Information Sciences 545 (2021) 223–240

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Fine-grained learning performance prediction via adaptive sparse self-attention networks

Xizhe Wang [a], Xiaoyong Mei [a], Qionghao Huang [b], Zhongmei Han [b], Changqin Huang [a,b,*]

[a] Department of Educational Technology, Zhejiang Normal University, Jinhua, China
[b] School of Information Technology in Education, South China Normal University, Guangzhou, China

ARTICLE INFO

Article history:
Received 30 March 2020
Received in revised form 3 August 2020
Accepted 7 August 2020
Available online 14 August 2020

Keywords:
Fine-grained performance prediction
Long sequence generation
Adaptive sparse self-attention networks
E-learning

ABSTRACT

Deep learning (DL) techniques have shown good potential in building favorable predictive models in e-learning environments. It is important to build a DL-based, fine-grained student performance prediction model to predict students' outcomes and learning status at every stage of their course, rather than merely predicting the students' final score and drop-out rate, which is referred to as coarse-grained performance. In this paper, we tackle the problem of fine-grained student performance prediction in an online course using DL-based long sequence generation by formulating the students' feature as a matrix where elements in certain parts are missing. We propose an adaptive sparse self-attention network to generate the missing values and simultaneously predict the fine-grained performance. The matrix representation of the features facilitates position-wise feature selection, which helps to find the most correlated components from the past learning stage, leading to an embedding with both the original features and their spatial relationships. We build a deep neural network model with several sparse self-attention layers stacked together to achieve sequence generation. Experimental studies on three real-world datasets from different e-learning platforms demonstrate the effectiveness and advantages of our proposed method.

1. Introduction

The development of computer-assisted learning systems and communications technology has led to a significant increase in numbers of learners, resources and services [20] on the platform of massive open online courses (MOOC). In particular, student performance prediction has received much attention in recent years because the predictive learning outcomes of online students can potentially offer useful information for teachers/instructors to estimate the students' final performance at an early stage in the course [46]. However, in this research direction, many if not most studies focus on predicting the students' final performance, regarded as coarse-grained performance, but pay less attention to periodical outcomes, regarded as fine-grained performance. The difference between coarse-grained performance and fine-grained performance is shown in Fig. 1. In practice, the accurate prediction of students' fine-grained performance can enhance learning efficiency, e.g., increasing student engagement [41] and/or preventing student dropout [8,10], which is of significant importance for e-learning. Overall, a good predictive model that not only reliably forecasts students' fine-grained performance early in each learning stage during an online course, but also effectively embodies the key components throughout the learning process that have

* Corresponding author at: Department of Educational Technology, Zhejiang Normal University, Jinhua, China.

https://doi.org/10.1016/j.ins.2020.08.017
0020-0255/© 2020 Elsevier Inc. All rights reserved.

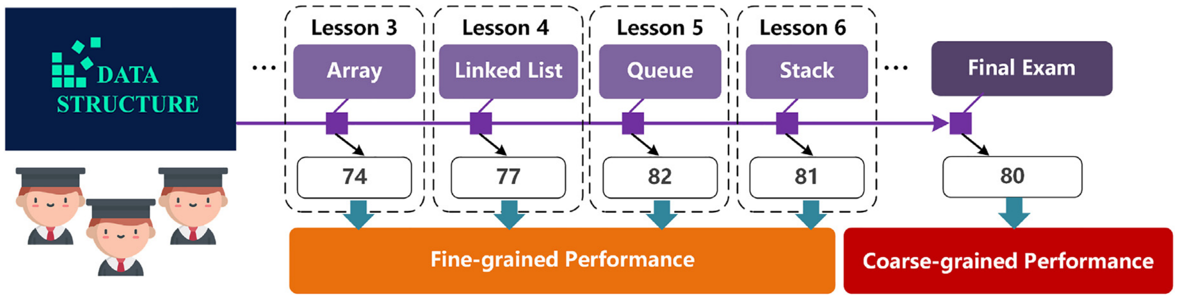


Fig. 1. Sample of student performance in an online course, showing the difference between fine-grained (periodical) performance and coarse-grained (final) performance.

an explicit/implicit influence on their performance, which would assist decision making in relation to when (and with whom) to offer timely educational interventions and thereby empower students to succeed.

There is a significant difference between coarse-grained and fine-grained performance prediction tasks. First, in terms of the modeling objective, coarse-grained performance prediction aims at building a model that can simulate the functional relationship between the learning features (e.g., attendance rates, learning attitudes, learning liveness, etc.) and the final outcomes of the students [13,25,47]. The premise behind the problem formulation, as adopted in some existing deep learning models/algorithms [46,7], is that all the feature components/information are already available since the one and only target is the final performance. In contrast, the purpose of fine-grained performance prediction is to predict the periodical outcomes at each learning stage. For instance, for a given online course $\mathcal{A}$, there are in total m lessons scheduled in the syllabus, meaning that $p_1, p_2, \ldots, p_m$ performances/scores should be considered at the end of each lesson. As such, one immediate challenge is that some features at a given stage are not known until the end of the lesson, which makes it difficult (but not unachievable) to build a predictive model as far as possible before the end of the current lesson. Technically, some advanced deep learning techniques, including the data-driven approximator construction for regression/classification [44,42] and sequence generation techniques [29,22,12] are needed for problem solving. Second, from the perspective of utility in real e-learning environments, it is natural to think that fine-grained performance prediction is more useful than coarse-grained performance prediction since it has the potential to provide timely proactive and actionable feedback to the students at each stage of the online course, rather than one-shot prediction which only concerns the final outcomes. A good fine-grained performance prediction model based on advanced deep learning techniques can provide valuable clues in good time for necessary interventions to enhance the students' motivation and engagement in e-learning environments.

In this paper, we propose a novel framework for building a fine-grained performance prediction model with deep neural network architecture, named the adaptive sparse self-attention network (AS-SAN), to cope with the aforementioned technical issues. To begin with, for a given student studying online course $\mathcal{A}$ comprising m lessons, we propose a matrix representation of the student's learning features, facilitating position-wise feature selection which helps the following process of long sequence generation. In particular, matrix representation and correlation-based feature selection make it adaptive to find the most relevant feature information reflecting the mutual influence between the different learning behaviors (students' engagement, course duration, motivation, etc.) presented by the students. Then, we develop a deep neural network with several adaptive sparse self-attention layers stacked together to model the functional relationship between the embedding obtained from the feature selection stage and the objective targets. Importantly, the utilized self-attention strategy can potentially find the internal correlation of elements at any position in the learning feature matrix by calculating and identifying their similarities, by which the proposed decoder-only architecture can effectively explore the historical data to capture the implicit association of each feature in the course semantic space. Based on the prediction result, student performance can be analyzed in a timely manner and some early warning tips can be offered accordingly. We test the proposed AS-SAN model on three real-world datasets collected from three e-learning systems and compare its performance with five existing baselines. Our experiment results show the effectiveness and efficiency of AS-SAN in terms of different evaluation metrics. Specifically, based on our experiment results for fine-grained performance prediction, we further discuss learning trend analysis by which the favorable potential of our proposal to provide timely proactive and actionable feedback to students participating in an online course is clearly demonstrated. Our main contributions are summarized as follows:

- Data representation: A matrix representation of the student's learning features at each stage of an online course is presented, in which the difference between the general problem formulation of fine-grained performance prediction and coarse-grained performance prediction, is elaborated. We believe that our proposed strategy of matrix representation for learning features is universal and is of some referential value for other kinds of timely learning performance prediction problems with missing/unknown features.

- Model: A novel fine-grained performance prediction model is proposed via performing adaptive correlation-based feature engineering and several adaptive sparse self-attention layers with decoder-only architecture. To the best of our knowledge, this is the first attempt to consider position-wise feature selection and deep learning techniques with a focus on fine-grained student performance prediction.
- Application: An experimental study conducted on three real-world datasets along with several case studies shows the good potential of our framework in both industrial research and practical needs, in light of the fast development of online courses.

The remainder of this paper is organized as follows: Section 2 briefly reviews some existing work on student performance prediction, and some techniques on long sequence generation to support fine-grained performance prediction. Section 3 gives the preliminaries including the basic assumptions behind our proposal, the notations and definitions, and the problem formulation. Section 4 details our proposed framework of building an adaptive sparse self-attention network for fine-grained student performance prediction. An experimental study on three real-world e-learning datasets is conducted and the results of the proposed method are compared with several baseline approaches, as detailed in Section 5. Section 6 provides the conclusion.

2. Related work

The main purpose of this section is to review the current research activities in relation to student performance prediction. Then, with a focus on developing a fine-grained prediction method, we briefly introduce the line of research on long sequence generation.

2.1. Student performance prediction

As learning analytics is a hot research issue, there are a number of related studies on student performance prediction which have been published to date. Many scholars have proposed various methods to build a student performance prediction model with different platforms (e.g. MOOC [13], Moodle [44], etc.), and goals (e.g. final performance [39], dropout rate [8], and engagement [41], etc.). With the development of deep learning techniques, all kinds of student performance prediction tasks can be realized in a more accurate way, and have become the main basis of learning early warning [4], learning status recognition [47], etc. In previous studies, Holmes et al. [17] presented a novel descriptive analysis of learner behavior, image processing techniques, and artificial neural networks (ANN) to model and classify comprehension and non-comprehension states in near real-time. Polyzou et al. [34] used a feature extraction method for next-term performance prediction. Yang et al. [46] proposed a behavior-based grade prediction method for MOOCs via time series neural networks (TSNN). These models achieved the best prediction performance, but most of these studies only target fiworkperformance, which known as coarse-grained prediction.

Corresponding to the formative evaluation in education, fine-grained prediction is more helpful for students' self-understanding as it supports more learning stages and provides more information. In the latest relevant research, Kolog et al. [25] proposed a fine-grained emotion detection system through support vector machines. Applications show that this system can be implemented to track emotions, especially from students' generated content. Kime et al. [23] applied a performance factors analysis (PFA) approach to provide a more fine-grained analysis of the student's mastery of content. To some extent, fine-grained prediction has become the coming trend in student performance prediction.

Due to the advantages of sequential tasks, recurrent neural networks (RNNs) and their variant models have become the most popular methods for fine-grained prediction. The structure of RNNs leads to the problem of gradient disappearance in long series prediction [26]. Although long short-term memory networks (LSTM) and gated recurrent unit (GRU) can alleviate gradient disappearance to a certain extent through the internal gate structure, there has to be a certain training epoch which cannot be avoided. After this, attention-based methods, especially the self-attention mechanism, achieved the state-of-the-art in many tasks, e.g., long text generation [29], image completion [1], etc., which can be used as the first choice for fine-grained student performance prediction.

2.2. Long sequence generation

Long sequence generation has broad applications for sequential data. For fine-grained prediction tasks, to ensure the validity of the prediction results, the prediction processes are always accompanied by the generation of missing features, which results in fine-grained prediction tasks transforming into sequence generation tasks [21].

Roughly, sequence generation techniques are classified into two categories: The first category is conditional sequence generation, which means the generation process of these tasks has reference information, e.g., machine translation [28], text summarization [29], etc. The other category is unconditional generation, which means the whole generation process is only based on existing features, and no additional information is added to the generation processes, e.g., image incompletion [1], audio generation [19]. Fine-grained prediction belongs to unconditional generation tasks.

Therefore, designing unconditional generative models for long sequence generation is highly challenging, as generative modeling requires estimating the joint probability over the entire sequence, which is susceptible to covariant shift and compounding errors, hence the long-range dependence of the model has to be considered. The whole generative process can be formalized by an autoregressive generation model [12]:

$$p(x) = \prod_{i=1}^{n} p(x_i|x_1, \ldots x_{i-1}; \theta), \tag{1}$$

where $p()$ is the conditional probability distributions, and is parameterized by a network θ.

Conventional time series methods such as ARIMA [30], RNN [31] can model this task well. But most of these methods are context-sensitive, and the elements of each row in the learning feature matrix are not directly related, which may have an impact on long-range generation. Recently, various works based on the attention mechanism have achieved significant improvements in text/image generative models. From seq2seq [29], to attention mechanism [33], to self-attention [40], the attention mechanism has become one of the most popular research areas of sequence generation. With the introduction of transformers, the self-attention model can no longer be based on the CNN/RNN framework. Self-attention first involves machine translation, to replace RNN in a seq2seq structure as an entire model framework, and has been widely adopted in many state-of-the-art pre-training language models (BERT [14], GPT [36], etc.). Compared with RNN and other time series prediction models, SAN has advantages in long-term dependence and has achieved good results in many tasks, e.g., item recommendation [32] and traffic flow prediction [27], etc. Technically, a number of studies are proposed to enhance the effect and efficiency of the self-attention mechanism. Lin et al. [28] presented a temperature mechanism to change the softness of attention distribution. Yang et al. [45] proposed local attention for self-attention to reduce the number of samples and improve efficiency. Shen et al. [38] integrated both soft and hard attention into one context fusion model, named the reinforced self-attention network (ReSAN), which achieved state-of-the-art performance.

In this study, we proposed a self-attention based fine-grained feature prediction method to generate the whole learning situation feature matrix. This study predicts lesson-level performance and provides a more valuable reference for learning services, which potentially helps increase student engagement and empowers students to succeed.

3. Preliminaries and problem formulation

In this section, we first introduce some assumptions and definitions, then we present the problem formulation in detail. To facilitate the understanding of various symbol representations, some important notations are shown in Table 1.

3.1. Assumptions and definitions

First of all, four assumptions are made to eliminate the interfering factors:

Assumption 1. The online course is iterative and all of the historical learning data is available and can be used with permission.

Assumption 2. Students are asked to complete this online course to obtain course credit, to ensure there are no or very few dropouts and that they treat their study seriously.

Assumption 3. Course learning is a continuous process, composed of multiple lessons with a fixed cycle, and each lesson has its corresponding knowledge points.

Assumption 4. At the end of each lesson, teachers will provide learners with a learning evaluation of the current course.

For online course students, their performance in one lesson is mainly determined by their personality learning behaviors [35], learning content [18], etc., which can be represented by a set of learning features, e.g., knowledge level, engagement, liveness, etc. The corresponding learning features in each lesson can form a learning feature matrix, and its definition is given as follows:

Definition 1 (*Learning feature matrix*). For a student in the i-th lesson of an online course $\mathcal{A}$, assign a fixed size feature-lesson matrix $\mathbb{V} \in \mathbb{R}^{m\times d}$, where m and d are the numbers of lessons and learning features, $V_k = [v_{1,k}, \ldots, v_{m,k}]$ is the k-th leaning feature sequential list.

According to the feature status, there are two types of learning features in this study, i.e., static learning features and dynamic learning features, which are detailed as follows.

Static learning features represent the personalized information of students with a fixed value for the whole course, e.g., student ID, age, gender, knowledge level and learning expectation, etc. Thus we have the definition of static learning features:

Table 1
Some important notations.

Notation	Description
$\mathcal{A}$	An online course
$\mathbb{V}$	The learning feature matrix including $\mathbb{V}_{sta}$ and $\mathbb{V}_{dyn}$ with d dimensions
P	The sequential list of student stage performance in $\mathcal{A}$
$\mathcal{C}$	A joint matrix combining $\mathbb{V}$ with P
$\mathcal{S}$	A collection of selected learning features
MB	A memory block sets to mark the position of the selected features
X	Input data of AS-SAN
$\hat{y}_i$	Predicted fine-grained performance of i-th lesson
E	A embedding matrix
$\mathcal{M}$	Sparse attention masking function
$i \in [1, m]$	The index of lessons in $\mathcal{A}$
$j \in [1, n]$	The index of students in $\mathcal{A}$
$k \in [1, d+1]$	The index of learning features in each lesson

Definition 2. [Static learning features]A static learning feature matrix, denoted as $\mathbb{V}_{sta} = \{V_1, \ldots, V_s\}, \mathbb{V}_{sta} \in \mathbb{R}^{m \times s}$ consists of s sets of static learning feature in $\mathcal{A}$, which are the first few columns of $\mathbb{V}$.

Different to static learning features, dynamic learning features are adjusted according to the status of students' activities and behaviors in each lesson, e.g., engagement, liveness, sentiment, relevance of content, course during, etc. In our previous study [43], we defined and quantified these features. Thus in this work, we denote the dynamic learning features as follows:

Definition 3 (*Dynamic learning features*). Except for the $\mathbb{V}_{sta}$ in $\mathbb{V}$, the dynamic learning feature matrix is defined as $\mathbb{V}_{dyn} = \{V_{s+1}, \ldots, V_d\}, \mathbb{V}_{dyn} \in \mathbb{R}^{m \times (d-s)}$, to represent the dynamic learning features in this work.

After this, different from general learning prediction tasks, our goal is to predict student performance for each lesson in an online course, instead of the student's final score for the whole course. Hence, we denote fine-grained student performance as follows.

Definition 4 (*Fine-grained student performance*). Instead of the student's final performance for the whole course, student performance for each learning stage is denoted as fine-grained student performance. With a sequential list $P = [p_1, \ldots, p_i, \ldots, p_m]$ for lesson-level student performance, p_i represents the fine-grained performance of the i-th lesson.

Remark 1. Generally, the formative assessment of a lesson is determined by the performance of learners in the current lesson, i.e., $p_i \sim L_i = [v_{i,1}, \ldots v_{i,d}]$. However, the assessment results also inevitably have an implicit impact on the learning features of the next lessons, such as the degree of mastery of the learned knowledge, the attainment rate of the previous tasks, etc., that is, formative assessment represents the learning situation of the course and will affect the learning process in the future.

Therefore, P is not only the prediction goals, but also the features to perform subsequent predictions in on-going lessons. Thus we combine $\mathbb{V}$ with P into a new matrix $\mathbb{C}$:

$$\mathcal{C}^{(j)} = Concat(V_1, \ldots, V_d, P), \mathcal{C}^{(j)} \in \mathbb{R}^{m \times (d+1)}. \tag{2}$$

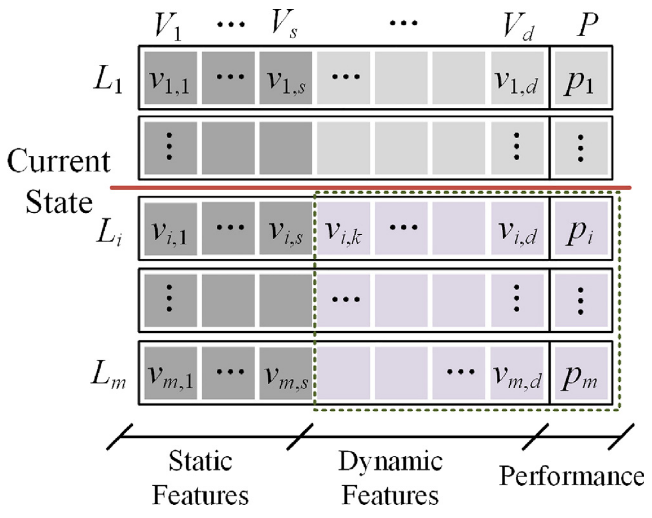
The details of the learning feature matrix are shown in Fig. 2.

3.2. Problem formulation

By virtue of the aforementioned matrix representation of the student's learning feature, we formulate the problem of fine-grained learning performance prediction as follows. Given n students studying the same online course $\mathcal{A}$, denote $\mathcal{C}^{(j)} \in \mathbb{R}^{m \times (d+1)}$ as the feature matrix of student $j, j = 1, 2, \ldots, n$, as shown in Fig. 2. The training objective can be expressed (informally) as

$$\min_{\theta} \sum_{j=1}^{n} \sum_{k=1}^{d+1} \left(\mathcal{C}_{i,k}^{(j)} - \mathrm{DLM}_{\theta}(\mathcal{C}_{i,1:k-1}^{(j)}, \mathcal{C}_{1:i-1,:}^{(j)}) \right)^2, \tag{3}$$

where $i = 1, 2, \ldots, m, \mathcal{C}_{i,k}^{(j)}$ denotes the element value in position (i, k) of the matrix $\mathcal{C}^{(j)}, \mathrm{DLM}_{\theta}(\cdot)$ stands for a deep learning model with trainable parameters θ. In particular, $\mathcal{C}_{i,1:k-1}^{(j)}$ denotes the slice with all the $(k-1)$ elements in the i-th row and the $1, 2, \ldots, k-1$-th column. For the special case where $k = 1, \mathcal{C}_{i,1:k-1}^{(j)}$ is nothing and can be omitted. In the same manner, $\mathcal{C}_{1:i-1,:}^{(j)}$ is the slice of all the elements within the first $(i-1)$ rows. It is logical to suppose that $i \geqslant i_0 > 1$, which means that at least some features are already known during the online learning process course $\mathcal{A}$.



V_1 ⋯ V_s ⋯ V_d P
L_1
$v_{1,1}$ ⋯ $v_{1,s}$ ⋯ $v_{1,d}$ p_1
Current State
L_i
$v_{i,1}$ ⋯ $v_{i,s}$ $v_{i,k}$ ⋯ $v_{i,d}$ p_i
L_m
$v_{m,1}$ ⋯ $v_{m,s}$ ⋯ $v_{m,d}$ p_m
Static Features
Dynamic Features
Performance

Fig. 2. Different part of the learning feature matrix for *j*-th student. Features in the bottom right corner of the matrix (in light blue) are to be predicted. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Specifically, when $k = d + 1$, the output of the model DLM_θ is the prediction of the fine-grained learning performance, i.e., $p_i, i = 1, 2, \dots, m$, as shown in Fig. 2.

The above problem formulation motivates us to think about two key technical issues. First, some feature selection strategy is needed to find the most useful components among the set $C^{(j)}_{i,1:k-1} \bigcup C^{(j)}_{1:i-1,:}$, otherwise time and memory usages may grow quadratically with sequence length. Second, an advanced deep learning model is needed that can attain equivalent or better performance on modeling the unknown functional relationship between the obtained embedding of the original long sequences and the prediction targets. We detail our proposal to address these two issues in Section 4.

4. Adaptive sparse self-attention network based fine-grained student performance prediction

Technically, the self-attention network introduces an attention mechanism relating different positions of a single sequence mechanism over the entire sequence, and demonstrate good potential in many different tasks. However, n elements are needed to compute the weights n times, leading to a $O(n^2)$ time complexity, which will significantly reduce the efficiency of the model when the sequence is too long. This makes the timeliness a bottleneck for the learning prediction and early warning task. Sparse transformers [12] provides an efficient solution to this problem. With the unique factorized attention heads, it can effectively reduce the amount of computation by capturing local feature information.

Thus in this section, we propose an adaptive sparse self-attention network (AS-SAN) for fine-grained student performance prediction. Compared to the ordinary self-attention mechanism adopted in prediction models, AS-SAN employs an adaptive feature selection method to efficiently capture the mutual influence between the eigenvalues of learning behaviors. Although the numerical range of each learning feature (engagement, duration, etc.) is quite different, these features may have an implicit correlation between them. Based on a decoder-only architecture, self-attention can mine the historical data to capture the implicit association of each feature in the course semantic space. It is necessary to compare this with other elements to obtain the feature distribution of the current position in the feature matrix. After this, the prediction of a vacant position is realized by a maximum possible distribution value. As aforementioned, this section consists of the following two parts: (i) a pair-wise feature correlation measurement method is proposed for local feature selection before self-attention calculation; (ii) using AS-SAN for autoregressive learning feature generation, to predict lesson-level performance for the following lessons. Finally, we introduce each component of our model in detail.

4.1. Model architecture

The architecture of the AS-SAN model is shown in Fig. 3. The model consists of three components, i.e., correlation-based feature selection layer, decoder-only self-attention network, and early warning layer. It takes the historical data of the completed lessons in the current course as input, and outputs the learning features one by one until the learning feature matrix is complete. The structure is detailed in the following sub-sections.

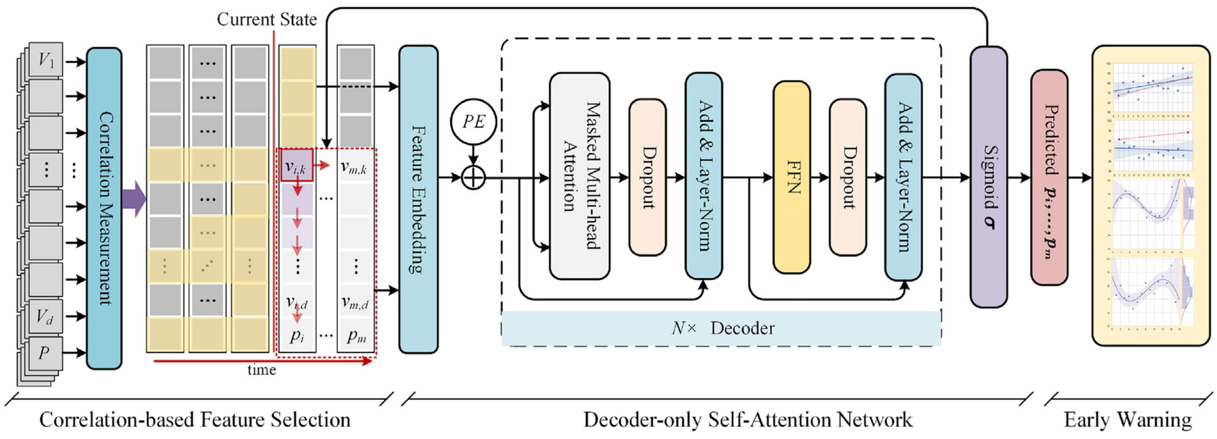


Fig. 3. Detailed architecture of the proposed AS-SAN model.

4.2. Correlation-based feature selection

The goal of the feature selection layer is to find the most contributive elements of all the learning features in the initial list as the input of local self-attention. In order to simply describe the range of feature selection, we use memory block (MB) as the position sets of the selected features. Based on the characteristics of the learning feature matrix, the construction process of the memory block can be divided into two steps:

Step 1. According to Definition 2, the static features remain unchanged throughout the whole course, which makes the pair-wise calculation between the target feature and most of the static features unnecessary. Given the target feature $v_{i,k}$, we selected the s static features to the left of $v_{i,k}$, i.e., the static feature selection sets are denoted by $\mathcal{S}_{sta} = \{v_{i,1}, v_{i,2}, \dots, v_{i,s}\}$.

Step 2. With respect to the dynamic features, each column of the matrix is a time series. As a time series, the historical state $\{v_{1,k}, v_{2,k}, \dots, v_{i-1,k}\}$ is directly related to $v_{i,k}$, but the correlation between two different rows is not clear. In this premise, we consider three feature selection methods, i.e., stride feature selection, similarity-based feature selection, and joint feature selection. Three examples are given in Fig. 4 to illustrate the difference among these three selection methods.

Stride Feature Selection. Inspired by the work in [12], we give a threshold $\gamma \in [d+1, (i-1) \times (d+1)]$ as a candidate basis for adjacent features. Thus, for the γ-nearest neighbor features, we have

$$\mathcal{S}(v_{i,k}, \gamma) = \{v_{i-\gamma/(d+1),k}, \dots, v_{i-\gamma/(d+1),d+1}, \dots, v_{i,k-1}\}. \quad (4)$$

Taking the historical state before the target feature $v_{i,k}$ as the candidate region, the Stride Memory Block $MB_{stride}(v_{i,k})$ can be expressed as:

$$MB_{stride}(v_{i,k}) = \mathcal{S}(v_{i,k}, \gamma) \cup \{v_{i,1}, v_{i,2}, \dots, v_{i,k-1}\}. \quad (5)$$

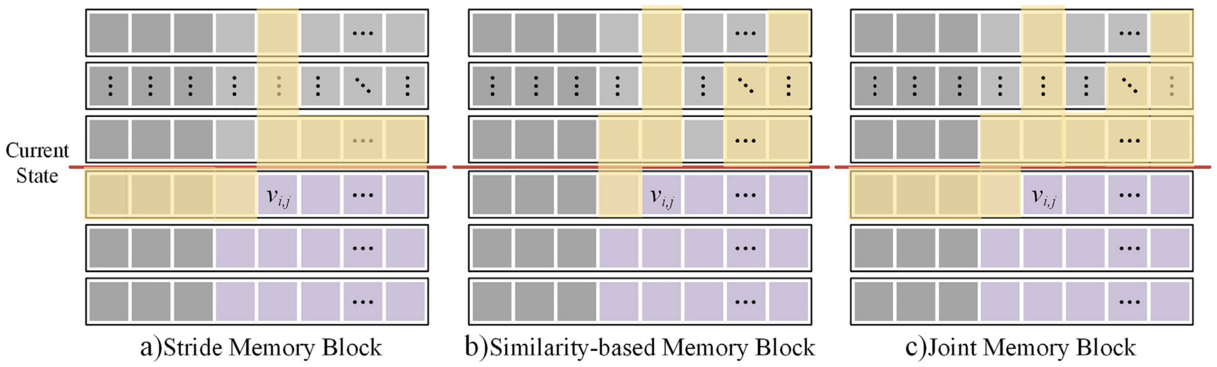


Fig. 4. Visualization samples of three dynamic feature selection methods. Blocks in yellow represent the selected input features. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

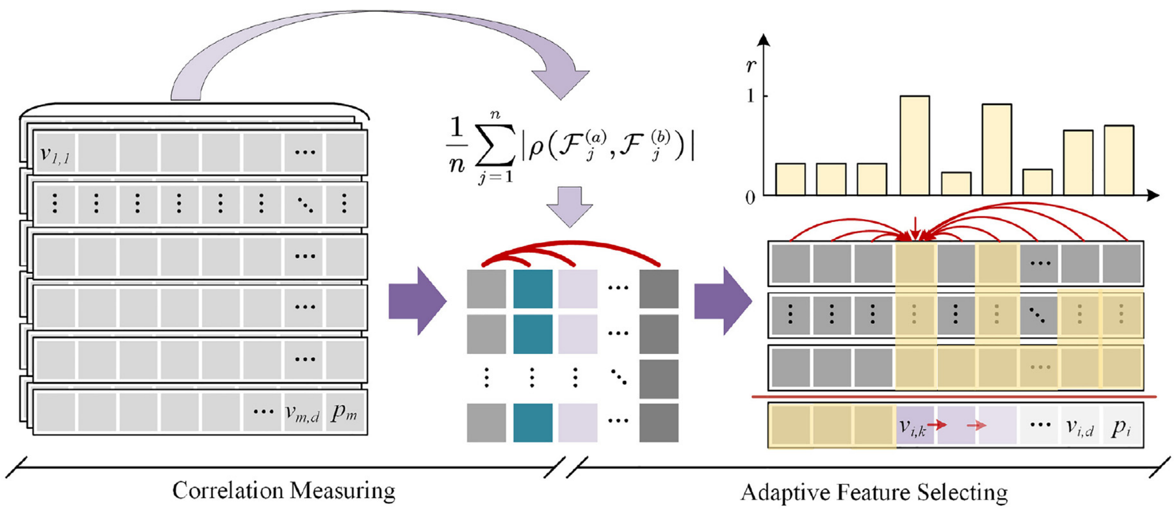

$v_{1,1}$
$v_{m,d}$
p_m
$\frac{1}{n}\sum_{j=1}^{n}|\rho(\mathcal{F}_j^{(a)},\mathcal{F}_j^{(b)})|$
r
1
0
$v_{i,k}$
$v_{i,d}$
p_i
Correlation Measuring
Adaptive Feature Selecting

Fig. 5. Process of similarity-based feature selection.

Similarity-based Feature Selection. The similarity-based feature selection process can be summarized in two steps: similarity measuring and dynamic feature selecting, which is shown in Fig. 5.

Since self-attention is mainly based on the similarity between features, we first identify the different kinds of students based on the consistency of the static features and then calculate the similarity of the features of the historical data. We give $\mathcal{F}^{(a)}, \mathcal{F}^{(b)}$ as two different sets of learning features in a course, then the feature similarity $r(\mathcal{F}^{(a)}, \mathcal{F}^{(b)}) \in [0,1]$ can be calculated as follows.

$$r(\mathcal{F}^{(a)}, \mathcal{F}^{(b)}) = \frac{1}{n}\sum_{j=1}^{n}\left|\rho\left(\mathcal{F}_j^{(a)}, \mathcal{F}_j^{(b)}\right)\right|, \tag{6}$$

where n is the number of historical data on an online course dataset. $\rho\left(\mathcal{F}_j^{(a)}, \mathcal{F}_j^{(b)}\right) \in [-1,1]$ is the Pearson correlation [5] between $\mathcal{F}^{(a)}$ and $\mathcal{F}^{(b)}$, which can be computed as:

$$\begin{aligned}\rho(\mathcal{F}^{(a)}, \mathcal{F}^{(b)}) &= \frac{cov\left(\mathcal{F}^{(a)}, \mathcal{F}^{(b)}\right)}{\sigma_{\mathcal{F}^{(a)}}\sigma_{\mathcal{F}^{(b)}}} \\ &= \frac{E\left[\left(\mathcal{F}^{(a)}-\mu_{\mathcal{F}^{(a)}}\right)\left(\mathcal{F}^{(b)}-\mu_{\mathcal{F}^{(b)}}\right)\right]}{\sqrt{\sum_{k=1}^{d+1}\left(\mathcal{F}_i^{(a)}-\mu_{\mathcal{F}^{(a)}}\right)^2}\sqrt{\sum_{k=1}^{d+1}\left(\mathcal{F}_i^{(b)}-\mu_{\mathcal{F}^{(b)}}\right)^2}},\end{aligned} \tag{7}$$

where $cov(\mathcal{F}^{(a)}, \mathcal{F}^{(b)})$ is a covariance between $\mathcal{F}^{(a)}$ and $\mathcal{F}^{(b)}$. $\sigma_{\mathcal{F}^{(a)}}, \sigma_{\mathcal{F}^{(b)}}$ are the standard deviation of $\mathcal{F}^{(a)}$ and $\mathcal{F}^{(b)}$, respectively.

After this, the value of $r \in [0,1]$ can be used as a basis for the number of vertical elements of the corresponding feature filtering, i.e., $(i-1)r \in Z$. For the k'-th column features, we have

$$\mathcal{S}(v_{i,k}, k') = \begin{cases}\left\{v_{i-(i-1)r,k'}, \ldots, v_{i,k'}\right\}, & \text{if } k' < k \\ \left\{v_{i-ir,k'}, \ldots, v_{i-1,k'}\right\}, & \text{if } k' \geqslant k\end{cases}. \tag{8}$$

For the target feature $v_{i,k}$, the whole similarity-based memory block $MB_{sim}(v_{i,k})$ can be expressed as:

$$MB_{sim}(v_{i,k}) = \bigcup_{k'=1}^{d+1}\mathcal{S}(v_{i,k}, k'). \tag{9}$$

Joint Feature Selection. Of the aforementioned two methods, the former ensures that the model obtains more extensive horizontal features, while the latter obtains the associated temporal features more effectively. Therefore, we combine these two methods and consider the similarity-based method as well as the adjacent sequence features. Then, we have the $MB_{joint}(v_{i,k})$ defined as

$$MB_{joint}(v_{i,k}) = \mathcal{S}(v_{i,k}, \gamma) \cup \bigcup_{k'=1}^{d+1} \mathcal{S}(v_{i,k}, k'). \tag{10}$$

Remark 2. Although the range of feature selection is different, all three kinds of feature extraction strategies can be regarded as a constrained correlation-based feature selection method. We define two threshold $\xi_{\min}, \xi_{\max}(\xi_{\min}, \xi_{\max} \in [0, r], \xi_{\min} < \xi_{\max})$ to limit the range of feature values, e.g., $\xi_{\min} > \frac{\gamma}{d+1} - 1$ and $\xi_{\max} = 1$ indicates the stride feature selection; $\xi_{\min} \geqslant 0$ and $\xi_{\max} \leqslant 1$ indicates the similarity-based feature selection; $\xi_{\min} > \frac{\gamma}{d+1} - 1$ and $\xi_{\max} \leqslant 1$ indicates the joint feature selection.

Further combining the two steps of the memory block construction process, which is applicative to all the feature selection scenarios, the final memory block of the target feature is computed in Eq. (11):

$$MB_{\xi_{\max}}^{\xi_{\min}}(v_{i,k}) = \left(\bigcup_{k'=1}^{d+1} \mathcal{S}_{\xi_{\max}}^{\xi_{\min}}(v_{i,k}, k') - \mathbb{V}_s \right) \cup \mathcal{S}_{static}. \tag{11}$$

We summarize the aforementioned steps of the adaptive feature selection method as follows:

Algorithm 1 Adaptive Feature Selection.

```
Input: Learning matrix $\mathcal{C}^{(j)} \in \mathbb{R}^{m\times(d+1)}$, target feature $v_{i,k}$, training data $\mathcal{D}$, threshold $\gamma$, $\xi_{\min}$ and $\xi_{\max}$.
Output: Input $X$ for prediction model.
1: Initialize $w \leftarrow \{1, 2, \ldots, d+1\}$;
2: for $\mathbb{D} \in \mathbb{R}^{m\times(d+1)}$ in $\mathcal{D}$ do
3:    $\boldsymbol{r}_w \leftarrow r(\mathcal{F}_k, \mathcal{F}_{k'})$ via Eq. (6), $\mathcal{F}_k, \mathcal{F}_w \in \mathbb{D}$ // Calculating feature similarity between $\mathcal{F}_k$ and $\mathcal{F}_w$;
4:    if $\boldsymbol{r}_w \geqslant \frac{1}{i}$ then
5:       Calculate $S_w$ via Eq. (8) // Calculating the number of values in the target interval of similar features;
6:    else $S_w = \emptyset$;
7:    end if
8: end for
9: Calculate $MB_{\xi_{\max}}^{\xi_{\min}}(v_{i,k})$ via Eq. (11) // Get the position of all elements associated with $v_{i,k}$;
10: $X \leftarrow MB_{\xi_{\max}}^{\xi_{\min}}(v_{i,k})$;
11: return $X$
```

4.3. Adaptive sparse self-attention network

In this section, AS-SAN is proposed to predict the fine-grained leaning performance. We use a multi-layer decoder-only transformer [40] for student performance prediction. This model consists of three parts: feature embedding layer, decoder layer, and output layer, which is shown in Fig. 4.

4.3.1. Embedding layer

For an autoregressive task, the length of the input features will change due to the different positions of predicted results, which need to transform discrete variables into constant length continuous vector representation through the embedding layer. Furthermore, the position of each feature in the learning feature matrix also determines the importance of the element to the predicted target. Therefore, a feature embedding generally defines a map $f_{fe}()$ from a discrete feature index to a D-dimensional vector. Similarly, position embedding [40] defines another map $f_{pe}()$ from a discrete position index to a D-dimensional vector. The final embedding of the input feature in the pos-th is usually calculated as follows:

$$f(x, pos) = f_{fe}(x) + f_{pe}(pos), \tag{12}$$

With respect to the embedding of the input in AS-SAN, an additional positional embedding is typically used to encode the spatial relationships of data. For the aforementioned correlation-based adaptive feature selection method, we have the initial list of input $X \in \mathbb{R}^{n'\times d_{input}}$, where n' is the sequence length and d_{input} denotes the dimension of each input element. In order to utilize the sequential information in the initial list, a position embedding $PE \in \mathbb{R}^{n'\times d_{input}}$ is used to represent the absolute position of the feature matrix. Then the embedding matrix for model input can be calculated by

$$E = emb(X, PE) = \left(x_{i'} W_e + pe_{i'}^r W_r + pe_{i'}^c W_c\right)_{x_{i'} \in X, pe_{i'} \in PE}, \tag{13}$$

where $emb()$ denotes the embedding function including feature embedding $f_{fe}()$ and position embedding $f_{pe}()$. W_e denotes the parameter matrix of the input data. pe indicates a two-dimensional position embedding, r and c refer to the row and column of each input element as attention embeddings, where W_r and W_c correspond to the parameter matrix of each position's row and column. Here W_e, W_r and W_c are all learned by training data.

4.3.2. Decoder layer

In decoders, a stack of N identical decoders is given to capture the inner-relationships information. Each decoder contains a multi-head attention (MHA) layer and a feed-forward network (FFN) layer. The entire self-attention operation can be implemented using a highly optimized matrix multiplication code and is executed in parallel for all the features' channels. $\mathcal{I} = \{I_1, I_2, \ldots, I_{n'}\}$ denotes the connectivity patterns for each input, where $I_{i'}\left(i' \in [1, n']\right)$ is the set of the input vectors to which the i'-th output vector attends after calculating the attention score.

The main differences between regular self-attention and AS-SAN are as follows: (i) in a regular self-attention network, we have $I_{i'} = \left\{l : l \leqslant i'\right\}$ to ensure every element of the input can contact all previous positions and itself. (ii) in AS-SAN, based on factorized attention [12], h is defined as the number of attention heads, and $A^{(h)}$ is the set of stochastic subset of E, where

$$A^{(h)}_{j'} \subset E, A^{(h)} \in \mathbb{R}^{j' \times h}, \tag{14}$$

where $j'(j' < i' - h)$ is the length of each subset $A^{(h')}, h' \in [1, h]$.

Masked Multi-head Attention. Then for multi-head attention, according to Transformers [40], h sets of matrices $W^{(h)}_{\boldsymbol{q}}, W^{(h)}_{\boldsymbol{k}}, W^{(h)}_{\boldsymbol{v}}$ represent queries, keys and values respectively. Then, a masking function $\mathcal{M}()$ is given before calculating the attention in order to select the top-κ ($\kappa \in \{4, 8, 16, 32\}$) contributive elements. In accordance with the work in [48], we concatenate the threshold ς as the κ-th (κ = 8) largest value of z, thus we have

$$\mathcal{M}_{\varsigma}(z) = \begin{cases} z, \text{if } z \geqslant \varsigma \\ -\infty, otherwise \end{cases} \tag{15}$$

The attention function is given as follows.

$$head_h\left(E, A^{(h)}\right) = softmax\left(\mathcal{M}_{\varsigma}\left(\frac{W^{(h)}_{\boldsymbol{q}} e_{i'}\left(\left(A^{(h)} W^{(h)}_{\boldsymbol{k}}\right)^T\right)}{\sqrt{\boldsymbol{d}}}\right)\right) A^{(h)} W^{(h)}_{\boldsymbol{v}}, \tag{16}$$

$$MH_h\left(E, A^{(h)}\right) = concat\left(head_1\left(E, A^{(h)}\right), \ldots, head_h\left(E, A^{(h)}\right)\right) W^o, \tag{17}$$

where $\boldsymbol{d}$ is the dimensionality of queries and keys, $W^o \in \mathbb{R}^{hn' \times d_{input}}$ is the projection matrix, e'_i is the i'-th element in E. The output is the sum of the values weighted by the scaled dot-product similarity of the keys and queries.

Feed-Forward Network. Gaussian Error Linear Unit(GELU) [15] is a smooth version of ReLU, $f(X) = X \odot \text{sigmoid}(1.702 \cdot X)$. Several studies [15,3] have proved GELU outperforms previous nonlinearities, e.g. sigmoid, tanh and ReLU, etc, especially for the sparse self-attention network [12]. Thus we use Gaussian Error Linear Unit as the activation function $f()$. After this, FNN is used to enhance the model with non-linearity and interactions of the input vectors as

$$FFN(x) = W_2 f(W_1 x + b_1) + b_2. \tag{18}$$

4.3.3. Output layer

The output layer generates a feature value for each incomplete matrix element. We use one linear layer followed by a sigmoid layer. The output of the sigmoid layer is the probability of the feature value, which is labeled as

$$O = LN(Dropout(MH_h) + E), \tag{19}$$

$$\hat{y} = \sigma(LN(Dropout(FFN(O) W_{out}) + O)), \tag{20}$$

where $LN()$ stands for the layer normalization [2], which is normalizes the data of each sample in each dimension. O is the output of the MHA layer and W_{out} is the output weight matrix.

For our learning feature autoregressive generation task, we use the MSE loss function on the training of AS-SAN, and to minimize the loss function, denote $\varphi = (i - 1) \times d + k$, that is,

$$\mathcal{L}(y_{i'}, \hat{y}_{i'}) = \frac{1}{n'} \sum_{i'=\varphi}^{n'} (\hat{y}_{i'} - y_{i'})^2. \tag{21}$$

Therefore, we summarize the above steps of fine-grained performance prediction via AS-SAN using Algorithm 2.

Algorithm 2 Fine-grained Performance Prediction.

```
Input: Training data $\mathcal{D}$, model input $X$, learning feature matrix $\mathcal{C}^{(j)} \in \mathbb{R}^{m\times(d+1)}$, first target feature $v_{i,k}$, related input feature sets $\mathcal{I}$, number of attention heads $h$, mask threshold $\varsigma$.
Output: Fine-grained performance $P = \{p_i, \dots, p_m\}$.
1: $\mathbb{W}_{PE}, \mathbb{W}_{MHA}, \mathbb{W}_{FFN}, W_{out} \leftarrow$AS-SAN$(\mathcal{D})$ // Training AS-SAN model to learn the Parameter matrix;
2: $i' \leftarrow (i-1)\times(d+1)+k$ // Initialize the index of input;
3: for $i' \leqslant m\times(d+1)$ do
4:    $E \leftarrow \left(x_{i'} + pe^r_{i'}W_r + pe^c_{i'}W_c\right)_{x_{i'}\in X, pe_{i'}\in PE}$ // Embed position information into input features;
5:    $A^{(h)}_{j'} \overset{stochastic}{\leftarrow} I_{i'}, i' \in [1, n'], j' < i' - h$ // Select $h$ sets of factorized stochastic subset;
6:    for $t \leqslant h$ do
7:       $head_t\left(E, A^{(t)}_{j'}\right) \leftarrow softmax\left(\mathcal{M}_\varsigma\left(\frac{W^{(t)}_{\boldsymbol{q}} e_{i'}\left(\left(A^{(t)}_{j'}W^{(t)}_{\boldsymbol{k}}\right)^T\right)}{\sqrt{\boldsymbol{d}}}\right)\right)A^{(t)}_{j'}W^{(t)}_{\boldsymbol{v}}$ // Calculate $t$-th head attention;
8:       $MH_t\left(E, A^{(t)}_{j'}\right) \leftarrow concat\left(MH_{t-1}\left(E, A^{(t-1)}_{j'}\right), head_t\left(E, A^{(t)}_{j'}\right)\right)$ // Concatenate all the heads;
9:    end for
10:   $O_{i'} \leftarrow LN\left(Dropout\left(MH_h\left(E, A^{(t)}_{j'}\right)\right) + E\right)$ // Calculate MHA layer output;
11:   $\hat{y}_{i'} \leftarrow \sigma(LN(Dropout(FFN(O_{i'})W_{out}) + O_{i'}))$ // Predict target feature value;
12:   $X \overset{append}{\leftarrow} \hat{y}_{i'}$ // Update input list;
13:   if $i' \mathrm{mod}(d+1) = 0$ then
14:      $p_i \leftarrow \hat{y}_{i'}, i \leftarrow i+1$ // Update output list;
15:   If
16:   $i' \leftarrow i' + 1$ // Update index of input;
17: end for
8: return $P$.
```

5. Experimental results

In this section, we first introduce three real-world online course datasets and five baselines for evaluation. Then we compare the proposed method with the baselines on these datasets.

5.1. Datasets and baselines

We evaluate our approach based on three datasets: WorldUC dataset, Liru dataset and Junyi dataset. These datasets are from three real-world e-learning system WorldUC,[1] Liru Online Course,[2] and Junyi Academy.[3] An overview of the three datasets is shown in Table 2.

In our previous work [43], we introduced the WorldUC and Liru datasets and proposed the methods of feature quantification. Through further data collection and preprocessing, we extended these datasets, which are detailed as follows.

WorldUC dataset. This dataset has 10,523 records with 10 continuous lessons in an online course for three years. For each student in one lesson, there are eight input features including three static features (gender, age and expectation score), five dynamic features (engagement, liveness, the relevance of learning resources and content, sentiment and duration time) and one output feature (assessment score).

Liru dataset. This dataset contains data on 1046 students in 18 consecutive lessons. Except for the static features (gender, grade, class and college) and 2 dynamic features(knowledge level and assignment score), the other five dynamic features and the output feature are as same for the WorldUC dataset.

Junyi dataset [11]. This is a public dataset from the Junyi Academy (an e-learning platform similar to the Khan Academy), containing information on 2063 students with 29 input features from 6 categories, i.e. student modeling (4 features), answer time duration (6 features), problem taken in exercises (4 features), answering accuracy (6 features), user taking exercises (3

1 http://www.worlduc.com.
2 https://moodle.scnu.edu.cn.
3 https://www.junyiacademy.org.

Table 2
Overview of the Datasets.

	WorldUC	Liru	Junyi
#Students	10523	1046	2,063
#Lessons	10	18	18
#Features	3(s)+5(d)+1(o)	4(s)+7(d)+1(o)	4(s)+20(d)+1(o)

#Features, s, d, o denote static features, dynamic features and fine-grained performance.

features) and user answering orders (6 features). Similar to our definition, we use four student modeling features as static features, and combine six answering accuracy features into 1 as the output feature, and the others are dynamic features.

Before the evaluation, we standardized these datasets through data preprocessing to keep the value range of the dynamic features and output feature between [0,100].

In this work, five similar autoregressive generation methods are selected for comparison as baselines:

(i) **Historical average (HA)** [9] uses the average of the historical records as the prediction result.
(ii) **Auto-regressive integrated moving average model (ARIMA)**[30]. ARIMA adds an integrated part to the ARMA [37] model, which can be applied one or more times to eliminate the non-stationarity for time series analysis.
(iii) **Significance-Offset Convolutional Neural Network (SOCNN)** [6] is an autoregressive convolutional network. With a weighted sum of adjusted regressors, the weights are data-dependent functions learned through a convolutional network.
(iv) **Pixel recurrent neural networks (P-RNN)**[31] uses a bi-LSTM structure to scan the feature matrix in a diagonal fashion starting from a corner at the top and reaching the opposite corner at the bottom.
(v) **Self-attention network (SAN)**[40] gives a multi-head self-attention mechanism and makes it possible to perform seq2seq modeling without CNN or recurrent network units.

After this, based on the different types of feature selection introduced in Section 4, three variants of AS-SAN models are implemented to the baselines, i.e., **Stride-AS-SAN**, **Sim-AS-SAN**, and **Joint-AS-SAN**, presenting the specific AS-SAN model with stride feature selection, similarity-based feature selection, and joint feature selection, respectively.

5.2. Experimental setup

Data Preprocessing. According to the datasets, the size of the regular learning feature matrix can be represented as T $\times$ F, where T and F are the number of lessons and features, and t and f denote the feature indices. Thus in the WorldUC, Liru and Junyi datasets, the sizes of the learning matrices are 10 $\times$ 9, 18 $\times$ 12 and 18 $\times$ 25, respectively. We randomly selected 75% of the samples as the training samples and the remaining 25% as the testing samples. In order to simulate learning prediction in real application scenarios, we mask the last *b* rows in the testing samples as the generation target, e.g., except for the static features. If we mask half (b = 5,9,9 for three datasets) of each regular matrix, the task is to predict the 5 $\times$ 6, 9 $\times$ 8 and 9 $\times$ 21 incomplete features of the learning matrix in three datasets, sequentially.

Parameter setting. All the experiments are conducted on a Tesla P4 GPU with Python 3.6 and Keras 2.25 + Tensorflow 1.15.0. For the baselines and our AS-SAN model, we use the same value for those critical hyperparameters. In our work, the initialization parameters (learning rate, weight, etc.) use the default setting in Keras. To avoid overfitting, we use GELU as the activation function of the FNN. All the parameters are detailed in Table 3.

5.3. Evaluation metrics

In our experiment, three different metrics are used in the performance evaluation, detailed as follows.

Coefficient of determination (R^2). R^2 is used to measure the accuracy of the generated feature value. Normally, the value range of R^2 is [0, 1] and the closer the R^2 value is to 1, the better the fit between the model prediction and the true value. R^2 is computed as:

$$R^2 = 1 - \frac{\sum_{i=1}^{K}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{K}(y_i - \bar{y}_i)^2}. \quad (22)$$

Mean absolute error (MAE) and **Root mean square error (RMSE)** are used to measure the deviation of the prediction results. Different from R2, the higher this value of MAE and RMSE gets, the worse the model is. MAE and RMSE are defined as:

$$MAE = \frac{1}{K}\sum_{i=1}^{K}|\hat{y}_i - y_i|. \quad (23)$$

Table 3
Training parameter setup.

Para.	Conf.	Para.	Conf.
Dropout [16]	0.1	Optimizer	Adam [24]
Attention head	8	FFN Hidden layer	128
γ	$d + 1$	Active function	GELU [15]
N	4	Loss function	MSE
κ	8	ϵ offset in LN	1e−05

$$RMSE = \sqrt{\frac{1}{K}\sum_{i=1}^{K}(\hat{y}_i - y_i)^2}. \tag{24}$$

In Eq. (22)–(24), K represents the size of the test dataset, $\hat{y}$ is the prediction and y is the true value.

5.4. Results and Discussion

5.4.1. Long sequences learning feature generation

In this section, we conduct an evaluations on the WorldUC dataset (b = 5) and Junyi dataset (b = 9) to discuss the following question:

- **RQ1**: Does the proposed AS-SAN model achieve state-of-the-art performance? Can it deal with long sequences learning feature generation?

We first compare all the baselines and our AS-SAN models on the WorldUC dataset. With a setting (γ = 6) on Stride-AS-SAN and Joint-AS-SAN, the results in Fig. 6 show that the R^2 result of HA is close to 0, which means the results have no referential significance. ARIMA only has a certain effect on very few elements at the beginning. These expose the weakness of the methods which exclusively consider the relation of historical statistic values and ignore the complicated long-term dependency. Through the effective collection of local information using a convolution layer, good results are achieved for the first few predictions in SOCNN. However, with the gradual lengthening of the sequence, its performance also declines rapidly. The difference in the prediction effect for P-RNN, SAN and the proposed SA-SAN are not significant. We speculate that the reason for this is that the prediction target area of WorldUC is a matrix of size 6 × 5, and its sequence length is relatively short. Therefore, in order to further verify the effect of these models, we compare the four methods (P-RNN, SAN, Stride-AS-ASN(γ = 21) and Joint-AS-ASN(γ = 21)) of the Junyi dataset, and the results are shown in Fig. 7.

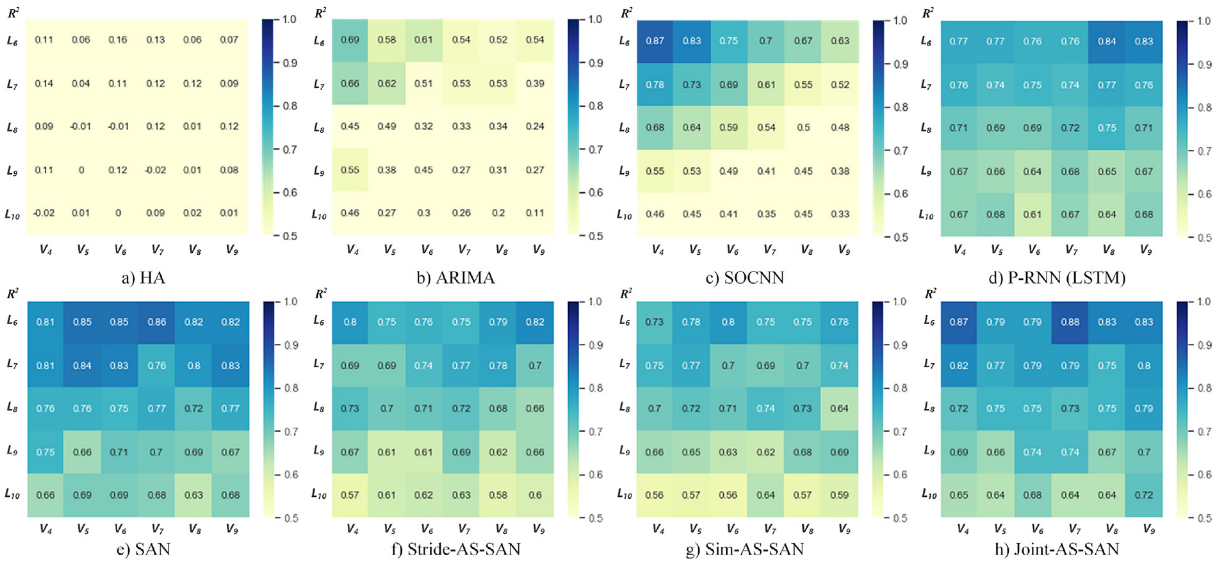


Fig. 6. Visualization of learning feature generation on WorldUC. Darker colors represent better results.

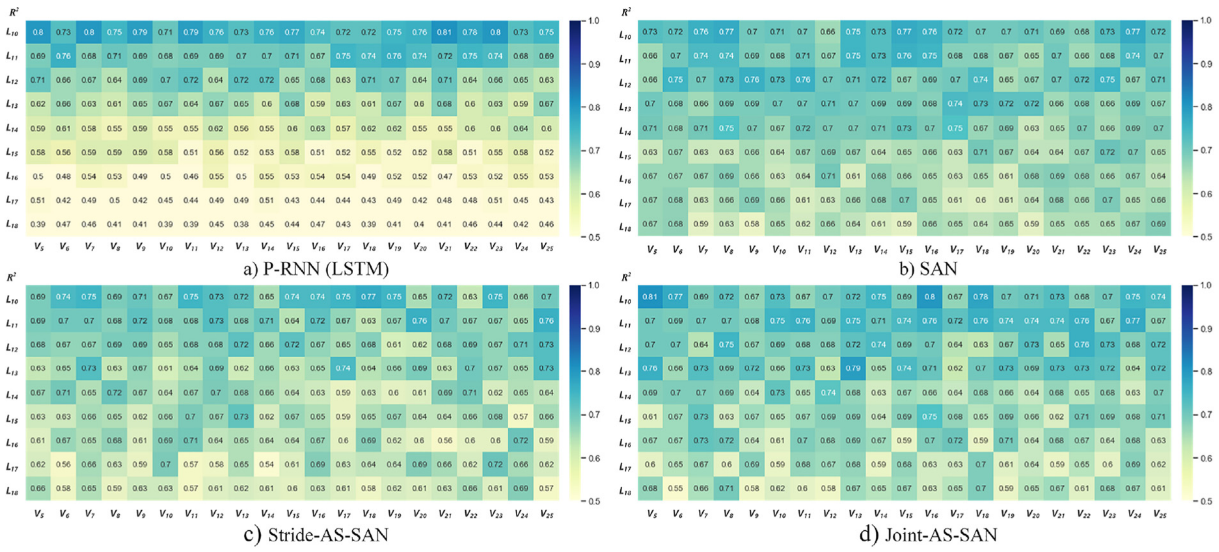


Fig. 7. Visualization of learning feature generation on Junyi. Darker colors represent better results.

Fig. 7 compares the prediction results on long sequence generation. Although P-RNN has a bi-directional network structure and gating mechanisms, a feature sequence which is too long will affect the prediction effect of bi-LSTM. The prediction results of P-RNN are even better than SAN in the early stage, but when the prediction length increases, the decrease in the prediction results is clearly faster than SAN. When only limited information is collected, Stride-AS-SAN, is not as effective as regular SAN and Joint-AS-SAN in feature generation, but it still outperforms P-RNN. The results of regular SAN and Joint-AS-SAN on learning feature generation are quite similar, and achieve significant performance improvements compared with the other baselines for long sequence generation.

5.4.2. Fine-grained student performance prediction

We conduct the performance prediction evaluation on three datasets to answer the following question.

- **RQ2**: What is the student performance prediction effect of our AS-SAN model?

To show the deviation for the baselines and proposed methods on different stages of performance prediction, we use MAE and RMSE to measure their deviation for performance prediction on three datasets with a fixed *b*. The evaluation results are shown in Table 4.

The experimental results show that traditional statistic time-series prediction methods (HA, ARIMA) are significantly less effective. SOCNN and P-RNN achieved non-trivial improvement compared to traditional time-series methods. However, with the limitation of capturing long-term correlations, their performance failed to improve further. Consistent with the conclusion in the previous section, the SAN-based methods have the best effect compared with the other methods. Comparing the results of SAN and AS-SAN, for long-term feature generation, regular SAN has more advantages, because it can capture global features with more information for prediction. However, for short-term feature generation, Joint-AS-SAN has a better effect. We speculate that the reason for this is that the noise in the data is reduced to some extent by filtering out irrelevant features.

We also tried different mask settings of 25%, 50% and 75% for the testing dataset to evaluate the result of all the baselines and our models, respectively. The results of the three datasets are shown in Figs. 8–10. In general, the results are consistent with the results of the previous experiments. It can be seen that, for all three datasets, the lower the mask setting of each dataset, the lower the value of MAE and RMSE. This also shows that a prediction made in a later stage of a course will be slightly better than predictions made in an earlier stage.

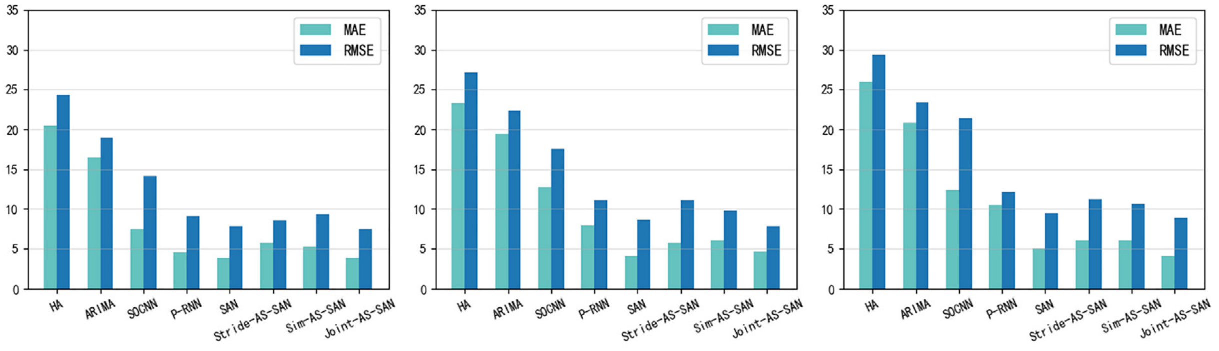
5.4.3. Model efficiency comparison

In this part, we describe the evaluation between regular SAN and the proposed methods to answer the following question:

Table 4
Comparison of student performance prediction.

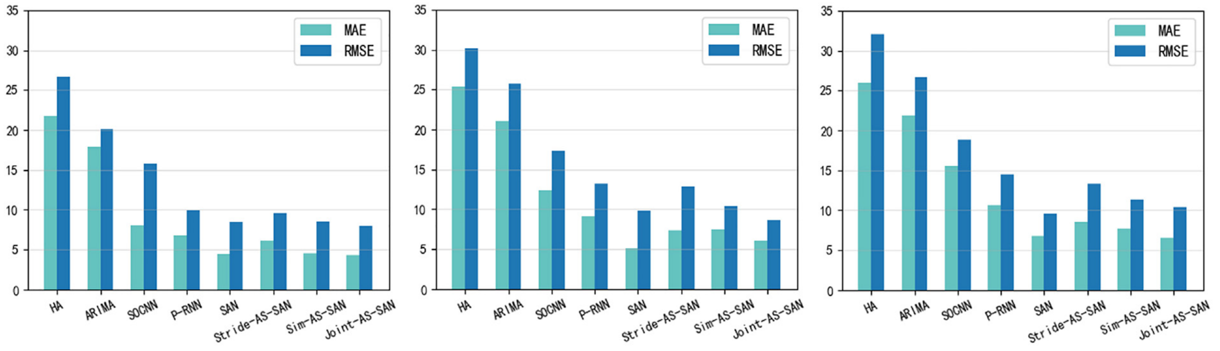
Datasets	Metrics	Feature Index*	HA	ARIMA	SOCNN	P-RNN	SAN	Stride-AS-SAN	Sim-AS-SAN	Joint-AS-SAN
	MAE	@(t = 3,f = 6)	20.47	16.56	7.51	4.61	3.97	5.75	5.32	**3.91**
WorldUC		@(t = 5,f = 6)	–	17.5	13.53	6.53	**4.38**	6.85	5.86	4.63
(b = 5)	RMSE	@(t = 3,f = 6)	23.81	19.94	14.26	9.21	7.84	10.69	9.55	**7.5**
		@(t = 5,f = 6)	–	21.26	15.2	11.43	**9.49**	11.08	10.45	9.82
	MAE	@(t = 3,f = 8)	22.4	12.52	10.89	8.73	6.88	7.02	7.46	**6.87**
		@(t = 5,f = 8)	–	15.76	11.72	10.11	**7.66**	9.43	9.92	7.68
Liru		@(t = 9,f = 8)	–	17.74	14.51	12.04	**10.06**	11.72	11.43	10.92
(b = 9)	RMSE	@(t = 3,f = 8)	25.95	12.52	12.04	9.63	8.23	7.74	7.59	**7.73**
		@(t = 5,f = 8)	–	17.59	13.04	11.23	**8.6**	9.68	9.52	8.9
		@(t = 9,f = 8)	–	19.47	17.17	15.42	**14.05**	15.07	15.22	14.82
	MAE	@(t = 3,f = 21)	21.25	15.05	12.35	9.98	9.33	8.78	9.77	**8.55**
		@(t = 5,f = 21)	–	17.66	13.25	12.74	9.24	9.63	10.2	**8.82**
Junyi		@(t = 9,f = 21)	–	19.93	15.54	13.95	**10.98**	11.21	11.34	11.06
(b = 9)	RMSE	@(t = 3,f = 21)	24.89	20.42	17.8	16.67	11.69	11.47	11.63	**11.06**
		@(t = 5,f = 21)	–	21.64	19.4	18.91	12.43	13.78	14.69	**12.03**
		@(t = 9,f = 21)	–	23.35	21.45	19.31	**15.07**	15.64	16.37	15.66

* Feature index refers to the positions of the generated feature matrix which are the target of student performance prediction. The best results for each metric (row) are bolded.



a) 25% mask(b=3) b) 50% mask(b=5) c) 75% mask(b=8)

Fig. 8. The performance on WorldUC with varying mask ratios.



a) 25% mask(b=5) b) 50% mask(b=10) c) 75% mask(b=15)

Fig. 9. The performance on Liru with varying mask ratios.

- **RQ3**: Compared to regular self-attention networks, by how much can efficiency be improved using the methods proposed in this study?

The evaluation results, as shown in Fig. 11. The result demonstrates a much better efficiency of our proposed method than the regular SAN. This is because our method utilizes a correlation-based feature selection method and adopts a factorized attention for prediction, reducing the number of SAN parameters. Of the three varieties of AS-SAN, Stride-AS-SAN and Sim-AS-SAN incur less time compared to Joint-AS-SAN. However, this significantly reduces their prediction accuracy. Therefore, considering the tradeoff between effectiveness and efficiency, Joint-AS-SAN is more suitable for fine-grained performance prediction.

5.4.4. Case study on the application of fine-grained performance prediction

To clearly describe fine-grained student prediction in real-world application scenarios, we explore the following question.

- **RQ4**: What are the application scenarios for fine-grained student performance prediction results?

In this section, four online course learners are selected as cases, and their learning trends are shown in Fig. 12. The red dotted line shows the trend of the learners based on the results of the pre-test and post-test for conventional learning, and the blue dots and connections represent fine-grained student performance prediction results.

Compared with traditional student performance prediction which only predicts the final score, more implicit patterns can be found to help learners attain higher achievements through fine-grained student performance prediction. Through the prediction of fine-grained learning achievements, richer learning services can also be carried out in a more accurate way. For instance, (i) early warning (Fig. 12(a)). Based on the prediction results, it is much easier to find the downward stage of the learning trend for a student, which is the best timing for an early warning; (ii) weak point forecasting (Fig. 12(b)). Each lesson corresponds to different knowledge points. Using fine-grained learning prediction, it is able to assist in detecting the weak point of these learners, and they can be given some helpful suggestions or recommendations; (iii) abnormal recognition(Fig. 12(c)). Anomaly detection discovers an uncommon state, e.g., abnormal performance or fluctuations in the learning trend, which indicates the learner is having problems or external factors are affecting the learner. Therefore, through fine-grained learning prediction, we can make effective prediction for each lesson in an online course from a new perspective, as the basis for early warning, anomaly detection and weak point forecasting, to ensure more effective learning.

6. Conclusions

In this paper, we propose an adaptive sparse self-attention network for fine-grained student performance prediction. Technically, we treat the problem of interpretable student performance prediction in an online course as an autoregressive feature generation task, and take into account the domain adaptation by stacking an adaptive feature selection layer and several adaptive sparse self-attention layers to build the deep learning model. Empirical validations on three real-world datasets collected from different e-learning platforms show that our method leads to better prediction results and can potentially provide valuable information with regard to the formative assessment and personalized feedback during the online learning process. Both the algorithmic design and the experimental studies confirm the possibility of its application to real educational environments. We believe this work could result in more efficient learning by increasing student engagement and/or preventing student dropout, in diverse settings in e-learning platforms possibly.

Future research can be extended in the following directions: (i) our current framework can be extended to generalize well and quickly from one course to another (courses may vary in length, content, format, etc.) across different learning programs; (ii) an advanced model can be developed that can deal with multimodal educational data and which has good potential in measuring the learner's engagement in individual/collaborative conditions and predicting an individual learner's performance in more diverse set-ups within the e-learning environment; and (iii) a pre-training strategy of our proposed AS-SAN model can be designed which would further boost the prediction process by fine-tuning on a given target course. Similar to this line of research, it is interesting to combine our framework with transfer learning, aiming to build a general pipeline for real-time student performance prediction with domain adaptation.

CRediT authorship contribution statement

Xizhe Wang: Conceptualization, Methodology, Writing - original draft. **Xiaoyong Mei:** Formal analysis, Investigation. **Qionghao Huang:** Software, Visualization. **Zhongmei Han:** Data curation, Validation. **Changqin Huang:** Supervision, Writing - review & editing, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (No.61877020, 61802132), the Science and Technology Projects of Guangdong Province, China (No.2018B010109002), and the Science and Technology Project of Guangzhou Municipality, China (No.201904010393).

References

[1] N. Armar, A. Vaswani, J. Uszkoreit, Ł. Kaiser, N. Shazeer, A. Ku, D. Tran, Image transformer, in: Proceedings of the International Conference on Machine Learning, 2019, pp. 4055–4064.

[2] J.L. Ba, J.R. Kiros, G.E. Hinton, Layer normalization, arXiv preprint arXiv:1607.06450, 2016.

[3] S. Balaji, T. Kavya, N. Sebastian, Learn-able parameter guided Activation Functions, arXiv preprint arXiv:1912.10752, 2019.

[4] D. Baneres, M.E. Rodríguez Gonzalez, M. Serra, An early feedback prediction system for learners at-risk within a first-year higher education course, IEEE Trans. Learn. Technol. 12 (2) (2019) 249–263.

[5] M.R. Berthold, F. Höppner, On clustering time series using euclidean distance and pearson correlation, arXiv preprint arXiv:1601.02213, 2016.

[6] M. Bińkowski, G. Marti, P. Donnat, Autoregressive convolutional neural networks for asynchronous time series, in: Proceedings of 35th International Conference on Machine Learning, 2018, pp. 580–589.

[7] A.F. Botelho, A. Varatharaj, T. Patikorn, D. Doherty, S.A. Adjei, J.E. Beck, Developing early detectors of student attrition and wheel spinning using deep learning, IEEE Trans. Learn. Technol. 12 (2) (2019) 158–170.

[8] C. Burgos, M.L. Campanario, D. de la Peña, J.A. Lara, D. Lizcano, M.A. Martínez, Data mining for modeling students' performance: a tutoring action plan to prevent academic dropout, Comput. Electr. Eng. 66 (2018) 541–556.

[9] J.Y. Campbell, S.B. Thompson, Predicting excess stock returns out of sample: can anything beat the historical average?, Rev Financial Stud. 21 (4) (2008) 1509–1531.

[10] R.L.U. Cazarez, C.L. Martin, Neural networks for predicting student performance in online education, IEEE Latin Am. Trans. 16 (7) (2018) 2053–2060.

[11] H.S. Chang, H.J. Hsu, K.T. Chen, Modeling exercise relationships in e-learning: a unified approach, in: Proceedings of the 8th International Conference on Education Data Mining, 2015, pp. 532–535.

[12] R. Child, S. Gray, A. Radford, I. Sutskever, Generating long sequences with sparse transformers, arXiv preprint arXiv:1904.10509, 2019.

[13] P. De Barba, G.E. Kennedy, M. Ainley, The role of students' motivation and participation in predicting performance in a MOOC, J. Comput. Assisted Learn. 32 (3) (2016) 218–231.

[14] J. Devlin, M.W. Chang, K. Lee, K. Toutanova, Bert: pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the North American Chapter of the Association for Computational Linguistics, 2018, pp. 4171–4186.

[15] D. Hendrycks, K. Gimpel, Gaussian error linear units (GELUs), arXiv preprint arXiv:1606.08415, 2016.

[16] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, arXiv preprint arXiv:1207.0580, 2012.

[17] M. Holmes, A. Latham, K. Crockett, J.D. O'Shea, Near real-time comprehension classification with artificial neural networks: decoding e-learner non-verbal behavior, IEEE Trans. Learn. Technol. 11 (1) (2017) 5–12.

[18] Q. Hu, Z. Han, X. Lin, Q. Huang, X. Zhang, Learning peer recommendation using attention-driven CNN with interaction tripartite graph, Inf. Sci. 479 (2019) 231–249.

[19] C.Z.A. Huang, A.V.J.U.N. Shazeer, M.D.D. Eck, Music transformer: generating music with long-term structure, arXiv preprint arXiv:1809.04281, 2018.

[20] M. Huang, A. Liu, N.N. Xiong, T. Wang, A.V. Vasilakos, An effective service-oriented networking management architecture for 5G-enabled internet of things, Comput. Netw. 107208 (2020).

[21] T. Huang, Y. Mei, H. Zhang, S. Liu, H. Yang, Fine-grained engagement recognition in online learning environment, in: Proceedings of the 9th IEEE International Conference on Electronics Information and Emergency Communication, IEEE, 2019, pp. 338–341.

[22] M. Jang, S. Seo, P. Kang, Recurrent neural network-based semantic variational autoencoder for sequence-to-sequence learning, Inf. Sci. 490 (2019) 59–73.

[23] K. Kime, T. Hickey, R. Torrey, The calculus dashboard-leveraging intelligent tutor techniques to provide automated fine-grained student assessment, in: Proceedings of the 2017 IEEE Frontiers in Education Conference, 2017, pp. 1–8.

[24] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, in: Proceedings of the International Conference on Learning Representations, 2015, pp. 1–15.

[25] E.A. Kolog, S.N.O. Devine, K. Ansong-Gyimah, R.O. Agjei, Fine-grained affect detection in learners' generated content using machine learning, Educ. Inf. Technol. 24 (6) (2019) 3767–3783.

[26] Y. Li, Z. Zhu, D. Kong, H. Han, Y. Zhao, EA-LSTM: Evolutionary attention-based LSTM for time series prediction, Knowl.-Based Syst. 181 (2019), 104785.

[27] H. Lin, W. Jia, Y. Sun, Y. You, Spatial-temporal self-attention network for flow prediction, arXiv preprint arXiv:1912.07663, 2019.

[28] J. Lin, X. Sun, X. Ren, M. Li, Q. Su, Learning when to concentrate or divert attention: self-adaptive attention temperature for neural machine translation, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 2985–2990.

[29] T. Liu, K. Wang, L. Sha, B. Chang, Z. Sui, Table-to-text generation by structure-aware seq2seq learning, in: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, 2018, pp. 4881–4888.

[30] W. Lu, W. Zhou, D. Shan, L. Zhang, J. Yang, X. Liu, The linguistic modeling of interval-valued time series: a perspective of granular computing, Inf. Sci. 478 (2019) 476–498.

[31] A.v.d. Oord, N. Kalchbrenner, K. Kavukcuoglu, Pixel recurrent neural networks, in: Proceedings of the International Conference on Learning Representations, 2016, pp. 1747–1756.

[32] C. Pei, Y. Zhang, Y. Zhang, F. Sun, X. Lin, H. Sun, J. Wu, P. Jiang, J. Ge, W. Ou, et al, Personalized re-ranking for recommendation, in: Proceedings of the 13th ACM Conference on Recommender Systems, 2019, pp. 3–11.

[33] Y. Peng, Y. Fang, Z. Xie, G. Zhou, Topic-enhanced emotional conversation generation with attention mechanism, Knowl.-Based Syst. 163 (2019) 429–437.

[34] A. Polyzou, G. Karypis, Feature extraction for next-term prediction of poor student performance, IEEE Trans. Learn. Technol. 12 (2) (2019) 237–248.

[35] J. Qiu, J. Tang, T.X. Liu, J. Gong, C. Zhang, Q. Zhang, Y. Xue, Modeling and predicting learning behavior in MOOCs, in: Proceedings of the 9-th ACM International Conference on Web Search and Data Mining, 2016, pp. 93–102.

[36] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, OpenAI Blog 1 (8) (2019) 9.

[37] S.E. Said, D.A. Dickey, Testing for unit roots in autoregressive-moving average models of unknown order, Biometrika 71 (3) (1984) 599–607.

[38] T. Shen, T. Zhou, G. Long, J. Jiang, S. Wang, C. Zhang, Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018, pp. 4345–4352.

[39] N. Tomasevic, N. Gvozdenovic, S. Vranes, An overview and comparison of supervised data mining techniques for student exam performance prediction, Comput. Educ. 143 (2020), 103676.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Proceedings of 2017 Annual Conference on Neural Information Processing Systems, 2017, pp. 5998–6008.

[41] H. Wan, K. Liu, Q. Yu, X. Gao, Pedagogical intervention practices: Improving learning engagement based on early prediction, IEEE Trans. Learn. Technol. 12 (2) (2019) 278–289.
[42] D. Wang, M. Li, Stochastic configuration networks: fundamentals and algorithms, IEEE Trans. Cybern. 47 (10) (2017) 3466–3479.
[43] X. Wang, P. Wu, G. Liu, Q. Huang, X. Hu, H. Xu, Learning performance prediction via convolutional GRU and explainable neural networks in e-learning environments, Computing 101 (6) (2019) 587–604.
[44] J. Xu, K.H. Moon, M. Van Der Schaar, A machine learning approach for tracking and predicting student performance in degree programs, IEEE J. Sel. Top. Signal Process. 11 (5) (2017) 742–753.
[45] B. Yang, Z. Tu, D.F. Wong, F. Meng, L.S. Chao, T. Zhang, Modeling localness for self-attention networks, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 4449–4458.
[46] T.Y. Yang, C.G. Brinton, C. Joe-Wong, M. Chiang, Behavior-based grade prediction for MOOCs via time series neural networks, IEEE J. Sel. Top. Signal Process. 11 (5) (2017) 716–728.
[47] J.W. You, Identifying significant indicators using LMS data to predict course achievement in online learning, Internet Higher Educ. 29 (2016) 23–30.
[48] G. Zhao, J. Lin, Z. Zhang, X. Ren, Q. Su, X. Sun, Explicit sparse transformer: concentrated attention through explicit selection, arXiv preprint arXiv:1912.11637, 2019.