	.407
		CPU-tf2.6.2	.754	.824	.755	.753	.753	.483	.408
		GPU	.754	.824	.754	.753	.753	.483	.407
	Vanilla-DKT	CPU-tf2.1.0	.752	.821	.756	.746	.751	.483	.410
		CPU-tf2.6.2	.754	.823	.749	.764	.756	.485	.408
		GPU	.753	.822	.746	.766	.756	.483	.409
Statics	DKVMN	CPU-tf2.1.0	.807	.825	.835	.932	.881	.393	.364
		CPU-tf2.6.2	.807	.823	.837	.929	.880	.394	.365
		GPU	.807	.823	.837	.929	.880	.394	.365
	DKVMN-Paper	CPU-tf2.1.0	.803	.812	.831	.933	.879	.374	.369
		CPU-tf2.6.2	.802	.809	.832	.930	.878	.373	.371
		GPU	.802	.809	.832	.930	.878	.373	.371
	LSTM-DKT	CPU-tf2.1.0	.807	.824	.832	.937	.881	.383	.365
		CPU-tf2.6.2	.806	.823	.832	.936	.881	.382	.366
		GPU	.806	.821	.832	.934	.880	.382	.366
	LSTM-DKT-S+	CPU-tf2.1.0	.806	.823	.832	.936	.881	.383	.365
		CPU-tf2.6.2	.806	.822	.833	.934	.881	.384	.366
		GPU	.807	.822	.835	.932	.881	.390	.366
	SAKT	CPU-tf2.1.0	.801	.808	.827	.936	.878	.363	.371
		CPU-tf2.6.2	.801	.808	.828	.935	.878	.364	.371
		GPU	.801	.811	.830	.930	.878	.370	.371
	Vanilla-DKT	CPU-tf2.1.0	.804	.815	.828	.938	.880	.372	.369
		CPU-tf2.6.2	.804	.818	.830	.936	.880	.378	.368
		GPU	.805	.819	.832	.934	.880	.380	.367
Synthetic-K2	DKVMN	CPU-tf2.1.0	.806	.872	.852	.867	.860	.546	.365
		CPU-tf2.6.2	.806	.872	.852	.868	.860	.546	.365

Table 30: Hardware result comparison.

Dataset	Model	Hardware	Acc	AUC	Prec	Recall	F1	MCC	RMSE
		GPU	.806	.872	.852	.868	.860	.546	.365
Synthetic-K2	DKVMN-Paper	CPU-tf2.1.0	.806	.872	.853	.865	.859	.547	.365
		CPU-tf2.6.2	.806	.872	.852	.867	.859	.546	.365
		GPU	.806	.872	.852	.867	.859	.546	.365
	LSTM-DKT	CPU-tf2.1.0	.805	.871	.848	.872	.860	.541	.366
		CPU-tf2.6.2	.805	.871	.853	.865	.859	.545	.366
		GPU	.805	.870	.852	.865	.859	.544	.366
	LSTM-DKT-S+	CPU-tf2.1.0	.806	.871	.851	.868	.859	.545	.366
		CPU-tf2.6.2	.806	.871	.853	.865	.859	.546	.365
		GPU	.806	.871	.853	.866	.859	.547	.365
	SAKT	CPU-tf2.1.0	.806	.872	.855	.862	.859	.548	.365
		CPU-tf2.6.2	.806	.872	.857	.860	.858	.549	.366
		GPU	.806	.872	.857	.860	.858	.549	.366
	Vanilla-DKT	CPU-tf2.1.0	.804	.869	.851	.867	.858	.542	.367
		CPU-tf2.6.2	.804	.868	.854	.860	.857	.543	.368
		GPU	.803	.868	.856	.857	.857	.544	.367
Synthetic-K5	DKVMN	CPU-tf2.1.0	.754	.829	.812	.775	.793	.491	.404
		CPU-tf2.6.2	.754	.829	.812	.775	.793	.491	.404
		GPU	.754	.829	.812	.775	.793	.491	.404
	DKVMN-Paper	CPU-tf2.1.0	.753	.829	.812	.773	.792	.490	.405
		CPU-tf2.6.2	.753	.829	.813	.771	.791	.490	.405
		GPU	.753	.829	.813	.771	.791	.490	.405
	LSTM-DKT	CPU-tf2.1.0	.752	.827	.807	.779	.793	.485	.406
		CPU-tf2.6.2	.751	.825	.803	.782	.792	.482	.407
		GPU	.751	.825	.804	.780	.792	.483	.407
	LSTM-DKT-S+	CPU-tf2.1.0	.751	.826	.805	.779	.792	.483	.406
		CPU-tf2.6.2	.751	.825	.804	.780	.792	.482	.407
		GPU	.751	.825	.804	.780	.792	.482	.407
	SAKT	CPU-tf2.1.0	.753	.828	.809	.777	.793	.488	.405
		CPU-tf2.6.2	.752	.828	.817	.764	.789	.492	.406
		GPU	.753	.828	.807	.779	.793	.486	.405
	Vanilla-DKT	CPU-tf2.1.0	.750	.822	.805	.778	.791	.481	.409
		CPU-tf2.6.2	.748	.822	.804	.774	.789	.478	.409
		GPU	.749	.821	.804	.777	.790	.479	.409