Evaluation of the Computational Efficiency of Recurrent Architectures for Student Performance Prediction

Darazha N. Issabayeva [1,*], Shirinkyz Shekerbekova [2], Ainur Bazhibayeva [1], Akzhol Bakytuly [1], and Sarsengali M. Aldashev [3]

[1] Department of Artificial Intelligence and Big Data, Faculty of Information Technology and Artificial Intelligence, al-Farabi Kazakh National University, Almaty, Kazakhstan
[2] Department of Informatics and Informatization of Education, Faculty of Information Technology and Artificial Intelligence, Abai Kazakh National Pedagogical University, Almaty, Kazakhstan
[3] Faculty of Civil Law and Civil Procedure, Labor Law, Faculty of Law, Al-Farabi Kazakh National University, Almaty, Kazakhstan
Email: daraja_78@mail.ru (D.N.I.); sh.shekerbekova@abaiuniversity.edu.kz (S.S.); bakytbekkyzyainur@gmail.com (Ai.B.); akzholl.bakytuly@gmail.com (Ak.B.); aldashev.sarsengaly@gmail.com (S.M.A.)
*Corresponding author

Manuscript received November 19, 2025; revised January 12, 2026; accepted February 26, 2026; published August 21, 2026

***Abstract*—With the rapid growth of digital educational platforms, early prediction of student academic performance has become increasingly important for timely risk identification and effective instructional support. This study investigates the computational efficiency and predictive performance of various deep learning recurrent architectures for student performance classification based on time-series data of learning activities. Experiments are conducted using the Open University Learning Analytics Dataset (OULAD), where student interactions with the learning management system are transformed into multidimensional time series combining dynamic behavioral features and static demographic attributes. Multiple recurrent, hybrid, and attention-based models are evaluated in an early prediction setting across different stages of course progression using standard classification metrics and statistical significance testing. In addition to predictive performance, the computational cost of each architecture is systematically assessed in terms of training time, model complexity, memory usage, and storage requirements. The results indicate that while predictive performance across architectures is largely comparable, their computational demands differ substantially, highlighting the importance of cost–performance trade-offs. Based on these findings, the study recommends favoring simpler recurrent models that achieve competitive accuracy with significantly lower computational overhead for deployment in real-world educational systems.**

***Keywords*—recurrent networks, deep learning in education, computational efficiency, neural sequence models, learning analytics**

I. Introduction

The modern education system is undergoing rapid transformation thanks to the widespread introduction of digital educational platforms and online courses. This process has intensified during the COVID-19 pandemic and the years that followed, when distance and hybrid learning formats became the main tool for interaction between teachers and students [1, 2]. The widespread use of virtual educational environments has led to the emergence of large amounts of data on student behavior: activity in Learning Management System (LMS), sequence of actions, engagement, task completion dynamics, and assessment results. This data opens up opportunities for building early prediction and student support systems capable of analyzing individual trajectories and identifying students at risk of low academic performance.

Despite a significant amount of research and the availability of open educational datasets, the task of early prediction remains challenging. The main difficulties are related to the high individual variability of student behavior, the irregularity of time sequences, and the pronounced dependence of activity on time. Classical machine learning methods and statistical approaches are poorly adapted to such dynamic data structures, as they are unable to correctly account for long-term and local time dependencies.

In this regard, recurrent neural networks and their modern variations, including bidirectional models, combining Convolutional Neural Network-Recurrent Neural Network (CNN-RNN) hybrids, and attention-oriented architectures, are of particular interest. These methods can extract complex temporal patterns and provide stable predictions on sequential data. However, the question remains: do complex architectures truly outperform simpler RNN models in this domain, or are the performance gains reported in the literature disproportionate to the actual complexity of the specific task within an LMS context?

The goal of this work is to conduct a comprehensive and objective analysis of various deep learning architectures for predicting academic performance on the Open University Learning Analytics Dataset (OULAD). The study aims to:

(1) Evaluate the ability of the student performance prediction of models,
(2) Their ability to make early predictions with a limited temporal depth of input,
(3) The practical applicability of models, considering their computational cost per performance.

This approach allows us not only to identify the most effective models for educational analytics tasks, but also to determine whether the differences between architectures are significant from a practical and statistical point of view, which is especially important when developing early warning systems in real educational environments, which is rarely taken into account when comparing and developing architectures.

The contributions of this study are fourfold. First, we frame student performance prediction as a cost–performance trade-off problem by jointly analyzing predictive accuracy and computational efficiency across recurrent neural architectures. Second, we empirically demonstrate diminishing returns of architectural complexity, showing that more complex recurrent and attention-based models do not yield statistically meaningful performance improvements

over simpler baselines within the studied LMS context. Third, we provide a systematic evaluation of early prediction under truncated time-series inputs, illustrating how predictive performance evolves across different stages of course progression. Finally, we offer practical insights for real-world deployment by identifying lightweight models that achieve competitive accuracy with substantially lower computational requirements.

II. Literature Review

The application of deep learning models demonstrates a wide variety of methods for solving this problem. Hybrid recurrent architectures show high results when working with data from LMS, where the temporal structures of student activity in the learning environment play an important role. For example, the Hierarchical Recurrent Neural Network-Bidirectional Long Short-Term Memory (HRNN-BiLSTM) model, which uses behavioral factors from social networks, achieved an accuracy of 99.87% [3], significantly surpassing the results of classical machine learning models. Similar results were obtained when working with the OULAD dataset, where a hybrid of convolutional and bidirectional Gated Recurrent Unit (GRU) achieved 97.48% [4].

However, architecture is not the only determining factor. The choice of optimizer can radically change the performance: switching from Adagrad to Adam increased the accuracy of GRU from 88% to 97.66% [5]. Equally important is the issue of early prediction. The Artificial Neural Network-Long Short-Term Memory (ANN-LSTM) model showed an increase in accuracy from 43% on the first day of observation to 72% by the 270th day [6], indicating the critical importance of the length of the historical window.

CNN proves to be effective even beyond classic computer vision tasks. When predicting student grades in regression mode, CNN achieved $R^2 \approx 0.84$, outperforming BiLSTM and LSTM with attention mechanism [7]. Similarly, converting OULAD tabular data into a two-dimensional representation and then applying 2D-CNN yielded 88% accuracy [8]—higher than standard ML algorithms, despite the atypical method of working with data. We will use hybrids of CNN-LSTM to see if such a combination of architecture can improve performance, and if it does, is it really applicable by analyzing its cost to compute.

Generally, deep architectures demonstrate an advantage when working with text data and event sequences, while simple fully connected networks are competitive where the temporal structure is weak: Deep Neural Network (DNN) achieved 94.2% versus 87.4% for Random Forest [9, 10].

Thus, the choice of architecture has to take into account the nature of the data: recurrent networks are effective with pronounced temporal dependencies, convolutional networks with structured features, including non-standard representations of tabular data, and hybrids of different layers are capable of capturing connections that basic recurrent models do not see.

III. Materials and Methods

A. Dataset Description

For this study, we used the Open University Learning Analytics Dataset (OULAD), which has info on 32,593 students who took 22 Open University courses between 2013 and 2014 [11]. A range of online activities, including clicks on various resources within the Virtual Learning Environment (VLE), assignment submissions, test completions, and grade receipts, were used to characterize each student. The data covers the entire module period, allowing for a detailed study of student behavior dynamics throughout the course. The dataset is available at: https://analyse.kmi.open.ac.uk/open-dataset

The data structure included the following tables: studentVLE with records of student activity in the virtual learning environment, studentAssessment with individual assignment results, assessments with assessment metadata, studentInfo with demographic information about students, and VLE with types of activities in the learning environment.

B. Data Preprocessing

The data preprocessing process involved several sequential steps to transform the raw tables into structured time series suitable for training deep learning models.

All student activity records were aggregated by day, forming time series of variable length. Basic activity metrics were calculated for each student, including the total number of clicks in the VLE per day, as well as clicks by activity type, aggregated from the VLE table. Assessment metrics included the average score for assignments submitted on a given day, the total weight of assignments, the average late submission, and the number of assignments submitted. Additionally, derivative features were calculated based on sliding windows: the sum of clicks for the last 3 and 7 days, the exponentially weighted moving average of clicks with a span parameter of 5, and the change in activity intensity from day to day. As a result, 21 temporal features were formed for each day of student activity. All models were trained using the same input representation, preprocessing pipeline, feature normalization, and padding strategy. No architecture-specific preprocessing was applied.

Eight static features characterizing the demographic and educational characteristics of students were extracted from the studentInfo table. These features included gender, region of residence, level of education, age group, disability status, number of previous attempts to complete the course, number of credits, and multiple deprivation index. All categorical features were numerically encoded using LabelEncoder [12], with missing values filled with zeros.

The final results of the students were converted into a binary classification, where class 1 represented successful completion of the course, combining the categories “Pass” and “Distinction,” and class 0 represented unsuccessful completion, combining the categories ‘Fail’ and “Withdrawn.” This allowed the task to be formulated as a binary prediction of academic performance.

Analysis of the lengths of the time series showed significant variability, with a minimum length of 1 day, a maximum length of 286 days, an average length of 68.6 days, and a median of 54 days. To standardize the input data, a padding approach was used, in which sequences longer than the maximum length were truncated at the end, and sequences shorter than the maximum length were padded with zeros. The 95th percentile of the length distribution, which was 182

days, was used as the maximum sequence length. This approach ensured a balance between preserving information for 95% of students and computational efficiency.

All features were normalized to ensure training stability. Standard scaler, which helps to correctly compare features with each other and minimize the impact of sudden surges in the data [13], was applied to temporal features using the formula:

$$x_{norm} = \frac{x - \mu}{\sigma} \quad (1)$$

where the parameters μ and σ were calculated only on the training sample.

Standard scaler was applied independently to static features. Normalization was performed strictly on the training set, then the same parameters were applied to the test set to prevent data leakage.

C. Formation of Training and Test Samples

The final dataset consisted of 26099 students with complete data. The temporal data had a dimension of 26099 × 182 × 24, the static data had a dimension of 26099 × 8, and the target variable was an array of 26099 binary labels. The dataset was split into a training sample of 20879 students and a test sample of 5220 students in an 80/20 ratio using stratification by the target variable. The class balance was 51.64% of successful students in both samples, which ensured the representativeness of the class distribution when evaluating the models.

D. Assessment of Early Prediction Ability

To systematically evaluate the models' ability to predict student performance early on, four versions of the dataset were created with different lengths of time series. The first version included the last 45 days of activity, which corresponded to approximately 25% of the total course length. The second version contained the last 91 days, corresponding to 50% of the course. The third version included the last 136 days, representing 75% of the course. The fourth version contained the full sequence of 182 days. This approach made it possible to evaluate how accurately each architecture could predict a student's final result using only partial information about their activity and performance during the course.

After all stages of preprocessing, for each sequence length option, arrays of temporal features with dimensions 20879 × SQ × 24 were formed for the training sample and 5220 × SQ × 24 for the test sample, where SQ is sequence length, which took values 45, 91, 136, or 182 depending on the dataset variant. This processed dataset became the input data for the deep learning models described in the next section.

E. RNN Architectures

The study examined architectures designed for processing sequential data. Each student is represented as a time series of weekly activity, which allows tracking the dynamics of engagement throughout the course.

The basic models were the classic recurrent networks, LSTM, and GRU. Their popularity in time series tasks is due to their ability to capture long-term dependencies and changing behavior patterns. Additionally, their bidirectional versions, Bi-LSTM and Bi-GRU, were tested, which process the sequence in both directions to obtain a more complete temporal representation. Hybrid architectures combining convolutional/self-attention and recurrent layers were also tested. In such models, a one-dimensional convolution extracts local patterns, after which the recurrent layers integrate them into the temporal context. This combination makes it possible to capture both short-term dynamics and long-term trends.

These architectures were chosen based on the hypothesis that explicit modeling of sequential data structures may offer an advantage over standard methods in the task of predicting academic performance.

LSTM was originally introduced to solve the vanishing gradient problem present in earlier recurrent neural networks. The main idea behind LSTM is to explicitly separate the hidden state and the dedicated memory cell (Fig. 1). The cell carries information across time intervals, while three gates (input, forget, output) determine when to record, store, or pass information to the next layer [14]. This explicit long-term memory control mechanism makes LSTM particularly suitable for data where temporal relationships extend over long horizons, such as the dynamics of student engagement over many school weeks. In educational prediction tasks, LSTMs often outperform simpler feedforward architectures precisely because they do not rely on the assumption of temporal independence between weekly features. Therefore, LSTMs are widely accepted as a "classic" basic deep learning model for the problem of forecasting.

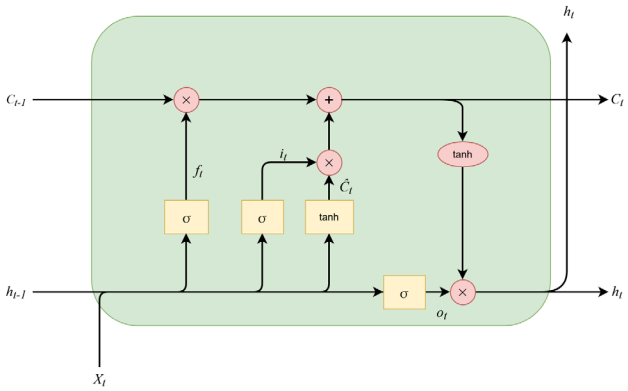

Fig. 1. LSTM cell architecture.

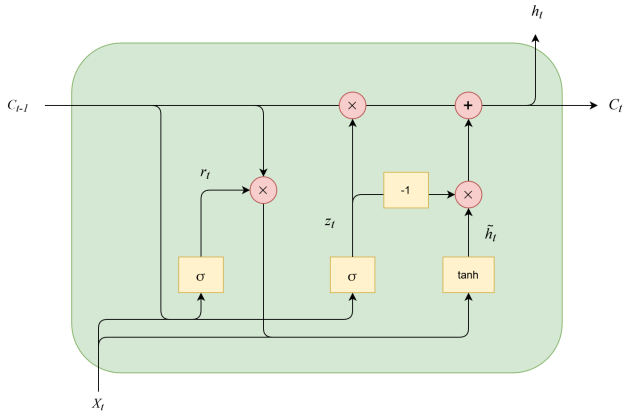

Fig. 2. GRU cell architecture.

Gated Recurrent Units (GRUs) are a more compact alternative to LSTMs, proposed in 2014. Instead of maintaining a separate internal memory cell, GRU combines

gating operations into only two key elements: an update gate and a reset gate (Fig. 2). This reduces the number of parameters compared to LSTM, which often results in shorter training times on the same data and better generalization on small datasets, but performs worse in other scenarios [15].

In addition to the basic models, bidirectional BiLSTM and BiGRU architectures were implemented. Thanks to the ability of these systems to analyze time series in both directions (Fig. 3), both previous and subsequent dependencies between time steps can be taken into account [16]. This is particularly important when demonstrating inconsistent student behavior, as tasks performed later in the week may be related to earlier stages of learning.

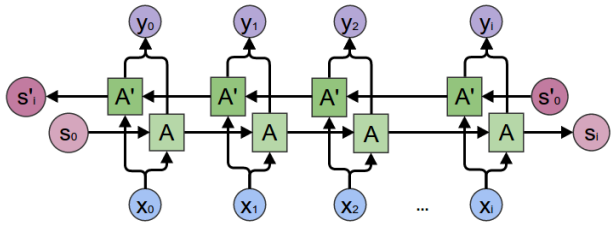

Fig. 3. Bidirectional recurrent network architecture.

Hybrid CNN-LSTM and CNN-GRU models combine convolutional and recurrent neural networks to simultaneously analyze short-term and long-term patterns in sequential data. Convolutional layers apply trainable filters to the original time series, recognizing local patterns in student behavior over a short time interval. A subsampling operation placed after the convolutional layer reduces the dimensionality of the features and provides robustness to small temporal variations in the input data. The resulting high-level representations are sent to a recurrent block, which establishes relationships between subsequent stages of the learning process. This organization allows for the processing of sudden changes in activity through the convolutional layer, as well as the ability of the recurrent part to track extended trends in student behavior [17, 18].

Architectures with attention mechanisms modify standard recurrent structures by adding adaptive weighting of different moments during the formation of the final choice. A multi-head attention mechanism determines the importance of coefficients for each step in the input sequence, allowing the model to selectively focus on the defining steps of an individual student's learning process. In contrast to classical recurrent networks that sequentially process input data with the risk of signal degradation in later steps, the attention mechanism establishes a direct connection between all time positions. In this experiment, Attention-RNNs combine bidirectional recurrent processing to form contextual embeddings with multi-head attention to selectively connect information from different temporal domains.

The common components of all models were the last fully connected layer with ReLU activation, one recurrent layer, and one convolutional layer for hybrid models, Dropout 0.3–0.4 layers, Layer normalization, which normalizes activations across the feature dimension [19] and, to stabilize gradients by normalizing activations across the batch dimension, Batch Normalization were implemented after the convolutional layer in hybrid models [20]. Two dense layers with ReLU activation functions, as shown in Eq. (2), are widely used in deep learning and demonstrate improved convergence compared to classical sigmoidal nonlinearities, as well as a sigmoid function for solving binary classification problems [21].

$$ReLU(x) = \max(0, x) \tag{2}$$

All models use identical training settings to ensure a fair architectural comparison. Training parameters for all models:

- Loss: Binary cross-entropy
- Optimizer: Adam (learning rate = 1e-3, clipnorm = 1.0)
- Metrics: Accuracy, Precision, Recall, ROC-AUC
- Callbacks: EarlyStopping (monitor = val_loss, patience = 10, restore_best_weights = True), ReduceLROnPlateau (monitor = val_loss, factor = 0.5, patience = 5, min_lr = 1e-6)
- Output: Dense(1, sigmoid)

Table 1 shows the architecture of all models that were used for this study.

Table 1. Architectures of models

Model	Temporal input branch	Static input branch	Fusion + classifier head
LSTM	LSTM(64, return_sequences=False) > LayerNorm > Dropout(0.3)	Dense(32, ReLU) > Dropout(0.3)	Concatenate > Dense(64, ReLU) > Dropout(0.4) > Dense(1, Sigmoid)
GRU	GRU(64, return_sequences=False) > LayerNorm > Dropout(0.3)	Dense(32, ReLU) > Dropout(0.3)	Concatenate > Dense(64, ReLU) > Dropout(0.4) > Dense(1, Sigmoid)
BiLSTM	Bidirectional(LSTM(64, return_sequences=False)) > LayerNorm > Dropout(0.3)	Dense(32, ReLU) > Dropout(0.3)	Concatenate > Dense(64, ReLU) > Dropout(0.4) > Dense(1, Sigmoid)
BiGRU	Bidirectional(GRU(64, return_sequences=False)) > LayerNorm > Dropout(0.3)	Dense(32, ReLU) > Dropout(0.3)	Concatenate > Dense(64, ReLU) > Dropout(0.4) > Dense(1, Sigmoid)
CNN-LSTM	Conv1D(64, k=5, same, ReLU) > BatchNorm > MaxPool1D(pool=2) > Dropout(0.3) > LSTM(64, return_sequences=False) > LayerNorm > Dropout(0.3)	Dense(32, ReLU) > Dropout(0.3)	Concatenate > Dense(64, ReLU) > Dropout(0.4) > Dense(1, Sigmoid)
CNN-GRU	Conv1D(64, k=5, same, ReLU) > BatchNorm > MaxPool1D(pool=2) > Dropout(0.3) > GRU(64, return_sequences=False) > LayerNorm > Dropout(0.3)	Dense(32, ReLU) > Dropout(0.3)	Concatenate > Dense(64, ReLU) > Dropout(0.4) > Dense(1, Sigmoid)
Attention-LSTM	Bidirectional(LSTM(64, return_sequences=True)) > Dropout(0.3) > MultiHeadAttention(heads=4, key_dim=32, dropout=0.2) (self-attn) > Residual Add > LayerNorm > GlobalAvgPool1D > Dropout(0.3)	Dense(32, ReLU) > Dropout(0.3)	Concatenate > Dense(128, ReLU) > Dropout(0.4) > Dense(64, ReLU) > Dropout(0.4) > Dense(1, Sigmoid)
Attention-GRU	Bidirectional(GRU(64, return_sequences=True)) > Dropout(0.3) > MultiHeadAttention(heads=4, key_dim=32, dropout=0.2) (self-attn) > Residual Add > LayerNorm > GlobalAvgPool1D > Dropout(0.3)	Dense(32, ReLU) > Dropout(0.3)	Concatenate > Dense(128, ReLU) > Dropout(0.4) > Dense(64, ReLU) > Dropout(0.4) > Dense(1, Sigmoid)

F. Learning Procedure and Assessment Metrics

The experiments were conducted on the Google Colab platform using the T4 TensorFlow Keras graphics processor. All models were trained with the same hyperparameters: batch size of 256, Adam optimizer, and binary cross-entropy loss function [5, 17]. The maximum number of epochs was set to 50, which seems optimal in the case of a small training dataset and a big batch size.

Each architecture was trained separately on data from each segment, which made it possible to track the dynamics of prediction accuracy as information about the student accumulated.

Notations: True Positive (TP) is a correctly predicted positive result, True Negative (TN) is a correctly predicted negative result, False Positive (FP) is a negative case predicted as positive, and False Negative (FN) is a positive case predicted as negative.

The evaluation was performed on a pre-separated test sample using the most common metrics in classification tasks: Accuracy, Precision, and Recall, which are shown in Eqs. (3)–(5), respectively [10, 22]. This makes it possible to compare architectures with each other and determine at what stage of the course acceptable prediction quality is achieved.

$$Acc. = \frac{TP+TN}{TP+TN+FP+FN} \in [0,1] \quad (3)$$

$$Rec. = \frac{TP}{TP+FN} \in [0,1] \quad (4)$$

$$Pre. = \frac{TP}{TP+FP} \in [0,1] \quad (5)$$

Based on the above metrics, the F1-score is calculated, which attempts to account for accuracy in situations where classes are unbalanced, focusing on the accuracy of positive predictions and actual positive records [23]. The formula of $F1$ is shown in Eq. (6) below.

$$F1 = \frac{2\times Pre.\times Rec.}{Pre.+ Rec.} \in [0,1] \quad (6)$$

Also, another important metric that is popular to use for classification problems is Area Under the Receiver Operating Characteristic Curve (ROC-AUC).

The ROC curve is constructed based on the true positive rate and false positive rate at different classification thresholds. TPR is plotted on the Y-axis, and FPR is plotted on the X-axis. The curve shows how the relationship between sensitivity and specificity changes as the threshold value changes [24].

AUC is the area under the ROC curve. The AUC range covers from 0.5, when the model essentially just guesses, to 1.0, when the model predicts perfectly [24, 25].

The statistical significance of differences between models was assessed using the McNemar's criterion, using paired predictions on the same data. For each pair, a 2 × 2 contingency table was constructed based on correct and incorrect answers [26]. The use of the McNemar's test ensured the reliability of the results. Paired comparisons were used for a qualitative assessment of the significance of differences, rather than to establish a strict global ranking of architectures.

G. Computational Cost Assessment

To evaluate the computational efficiency of each architecture, we selected four metrics that will be used: training time, number of parameters, saved model size, and memory consumption during training. These metrics characterize multiple aspects of resource consumption and let us compare models no matter of prediction performance. Time to train was evaluated as the duration of a complete training under identical conditions [27]. It allows for comparison of recurrent models with different architectures and numbers of layers. The number of parameters and model size shows the complexity of the structure and requirements for disk space for saving and loading the model [27, 28]. Memory consumption was recorded as the average value used during training, which is a critical parameter when working with limited computing power [29]. All these metrics will provide us with a more complete understanding of computational cost and assess how effective it is to implement such architectures in real educational systems.

IV. Result and Discussion

A. General Characteristics and Quality Assessment of Models

After all architectures have been tested for the early prediction of student performance. All accuracy results were in a diapason of 89.97–91.03%, which is shown in Table 1. All models have shown the same efficiency: the difference between the best result and worst is 0.0057 points, which suggests that the data has consistent and pronounced effects in the behavioural data of students, which are being extracted very similarly by all recurrent, hybrid, and attention-implemented architectures. In Table 2, it is shown that on the full length of time-series, bidirectional and attention-based architectures have shown the highest results in accuracy and F1-score, but still, all models show a minimum difference in metrics. Pairwise McNemar's tests were conducted across all eight recurrent architectures to assess differences in error patterns on the same test instances. With the exception of a single marginal comparison (GRU vs BiLSTM, $p = 0.044$), no statistically significant differences were observed. Importantly, this isolated result does not remain significant after correction for multiple comparisons, indicating that the evaluated architectures exhibit highly similar error distributions.

Table 2. Metrics score on the full length of the dataset

Model	Accuracy	F1-Score	Precision	Recall	AUC
LSTM	0.9000	0.9091	0.8569	0.9681	0.9514
GRU	0.8983	0.9072	0.8581	0.9622	0.9512
BiLSTM	0.9040	0.9115	0.8704	0.9566	0.9522
BiGRU	0.9015	0.9098	0.8632	0.9618	0.9525
CNN-LSTM	0.9021	0.9093	0.8715	0.9507	0.9549
CNN-GRU	0.9010	0.9087	0.8670	0.9547	0.9540
Attention-GRU	0.9038	0.9125	0.8609	0.9707	0.9592
Attention-LSTM	0.9034	0.9123	0.8591	0.9726	0.9601

To show the dynamic of how fast models train and become

stable, a graph of the AUC dynamics (Fig. 4) was plotted over the full dataset. The graph clearly shows that the models quickly pick up on student behavior patterns and quickly stop learning. All models spent less than half of the assigned epochs reaching the limiting metric values and starting to overfit. It may be due to the overly aggressive feature engineering or of the OULAD itself, which already consisted of clear, non-noisy data.

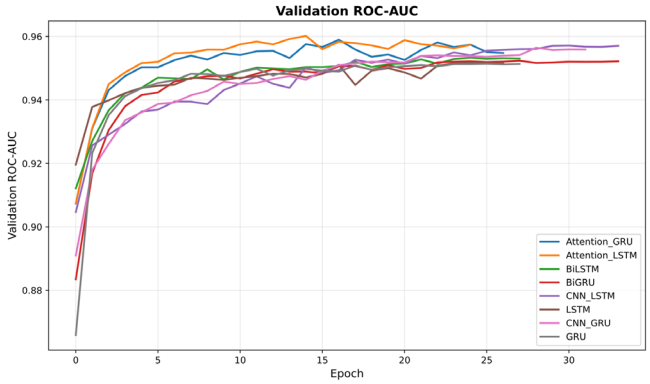

Fig. 4. Validation AUC dynamic on the full length of the dataset.

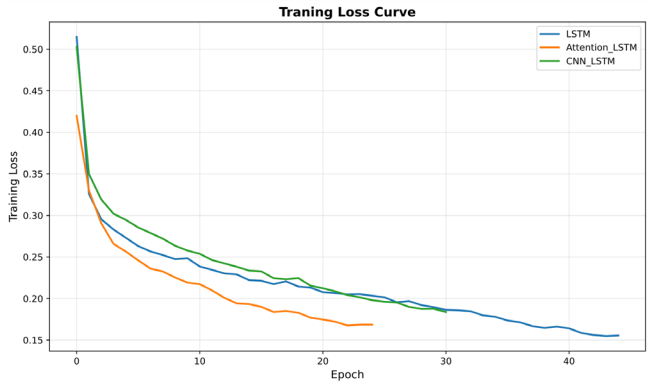

Fig. 5. Training loss dynamic on full-length dataset length.

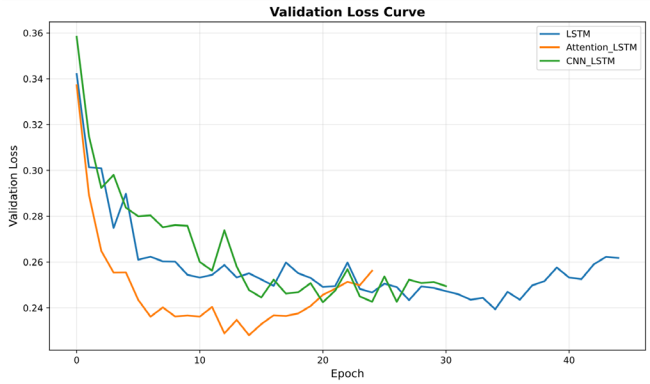

Fig. 6. Validation loss dynamic on full-length dataset length.

Using the same data split, preprocessing pipeline, and training protocol as in the main experiments, we also trained three representative architectures (LSTM, CNN-LSTM, and Attention-LSTM) in order to analyze convergence behavior. We also recorded the training history in order to plot training/validation curves (Figs. 5 and 6). There is no discernible generalization benefit of the more sophisticated architectures over the baseline LSTM, and the curves exhibit fast convergence during the first 10 to 15 epochs before plateauing. The attention-based model needed a increased architectural complexity (141k parameters and 186 s training time against 29k parameters and 85 s for LSTM), even if the validation performance was comparable (AUC = 0.956–0.962 and F1 = 0.913–0.919). These findings provide credence to the idea that, in this context, greater architectural complexity results in declining returns.

B. The Influence of Time Sequence Length on Prediction Quality

Results of experiments on all four variants of series length are shown in Table 3. Even when using only 25% percent of full length model, achieve 89.75% accuracy score, proving the feasibility of early prediction and intervention. As the sequence length increased, the quality improved very slightly: the increase of accuracy from 45 to 182 days for BiGRU is only 0.4%, showing diminishing impact of additional weeks of active accumulation. Interestingly, the maximum result for BiLSTM pf 91.03% was on 136 days of length of the time-series, while accuracy decreased slightly, suggesting the growth of noise in the later stages of the course.

Table 3. F1-score of models on all 4 time segments

Model	45 days	91 days	136 days	182 days
LSTM	0.9022	0.9112	0.9116	0.9091
GRU	0.9044	0.9092	0.9108	0.9072
BiLSTM	0.9062	0.9107	0.9167	0.9115
BiGRU	0.9071	0.9115	0.9092	0.9098
CNN-LSTM	0.9006	0.9005	0.9079	0.9093
CNN-GRU	0.9005	0.9076	0.9141	0.9087
Attention-LSTM	0.9024	0.9141	0.9146	0.9123
Attention-GRU	0.8993	0.9071	0.9167	0.9125

C. Computational Efficiency of Models

For a more comprehensive evaluation, a computational analysis was conducted (Table 4). The variation in resource consumption was much greater than in prediction quality:

- The number of parameters ranged from 23,969 GRU to 141,089 LSTM with attention mechanisms,
- training time ranged from 52.3 to 195 seconds,
- model size ranged from 0.33 MB to 1.71 MB.

Table 4. Computation metrics

Model	Time to train (sec)	Parameters count	Model size (MB)	Memory usage (MB)
LSTM	54.6 s	29,473	0.39	4839
GRU	52.3 s	23,969	0.33	4838
BiLSTM	81.2 s	56,481	0.71	5197
BiGRU	89.9 s	45,473	0.58	5196
CNN-LSTM	71.8 s	47,713	0.61	5252
CNN-GRU	67.2 s	39,649	0.52	5556
Attention-LSTM	185.1 s	141,089	1.59	5742
Attention-GRU	195.0 s	130,081	5742	1.59

This demonstrates that, while delivering comparable performance, different architectures differ significantly in their operational requirements. In resource-constrained environments, preference may be given to simpler architectures, which provide similar performance at significantly lower computational cost.

D. Early Prediction Perspective

Early prediction is implemented through truncated

sequences of interactions that reflect partial progress in the course. The practical value of such predictions lies in the timely support of students (consultations, tutoring) to prevent dropouts. At the same time, the cost of error in the early stages is asymmetric: a "false positive" result only leads to excessive support, while a "false negative" means a missed opportunity to help. Thus, early prediction models are not considered as definitive predictors, but as decision support tools, where priority is given to completeness (recall) and stability when working with incomplete data.

E. Generalisability and Limitations

The study is limited to OULAD data, which reflects the specifics of one university and a specific LMS. This allowed for a systematic comparison of architectures but limits the transferability of conclusions to other educational environments with different interaction patterns. The results should be considered in the context of LMS analytics rather than as universal patterns. To confirm the reliability of the identified cost-effectiveness patterns, verification using data from other educational institutions is necessary.

V. Conclusion

In this study, we conducted a comprehensive experimental analysis of the performance of eight recurrent architectures for the task of early student dropout risk prediction based on OULAD data. The results demonstrate high model robustness: regardless of the time sequence length and architecture type, classification accuracy remained within the same range, with only minor differences in results. Statistical analysis using the McNemar test confirmed the absence of significant differences between the models, indicating that key student behavioral patterns are sufficiently pronounced to be reliably extracted by a wide range of architectures, including recurrent, convolutional-recurrent, and attention-based approaches. Metric trend plots showed that all models identified these patterns very early. Therefore, metric performance is not the determining factor when choosing an architecture for use in such tasks.

The consistently high performance observed across all evaluated architectures, ranging from basic RNNs to complex attention-based models suggests that the predictive signal is primarily driven by the underlying feature engineering rather than specific model heuristics. Our empirical results demonstrate that while complex models like Attention-LSTM involve a significantly larger parameter space (141k vs. 29k for LSTM), they do not yield statistically significant gains. This indicates that the carefully designed temporal and static features effectively capture the variance in student engagement, leading to a performance plateau where architectural complexity offers diminishing returns. While a full ablation study of the feature set is beyond the scope of this paper, our results provide a clear baseline for the cost-performance trade-off in the LMS context, where simpler models achieve competitive accuracy with lower computational overhead.

It was clear that the resulting metrics and computational performance of the models exhibit significant differences. The number of parameters varied almost sevenfold, from 23,969 for the base GRU to 141,089 for the LSTM with an attention mechanism. Training times varied by more than three times. These differences are crucial for practical educational analytics systems, where infrastructure resource requirements can be a limiting factor. Therefore, when deploying early risk group detection systems, preference should be given to models that provide an optimal balance between prediction quality and computational efficiency. In particular, light basic models with no modifications to GRU and LSTM demonstrate similar performance rates with significantly lower resource consumption, making them the most rational choice for such problems.

Despite the high stability of the results, this study has several limitations. The OULAD dataset represents data from a single platform and covers a limited time period, which may reduce the generalizability of the models. Future work would help expand the experimental design to include multiple independent educational platforms, courses, and institutional contexts, and in examining the transferability of the trained models. An additional direction of development is the inclusion of a broader set of features—demographic, contextual, and behavioral—which will allow for a more complete description of risk factors.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

AB conducted the research; DN supervised the research; ABB and SS developed methodology; SS and SA collected data; SS and ABB studied research methods; DN and SA reviewed and edited the draft; all authors had approved the final version.

References

[1] L. Mishra, T. Gupta, and A. Shree, "Online teaching-learning in higher education during lockdown period of COVID-19 pandemic," *International Journal of Educational Research Open*, vol. 1, Jan. 2020. doi: 10.1016/j.ijedro.2020.100012

[2] L. M. D. Santos, "Online learning after the COVID-19 pandemic: Learners' motivations," *Front. Educ.*, vol. 7, Sept. 2022. doi: 10.3389/feduc.2022.879091

[3] S. Senthamaraiselvi, "Hybrid RNN-BiLSTM approach for student success estimation in online social networks," *Journal of Information Systems Engineering and Management*, vol. 10, no. 2, pp. 507–519, Feb. 2025. doi: 10.52783/jisem.v10i2.2402

[4] K. O. Adefemi and M. B. Mutanga, "A robust hybrid CNN–LSTM model for predicting student academic performance," *Digital*, vol. 5, no. 2, 16, June 2025. doi: 10.3390/digital5020016

[5] A. E. Hajoui, O. Y. Alaoui *et al.,* "A student performance prediction using RNNs models with variety of optimization techniques in deep learning," *IJIET*, vol. 15, no. 6, pp. 1289–1301, 2025. doi: 10.18178/ijiet.2025.15.6.2331

[6] F. A. Al-azazi and M. Ghurab, "ANN-LSTM: A deep learning model for early student performance prediction in MOOC," *Heliyon*, vol. 9, no. 4, Apr. 2023. doi: 10.1016/j.heliyon.2023.e15382

[7] G. Airlangga, "Predicting student performance using deep learning models: A comparative study of MLP, CNN, BiLSTM, and LSTM with attention," *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, vol. 4, no. 4, pp. 1561–1567, Oct. 2024. doi: 10.57152/malcom.v4i4.1668

[8] S. Poudyal, M. J. Mohammadi-Aragh, and J. E. Ball, "Prediction of student academic performance using a hybrid 2D CNN model," *Electronics*, vol. 11, no. 7, 1005, Jan. 2022. doi: 10.3390/electronics11071005

[9] F. Shiri, T. Perumal, N. Mustapha, and R. Mohamed, "A comprehensive overview and comparative analysis on deep learning models," *JAI*, vol. 6, pp. 301–360, 2024. doi: 10.32604/jai.2024.054314

[10] B. K. Yousafzai *et al.*, "Student-Performulator: Student academic performance using hybrid deep neural network," *Sustainability*, vol. 13, no. 17, Jan. 2021. doi: 10.3390/su13179775
[11] J. Kuzilek, M. Hlosta, and Z. Zdrahal, "Open University Learning Analytics dataset," *Sci. Data*, vol. 4, no. 1, Nov. 2017. doi: 10.1038/sdata.2017.171
[12] J. T. Hancock and T. M. Khoshgoftaar, "Survey on categorical data for neural networks," *Journal of Big Data*, vol. 7, no. 1, 28, Apr. 2020. doi: 10.1186/s40537-020-00305-w
[13] F. Aldi, F. Hadi, N. A. Rahmi, and S. Defit, "Standardscaler's potential in enhancing breast cancer accuracy using machine learning," *Journal of Applied Engineering and Technological Science (JAETS)*, vol. 5, no. 1, pp. 401–413, Dec. 2023. doi: 10.37385/jaets.v5i1.3080
[14] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, July 2019. doi: 10.1162/neco_a_01199
[15] S. Yang, X. Yu, and Y. Zhou, "LSTM and GRU neural network performance comparison study: Taking Yelp review dataset as an example," in *Proc. 2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI)*, June 2020, pp. 98–101. doi: 10.1109/IWECAI50956.2020.00027
[16] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997. doi: 10.1109/78.650093
[17] K. O. Adefemi, M. B. Mutanga, and V. Jugoo, "Hybrid deep learning models for predicting student academic performance," *Mathematical and Computational Applications*, vol. 30, no. 3, June 2025. doi: 10.3390/mca30030059
[18] H. Farhood, I. Joudah, A. Beheshti, and S. Muller, "Evaluating and enhancing artificial intelligence models for predicting student learning outcomes," *Informatics*, vol. 11, no. 3, Sept. 2024. doi: 10.3390/informatics11030046
[19] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," arXiv preprint arXiv:1607.06450, July 2016.
[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd International Conference on Machine Learning (ICML)*, vol. 37, July 2015, pp. 448–456.
[21] A. F. Agarap, "Deep learning using rectified linear units (ReLU)," arXiv preprint, arXiv:1803.08375, Feb. 2019.
[22] O. Rainio, J. Teuho, and R. Klén, "Evaluation metrics and statistical tests for machine learning," *Sci. Rep.*, vol. 14, no. 1, 6086, Mar. 2024. doi: 10.1038/s41598-024-56706-x
[23] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, 6, Jan. 2020. doi: 10.1186/s12864-019-6413-7
[24] C. Marzban, "The ROC curve and the area under it as performance measures," *Weather and Forecasting*, Dec. 2004. doi: 10.1175/825.1
[25] Ş. K. Çorbacıoğlu and G. Aksel, "Receiver operating characteristic curve analysis in diagnostic accuracy studies: A guide to interpreting the area under the curve value," *Turkish Journal of Emergency Medicine*, vol. 23, no. 4, 195, Dec. 2023. doi: 10.4103/tjem.tjem_182_23
[26] C. Militello, L. Rundo, S. Vitabile, and V. Conti, "Fingerprint classification based on deep learning approaches: Experimental findings and comparisons," *Symmetry*, vol. 13, no. 5, May 2021. doi: 10.3390/sym13050750
[27] S. Angerbauer, A. Palmanshofer, S. Selinger, and M. Kurz, "Comparing human activity recognition models based on complexity and resource usage," *Applied Sciences*, vol. 11, no. 18, Jan. 2021. doi: 10.3390/app11188473
[28] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," arXiv preprint, arXiv:1605.07678, Apr. 2017.
[29] S. Jin *et al.*, "COMET: A novel memory-efficient deep learning training framework by using error-bounded lossy compression," arXiv preprint, arXiv:2111.09562, Nov. 2021.