EasyDKT: an easy-to-use framework for Deep Knowledge Tracing

Gabriella Casalino[1,*], Mattia Di Gangi[2,*], Francesco Ranieri[2], Daniele Schicchi[3,*] and Davide Taibi[3,*]

[1]*Computer Science Department, University of Bari, Bari, Italy*
[2]*AppTek GmbH, Aachen, Germany*
[3]*Institute for Education Technology, National Research Council of Italy, Palermo, Italy*

Abstract
The goal of knowledge tracing (KT) is to track a students' progress over time by analyzing their historical data, so as to predict their future performance on tests related to the topics they have covered. The rise of online platforms for education, where the learning process is embedded, unlocked the potential of customized teaching such as in intelligent tutoring systems. Thanks to ongoing advancements in KT algorithms, teachers can now be aware of students' needs and recommend appropriate learning resources. They can also rank learning content, skipping or delaying content based on difficulty. In recent years, Deep Knowledge Tracing (DKT) has proven highly effective in solving KT tasks due to its ability to model complex long-range dependencies in test sequences, resulting in better prediction quality. The field of DKT is expanding, with numerous algorithms being proposed and implemented using various technologies. This paper introduces a new framework called EasyDKT, which simplifies the development and evaluation process for DKT algorithms. The framework aims at offering users a high level of technological abstraction, with a modular structure that considers data processing, evaluation metrics, and neural network models to be trained on custom datasets. Currently, EasyDKT supports PyTorch and TensorFlow, with plans to incorporate additional technologies in the future. Experiments on the ASSISTments skill-builder dataset 2009-2010 show a case study of students' data analysis through EasyDKT.

Keywords
Deep Knowledge Tracing, Education, Deep Learning, Artificial Intelligence

1. Introduction

Learning is the process of obtaining fresh knowledge, adopting new behaviors, acquiring new skills, and developing new values, attitudes, and preferences. A meaningful learning experience concerns integrating recently acquired information with existing knowledge to exploit the acquired knowledge in several situations and contexts.
The learning process is a highly personalized experience encompassing many activities, such as

AIxEDU: 1st International Workshop on High-performance Artificial Intelligence Systems in Education, November 06–09, 2023, Rome, Italy
*Corresponding author.
gabriella.casalino@uniba.it (G. Casalino); daniele.schicchi@itd.cnr.it (D. Schicchi); davide.taibi@itd.cnr.it (D. Taibi)
https://orcid.org/0000-0003-0713-2260 (G. Casalino); https://orcid.org/0000-0003-4491-0025 (M. Di Gangi); https://orcid.org/0000-0003-0154-2736 (D. Schicchi); https://orcid.org/0000-0002-0785-6771 (D. Taibi)

reading, writing, listening, observing, thinking, and testing. Recognizing that every student has a unique learning pace and requirements is crucial, so personalized learning is essential to optimize the learning experience and ensure maximum benefit. Moreover, each student has followed a personal learning path. In this sense, Knowledge Tracing (KT) supports personalized learning by analyzing students' previous interactions with specific topics to predict their performance on future tests. Teachers can use KT algorithms to pinpoint their students' learning needs and suggest relevant materials accordingly. This also allows them to prioritize learning content by postponing or skipping material that may be particularly challenging. These advancements have significantly improved the educational experience for students and have enabled teachers to provide more individualized and effective instruction [1].
According to the Beijing consensus [2], Artificial Intelligence (AI) can support personalized learning, offering systems capable of recognizing the students' needs and offering them valid support [3, 4, 5, 6, 7]. New studies on KT have leveraged AI to develop self-governing systems to monitor student competencies. Deep Knowledge Tracing (DKT) utilizes deep learning to enhance the analysis of intricate, far-reaching connections in assessment sequences that depict a student's abilities [8, 9]. DKT is an expanding area investigating many algorithms developed through various technologies. To make the usage of DKT easier, this paper proposes EasyDKT, an innovative framework that offers users a high level of technological abstraction in implementing DKT. We leveraged a modular design that makes the framework easy to extend with other DKT models, integrating and combining custom datasets. In addition, EasyDKT abstracts the data processing and the evaluation stage, facilitating tasks such as comparing several DKT models. It has been implemented in Python and supports PyTorch [10], and TensorFlow [11].
Currently, EasyDKT implements the original DKT model proposed by Piech et al. [12]. Such a choice is due to the model's importance and the difficulties for the scientific community to implement it with modern frameworks since the original software libraries no longer work. In this way, we contribute to the research field by offering a modern version of the original DKT model that achieves the same performance and that is easily accessible. To validate the implementation, we presented a case study using EasyDKT to analyze the well-known ASSISTments skill-builder dataset 2009-2010. Experiments have been conducted varying the model's hyperparameters to validate the effectiveness of the proposed tool. We achieved a maximum value of AUC of 0.84, very close to 0.86 reported in [12].

The paper is organized as follows: Section 2 briefly introduces the main ideas behind Knowledge Tracing. Section 3 reviews the literature of Deep Knowledge Tracing, from the first model to more recent advancement. The EasyDKT framework is then presented in Section 4, together with details of the modules it is composed by. The experimental design and the evaluation results are presented in Section 5. Finally, section 6 concludes the paper and depicts future direction for this research.

2. Knowledge Tracing

Corbett and Anderson proposed the former model of Knowledge Tracing (KT) in their ACT Programming Tutor (APT), and it was intended to guide students in Lisp programming activities [13]. As illustrated in figure 1, the idea behind the KT theory is that the student's knowledge

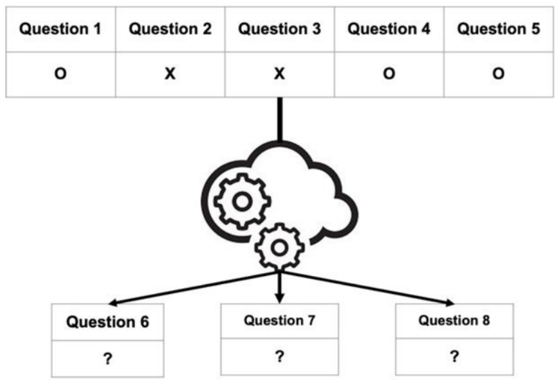


Figure 1: Base schema of Knowledge Tracing.

can be modeled if the domain knowledge is organized into a hierarchical structure of skills that is proposed during the learning experience, so as students can master low-level skills before approaching to the highest level skills. It is assumed that students first acquire knowledge through declarative form, followed by the acquisition of domain-specific procedural knowledge through practical tasks. In particular, a set of the rules (skills and sub-skills) that the student should have known is defined, and for each exercise, the probability that the student has learned each rule is evaluated. Analyzing the student's interactions the system can recommend activities that improve the student's competencies, such as analyzing and studying simpler topics preparatory to the main topic [13].
Autonomous Knowledge Tracing belongs to the Intelligent Tutoring System (ITS) field, which utilizes cognitive models to evaluate students' understanding. The system adapts its feedback and guidance based on the student's knowledge, thereby enhancing the quality and speed of learning. A model is developed for the student by analyzing their progress in a sequence of tasks. The algorithm closely monitors each exercise's outcomes, noting any successful or unsuccessful attempts. This data is then used to predict how well the student will perform in the subsequent exercises.
Probabilistic models based on Bayes theory were mostly used to to estimate learners' knowledge states over time [14]. However, recent advancements in Deep Learning have shown its effectiveness in tackling KT. In Deep Knowledge Tracing students are modeled individually. Each student's abilities are represented by predicted probabilities of using specific skills to solve exercises. The model automatically extracts hidden skills from the student's past interactions and uses historical student data to predict their likelihood of mastering the next item and the skills involved, along with their probability. This enables identifying students who require extra assistance or recommending learning resources based on the acquired skills. For further information on this topic, please refer to [15].

3. Literature Review

Recurrent Neural Networks have been commonly used to address the KT problem, demonstrating impressive results in forecasting a student's performance by reviewing their previous interactions. These models examine the sequence of question-answer pairs $\{q_t, a_t\}$ over a period to forecast the student's response at time $t + 1$.
Piech et al. [12] were the first to introduce the concept of "Deep Knowledge Tracing". They suggested that the task should be formulated as a temporal application because the student's knowledge increases over time. The authors experimented with both classical and LSTM RNNs to analyze the data representing the student's history. Their goal was to trace the acquired knowledge and predict future performances without hard-coding the student competencies, which is a demanding task that requires expert annotators. This work leverages three different sets of data: *Simulated-5*, *Khan Math*, and *ASSISTments "skill builder"*. Simulated-5 involves 4,000 virtual students answering 50 exercises based on 5 concepts. The students' knowledge is modeled by the Item Response Theory, and the skills improve gradually. Each exercise covers a specific concept and is labeled with a level of difficulty. The Khan Math is a collection of data from Khan Academy, consisting of 1.4 million exercises completed by 47,495 students across 69 categories. ASSISTments is a dataset used for building an Intelligent Tutor System that helps students with math problems. The tutor outputs a log of actions performed by the student every time they correctly complete an exercise. This publicly available data covers the time period of 2009-2010 and is a significant resource for addressing the KT problem using ML.

Subsequently, Xiong et al. [16] have revised the preliminary score presented by Piech et al. [12], uncovering issues that were not considered. The authors have considered the issues and tested the RNNs on more reliable data sets. The final results show high performance achieved by using RNNs, but the performance gap with previous models was reduced.
Scientists have been studying the qualities of DKT after its innovative introduction. According to Khajah et al. [17], RNNs incorporate the recency effect, meaning the model aligns with human reasoning processes as it gives more importance to recent events over past ones. Since DTK's input is the sequence of exercises a student receives in the same order, it can contextualize trial sequences. This helps us understand how the exercise sequences impact the student's learning. DKT can predict a student's performance on the next exercise based on their achievement history and can also determine the degree of relatedness among skills.

It is not possible for the DKT system to determine if a student has fully grasped a particular concept. To tackle this problem, a new model called Dynamic Key-Value Memory Networks (DKVMN) was suggested by Zhang et al. [18]. This model has the ability to learn the relationships between different concepts and give an accurate assessment of a student's understanding level for each individual concept. DKVMN is inspired by the *Memory-Augmented Neural Network*[19], a particular neural network (NN) which exploits an external memory module to enhance the ability of the model to capture long-term dependencies. DKVMN uses two memory modules: a static matrix for knowledge concepts (key) and a dynamic matrix for student competencies (value) for each concept. A comparison was made between the approach used in the DKVMN model and the classical DKT model introduced by Piech et al. [12] using four distinct sets of data. The results indicate that the DKVMN model performs better in tracking the student's knowledge and provides a comprehensive outline of the level of mastery of each concept for every student.

Zhang et al. [18] have suggested an alternative method to enhance the performance of RNNs for the KT problem. They recommend analyzing the range of additional features captured by computer-based learning platforms and incorporating them into DKT models. The authors conducted an experiment to determine the impact of three factors on student performance: response time, number of attempts, and whether the first action was to request help. They then proposed a method for incorporating this information into RNN analysis. The results showed that augmenting the features considered led to better outcomes than the original DKT model. Minor changes were made to the original DKT structure to accommodate the richer set of student information and contextual insights that were deemed important for achieving improved results.

Currently, deep learning models utilizing Transformers are at the forefront of solving tasks involving temporal sequences. Tackling the KT has exploited such models in several respects. A study conducted by Pandey et al. [20] focused on improving Deep Knowledge Tracing (DKT) by utilizing self-attention-based neural networks. Their approach, called SAKT, analyzes a student's previous actions to determine which concepts they have mastered. SAKT outperforms previous deep learning-based systems in addressing the issue of sparse data, where students only interact with a few concepts, resulting in limited information. The SAKT system calculates attention weights to determine the importance of completed exercises when predicting a student's performance on a given exercise. By visualizing these attention weights, it becomes easier to see which completed exercises the network relied on to make a prediction. This helps to identify the relevant past exercises that the student used to solve the current exercise. SAKT has been extensively tested on real-world datasets and has shown an average improvement of 4.43% in AUC compared to previous DL models.

The use of feedback connection tracking time through positional encoding is not utilized by Transformers according to a study by [21]. However, recent developments have resulted in a new architecture called Transformer-XL. This architecture includes a recurrence mechanism and an updated positional encoding scheme, which allows for better capturing of longer-term dependencies compared to both RNNs and traditional Transformer models. In their work, He et al. [22] utilized the unique features of Transformer-XL to address issues arising from analyzing lengthy input exercise sequences, which have negatively impacted past DL models' performance. The system they developed, KT-XL, was thoroughly tested on three real-world datasets and compared with previous models such as DKT, DKVMN, and SAKT. KT-XL outperformed all other models across the datasets, with an average improvement of 3.6%.

New directions of Deep Knowledge Tracing research aim at improving students' knowledge modeling. A cognitive representation of students' skills that overcomes the common assumption of questions equivalent contributions has been proposed in [23]. A module to interpret the prediction results has also been included, to facilitate the use of DKT for the analysis of students behavior. The use of augmented knowledge have been proposed to better model students' skills. In [24] hierarchical heterogeneous knowledge structures are modeled through knowledge-graphs, whilst Tato et al. explored the use of multi-modal data to enhance the latent representations of students [25]. Spatial and temporal features obtained from students' activities history has been used to extract deeper hidden information in [26]. Students' exercises are used to derive spatial information that is then connected with temporal characteristics. Results shown that using more informative representations of students' knowledge helps in creating effective user models, leading to better predictive results than state-of-the-art algorithms for similar tasks.

4. Framework

In this paper, we introduce EasyDKT - a user-friendly framework that enables users to experiment with DKT algorithms through a convenient command-line user interface. The framework, shown in figure 2, consists of four modules for data management, creation of a neural network model, experimentation and validation. The framework allows to extend its modules with interchangeable classes and functions to further enhance the functionalities, which can be easily selected through a configuration file. EasyDKT implements Piech et al. [12] original neural network. However, since the technologies they used are no longer available, their experiments are not reproducible. Thus, the code has been refactored by using two of the most used neural network libraries, that are TensorFlow and PyTorch. User could select the preferred library, and could compare results obtained with different libraries and settings (as we did in the experimental part). The configuration file outlines the necessary data for training and evaluating the model and which module to use for managing data loading and preprocessing. It also includes the DKT algorithm, its hyperparameters, and the evaluation metrics used to monitor training progress and final results.

Particularly, we considered the following hyperparameters and the relative values, used to create and tune Deep Knowledge Tracing models:

- Library: Deep Learning Library (Pythorch, Tensor Flow - TF)
- Optimizer: RMSProp, Adam
- Dropout: dropout rate;
- Hidden Units: Number of LSTM hidden units;
- Batch size: Number of sequences to process in a batch;
- Learning rate;
- Epochs: Number of epochs;
- Time Window: Number of timesteps to process in a batch;

The framework has meant to be used as a baseline for comparisons, and as a basis technology to build new algorithms on. For this reason we separated four modules, incapsulating the functionalities required during a Deep Knowledge Tracing process. The four modules have been further detailed in the following:

Data managing

The first module is devoted to prepare student's data in the format that is required by the given library. Then, only information related to the student and the answer to a given question are considered for the processing.

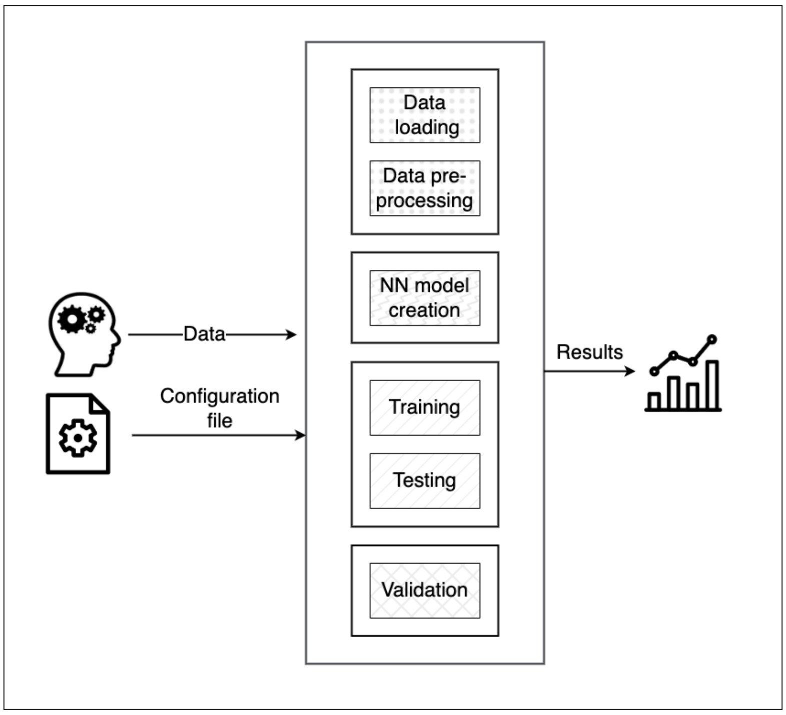


Figure 2: EASY-DKT framework.

Neural network model creation

Based on the configuration settings, the neural network model is created. The technologies are hidden in this module, which exposes an interface to communicate with the user through the configuration file.

Experimentation

Data is then divided in a training set to create the model, and a testing set to evaluate it. Since data is sequentially analysed, and this sequence is crucial for the deep learning models, we considered a train-test setting, rather than a more general cross-validation setting.

Validation

The predictive task has been evaluated in terms of the standard classification measure Area Under the Curve (AUC). Also, graphs with AUC values over epochs are generated in order to compare the stability and robustness of different configuration settings.

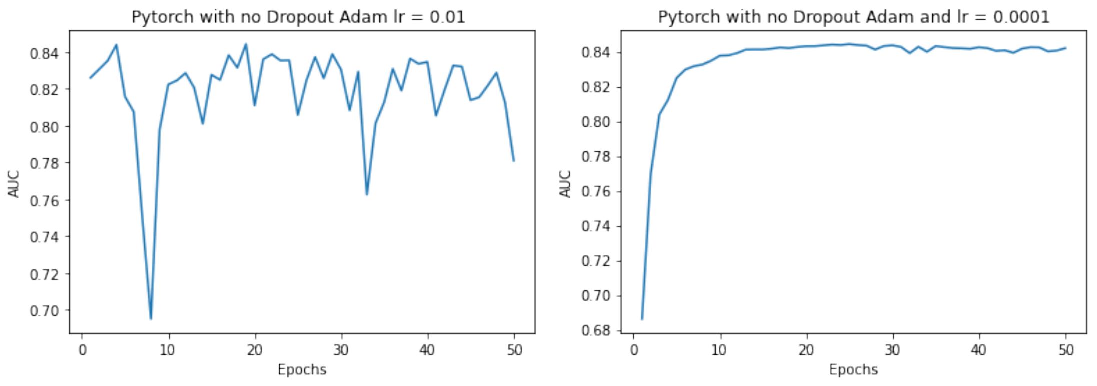


Figure 3: *On the left:* the worst Case achieved by using pytorch, the Adam optimizer, no dropout, and a learning rate of 0.01. *On the right:* The best performance achieved with the same configuration as the worst case but reducing the learning rate to 0.0001.

5. Experiments

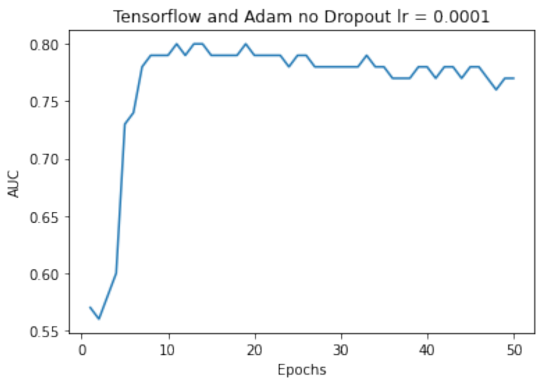
5.1. Data

The exploited data is the ASSISTments skill-builder dataset 2009-2010, created through an online platform and made it available for free [1]. This dataset includes mathematical skill-builder problems that students can solve, and their answers are recorded. The problems presented are designed to test specific skills, and some questions may require knowledge of multiple skills. To complete the test, students must answer three consecutive questions correctly. It's important to note that if a student uses any support or tutoring system provided by the platform itself, the question will be marked as incorrect. Additionally, students receive instant feedback to know if they answered the question correctly.
The dataset contains 4217 problems and a total of 124 skills. As some students may solve the same problem, the dataset actually consists of 522,000 tuples. Each tuple comprises three parts: a student identifier (id), a skill identifier, and the answer to the problem. If a problem relates to multiple skills, there will be multiple tuples with the same student id and answer but different skill identifiers.
The dataset was divided within the framework to use 3361 items for Deep Learning model training and 856 items for testing.

5.2. Results

During the experimental phase, our main objective was to determine the most effective approach for implementing the DKT via PyTorch and TensorFlow deep learning libraries. To achieve this, we conducted a series of rigorous experiments, carefully adjusting various model parameters, including the dropout rate, learning rate, and optimizer (i.e. RMSProp and Adam), and ensuring a comprehensive evaluation.
The model evaluation process has been carried out by computing the AUC (Area under the

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



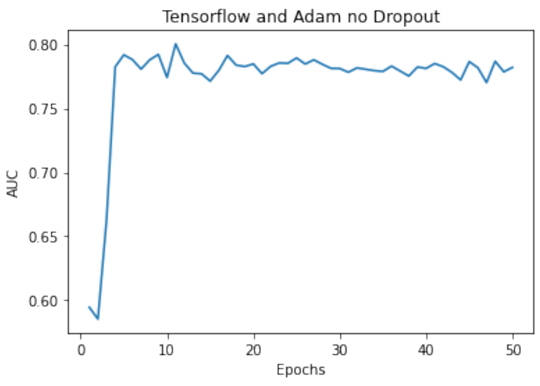


Figure 4: *On the left:* the worst Case achieved by using tensorflow, the Adam optimizer, no dropout, and a learning rate of 0.0001. *On the right:* The best performance achieved with the same configuration as the worst case but reducing the learning rate to 0.001.

ROC curve) on the tests performed on the ASSISTtments skill-builder data. The ROC curve is a statistical method that gauges the accuracy of a diagnostic test across the full spectrum of potential values. Measuring the area beneath the ROC curve is a widely recognized approach for assessing machine learning models.
The figures 3 and 4 display the highest and lowest performance results we obtained while conducting our experiments using the Pytorch and Tensorflow frameworks. We experimented with various configurations by adjusting the model's hyperparameters and looking at the AUC in the range of 50 epochs. A fully overview of the conducted experiments can be found in table 1.

Concerning pytorch, the worst performance was observed when Adam was the optimizer and had a learning rate of 0.01. In this case, there is an increase in the instability of the AUC value obtained at each iteration. The best performance was achieved with the same configuration but by reducing the learning rate to 0.0001. Starting from the twelfth training cycle, the AUC value stabilized at 0.84 until the end of the execution, resulting in a 2% increase in the final AUC value. With the best configuration, our framework achieves an AUC score of 0.84 comparable to the original score of 0.86 reported in [12], which was developed using outdated technologies that are now difficult to replicate. Instead, Figure 4 includes the results of our framework when using tensorflow. The lowest performance was observed with the configuration that employed Adam optimizer, did not use dropout, and had a learning rate of 0.0001. Despite using a high-performing configuration for Pytorch's algorithm, there was no improvement in the outcome compared to TensorFlow. In fact, the AUC value remained constant at 0.76, which is worse than the previous performance. On the other hand, the highest performance was achieved with the same configuration as the worst case but with a learning rate of 0.001. In this case, changing the optimizer from RMSProp to Adam does not significantly impact performance as the AUC fluctuates between 0.78 and 0.79.
We have observed that the learning rate has the most significant impact on improving the model's performance. In addition, even when using the same configuration, models developed with TensorFlow and PyTorch libraries show different performances. This highlights the

Table 1
Results of experiments with various configurations by adjusting the model's hyperparameters. AUC is the measure used to evaluate the model configuration in 50 epochs.

#	Lib	Opt	Dropout	Hiddenunits	Batchsize	LR	TW	Epochs	AUC
1	Pytorch	RMSProp	None	200	5	0.001	100	50	0.82
2	Pytorch	RMSProp	0.6	200	5	0.001	100	50	0.82
3	Pytorch	Adam	None	200	5	0.001	100	50	0.82
4	Pytorch	Adam	None	200	5	0.01	100	50	0.78
5	**Pytorch**	**Adam**	**None**	**200**	**5**	**0.0001**	**100**	**50**	**0.84**
6	TF	RMSProp	None	200	5	0.001	100	50	0.78
7	TF	Adam	None	200	5	0.001	100	50	0.79
8	TF	Adam	None	200	5	0.0001	100	50	0.76

differences between these two libraries, despite both aiming to achieve the same goal. Probably, different initialization parameters affect the training of the model leading to a result gap of 5%. In conclusion, the Pytorch implementation with Adam optimizer provided the best results. This highlights the importance of having an easy-to-use parameterizable workflow for DKT to identify the best configuration for a given problem quickly.

6. Conclusion and Future Works

We have introduced a modular framework that makes Deep Knowledge Tracing more accessible and efficient. Our framework incorporates Pytorch and Tensorflow, the two most significant deep-learning libraries, giving users the flexibility to choose their preferred one, while incapsulating the implementation details, so that users are not necessarily required to know how to use these libraries and their syntax. A simple configuration file has been used to define the experimental setup. EasyDKT is a first attempt to develop a simple tool for DKT, implementing all recent technologies. It has been conceived as the core of a more complex tool where more recent DKT methodologies are encapsulated as hierarchical building blocks. A case study exploring the use of EasyDKT with the ASSISTments skill-builder dataset has been presented. Particularly, we studied how the neural-network hyperparameters could affect the learning performance of the tool. Our experiments have demonstrated that when Pytorch is utilized, our framework attains state-of-the-art performance. However, some challenges are encountered when using Tensorflow.
Our future work involves improving the software structure of the framework to enable the execution of various algorithms with multiple execution parameters, which can result in more accurate outcomes. Additionally, we aim to modify the NN model structure based on the preferences of experienced users. We plan to analyze specific components of the tensorflow implementation, such as tensor initialization, and compare them with those of Theano to achieve superior results (AUC = 0.78). We also propose modifying the dataset read-from-file section to make it more adaptable to different datasets and implementations. This will make the framework more versatile and usable with different datasets, structures, and formats. Finally, we plan to enhance the tool by including more advanced DKT algorithms, and by providing an intuitive

interface to facilitate the user experience.

Acknowledgment

Gabriella Casalino acknowledges funding from the European Union PON project Ricerca e Innovazione 2014-2020, DM 1062/2021. This work is partially funded by Bando per Progetti di Ricerca GNCS 2023 - CUP E53C22001930001. G.C. is a member of the INdAM GNCS research group.

References

[1] A. T. Corbett, J. R. Anderson, Knowledge tracing: Modeling the acquisition of procedural knowledge, User modeling and user-adapted interaction 4 (1994) 253–278.

[2] UNESCO, Beijing consensus on artificial intelligence and education, 2019.

[3] D. Taibi, G. Fulantelli, V. Monteleone, D. Schicchi, L. Scifo, An innovative platform to promote social media literacy in school contexts, in: ECEL 2021 20th European Conference on e-Learning, Academic Conferences International limited, 2021, p. 460.

[4] G. Lo Bosco, G. Pilato, D. Schicchi, Deepeva: a deep neural network architecture for assessing sentence complexity in italian and english languages, Array 12 (2021) 100097.

[5] D. Schicchi, G. Pilato, G. L. Bosco, Attention-based model for evaluating the complexity of sentences in english language, in: 2020 IEEE 20th Mediterranean Electrotechnical Conference (MELECON), IEEE, 2020, pp. 221–225.

[6] G. Casalino, G. Castellano, G. Zaza, Neuro-fuzzy systems for learning analytics, in: International Conference on Intelligent Systems Design and Applications, Springer, 2021, pp. 1341–1350.

[7] G. Casalino, G. Castellano, C. Mencar, Incremental and adaptive fuzzy clustering for virtual learning environments data analysis, in: 2019 23rd International Conference Information Visualisation (IV), IEEE, 2019, pp. 382–387.

[8] G. Casalino, L. Grilli, P. Limone, D. Santoro, D. Schicchi, et al., Deep learning for knowledge tracing in learning analytics: an overview., TeleXbe (2021).

[9] X. Song, J. Li, T. Cai, S. Yang, T. Yang, C. Liu, A survey on deep learning based knowledge tracing, Knowledge-Based Systems 258 (2022) 110036.

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, Advances in neural information processing systems 32 (2019).

[11] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL: https://www.tensorflow.org/, software available from tensorflow.org.

[12] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, J. Sohl-Dickstein, Deep knowledge tracing, Advances in Neural Information Processing Systems (2015) 505––513.

[13] A. T. Corbett, J. R. Anderson, Knowledge tracing: Modeling the acquisition of procedural knowledge, User modeling and user-adapted interaction (1994) 253-–278.

[14] O. Bulut, J. Shin, S. N. Yildirim-Erbasli, G. Gorgun, Z. A. Pardos, An introduction to bayesian knowledge tracing with pybkt, Psych 5 (2023) 770–786.

[15] G. Abdelrahman, Q. Wang, B. Nunes, Knowledge tracing: A survey, ACM Computing Surveys 55 (2023) 1–37.

[16] X. Xiong, S. Zhao, E. G. V. Inwegen, J. E. Beck, Going deeper with deep knowledge tracing, Proceedings of the 9th International Conference on Educational Data Mining (2016) 545-–550.

[17] M. Khajah, R. V. Lindsey, M. C. Mozer, How deep is knowledge tracing?, arXiv:1604.02416v2 (2016).

[18] L. Zhang, X. Xiong, S. Zhao, A. Botelho, N. T. Heffernan, Incorporating rich features into deep knowledge tracing, LS '17: Proceedings of the Fourth (2017) ACM Conference on Learning (2017) 169-–172.

[19] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, et al., Hybrid computing using a neural network with dynamic external memory, Nature 538 (2016) 471–476.

[20] S. Pandey, G. Karypis, A self-attentive model for knowledge tracing, arXiv preprint arXiv:1907.06837 (2019).

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017) 5998–6008.

[22] Y. He, X. Hu, Z. Zu, G. Sun, Kt-xl: A knowledge tracing model for predicting learning performance based on transformer-xl, ACM TURC'20: Proceedings of the ACM Turing Celebration Conference - China (2020) 175–179. doi:10.1145/3393527.3393557.

[23] J. Chen, Z. Liu, S. Huang, Q. Liu, W. Luo, Improving interpretability of deep sequential knowledge tracing models with question-centric cognitive representations, arXiv preprint arXiv:2302.06885 (2023).

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

[25] A. Tato, R. Nkambou, Towards a multi-modal deep learning architecture for user modeling, in: The International FLAIRS Conference Proceedings, volume 36, 2023.

[26] L. Lyu, Z. Wang, H. Yun, Z. Yang, Y. Li, Deep knowledge tracing based on spatial and temporal representation learning for learning performance prediction, Applied Sciences 12 (2022) 7188.