Performance Analysis of Knowledge Tracing Models with Mixed Precision: A Comparative Study on Server and Raspberry Pi Environments

Junhyeong Park, Chengxing Zou, Inseo Kim, Myung Gyu Park, and Jinsung Kim
School of Computer Science and Engineering
Chung-Ang University
Seoul, Republic of Korea
{acorn0415, zcx787924758, inseo764, audrb1999, kimjsung}@cau.ac.kr

***Abstract*—With the recent advancements in artificial intelligence (AI) and the improved performance of mobile devices, Knowledge Tracing (KT) models have gained significant attention in the education area, where they play a crucial role in tracking students' learning progress and providing personalized learning experiences. This study aims to compare the performance of KT models on a Raspberry Pi and a GPU server while examining the feasibility of model lightweighting using the Mixed Precision technique in low-power environments. Specifically, we performed fine-tuning and inference of pretrained models on both low-power devices and high-performance servers. The experimental results showed minimal performance differences in terms of AUC between the two environments, while accuracy (ACC) varied across models. Additionally, Attention-based models experienced significantly longer inference times when AMP was applied on low-power devices, indicating that complex computations require more resources on such devices. Based on these findings, applying AMP to the DKT+ model was identified as the most suitable option for low-power environments.**

***Index Terms*—Knowledge Tracing, Mixed-Precision, low-power device**

I. Introduction

With the recent advancements in artificial intelligence (AI), the performance of mobile devices has significantly improved, leading to the release of various AI-powered mobile applications. In particular, AI-based learning apps such as Santa TOEIC have gained significant attention in the education sector. Among these, Knowledge Tracing (KT) models play a crucial role in assessing students' levels and providing personalized learning content. The importance of KT models continues to grow, and if these models could be trained and inferenced on mobile devices, they could offer more personalized learning experiences.

By performing KT models on mobile devices, the reliance on communication with central servers can be reduced, alleviating traffic congestion, and ensuring stable services even with a large number of users. In this context, this study proposes the training and inference of KT models on mobile and low-power devices, while also verifying the feasibility of lightweighting models using Mixed Precision techniques. We conducted experiments with various KT models to evaluate their performance.

II. Related Work

A. Knowledge Tracing

Knowledge Tracing (KT) is a model that predicts the likelihood of a student answering the next question correctly based on their learning history of correct and incorrect responses. This model measures the student's understanding or mastery level of a particular skill associated with the question. KT models have evolved in various forms to track individual students' learning progress and offer personalized learning experiences.

The initial form of KT, Bayesian Knowledge Tracing (BKT) [1], used a Bayesian network to track a student's learning progress. However, BKT had limitations in adequately reflecting past data in the student's learning history. To address this issue, the DKT (Deep Knowledge Tracing) model [2], based on LSTM (Long Short-Term Memory) [3], was introduced. DKT provided improved performance in KT tasks by accounting for long-term dependencies in learning sequences.

Following DKT, DKT+ (Deep Knowledge Tracing Plus) [4] was developed to further address the limitations of DKT. While DKT demonstrated strong performance, it lacked the ability to deeply capture the interaction between students and specific problems and underutilized the complexity of input data. DKT+ enhanced the input data by incorporating metadata such as problem IDs, difficulty levels, and concept tags to provide richer contextual understanding. It also introduced input reconstruction regularization and prediction consistency regularization to improve inference performance. These enhancements allowed DKT+ to achieve higher accuracy and deliver more consistent predictions compared to DKT.

Subsequent advancements in KT models introduced memory mechanisms, such as in the DKVMN (Differentiable Key-Value Memory Network) [5]. This model enhanced the storage and retrieval of past information, allowing more accurate predictions of student performance. Following DKVMN, the SAKT (Self-Attention Knowledge Tracing) model [6] emerged, which applied the Self-Attention mechanism [7]. This mechanism offered more flexibility and faster performance in processing and predicting learning data. The self-

attention mechanism was instrumental in overcoming the limitations of LSTM models by dynamically reflecting the importance of learning history.

Recent studies have highlighted the MAMBA model [8] as the next-generation successor to transformers, and with the introduction of MAMBA4KT [9], learning performance and efficiency have been significantly improved. MAMBA4KT, in particular, excels in the field of Knowledge Tracing by processing learning data more effectively. Compared to previous models, it provides faster and more accurate results, demonstrating a groundbreaking improvement in learning performance. Despite the development of these advanced models, collecting and processing data on central servers can lead to network bottlenecks. This problem becomes particularly severe when large numbers of students access the system simultaneously.

To solve this network bottleneck issue, one approach is to enable the models to perform training and inference directly on mobile or low-power devices. This approach allows real-time performance improvements on each device, reduces reliance on central servers, and alleviates traffic issues. Moreover, it can provide a more personalized learning environment for students.

In this study, we compare the performance of LSTM-based models such as DKT and DKVMN, and attention-based models like SAKT. Specifically, we evaluate the inference performance of these models after fine-tuning them on low-power devices such as the Raspberry Pi, and compare this to the performance on GPU servers. Additionally, we investigate the effect of model lightweighting using Mixed Precision techniques, analyzing the performance differences to assess the effectiveness of these models on low-power devices. Through this analysis, we explore the practical viability of performing training and inference on mobile environments.

B. Mixed-Precision

Mixed Precision is a technique that reduces the precision of floating-point operations from 32-bit to 16-bit to improve the speed of deep learning model training and inference, while also reducing memory usage. This allows for faster computations with fewer resources, making it particularly effective in low-power devices or constrained hardware environments. By applying Mixed Precision, some operations are processed in 16-bit instead of 32-bit, maximizing both speed and memory efficiency. In this experiment, we utilized PyTorch's AMP (Automatic Mixed Precision) functionality to implement Mixed Precision. AMP automatically selects the optimal precision during the computation process, aiding in model lightweighting, and it allows the use of Mixed Precision with minimal code modifications. In this study, we hypothesized that applying Mixed Precision would reduce the model size, making it more suitable for low-power devices, and decrease inference time. Based on this hypothesis, we conducted experiments comparing the performance of models with and without Mixed Precision.

III. Experiment

A. Experimental Environment

This experiment was conducted in two different environments. The first environment was a GPU server, where an NVIDIA RTX 4090 graphics card was used for training, fine-tuning, and inference using PyTorch. This high-performance hardware setup provided fast computations and high accuracy, ideal for demanding tasks.

The second environment was a low-power device, specifically the Raspberry Pi 400 based on the Raspberry Pi 4 model. To ensure smooth execution of the experiment, we installed Ubuntu 22.04 as the operating system and optimized the environment for seamless use of Python libraries. The Raspberry Pi had a total storage capacity of 32GB and 4GB of RAM, providing a constrained resource setting to evaluate model performance in low-power conditions. On the server, we utilized the GPU for training and inference, while on the Raspberry Pi, all fine-tuning and inference were conducted using the CPU. The ASSISTMENT2009 dataset was utilized to evaluate the performance during both training and inference.

B. Result and Analysis

TABLE I
Comparison of AUC and ACC between Raspberry Pi and GPU Server.

Model	GPU		Raspberry Pi	
	AUC	ACC	AUC	ACC
dkt	0.7818	0.7430	0.7837	0.7438
dkt_amp	0.7805	0.6942	0.7825	0.6953
dkt+	0.8085	0.7630	0.8066	0.7618
dkt+_amp	0.7996	0.5325	0.8005	0.5364
dkvmn	0.8084	0.7660	0.8088	0.7643
dkvmn_amp	0.8067	0.7649	0.8070	0.7647
sakt	0.7936	0.7566	0.7942	0.7574
sakt_amp	0.7908	0.7588	0.7917	0.7582

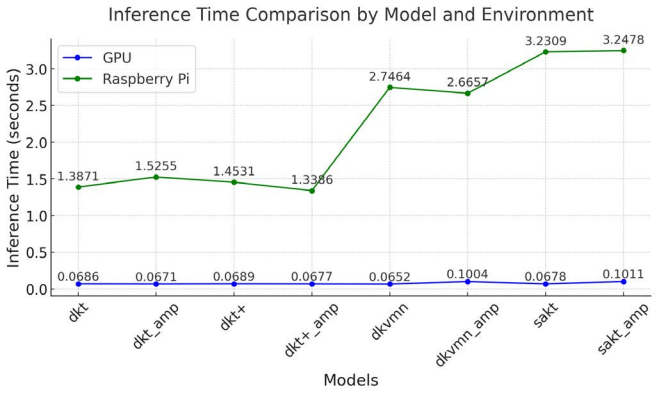

Fig. 1. Inference Time Comparison by Model with and without AMP.

In this study, we conducted experiments comparing the fine-tuning and inference performance on a Raspberry Pi [10] and a GPU server. The experimental setup utilized the CPU of the low-power Raspberry Pi and the high-performance GPU server

to evaluate the performance of the same models. Specifically, pretrained models were fine-tuned and subsequently evaluated in both environments to compare the results.

Table 1 summarizes the comparison of AUC (Area Under the Curve) and ACC (Accuracy) for each model between the Raspberry Pi and GPU server. This comparison clearly highlights the effects of AMP (Automatic Mixed Precision) application and the performance differences across the two environments. In most models, the AUC differences between the GPU and Raspberry Pi were within ±0.26%, indicating that high performance can be maintained even on low-power devices. However, in terms of ACC, some models exhibited significant differences. Notably, the DKT+ model showed a significant 30% decrease in ACC when AMP was applied, whereas the other LSTM-based model, DKT, showed a smaller difference of approximately 6.5%, demonstrating relatively stable performance. In contrast, Attention-based models (e.g., SAKT and DKVMN) exhibited minimal ACC differences of less than 0.3%, regardless of AMP application.

Additionally, when models trained on the GPU were fine-tuned on the Raspberry Pi and evaluated on the CPU, most models exhibited minimal performance degradation. The AUC differences between the two environments were mostly within ±0.26%, and ACC differences were similarly small across most models. However, in some cases, such as the DKT+ model, applying AMP resulted in a noticeable drop in accuracy.

Figure 1 illustrates the inference time comparison of each model between the GPU server and the Raspberry Pi. On the GPU server, applying AMP resulted in approximately a 50% increase in inference time for Attention-based models (e.g., SAKT, DKVMN), while LSTM-based models (e.g., DKT, DKT+) exhibited a relatively stable increase of within 2%. This is likely due to the additional data transformation and precision adjustment overhead caused by AMP, which more significantly affects the complex computations of Attention mechanisms on GPUs.

When comparing the inference time between the Raspberry Pi and the GPU, all models showed significantly longer execution times on the Raspberry Pi. For Attention-based models (e.g., SAKT, DKVMN), the Raspberry Pi had up to a 4000% increase in inference time compared to the GPU. This dramatic difference can be attributed to the computational complexity of Attention mechanisms, which rely heavily on parallel processing, a feature that is limited in low-power CPU-based environments. For LSTM-based models (e.g., DKT, DKT+), the difference was smaller but still significant, with the Raspberry Pi showing up to a 2000% increase in inference time compared to the GPU. This highlights the computational constraints of the Raspberry Pi relative to the GPU.

Additionally, the effect of AMP on inference time varied depending on the model type. On the Raspberry Pi, LSTM-based models showed a more noticeable increase of up to 10%, whereas Attention-based models demonstrated only minor differences of between 0.5% and 2%. This can be explained by the fact that Attention-based models inherently have longer baseline inference times, making the percentage impact of AMP-related overheads appear smaller. Moreover, in low-power CPU environments, the benefits of parallel optimization offered by AMP are limited, resulting in a smaller overall impact on inference time for Attention-based models.

In summary, the impact of AMP application and the differences between the GPU and Raspberry Pi environments depend significantly on the model structure (LSTM vs. Attention). On the GPU, AMP application led to significant increases in inference time for Attention-based models, while LSTM-based models remained relatively stable. Conversely, on the Raspberry Pi, Attention-based models had longer baseline execution times compared to LSTM-based models, but AMP application caused only minimal differences in their inference times. These results highlight the distinct hardware characteristics and computational behaviors of GPUs and CPUs.

IV. Conclusion

The impact of AMP (Automatic Mixed Precision) application and the performance differences between the GPU and Raspberry Pi varied significantly depending on the model structure (LSTM vs. Attention). In the GPU environment, applying AMP resulted in a roughly 50% increase in inference time for Attention-based models (e.g., SAKT, DKVMN), while LSTM-based models (e.g., DKT, DKT+) showed a relatively stable increase of within 2%. This can be attributed to the data transformation and precision adjustment overhead, which had a more pronounced effect on the complex computations of Attention-based models.

In contrast, in the Raspberry Pi environment, Attention-based models exhibited longer baseline inference times compared to LSTM-based models, but the percentage increase due to AMP application was minimal, ranging from 0.5% to 2%. LSTM-based models, on the other hand, showed a more noticeable increase of up to 10%, which can be explained by the limited parallel optimization benefits in CPU-based environments. Additionally, when comparing inference times between the GPU and Raspberry Pi, Attention-based models showed up to a 4000% increase, while LSTM-based models exhibited up to a 2000% increase, highlighting the computational constraints of low-power CPU environments.

In summary, AMP application had varying effects depending on the environment. On GPUs, the complex computational structure of Attention-based models resulted in increased inference times when AMP was applied. Conversely, on the Raspberry Pi, Attention-based models, despite having longer baseline execution times, showed minimal differences in inference times due to AMP. These results reflect the distinct hardware characteristics and computational behaviors of GPUs and CPUs, demonstrating that the impact of AMP application is highly dependent on both the model structure and the computational environment.

In future research, we plan to expand our experiments beyond the Raspberry Pi to include a variety of widely used mobile devices, enabling a broader evaluation of model

performance in low-power environments. Furthermore, by applying not only mixed precision techniques but also various other optimization methods, we aim to effectively narrow the performance gap between high-performance GPUs and low-power devices.

ACKNOWLEDGMENT

This work was partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. NRF-2022R1G1A1013586) and Korea Institute for Advancement of Technology(KIAT) grant funded by the Korean Government(MOTIE) (P0020632, HRD Program for Industrial Innovation).

REFERENCES

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

[2] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. J. Guibas, and J. Sohl-Dickstein, "Deep knowledge tracing," *Advances in neural information processing systems*, vol. 28, 2015.

[3] A. Graves and A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[4] C.-K. Yeung and D.-Y. Yeung, "Addressing two problems in deep knowledge tracing via prediction-consistent regularization," in *Proceedings of the fifth annual ACM conference on learning at scale*, 2018, pp. 1–10.

[5] J. Zhang, X. Shi, I. King, and D.-Y. Yeung, "Dynamic key-value memory networks for knowledge tracing," in *Proceedings of the 26th international conference on World Wide Web*, 2017, pp. 765–774.

[6] S. Pandey and G. Karypis, "A self-attentive model for knowledge tracing," *arXiv preprint arXiv:1907.06837*, 2019.

[7] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[8] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752*, 2023.

[9] Y. Cao and W. Zhang, "Mamba4kt: An efficient and effective mamba-based knowledge tracing model," *arXiv preprint arXiv:2405.16542*, 2024.