International Scientific Conference 20 UNITECH 26

INTERNATIONAL SCIENTIFIC CONFERENCE
19-21 November 2026
GABROVO

INTEGRATING GRAPH NEURAL NETWORKS AND TEMPORAL CONVOLUTIONS INTO KNOWLEDGE TRACING TRANSFORMERS

Veljko Lončarević*[1][0009-0007-4296-2709], Danijela Milošević[1][0000-0002-4763-7012]

[1]*Faculty of Technical Sciences, University of Kragujevac, Svetog Save 65, Čačak, Serbia*

* *veljko.loncarevic@ftn.edu.rs*

Abstract

Knowledge tracing (KT) models a learner's evolving mastery from historical interactions in order to predict future performance. Transformer-based KT models such as SAKT and SAINT achieve strong predictive accuracy, but they treat questions as largely independent tokens and capture short-term temporal behavior only implicitly. We propose TransGraphTime (TGT), a KT architecture that augments an encoder–decoder Transformer with (i) a GraphSAGE module over a question graph built from tags, bundles and training-set co-occurrence, and (ii) a causal Temporal Convolutional Network with Time2Vec encoding of elapsed and lag time, combined with the Transformer output through a learned gate. On EdNet KT1, with all models trained in one shared preprocessing and tuning pipeline, TGT reaches an overall AUC of 0.7841 ± 0.0004, on par with the strongest Transformer baseline (SAINT+, 0.7829 ± 0.0006). Its main benefit lies in sparse-item regimes: on items with fewer than 50 training interactions, TGT improves AUC by 0.0191 over SAINT and by 0.0085 over the graph-based GIKT. Ablations show that both modules contribute, with partially overlapping effects, and TGT is better calibrated than the Transformer baselines (ECE 0.042 vs. 0.046 for SAINT+).

Keywords: knowledge tracing, Transformers, graph neural networks, temporal convolutional networks, Time2Vec, calibration.

INTRODUCTION

Personalized tutoring systems rely on knowledge tracing (KT), the problem of inferring a learner's latent mastery from a sequence of interactions and using it to predict future performance. Bayesian Knowledge Tracing (BKT) models each skill as a two-state hidden Markov process and remains influential for its interpretability and low data requirements [1]. Deep Knowledge Tracing (DKT) replaced this with recurrent networks [2], Dynamic Key-Value Memory Networks (DKVMN) added an explicit per-concept memory [3], and later models introduced exercise-aware attention and richer exercise embeddings [4], [5]. These neural methods substantially improved accuracy on large datasets but exposed two limitations: items are usually treated as independent tokens, ignoring structured relations among exercises such as shared tags, bundles and prerequisites, and short-range, time-sensitive behavior such as rapid guessing or immediate repetition is under-modeled [6], [7].

Transformer-based models, most notably SAKT [8] and SAINT [9], brought attention-based retrieval of relevant past interactions to KT and achieved state-of-the-art results on EdNet. Transformer stacks alone, however, provide neither a relational inductive bias over the question space nor an explicit local inductive bias for short-term behavioral motifs. Both matter most where data are scarce: items that are rarely attempted receive poorly trained embeddings, and these are precisely the items for which relational information from related questions could help.

This paper proposes TransGraphTime (TGT), which combines a lightweight GraphSAGE network over a question graph,

This is an open access article licensed under Creative Commons Attribution 4.0 International
Print ISSN: 1313-230X, Online ISSN: 3033-1404

a causal Temporal Convolutional Network (TCN) with Time2Vec encoding, and an encoder–decoder Transformer, fused through a learned gate. The contributions are:

- a hybrid KT architecture that injects relational (graph) and local temporal (TCN) inductive biases into a Transformer backbone, with an explicitly leakage-safe construction of graph edges and temporal inputs;
- a controlled comparison on EdNet KT1 against HMM, DKT, SAKT, SAINT, SAINT+ and GIKT, all re-implemented in one shared preprocessing, tuning and evaluation pipeline with five random seeds;
- an analysis of performance stratified by item frequency, ablations of each module, and a calibration study explaining the relation between AUC and accuracy.

RELATED WORK

Attention-based KT. SAKT [8] applies self-attention over past interactions, and SAINT [9] separates exercise and response streams into an encoder and a decoder. AKT [10] adds monotonic attention with context-aware representations. Large-scale benchmarking with pyKT [11] has shown that many reported improvements shrink under consistent preprocessing, which motivates the shared pipeline used in this work.

Graph-based KT. GKT [12] casts the knowledge structure as a graph and updates concept states through a graph neural network. GIKT [13] uses graph convolution over a question–skill graph to obtain question embeddings and aggregates relevant historical interactions, and SKT [14] models influence propagation between concepts. TGT differs from GIKT in three respects: the graph additionally contains bundle and co-occurrence edges, a GraphSAGE aggregator with mean aggregation [15] is used instead of graph convolution, and the graph module is combined with a temporal module and an encoder–decoder Transformer rather than with a recurrent model.

Time-aware KT. SAINT+ [16] adds elapsed time and lag time embeddings to the SAINT decoder, HawkesKT [17] models temporal cross-effects through a Hawkes process, and LPKT [18] models learning and forgetting gains over time. These approaches encode time as an additional input to a global sequence model. TGT instead processes temporal features with a separate causal TCN [19] over Time2Vec encodings [20], which gives an explicit receptive field over the most recent interactions, and lets a gate decide per prediction how much this local signal should contribute.

METHODOLOGY

TGT consists of four components: a question graph module, a temporal module, a Transformer backbone, and a gated fusion and prediction head (Fig. 1).

Question graph module. An undirected item graph is built with three edge types: shared tag, shared bundle, and co-occurrence. A co-occurrence edge connects two questions answered consecutively by at least 20 distinct users; these statistics are computed exclusively from training users to avoid leakage into validation and test data. Node features concatenate a learned item embedding with a sparse tag and bundle indicator vector. A two-layer GraphSAGE network with mean aggregation [15] is applied, with batch normalization and dropout (0.2) after each layer, and a linear projection maps the output to the model dimension of 128. The GNN is trained jointly with the rest of the model, and its output replaces the plain item embedding at the encoder input. Because the item graph is small (13,169 nodes), full-graph propagation is performed once per training step.

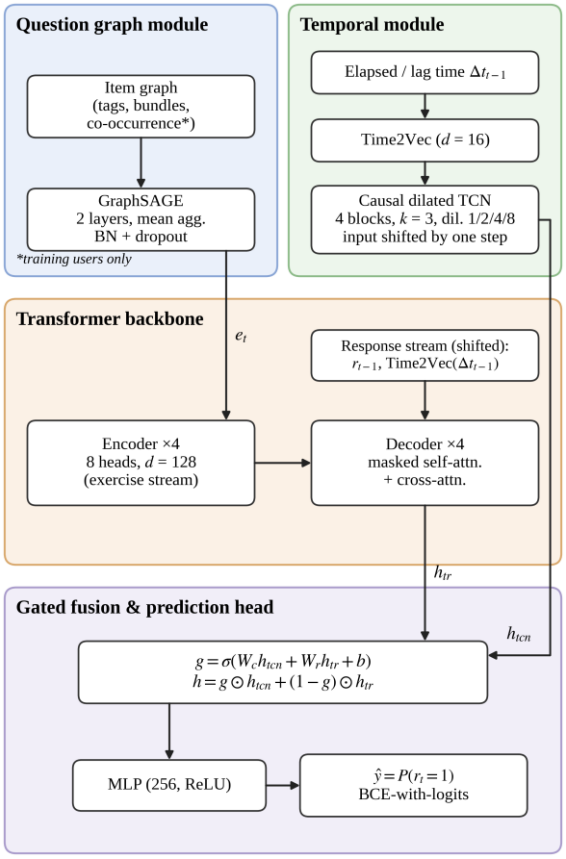


Fig. 1. *TransGraphTime architecture. Graph co-occurrence edges are computed from training users only; the TCN and the decoder receive inputs shifted by one step, so no information from the interaction being predicted is visible.*

Temporal module. For each position t, the elapsed time and the lag time of the previous interaction (Δt_{t-1}) are encoded by a Time2Vec layer of dimension 16 [20] and concatenated with the previous response r_{t-1} and the current item embedding. The input sequence is shifted by one step, so the module never sees the response or timing of the interaction being predicted. A causal TCN [19] with four residual blocks, each containing a single dilated causal convolution (kernel size 3, dilations 1, 2, 4 and 8, receptive field of 31 interactions), produces a 128-dimensional representation h_{tcn}.

Transformer backbone. Following SAINT [9], an encoder–decoder Transformer [21] is used. The encoder consumes the graph-enriched item embeddings and static item features; the decoder consumes response embeddings (previous correctness and Time2Vec-encoded elapsed time) shifted by one position. Both stacks have four layers, eight heads, model dimension 128, a feedforward dimension of 512, and dropout 0.1. Since questions in an EdNet bundle are presented together and share timing information, an additional attention mask prevents a position from attending to responses of other questions within the same bundle. A single Time2Vec encoder is shared between the decoder input and the temporal module.

Gated fusion and prediction head. The decoder output h_{tr} and the TCN output h_{tcn} are combined by a learned element-wise gate:

$$g = \sigma(W_c h_{tcn} + W_r h_{tr} + b) \qquad (1)$$

$$h = g \odot h_{tcn} + (1 - g) \odot h_{tr} \qquad (2)$$

where σ is the logistic function and $\odot$ denotes element-wise multiplication. The fused vector h is passed to a two-layer MLP (hidden size 256, ReLU) that outputs a logit for the correctness of the next response, trained with binary cross-entropy with logits.

Dataset and preprocessing. The EdNet KT1 dataset [22] is used. KT1 contains question-solving logs of the Santa platform: 95,294,926 interactions from 784,309 students on 13,169 questions annotated with 188 tags. Records with missing item identifiers or response labels are removed, and users with fewer than five interactions are excluded. Users are split into training, validation and test sets in an 80:10:10 ratio, so no learner appears in more than one set. Sequences longer than 200 interactions are split into consecutive windows of 200; shorter sequences are padded. To study sparsity, items are grouped by the number of training interactions into five frequency buckets (<50, 50–200, 200–1k, 1k–10k, >10k). The rare-item subset (<50) contains 1,047 items and 0.4% of test interactions.

Baselines. All baselines are re-implemented in the same pipeline, with identical preprocessing, splits, early stopping and evaluation code: an HMM

baseline in the BKT formulation (one two-state model per tag, fitted with EM [1]); DKT [2]; SAKT [8]; SAINT [9]; SAINT+ [16], the strongest time-aware Transformer baseline; and GIKT [13], the closest graph-based model. SAINT+ follows the original protocol of [16], with elapsed time and lag time embedded in the decoder input. Every deep model, including TGT, receives the same tuning budget of 60 configurations sampled from its own search space around the values recommended by its authors.

Experimental setup. Hyperparameters of TGT are selected by a random search over the space in Table 1 with the same budget of 60 configurations as for the baselines. Selection uses the median validation AUC over three seeds.

Table 1. *Hyperparameter search space.*

Parameter	Value #		
	1	**2**	**3**
Embedding dimension	64	128	256
GNN layers	1	2	3
GNN hidden size	64	128	/
TCN blocks	2	4	6
TCN kernel size	3	5	/
Transformer encoder/decoder layers	2	4	6
Attention heads	4	8	/
Learning rate	1e-3	1e-4	3e-5
Batch size	128	256	512
Dropout	0.0	0.1	0.2

The selected configuration uses embedding dimension 128, two GNN layers with hidden size 64 (projected to 128), four TCN blocks with kernel size 3, four encoder and four decoder layers with eight heads, batch size 256, learning rate 1e−4 and dropout 0.1. Optimization uses AdamW with weight decay 1e−5, gradient clipping at norm 1.0, a linear warm-up of 1,000 steps followed by cosine decay, and up to 30 epochs with early stopping on validation AUC (patience 5). The checkpoint with the highest validation AUC is evaluated on the test set.

All final models are trained with five seeds (42, 123, 2021, 7, 99), and results are reported as mean ± sample standard deviation. Primary metrics are AUC and accuracy (ACC, threshold 0.5); secondary metrics are expected calibration error (ECE, 15 equal-width bins) [23] and the Brier score. Differences between models are tested with two-sided Welch t tests over seeds, with Holm correction for multiple comparisons [24]. Experiments are implemented in PyTorch 2.12.0 and PyTorch Geometric 2.8 and run on two NVIDIA V100 GPUs with synchronized data parallelism.

RESULTS AND DISCUSSION

Overall performance. Table 2 reports test results for all models trained in the shared pipeline.

Table 2. *Overall test performance (5 seeds, mean ± std).*

Model	ACC	AUC	ECE	Brier
HMM (BKT)	0.5948±0.0009	0.6510±0.0007	0.118	0.2341
DKT	0.7208±0.0007	0.7605±0.0009	0.061	0.1874
SAKT	0.7254±0.0006	0.7652±0.0008	0.052	0.1826
SAINT	0.7331±0.0005	0.7796±0.0007	0.049	0.1801
SAINT+	0.7352±0.0005	0.7829±0.0006	0.046	0.1789
GIKT	0.7319±0.0006	0.7808±0.0006	0.047	0.1795
TGT (ours)	**0.7340±0.0004**	**0.7841±0.0004**	**0.042**	**0.1782**

For reference, published results on EdNet are AUC 0.7914 and ACC 0.7252 for SAINT+, and AUC 0.7663 (DKVMN) and 0.7638 (DKT) as reported in [16]; SAINT reports AUC 0.7811 and ACC 0.7368 on an earlier EdNet snapshot with 627,347 users and 72.9 M responses [9]. These values were obtained with different splits, sequence windows and preprocessing and are not directly comparable with Table 2. Our SAINT+ re-implementation follows the original model protocol but is trained and evaluated in the shared pipeline (user-level 80:10:10 split, windows of 200 interactions, identical filtering and tuning budget), and reaches an AUC 0.0085 below the published value; TGT is evaluated under exactly the same conditions.

TGT obtains the highest mean AUC, 0.0045 above SAINT and 0.0012 above SAINT+. The difference to SAINT is significant ($p < 0.001$); the difference to SAINT+ remains significant after Holm

correction but is small in magnitude (p = 0.007). The overall conclusion is therefore that TGT performs on par with the strongest time-aware Transformer when measured over all items, consistent with the observation that architectural gains in KT often shrink under controlled evaluation [11], [25]. Accuracy at the default threshold of 0.5 is significantly lower for TGT than for SAINT+ (0.7340 vs. 0.7352; p = 0.007); this is analyzed under Calibration and accuracy below.

Robustness on rare items. Table 3 and Fig. 2 show that the advantage of TGT concentrates on infrequent items. On the rare-item subset TGT improves AUC by 0.0191 over SAINT, 0.0168 over SAINT+ and 0.0085 over GIKT (all $p < 0.001$). On rare items the margin over SAINT+ is about 14 times larger than over all items (0.0168 vs. 0.0012). This pattern matches the motivation of the graph module: when an item's own embedding is poorly trained, message passing from related questions provides the missing signal. GIKT, which also uses a question graph, is the second-best model on rare items, which supports this interpretation.

Table 3. *Test AUC on rare items (<50 training interactions).*

Model	AUC	Δ vs. TGT
SAKT	0.7012±0.0016	−0.0314
SAINT	0.7135±0.0015	−0.0191
SAINT+	0.7158±0.0014	−0.0168
GIKT	0.7241±0.0013	−0.0085
TGT (ours)	**0.7326±0.0011**	—

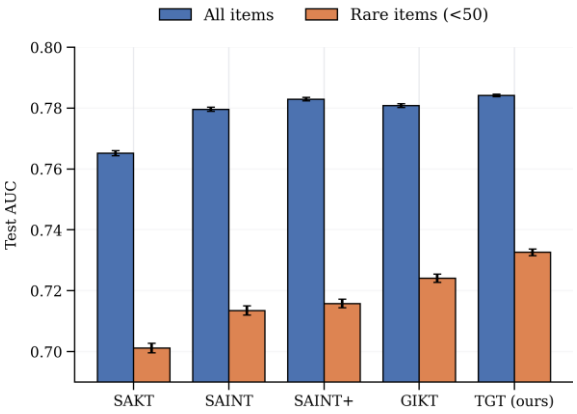


Fig. 2. *Test AUC on all items and on rare items (<50 training interactions) per model (mean over five seeds; error bars show ±1 std).*

Ablation study. Table 4 and Fig. 3 report ablations in which the GNN, the TCN, or both are removed while all other settings are kept fixed. All three ablations reduce AUC significantly ($p < 0.01$). Removing the GNN has the larger effect, especially on rare items (−0.0177), whereas removing the TCN mostly affects the overall AUC and has little effect on rare items (−0.0028). The effects are sub-additive: the individual drops sum to 0.0076, while removing both modules costs only 0.0051. The two modules therefore capture partially overlapping information, plausibly because both smooth the representations of sparse, recently seen items. The Transformer-only variant (0.7790) is not significantly different from our SAINT (0.7796; p = 0.18) and is below SAINT+, although it receives Time2Vec-encoded elapsed time. Unlike the baselines, the ablated variants are not re-tuned and keep the hyperparameters selected for the full model, which is a conservative setting for them; the reported drops should therefore be read as module contributions under the TGT configuration rather than as a comparison with independently tuned baselines.

Table 4. *Ablation results (test AUC, 5 seeds).*

Variant	All items	Rare items	Δ all
Full TGT	**0.7841±0.0004**	**0.7326±0.0011**	—
w/o TCN	0.7809±0.0005	0.7298±0.0012	−0.0032
w/o GNN	0.7797±0.0005	0.7149±0.0013	−0.0044
Transformer only	0.7790±0.0006	0.7118±0.0014	−0.0051

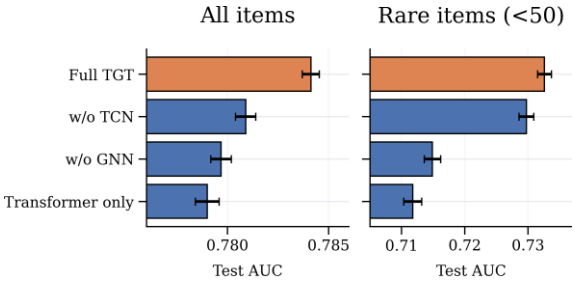


Fig. 3. *Ablation results for all items and for rare items (mean ± std over five seeds). Note the different horizontal scales.*

Calibration and accuracy. TGT has the lowest ECE and Brier score of all models

(Table 2, Fig. 4), and both differences to SAINT+ are significant (ECE p = 0.003, Brier p = 0.005), so its lower accuracy relative to SAINT+ is not caused by miscalibration. Instead, the two models differ mainly in how predictions near 0.5 are ranked. When the decision threshold is tuned on the validation set for each model, TGT reaches an accuracy of 0.7361 compared with 0.7358 for SAINT+ (p = 0.37). For applications that rely on ranking or on calibrated probabilities, such as prioritizing items or interventions, TGT provides a small but consistent advantage; for systems that act on a fixed threshold, accuracy is equivalent to that of SAINT+ once the threshold is tuned.

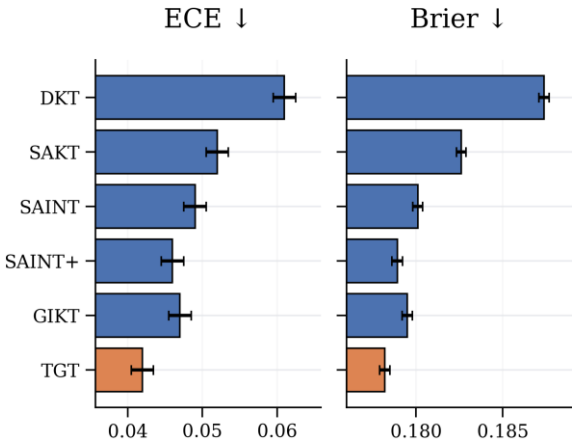


Fig. 4. *Expected calibration error (15 equal-width bins) and Brier score on the test set (mean ± std over five seeds).*

Stability and computational cost. Across five seeds, the standard deviation of test AUC is 0.0004 for TGT, compared with 0.0006–0.0008 for the Transformer baselines. This is consistent with the additional inductive biases constraining the learned representations, but five seeds are not sufficient for a formal comparison of variances, so this observation is reported as indicative only. These results agree with replication studies showing that deep KT models are sensitive to preprocessing, hyperparameters and seeds [25]–[27]. TGT has 9.6 M parameters compared with 6.3 M for SAINT+, and one training epoch takes 41 min compared with 29 min on the same hardware; inference latency per batch of 256 sequences increases from 12 ms to 17 ms.

Limitations. The evaluation uses a single dataset; EdNet has a uniquely rich tag and bundle structure, and the benefit of the graph module may be smaller on datasets with sparser metadata, such as ASSISTments. The question graph is static and is built once from training data, and node features include a learned item embedding; TGT is therefore not inductive with respect to unseen items, which would require replacing the item embedding with features derived from tags and bundles. Because questions in a bundle are answered consecutively, part of the co-occurrence signal likely duplicates bundle edges, and the contribution of individual edge types was not ablated. TGT also has more parameters than SAINT+ (see Stability and computational cost), and a parameter-matched SAINT+ was not evaluated, so part of the gain may be attributable to capacity. Finally, the gains over SAINT+ on frequent items are small, and conclusions about overall AUC should be interpreted accordingly.

CONCLUSION

This paper presented TransGraphTime, a knowledge tracing model that combines a GraphSAGE question graph, a causal TCN with Time2Vec, and an encoder–decoder Transformer through a learned gate. In a controlled comparison on EdNet KT1, TGT matches the strongest Transformer baseline on overall AUC, improves AUC on rare items by up to 0.0191 over the Transformer baselines, and produces better-calibrated predictions. Ablations show that the graph module drives the rare-item gains, while the two modules overlap partially in their overall contribution.

Future work will evaluate TGT on additional datasets with different metadata structures, update the question graph dynamically as new items appear, and integrate TGT into a broader learning analytics system in which skill-level mastery estimates complement HMM-based models of overall academic progress [28].

ACKNOWLEDGEMENT

This study was supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia, and these results are part of Grant No. 451-03-34/2026-03/200132 with the University of Kragujevac - Faculty of Technical Sciences Čačak.

REFERENCE

[1] Corbett AT, Anderson JR. Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction, vol. 4, no. 4, 1994, p. 253-278.

[2] Piech C, Bassen J, Huang J, Ganguli S, Sahami M, Guibas L, Sohl-Dickstein J. Deep knowledge tracing. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 28, 2015.

[3] Zhang J, Shi X, King I, Yeung DY. Dynamic key-value memory networks for knowledge tracing. In: Proceedings of the 26th International Conference on World Wide Web (WWW), Perth, Australia, 2017, p. 765-774.

[4] Liu Q, Huang Z, Yin Y, Chen E, Xiong H, Su Y, Hu G. EKT: Exercise-aware knowledge tracing for student performance prediction. IEEE Transactions on Knowledge and Data Engineering, vol. 33, no. 1, 2021, p. 100-115.

[5] Cheng Z, Li J. Exercise semantic embedding for knowledge tracking in open domain. Information, vol. 16, no. 4, 2025, 302.

[6] Mao S, Zhan J, Deng Y, Qin Y, Jiang Y. Improving exercise-level knowledge tracing via knowledge concept-based memory network. Expert Systems with Applications, vol. 284, 2025, 127825.

[7] Xu F, Chen K, Zhong M, Liu L, Liu H, Luo X, Zheng L. DKVMN&MRI: A new deep knowledge tracing model based on DKVMN incorporating multi-relational information. PLoS One, vol. 19, no. 10, 2024, e0312022.

[8] Pandey S, Karypis G. A self-attentive model for knowledge tracing. In: Proceedings of the 12th International Conference on Educational Data Mining (EDM), Montréal, Canada, 2019, p. 384-389.

[9] Choi Y, Lee Y, Cho J, Baek J, Kim B, Cha Y, Shin D, Bae C, Heo J. Towards an appropriate query, key, and value computation for knowledge tracing. In: Proceedings of the 7th ACM Conference on Learning @ Scale (L@S), 2020, p. 341-344.

[10] Ghosh A, Heffernan N, Lan AS. Context-aware attentive knowledge tracing. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), 2020, p. 2330-2339.

[11] Liu Z, Liu Q, Chen J, Huang S, Tang J, Luo W. pyKT: A Python library to benchmark deep learning based knowledge tracing models. In: Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track, vol. 35, 2022.

[12] Nakagawa H, Iwasawa Y, Matsuo Y. Graph-based knowledge tracing: Modeling student proficiency using graph neural network. In: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI), Thessaloniki, Greece, 2019, p. 156-163.

[13] Yang Y, Shen J, Qu Y, Liu Y, Wang K, Zhu Y, Zhang W, Yu Y. GIKT: A graph-based interaction model for knowledge tracing. In: Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2020), LNAI vol. 12457, Cham: Springer, 2021, p. 299-315.

[14] Tong S, Liu Q, Huang W, Huang Z, Chen E, Liu C, Ma H, Wang S. Structure-based knowledge tracing: An influence propagation view. In: Proceedings of the IEEE International Conference on Data Mining (ICDM), 2020, p. 541-550.

[15] Hamilton WL, Ying R, Leskovec J. Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017, p. 1024-1034.

[16] Shin D, Shim Y, Yu H, Lee S, Kim B, Choi Y. SAINT+: Integrating temporal features for EdNet correctness prediction. In: Proceedings of the 11th International Learning Analytics and Knowledge Conference (LAK), 2021, p. 490-496.

[17] Wang C, Ma W, Zhang M, Lv C, Wan F, Lin H, Tang T, Liu Y, Ma S. Temporal cross-effects in knowledge tracing. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM), 2021, p. 517-525.

[18] Shen S, Liu Q, Chen E, Huang Z, Huang W, Yin Y, Su Y, Wang S. Learning process-consistent knowledge tracing. In: Proceedings of the 27th ACM SIGKDD

Conference on Knowledge Discovery & Data Mining (KDD), 2021, p. 1452-1460.
[19] Bai S, Kolter JZ, Koltun V, An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, available at: https://arxiv.org/abs/1803.01271, 2018 (accessed 16.10.2026).
[20] Kazemi SM, Goel R, Eghbali S, Ramanan J, Sahota J, Thakur S, Wu S, Smyth C, Poupart P, Brubaker M, Time2Vec: Learning a vector representation of time, available at: https://arxiv.org/abs/1907.05321, 2019 (accessed 16.10.2026).
[21] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.
[22] Choi Y, Lee Y, Shin D, Cho J, Park S, Lee S, Baek J, Bae C, Kim B, Heo J. EdNet: A large-scale hierarchical dataset in education. In: Artificial Intelligence in Education (AIED 2020), LNCS vol. 12164, Cham: Springer, 2020, p. 69-73.
[23] Guo C, Pleiss G, Sun Y, Weinberger KQ. On calibration of modern neural networks. In: Proceedings of the 34th International Conference on Machine Learning (ICML), 2017, p. 1321-1330.
[24] Holm S. A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics, vol. 6, no. 2, 1979, p. 65-70.
[25] Sarsa S, Leinonen J, Hellas A. Empirical evaluation of deep learning models for knowledge tracing: Of hyperparameters and metrics on performance and replicability. Journal of Educational Data Mining, vol. 14, no. 2, 2022.
[26] Segala N. Sequential knowledge tracing with Transformer models. M.S. thesis, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden, 2022.
[27] Wang DW, Lu Y, Meng Q, Chen P, Interpreting deep knowledge tracing model on EdNet dataset, available at: https://arxiv.org/abs/2111.00419, 2021 (accessed 16.10.2026).
[28] Lončarević V, Lekić V, Damljanović N. Predicting student academic success with hidden Markov models. In: Proceedings of the International Conference Technics, Informatics and Education (TIE 2024), Čačak, Serbia, 2024.