Addressing Data Scarcity in Knowledge Tracing using Pre-Training on Synthetic Bayesian Knowledge Tracing Data

Ben Hocquet
Winchester Thurston
hocquetb@winchesterthurston.org

January 25, 2026

Abstract

This paper aims to combine the best of Bayesian Knowledge Tracing (BKT) and Deep Learning, training a model on synthesized BKT data then fine-tuning it based on real data using K-Split cross-validation to evaluate and predict a student's mastery through next-question prediction. The main objective of this work is to teach deep learning models the complex and often non-linear learning path students take on the road to mastery.

1 Introduction

Traditional learning aims to teach a student a topic in a given amount of time. At the end of the allotted time, the student is evaluated through a test and advanced, regardless of performance. Educational psychologist Benjamin Bloom popularized an innovated way of evaluating students; where instead the student is advanced based on a mastery score, this mastery score is compared to a percentage, which in Blooms study the passing criterion was 80-90%. Mastery learning is not without its advantages. Students who used it saw a 1.0σ improvement. Those who used one-on-one tutoring services saw this number rise to 2.0σ It's difficult to estimate a mastery score, as test results can certainly be deceiving. How can you know the student didn't guess on that question and got lucky, Or knew that question but simply slipped up? In short, tests are too simple a metric, and it's far too difficult to discover a mastery score through simple arithmetic or logic. In an attempt to simplify this problem, the standard of knowledge tracing, Bayesian Knowledge Tracing (BKT), was created. Using Bayes theorem to predict the students' mastery score, taking into account slip and guess probability. BKT is the standard to solving knowledge tracing problems because of its efficacy and lightweight nature. BKT in Python applications like PyBKT[1] are both accessible and easy, though not necessarily accurate. BKTs, inability to adapt to serve different students and simplicity in its parameters are serious concerns for accuracy, especially concerning Edtech. The idea of using Deep Learning to solve this problem because of its ability to adapt is present and common in the educational data mining community. This paper aims to combine the best of BKT and Deep Learning, training a model on synthesized BKT data then adjusting and fine-tuning it based on a real dataset using K-Split cross validation to eventually evaluate and predict a student's mastery through next question prediction.

2 Methods

2.1 Generating Data

Data is generated using randomly sampled Bayesian parameters through realistic ranges seen in 5.1. This data is randomized for each synthetic student, which the student then answers 25 questions either correctly or incorrectly. The true underlying knowledge level of each student is recorded, as well as data left out for validation & final test. Usable data generated never exceeds the splits seen in 5.5.

2.2 Models

Three models are trained on the same data: LSTM, MLP, and a self-attention model.

2.3 MLP

MLP uses a standard feedforward architecture with 3 hidden layers with decreasing size(128,64,32). Each hidden layer includes batch normalization and ReLU. Dropout rates are kept at 0.3 to avoid overfitting.

$$\mathbf{h}_1 = \text{Dropout}(\text{ReLU}(\text{BN}(W_1\mathbf{x} + b_1)))$$

Hidden layer transformation with batch normalization, ReLU activation, and dropout regularization.

$$\mathbf{h}_i = \text{Dropout}(\text{ReLU}(\text{BN}(W_i\mathbf{h}_{i-1}+b_i))) \quad \text{for } i \in \{2,3\}$$

Hidden layers 2,3 with decreasing dimensions

$$\hat{y} = \sigma(W_4 \mathbf{h}_3 + b_4)$$

Where $\mathbf{x} \in \mathbb{R}^7$ is the input feature vector, $\mathbf{h}_i$ is the i-th hidden layer, and $\hat{y} \in (0, 1)$ is the predicted probability of a correct response, often defined in BKT as P(L)

2.4 LSTM

The LSTM architecture is a sequential history of the students' responses using 2 layers of 128 each.

$$\mathbf{h}_t, \mathbf{c}_t = \mathrm{LSTM}(\mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{c}_{t-1}) \tag{1}$$

H_t and C_t are hidden cell states at time t. The input sequence uses a sliding window length of 5(meaning the model considers the 5 most recent responses), with the aim to capture the most recent responses. H_T, which is the final hidden state, is then passed through a two-layer feedforward network.

$$\hat{y} = \sigma(W_2 \cdot \mathrm{ReLU}(W_1 \mathbf{h}_T + b_1) + b_2) \tag{2}$$

2.5 Self-Attention

The self-attention head uses a multi-head using 4 total heads with a hidden dimension of 64.
Input features are first projected to the hidden dimension

$$\mathbf{H} = \mathrm{Linear}(\mathbf{X}) \quad \text{where } \mathbf{X} \in \mathbb{R}^{T \times 7}, \mathbf{H} \in \mathbb{R}^{T \times 64} \tag{3}$$

Self-attention is then defined as

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{4}$$

Where Learned query is $Q = \mathbf{H}W_q$, Key is $K = \mathbf{H}W_K$, and value projections $V = \mathbf{H}W_V$, and $d_k = 64/4 = 16$ hidden dimension/number of heads. The attention outputs are then averaged across the sequence and then passed through a feedforward network for a binary prediction.

2.6 Training Configuration

All models are trained using the Adam optimizer with the following parameters

- Learning rate: 1×10^-3 with no decay
- Weight decay: 1×10^-5 (L2 regularization)
- Batch size: 256
- Maximum epochs: 100
- Loss function: Binary cross-entropy loss.

Binary cross-entropy is then defined in the equation

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right] \tag{5}$$

Where $Y_i \in (0, 1)$ is the ground truth label and $\widehat{Y}_i = \sigma(z_i)$ is the predicted probability.

2.7 Regularization and early-stopping

Overfitting was a large concern in designing this, especially given the synthetic dataset. 3 regularization strategies are employed.

- Dropout: Applied after each hidden layer with a rate=0.3
- L2 weight decay: $\lambda = 110^-5$ added to loss
- Early stopping: Training terminated when validation loss fails to improve for 10 consecutive epochs.

Model checkpoints are also saved as a PyTorch file at each epoch where validation AUC achieved a new maximum. The final epoch is not used as the final model of the training set; the model with the highest validation AUC loss is.

2.8 7 Features

- Current response
- Cumulative correct count
- Historical accuracy
- Previous response
- Streak length
- Recent accuracy
- Question position

2.9 Evaluating the models

5 Evaluation metrics are used to judge the models based on their performance in the testing dataset.

Models were evaluated using five metrics computed on the held-out test set:

$$\mathrm{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy Score

$$\mathrm{Precision} = \frac{TP}{TP + FP}$$

Precision Score

$$\mathrm{Recall} = \frac{TP}{TP + FN}$$

Recall Score

$$\text{F1-Score} = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

F1-Score

$$\text{ROC-AUC} = \int_0^1 \text{TPR}(t)\, d[\text{FPR}(t)]$$

ROC-AUC Score

Where TP, TN, FP, and FN denote True positives, True negatives, false positives, and false negatives in that order. Predictions are converted to binary correct/incorrect using a probability threshold of 0.5. In summary, Models are evaluated based on their next-question prediction accuracy, then evaluated on multiple statistics: Precision, Recall, F1 and ROC-AUC.

3 Bayesian Knowledge Tracing

3.1 Calculating Mastery

Bayesian Knowledge tracing is described as a Hidden Markov Model, where the machine learns based on Y and outputs $\hat{Y}$ because X cannot be observed. In context, X is the students' learning of the domain; because the model cannot directly observe, we base the P(L) on probability through BKT. BKT begins at time=0 with (L_0) = prior for subsequent processes (L_T) = prior is the updated probability from the previous operation. When an observation is made, defined as observation O_T,the algorithm updates the master probability to a binary number, either 0 if it was incorrect or 1 if it was correct. If $O_T = 1$ or the observation by the model was that the student got the question correct, that observation is applied to the probability using this formula.

$$(P(L_t|O_t = 1) = \frac{P(L_t)\cdot(1-P(S))}{P(L_t)\cdot(1-P(S))+(1-P(L_t))\cdot P(G)}$$

Put simply, this is the probability that the student knows the skill that they got correct. Because learning is not linear, and students can guess in a domain they have never studied, and still get some correct through guessing. BKT takes this into account with $P(G)$ a constant in the model. Slips are also accounted for through $P(S)$, another constant which takes into account that the student knows the skill, but simply slipped and got it incorrect.[1]

$$P(L_t|O_t = 0) = \frac{P(L_t)\cdot P(S)}{P(L_t)\cdot P(S)+(1-P(L_t))\cdot(1-P(G)})$$

If the observation is that the student got the question wrong or $O_T = 0$, we apply this formula, which is nearly identical to the correct formula, with the exception $1-$ is applied to $P(G)$ in the denominator instead of $P(S)$ in the numerator. In the $O_T = 0$ formula, 1 is also subtracted from $P(S)$ in the denominator.[1]

Following the Calculation of the posterior probability, the student's knowledge is updated using

$$P(L_{n+1}) = P(L_n \mid O_n) + (1 - P(L_n \mid O_n))T$$

3.2 Parameters of BKT

$$P(L_T)$$

This is the latent probability that the student knows the skill given T for time. This is updated frequently based on the model's evaluations. This is a monotonic variable, meaning according to BKT, the students' knowledge can only stay the same or increase.

$$P(T)$$

This constant defines the learning rate, the probability that the student learns the domain in-between questions. This is the largest weakness of BKT, as it assumes 3 things: 1. Learning is purely binary 2. Learning is also Monotonic and 3. Learning rate is consistent across all time, students, and domains.

$$P(G)$$

This constant defines the guess probability. Which defines the probability that the student does not know the domain but answered the question correctly. This is commonly set at 25%. A sometimes unrealistic factor that can affect the model's confidence.

$$P(S)$$

This constant defines the probability that the student knows the domain but simply slipped up and selected the wrong answer. This is normally around 15%. This value not only limits the confidence of the model at 85% but in theory it prevents high-confidence mastery estimates. [1]

4 The Need For Deep Learning

4.1 Educational Accuracy

The primary concern for switching to deep learning knowledge tracing models is the improvement in many statistics, such as the AUC metric (area under curve), where a noticeable improvement is seen[11]. Learning is not a linear path and differs for everyone; Bayesian captures a very limited aspect of tracking it, and the inability for the machine to recognize new patterns in a dataset but simply remain monotonic is one of the biggest hindering factors of these models. The parameters, as discussed in 2.2, are the biggest show of this.

5 Synthetic Data with BKT

5.1 Parameters

In order to generate proper synthetic data, parameters are the most significant in this experiment. Using what was defined as default ranges for $P(L_0)$, $P(T)$, $P(S)$, and $P(G)$ per student sampling was used, assigning random parameters from the realistic ranges listed below.

- P(L_0): 10%-60%
- P(T): 5%-25%
- P(S): 5%-20%
- P(G): 15%-35%

Per student sequence, 25 questions are generated with their individually assigned parameters. The student then based on the simulation BKT model- Which is simply a reverse BKT, where given initial parameters, the model generates observed binary responses. This paper presents results using various dataset sizes, with the largest comprising 10,000 students. 10,000 · 24 leading to about 240,000 training samples. Including later discussed validation and test data, this brings the total to 336,000 samples or 14,000 students. It's important to note that since the task the models are being evaluated on is next-response prediction (predicting question $n+1$ based on responses through question n. Each 25-question set yields only 24 prediction points, as there is no question 26 to predict.

5.2 Validation Data

Validation data, which consists of 48,000 samples or 2,000 students, is held out during training to detect any overfitting and for early stopping decisions in training.

5.3 Ground Truth

The ground truth is stored in all 3 points of data (Training, Validation, Test). Unavailable in regular educational training data, the ground truth allows verification that the model is training off learning rather than patterns in the students' responses.

5.4 Test Data

A small portion of the data is exclusively generated for testing the model. This is a small set, only 15% of the training set size, but is plenty as is. This data is generated at the same stage as the rest, so there are no repetitions, and the data is truly unique. The data goes through an identical process as training and validation data.

5.5 Max Data Splits

- **Training**: 10,000 students (240,000 samples, 71%)
- **Validation**: 2,000 students (48,000 samples, 14.79%)
- **Test**: 2,000 students (48,000 samples, 14.79%)
- **Total**: 14,000 students (336,000 samples, 100%)

6 Models Used

6.1 Multilayer perceptron

Compared to LSTMs, the Multilayer perceptron is much simpler; it differs because it doesn't have a memory of specific time sequences as an LSTM does. It takes its inputs multiplies them by weights, adds bias, and passes them through a function

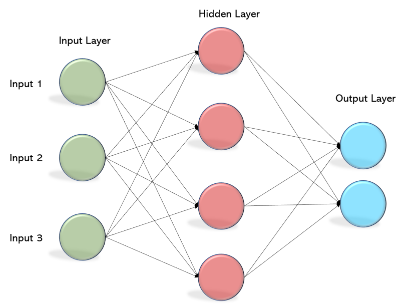

Figure 1: Inputs are passed through a hidden layer to calculate probability[7]

The above figure [7] is what's commonly seen as the conventional neural network. This architecture works exceptionally well if what you're looking for is training speed and an easy-to-understand model. MLPs are frequently called a feedforward network because the data like a river travels only in one direction; the simple model has no ability to recall previous interactive or measure time unlike the other models, MLPs simply processes all the data at once and applies weights to get an output.

6.2 Long Short Term Memory

At first glance, recurrent neural networks (RNNs) appear to be extremely appealing for multiple problems and use cases, primarily because they have the ability to connect past information to the present. This isn't always true, though. In short, as the gap between the past and present context for the model, it becomes difficult for the model to correctly identify the answer.

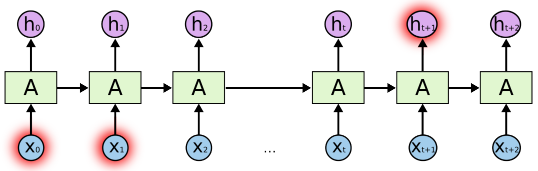

Figure 2: A standard RNN with a large gap between context [8]

Long Short Term Memory or LSTMs, as called in this paper, have a unique solution to track time. The architecture consists of 3 gates, a forget gate, which, simply put, disposes of information that is not relevant to the output. An input gate decides what new information to store that may be relevant to the output. The final gate is responsible for the Output, based on the model's current memory. This model works excellently for capturing the temporal dependencies of the problem. In context, the 3 gates work together to discard unnecessary data, input the most recent questions and the students' results, and finally decide what to output given its current memory.

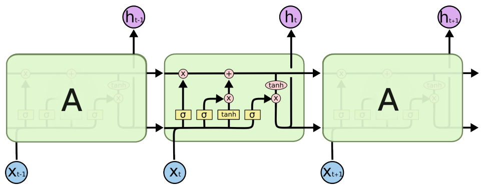

Figure 3: A chain of LSTMs working [8]

6.3 Self-Attention

Self-attention models are extremely unique because they weigh the importance of past responses when making predictions. In context, this means that the model focuses more on recent questions and simply ignores less relevant questions. The model learns which input matters the most for the desired output. [13]

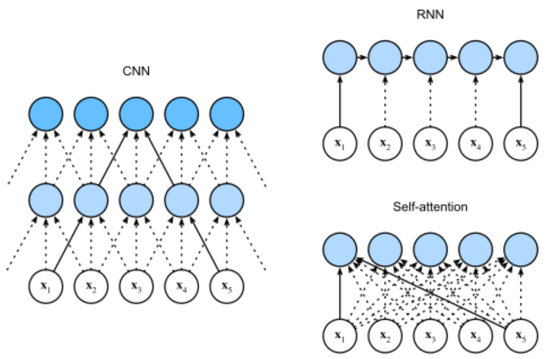

Figure 4: How Self-Attention models differ from CNNs or RNNs[15]

The Self-Attention architecture allows each token (shown here as X_1 through X_5) to look at all other tokens and evaluate the most relevant. This solves the earlier issue discussed in 6.2 of past and present. This approach makes it simple to model long-range relationships.

6.4 Control BKT

The control BKT uses non-adaptive constants as follows.

- $P(L_0) = 0.3$
- $P(T) = 0.1$
- $P(G) = 0.25$
- $P(S) = 0.1$

The control model uses fixed parameters that, unlike the other models, can not change capture skill specific differences or adapt its parameters during training or evaluation. This is a limitation spoken more on in section 9 that may produce relatively lower results. This is intentional and is used as a conservative comparison to the other models.

7 Results

7.1 Architecture Comparison

After training, we compared the architecture of LSTM , MLP & Attention as-well as normal BKT as a control. Using the metrics in 2.9.

Table 1: Model Performance Comparison with BKT Baseline

Model	**Acc.**	**Prec.**	**Recall**	**F1**	**AUC**	**Prec. Gain**
BKT	0.749	0.843	0.825	0.834	0.661	–
MLP	0.807	0.837	0.930	0.881	0.748	+7.8%
LSTM	0.825	0.850	0.951	0.898	0.703	+10.3%
ATTENTION	0.825	0.848	0.954	0.898	0.702	+10.2%

All 3 deep learning models that were trained on our synthesized data beat BKT, LSTM and attention stand out with the highest accuracy of 82.5%. Both models outperformed BKT with a significant 10% gain.
Though the models were trained on BKT data - they were still able to outperform BKT. This is most likely because the deep learning models were able to generalize parameters to approximate the student's learning progress. The BKT model is forced to use these linear assumptions to track learning, whilst the deep learning models can generalize and vary. The LSTM and Attention model had a significant advantage in accuracy over MLP, this is attributed due to the architecture differences. MLP has a lack of recurrence, meaning it has no way to carry information over from $t-1$ to t. The MLP maps $X \to Y$ directly, whilst LSTMs models and Attention models solve this by using a hidden state for the LSTM and weighted dependencies in the attention models. In simple terms, the MLP takes the data all at once whilst the LSTM and Attention models take the data sequentially.

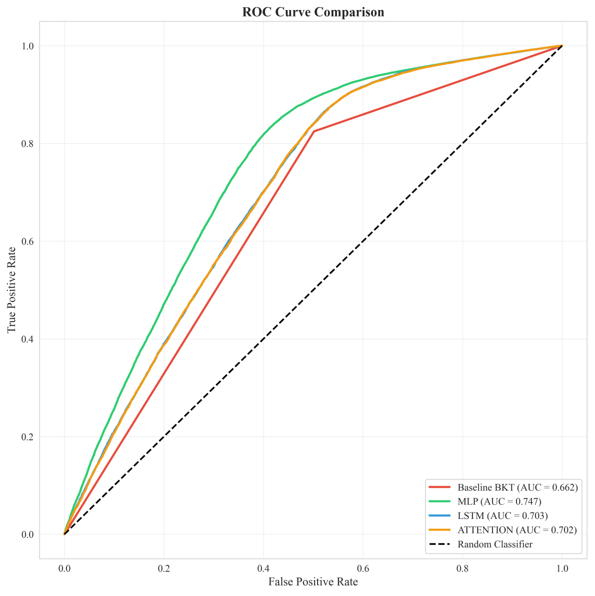

Figure 5: Architecture Comparison with ROC-AUC

However, it only gets more interesting from here, the AUC score was highest in the MLP, which has the lowest accuracy of the three. Both the LSTM and attention models have a higher accuracy in next question prediction however because of their sequential architecture the confidence shoots up, the model might notice several questions correct in a row and have a high confidence the student will continue this trend. When this ultimately fails with a high confidence level, the model's AUC score drops. The MLP doesn't see this trend as it takes the data all at once instead of sequentially as previously mentioned. In this case, the AUC score represents a difference in architecture when presented random slips. The MLPs higher AUC doesn't make it a more reliable model, it shows that the models blindness prevents the model from being overconfident.

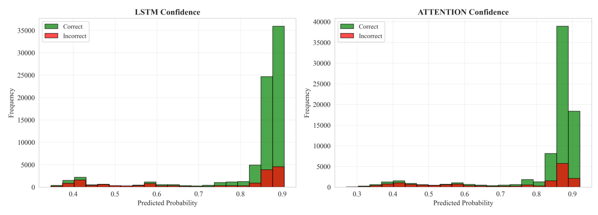

Figure 6: Confidence of LSTM(left) and Attention(right)

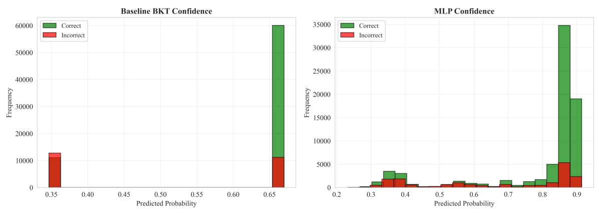

Figure 7: Confidence of BKT (left) and MLP(right)

The architectural difference is clear in these confidence graphs, the LSTM and Attention models cluster at the upper ranges around 0.9-1.0. The dense presence of incorrect answers contribute to the claim that these models extrapolate trends noticed in students success / non-success. These trends are reinforced and are more significant to the sequential models. The MLPs confidence is noticeably more distributed across the confidence, this indicates a more cautious approach as the MLP relies solely on the statistical average. Though being cautious doesn't mean being right.

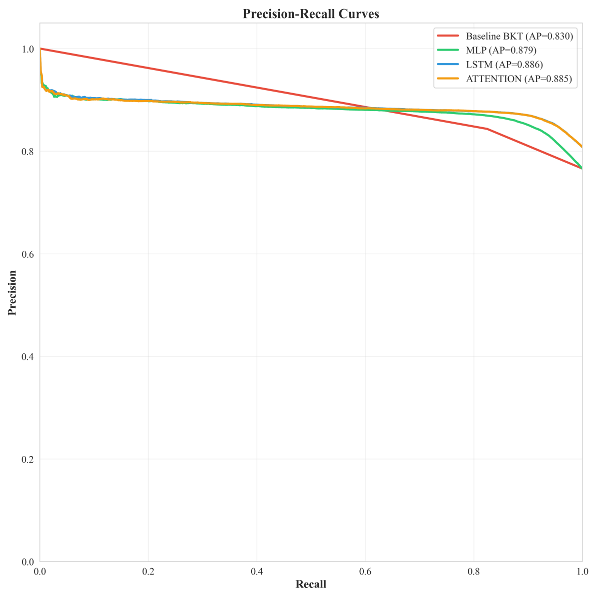

Figure 8: A precision recall curve of the 4 models

Figure 8 reveals BKTs conservative model, only being accurate with low recall, though it's more accurate it leaves out too many false negative. Simply put BKT given a pool of data is very accurate to the data it does identify, but it doesn't identify very much data. The 3 deep learning models are more accurate given higher recall. Beating out BKT in this graph as well. The MLP had the worst accuracy given a high recall, whilst LSTM and attention were nearly identical with a 0.001 difference.

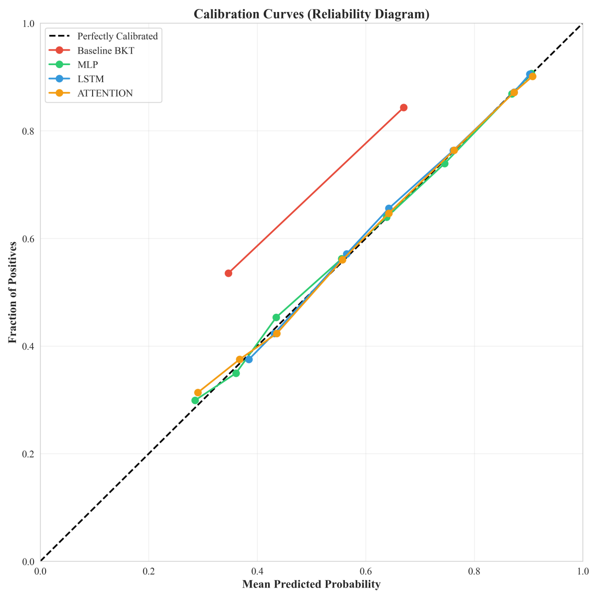

Figure 9: Calibration Curve with all 4 models

Another great figure to show the difference between the 3 deep learning models and BKT. BKT tends to underestimate even though students succeed, BKT's fixed parameters discussed in 3.2 limit an accurate representation. MLP , LSTM & Attention are extremely reliable on the test data. Even though this is BKT data the models they were trained because it was a dynamic range not just a constant set the models are able to stay flexible and adapt for an accurate probability.

7.2 Real Data with ASSISTments

All 3 models were then evaluated on the ASSISTments dataset from 2009-2010, a popular educational data mining dataset for evaluating KT models. The dataset contains 4,108 students and 521,317 samples. Using the same standard of a 5-fold cross validation

Model	Acc.	Prec.	Recall	F1	AUC	Prec. Gain
BKT	0.678	0.681	0.993	0.808	0.562	–
MLP	0.762	0.832	0.815	0.823	0.819	+12.4%
LSTM	0.762	0.771	0.928	0.842	0.789	+12.5%
ATTENTION	0.717	0.823	0.745	0.782	0.786	+5.7%

Table 2: Model Performance Comparison (Cross-Validation).

The main target of the real data benchmark was to see a significant increase in both AUC and F1 scores, which was accomplished. The MLP model achieved a 47.1% increase in AUC relative to the control BKT model. Across the board, significant increases were seen in all but one metric, recall. The BKT model held the highest recall of 0.993 which aligns perfectly with the idea that the model is often correct when predicting a correct answer. However, BKT exhibits the lowest precision at 0.681 meaning the model has a poor ability to discriminate between the correct and incorrect answers.

Overall result from the data table show that the simple feedforward architecture shown in the MLP yield the highest metrics across the board for the 3 neural networks. An interesting development as in 7.1 the MLP yielded the worst scores. This suggests that MLP KT models are able to better generalize on real noisy data. Whilst the LSTM and attention models can recognize the patterns in the synthetic data, MLPs simple architecture is marginally better.

8 Future Work

A future goal of this work is to add multi-skill data generation. Creating 15-45 independent skills for each student, with 200 samples per skill, per student. Generating BKT probabilities for each skill with the hopes of refining the model to work with noisier patterns and with the need for less fine-tuning.

A gradio web-app was also part of this work, a path towards making knowledge tracing and mastery learning accessible to teachers and tutors. In practice this works exceptionally well by simply running a python file a full web app can be accessed locally and from any other computer even on other networks.

9 Limitations

This work in its entirety was completed on limited hardware, specifically an Apple MacBook Air with M2 Apple Silicon chip boasting around 3.6 TFLOPS. Data generation, training & use/evaluation is all possible on everyday laptops. Because of this fact, the numbers posted in this paper may not represent the absolute best metrics of this architecture.

As mentioned in future work, this work only focuses on single-skill data generation. For subjects which focus on multiple skills and might provide more noise for a model to identify patterns, like the ASSISTments dataset this may be a stronger use of synthetic data. This is an easily added component to add to synthetic data generation, which will be included in future work.

These models also only evaluate next-question prediction, This paper does not look at long-term mastery, which is an important factor for educators reviewing the aspects of mastery learning.

Mentioned in 6.4, The control BKT is not a state-of-the-art model and is a simple, conservative representation of the vanilla theory of knowledge tracing. In practice, educational technology companies that employ intelligent tutoring systems and use BKT, use parameter estimations to change the four parameters mentioned in 3.2 based on student data.

The last noted discrimination was the baseline, this paper only compared standard BKT to the MLP , LSTM , Attention models which had been exposed to synthetic Bayesian data.There was no control neural network that was just fine-tuned on the evaluation data and not exposed to synthetic data.

10 Conclusion

This paper proved that using synthetic Bayesian data to pre-train a deep learning model is more effective than using standard Bayesian Knowledge Tracing regardless of architecture. This data is able to refine the principles and patterns of non-linear learning. Even models trained on single skill synthetic data can still accomplish great results on real data relative to the standard BKT. The most important metric seen though not the greatest increase was AUC score where a 26.5% increase was seen simply by pre-training and using unique architecture. This approach is both extremely accessible and lightweight with the capability to complete an entire workflow on limited hardware.

References

[1] Anusha Badrinath, Fan Wang, and Zachary A Pardos. Pybkt: An accessible python library of bayesian knowledge tracing models. Online. Accessed November 19, 2025.

[2] Benjamin S Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13(6):4–16, 1984.

[3] Albert T Corbett and John R Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, 1995.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[5] Rita Liao. Ai tutors are quietly changing how kids in the us study, and the leading apps are from china. *TechCrunch*, May 2024. Accessed: 2025-12-16.

[6] Xiang Lu, Xia Cheng, Yang Liu, Neil Heffernan, and Cristobal Heffernan. Towards interpretable deep learning models for knowledge tracing. In *Proceedings of the 29th International Conference on Artificial Intelligence in Education (AIED)*, pages 1–12, 2020. Accessed: 2025-12-16.

[7] Awhan Mohanty. Multi layer perceptron (mlp) models on real world banking data, 2019. Accessed: 2025-12-16.

[8] Christopher Olah. Understanding lstm networks, 2015. Accessed: 2025-12-16.

[9] Shalini Pandey and George Karypis. A self-attentive model for knowledge tracing. In *Proceedings of the 12th International Conference on Educational Data Mining (EDM)*, pages 384–389, 2019.

[10] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks, 2012. Accessed: 2025-12-16.

[11] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NeurIPS)*, pages 505–513, 2015.

[12] Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. Accessed: 2025-12-16.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, pages 6000–6010, 2017.

[14] Wikimedia Commons contributors. Multilayer perceptron. https://commons.wikimedia.org/wiki/File:MultiLayerPerceptron.svg, April 2012. Accessed: 2025-12-16.

[15] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Neural network diagram. Wikimedia Commons, 2017. Licensed under CC BY-SA 4.0.

A ASSISTments Full Table

Model	**Acc.**	**Prec.**	**Recall**	**F1**	**AUC**	**Gain**
BKT	0.678	0.681	0.993	0.808	0.562	–
MLP	0.762 ± 0.023	0.832 ± 0.021	0.815 ± 0.027	0.823 ± 0.021	0.819 ± 0.024	+12.4%
LSTM	0.762 ± 0.019	0.771 ± 0.021	0.928 ± 0.016	0.842 ± 0.013	0.789 ± 0.025	+12.5%
ATTENTION	0.717 ± 0.024	0.823 ± 0.014	0.745 ± 0.038	0.782 ± 0.025	0.786 ± 0.024	+5.7%

Table 3: 5 Fold CV for all models on the ASSISTments dataset, with standard deviations.