IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 19 December 2022, accepted 5 February 2023, date of publication 16 February 2023, date of current version 28 February 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3246122

RESEARCH ARTICLE

Transformer Encoder Model for Sequential Prediction of Student Performance Based on Their Log Activities

SRI SUNING KUSUMAWARDANI, (Member, IEEE), AND SYUKRON ABU ISHAQ ALFAROZI, (Member, IEEE)
Department of Electrical and Information Engineering, Universitas Gadjah Mada, Yogyakarta 55281, Indonesia
Corresponding author: Sri Suning Kusumawardani (suning@ugm.ac.id)

ABSTRACT Learning management systems (LMSs) have been used massively due to the growing utilization of distance learning. This advancement has led to increased educational data that can be analyzed to improve the quality of the learning process. Learning analytics (LA) is one of the most important methods that can be used to analyze student performance. In this paper, we proposed an LA method based on deep learning, i.e., transformer encoder, to sequentially predict the student's final performance based on log activities provided by an LMS. The objective is to predict at-risk students of failing so that they can be mitigated as soon as possible. The proposed model was evaluated on the Open University LA Dataset (OULAD) for daily or weekly prediction. The results show that the model could predict at the early stage with an accuracy of 83.17% on withdrawn versus pass-distinction classes. Meanwhile, for the other tasks, i.e., withdrawn-fail versus pass-distinction and fail versus pass-distinction tasks, the accuracy was at least 76% at the early stage. The proposed model was compared to the LSTM model. We found that the transformer encoder performed better than the LSTM, with the average difference values from 1% to 3% in terms of accuracy and from 3% to 7% in terms of F1-score for all tasks, based on the statistical testing. Furthermore, the ablation study using positional encoding, different feature aggregation methods, and weighted loss function for the imbalanced class problem was conducted. In OULAD, we found that model without positional encoding was better in all cases. Furthermore, the weekly feature aggregation and the use of a weighted loss function performed better in some cases.

INDEX TERMS Learning analytics, transformer encoder, student at-risk prediction, massive open online courses, sequential model, imbalanced dataset.

I. INTRODUCTION

The fast development of technology has impacted to our educational institutions to adopt and develop a sustainable educational system. In recent years, we have faced an enormous transformation of our education system due to the coronavirus disease (COVID-19) that led colleges, schools, and other education systems to conduct distance learning. Massive open online courses (MOOCs) are one of the distance learning technologies. With this technology, everyone can massively access course resources from anywhere and anytime. However, an online course system has several challenges. For instance, it is hard to maintain student engagement during a class activity, which is one of the important learning outcome factors [1]. Moreover, it is challenging to monitor at-risk students in a particular class due to the lack of interaction between the teacher and the student. Predicting at-risk students is important because it allows educators and policymakers to identify students who may be struggling academically or facing challenges outside of the classroom that could impact their academic performance. Early identification of at-risk students allows for interventions to be implemented, which can improve academic outcomes and prevent academic failure. Hence, it is very crucial to tackle

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Asif.

the weaknesses in MOOCs. Fortunately, with the advantage of technology in the education system, we can alleviate the weaknesses of an online course using learning analytics (LA). LA finds factors related to student performance so that we can predict, mitigate, and analyze the performance of students according to their activities during the course period. Thus, the course outcomes and student performances can be maintained and improved. Student performance prediction is important so that proper follow-up actions and mitigation could be conducted to help students who are in-need [2]

A blending technology of MOOCs and LA is fundamental to conducting an intelligent tutoring system (ITS). The system provides personalized learning experiences depending on the need of the students according to the recommendations of the LA. The implementation of LA is generally based on traditional machine learning and statistical models [3], [4], [5], [6], [7]. However, the merging of deep learning model [8], which is adopted on various applications in medical imaging [9], telecommunication [10], economics [11], also become a new standard to be adopted in LA [12], [13], [14].

In this paper, we focus on the application of LA, particularly student performance prediction, based on the activity data logs using one of the deep learning models, i.e., transformer encoder [15], for a publicly available benchmark LA dataset, Open University LA Dataset [16]. The dataset has several activity data that can be used for modeling an LA problem. We used this dataset for the sequential performance prediction from the student daily activities. The model predicts the final performance of students, i.e., withdrawn, failed, passed, or distinction, based on the student interactions with the LMS, either for course resource click streams or assignment activities. Student at-risk prediction (failed or withdrawn) is an important aspect of ITSs. It can mitigate and treat students based on their performance with proper measurements.

The sequential prediction of student performance on the OULAD is based on recurrent neural networks (RNNs) such as long short-term memory (LSTM) [13] and gated recurrent unit (GRU) [17]. These models are primarily utilized in natural language processing (NLP) tasks and have a memory state that can be used for selectively remembering or forgetting past information. However, time dependence makes these models slow to train and sophisticated. A Transformer [15] was designed to overcome the problem by replacing the hidden state memory with the attention layer so that the model can be trained using a parallel computation similar to an ordinary fully connected network with the memory retained in the attention layer. Therefore, we propose a method for predicting at-risk students using a transformer encoder on the OULAD. There are three tasks of predicting at-risk students, i.e., withdrawn-fail versus pass-distinction, withdrawn versus pass-distinction, and fail versus pass-distinction. The tasks are to predict at-risk students on each day or week, given all the past activities. Thus, a student might be predicted as a non-at-risk student in the early period of learning. However, the change in activity sequences in future features might change the prediction based on the pattern of what the model learns. We summarize our work as follows:

- An alternative model, i.e., transformer encoder, that is more accurate than the recurrent model, was proposed.
- A model structure in the transformer encoder was investigated so that the model can perform better with the OULAD.
- The effect of the preprocessing method, i.e., feature aggregation and the use of the weighted loss function, was compared so that we can use the appropriate model for the OULAD.

In addition, one can use the same idea of implementing the sequential prediction based on the student activities as long as the data from LMS can be formed as sequential feature vectors using the transformer encoder. From the experimental results, the transformer encoder without positional encoding performed better than the recurrent model, i.e., LSTM, on all tasks.

II. RELATED WORKS

The OULAD has become one of the standard benchmarks for the LA problem on predicting at-risk students on online courses based on their log activities. The problems that can be formulated in OULAD can be further implemented in other e-learning platforms to track student performance. Many researchers identified several problems on how to utilize the dataset. There were two main problems on the OULAD, i.e., regression of the final scores [13] and classification of student performance. Withdrawal prediction is the most common issue in the OULAD, which is a binary classification problem for predicting student withdrawal or not. For instance, Poitras et al. [18] predicted whether students tend to be withdrawn or not on a particular module (CCC2013B) based on their interaction activities until 40% of the total assignments are submitted. Similarly, other works [5], [14], [19], [20], [21] also worked with a withdrawal prediction either of a particular course or all available courses in the OULAD. The identification of the withdrawal problem is important because dropout is one of the most common problems in MOOCs. Thus, one can mitigate the students based on student performance prediction. By contrast, a multi-class classification problem was used to predict the final results of students, i.e., withdrawn, fail, pass, and distinction on some modules [22] based on the student demographic feature only. The Demographic feature has the advantage of predicting student performance even before a class starts. However, this prediction might lead to demographic bias. Different with [22], in this study, we focused on the student interaction activity feature only, where student performance is predicted purely based on their interaction with the learning system.

A variety of machine learning models are used to predict student performance on the OULAD. For instance, Hussain et al. [23] used a decision tree and its variants for

predicting student engagement on the OULAD. The study formulated an engagement based on the student's first assessment score, final student performance, and student interaction (number of clicks). This engagement class is predicted using a classification model, i.e., a decision tree which is a classifier that can be easily interpreted. Moreover, the study extracted the important features of factors affecting most to student engagement. The main focus of this study is a factor identification that affects student engagement. Heuer and Breiter [21] used a support vector machine (SVM) to predict the student performance using the activity features in a tabular form, such that the column of the table feature is the aggregate value of the i-th day and the row is the instance of a student. The authors compared several machine learning models, such as a decision tree, logistic regression, and random forests. However, this study utilized the whole feature set to predict the student performance, so it is hard to identify students that tend to be dropped out from a particular course in the early or middle of the course. Moreover, this study showed the importance of activity features compared to demographic features. The use of demographic features along with activity features has an insignificant effect on the model performance. Hence, we used the activity features for sequentially predicting the performance either on a daily or weekly basis.

The important thing to implement in a sequential prediction is the ability of a model to update its knowledge based on sequential inputs, i.e., in this case, daily or weekly activity features. RNN model can overcome this problem, such as LSTM or GRU. For example, Hassan et al. [14] utilized an LSTM model to predict student performance (withdrawn versus pass-distinction) on a weekly basis based on student activity features. Thus, students that tend to be dropped out can be sequentially predicted. Another work in sequential prediction used a GRU model [17]. The authors used all features available in the OULAD, such as demographics, assessments, and activities, to predict at-risk students (fail versus pass-distinction). In this case, the authors found that the GRU model performed better than the LSTM. Both studies focused on the early prediction of the at-risk student either failing or withdrawing. However, the main overhead of the recurrent model is a time-dependent computation which makes the training slow. Another approach to overcome the computational overhead in a sequential model is to use the attention methods such as the transformer model [15].

A transformer architecture helps the model bypass the recurrent mechanism so that it can be faster either in training or testing. It also solves the gradient and information flow problems when a deep recurrent model is used. One of the transformer models used in LA is called Self-AttentIve Neural Knowledge Tracing (SAINT) [12]. This model was proposed for knowledge tracing to model a student's knowledge through learning activities. Choi et al. [12] proposed the knowledge tracing SAINT model to predict the response $r_i \in \{0, 1\}$ of the next exercise question, either it is correct (1) or incorrect (0) from the set of exercise questions given interactions $I_i = \{E_i, R_i\}$ where R_i is the student response r_i to exercise information E_i. The SAINT transformer model consists of two parts i.e., an encoder for processing the interactions I_i and a decoder for processing the output of the encoder and the sequence of exercise information E_i. Furthermore, Shin et al. [24] proposed the SAINT+ model that added two temporal features i.e., elapsed time (time taken for answering a question) and lag time (time interval between adjacent learning activities). The model improved the performance of the successor model SAINT around 1% either for accuracy or area under the curve (AUC) metric. However, SAINT focuses on a single activity, such as a quiz activity, but it is not for the whole course activity.

In this paper, we propose a transformer encoder model only for predicting at-risk students based on their log activities. The activities were obtained from the whole course interaction not only from a particular exercise or activity. The encoder used a masking method so that the model could not see the next input information. Thus, our model is a simple version of the transformer architecture while still concerning high-performance accuracy.

III. OULA DATASET

This section introduces the OULAD which is one of the most popular open datasets in LA [16]. An open dataset can be used as a benchmarking and standardizing dataset, so we can compare and improve the result of models. There are many similar datasets such as HarvardX, MITx, and Coursera Forum. However, OULAD has relatively more features than other datasets in the case of the completeness of features. It consists of log data, such as registrations, demographics, assignments, and interactions data, from 32,593 students. This dataset is mainly used for LA to determine the relationship between the log activities of students and their personal information and learning performances.

The OULAD was gathered from student activity records from 2013 to 2014 of several courses, i.e., social sciences and science, technology, engineering, and mathematics (STEM) by the Open University. It consists of seven courses, which are called modules. Each module was presented at different times, i.e., B for February and J for October. For example, 2013J means that data were gathered from a module presentation that started on October 2013. The modules were coded as from AAA to GGG, as described in Table 1.

In short, the OULAD consists of seven data tables as depicted in Fig. 1. These tables were available in a comma separated value format. The tables can be distinguished into three groups of features, i.e., demographics features which contain the student detail information, course data features which contain the course detail information, and activity features which contain the activity log features that mainly used in this work as squential student performance prediction either daily or weekly.

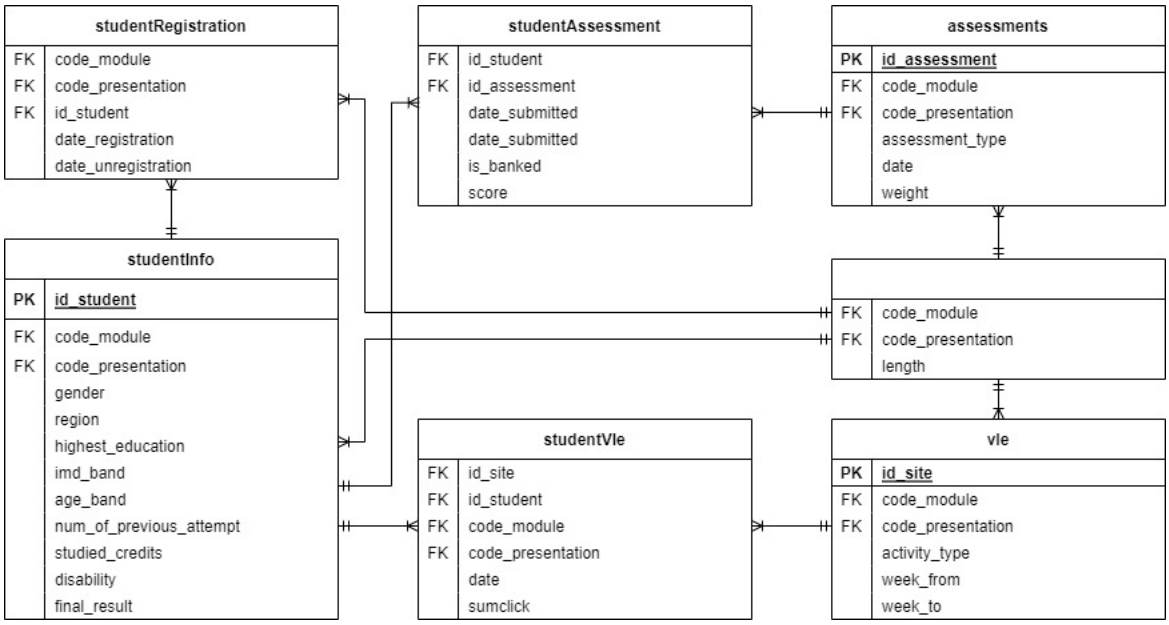

FIGURE 1. Entity relationship diagram of the OULAD.

A. DEMOGRAPHICS FEATURES

These features consists of `studentRegistration` and `studentInfo` tables. The `studentRegistration` table describes when the students were registered and unregistered to a particular course module. The information about a student's name, age, highest education, and region, among other, are included in the `studentInfo` table as depicted in Fig. 1. It also contains the final results which explain whether a student has a fail, dropout, pass, or distinction status of a particular course. In our case, most of these features were not used to minimize the bias related to demographic information.

B. COURSE DATA FEATURES

The description of the data can be seen in the `courses`, `assessments`, and `vle` tables. Each course module has each duration in days that were saved in the length field as shown in the courses table. Each course has several assessments consisting of Tutor-Marked Assessment (TMA), Computer-Marked Assessment (CMA) and Final Exam (Exam). Moreover, every course has resources in a virtual learning environment (VLE) where students will interact with (see `vle` table in Fig. 1).

C. STUDENT ACTIVITY FEATURES

The most important features used in our work is the activity features, which are presented in three tables, i.e., `studentRegistration`, `studentVle` and `studentAssessment`. We grouped the `studentVle` and `studentAssessment` as activity features because they were recorded during the course everyday. We can process the data according to the date when the student interacted. Interestingly, in the `studentAssessment` table, an attribute named `is_banked` indicates a particular assessment score transferred from the previous presentation if the value is 1. The VLE consists of 20 activities and the assessment consists of three activities. Thus, 23 activity types are used as activity features. The merging process is described in Section IV-A. Then, based on these features, the final result of each student is predicted without knowing the demographic information to alleviate the bias.

TABLE 1. Code modules based on its domain.

Module	Domain	Presentation	Students
AAA	Social Science	2	748
BBB	Social Science	4	7909
CCC	STEM	2	4434
DDD	STEM	4	6272
EEE	STEM	3	2934
FFF	STEM	4	7762
GGG	Social Science	3	2534
Total	7	22	32,593

In this paper, we mainly focus on two tables, i.e., `studentVle` and `studentAssessment`. From these features, the final result of students will be predicted. The prediction is based on daily/weekly predictions. Thus, we can update whether a student is predicted as at-risk or not based on the current activity of the current and previous days/weeks.

IV. METHOD

This section consists of several steps before feeding the dataset to our proposed model, i.e., preprocessing and model architecture of the transformer encoder used in our work.

A. PREPROCESSING

As stated in Section III, the OULAD mainly consists of three blocks of data, i.e., student personal information, stu-

dent assessment records, and student log activities. In this section, we describe the process of obtaining the activity features.

First, the `studentAssessment` and `assessments` tables were merged together using the left join method based on `id_assessment` to obtain additional information of the assessment activities. Thus, the result of this process is a new table that has several attributes, i.e., `id_student`, `id_assessment`, `code_module`, `code_presentation`, `assessment_type`, `score`, `is_banked`, `date_submitted`, `weight` and `date`. Because we want to merge these assessment activities with VLE activities, the attributes on both tables should be the same. Therefore, some attributes were dropped and renamed. The `date` and `id_assessment` had been removed, and `date_submitted` and `assessment_type` had been renamed to `date` and `activity_type`, respectively. We also added `sum_click` attribute set to 1 value so that it matches to the list of VLE attributes. Moreover, the value of the `weight` attribute had been modified, such as $weight = weight \times score$. We considered this modification due to the use of aggregate features. The final score is calculated as the weighting score of these assessments. In this case, the weighting score is more important than the score and weight individually. Furthermore, if the features were aggregated either on a daily or weekly basis, the weighting score is totally different from the multiplication of aggregated scores and weights.

Second, the `studentVle` and `vle` tables were merged together, similar to the previous process. After merging, several attributes that do not match to the merged assessment table were dropped, i.e., `week_from`,`week_to`, and `id_site`. We added new attributes, i.e., `score`, `is_banked`, and `weight`, so that both tables have the same number of attributes. These attributes were set to zero values as the VLE activities do not have scores or weights, and there is no transfer of activities from the previous presentation. Thus, at the end of these processes both tables have the same number of attributes, i.e., `id_student`, `code_module`, `code_presentation`, `activity_type`, `score`, `is_banked`, `weight`, `sum_click` and `date`. Finally, both tables were concatenated forming one big activity table.

Third, some activities had negative date values and were larger than the course length. Then, we normalized all these data so that all negative date values were set to zero, and all the date values that are larger than the course length were set equal to the course length. Assessments that had `is_banked` attribute set to 1, had negative date values which means that the assessment was performed before the class started. Finally, we aggregated the features based on a daily basis. Therefore, we had four features, excluding the ID attributes, i.e., is banked, score, weight, and sum click. Then, we normalized the attributes where the score was divided by 100, the weight was divided by 10,000 (the maximum possible value), and the sum of clicks was divided by 100.

TABLE 2. Distribution of the class labels.

Label	Number of Students
Withdrawn	7188
Fail	6706
Pass	12360
Distinction	3024
Total	29278

The attribute ID was used to obtain the class label from `studentInfo` table.

Fourth, we rearranged the table so that it can be formed as a multidimensional array with size (N, D, T, F) where $N = 29278$ is the number of students, $D = 270$ is the maximum number of days for each module, $T = 23$ is the number of activity type, and $F = 4$ is the number of features. Here the number of students N is less than that in Table 1, because some of the students did not have any activities. Thus, these students were excluded in our work. The distribution of the class labels is described in Table 2. Furthermore, because we used the transformer model in our task, the activity feature matrix was reshaped with the dimension of (N, D, E), where $E = 92$ is a flattened feature of all activity types. For the weekly features, we aggregated the input from the daily table of each seven days so that the dimension of the D axis became 39 instead of 270.

The Pandas framework was used in this preprocessing technique. Detailed implementation of this preprocessing is available in our repository.

B. TRANSFORMER ENCODER

A Transformer is a novel model that gains popularity in NLP due to the simple mechanism of using a fully connected layer and has a better performance than RNN models such as LSTM or GRU. Moreover, a transformer has better speed due to its parallelism without a chained calculation of the previous hidden state. Basically, this model is a fully connected network with the attention mechanism [15]. The attention mechanism can look back and forth so that the model knows the context of the current input features. There are two main components of the transformer, i.e., the encoder and the decoder. We only utilized a simple encoder to perform our task. This idea is the same as the popular architecture Bidirectional Encoder Representations from Transformers (BERT) [25] to address several tasks, such as classification for sentiment analysis. The encoder or decoder basically consists of a fully connected network with a normalization layer and an attention layer, which is the most important part of the transformer. Thus, the main difference between a transformer and a fully connected network is the existence of the attention layer.

A sequence of input activity features in each day of a student n is denoted as $E_n^l = [e_1^l, e_2^l, \ldots, e_D^l]^T$ on the l-th encoder layer which is a matrix of embedding vectors from the first day to the D-th day. Then, from these sequences of features, the transformer encoder generates the sequence of output features $O_n^l = [o_1^l, o_2^l, \ldots, o_D^l]^T$ through the attention,

normalization, and fully connected layers. A transformer encoder consists of several stacks of encoder layers. In our case, two layers were used. The self-attention operation inside can be described as follows: Query Q, key K, and value V are the inputs of the self-attention layer. In our case, the three inputs were generated from a single input matrix E_n with its own weights, such that,

$$\begin{aligned} Q_i^l &= E_n W_i^q, \\ K_i^l &= E_n W_i^k, \\ V_i^l &= E_n W_i^v. \end{aligned} \tag{1}$$

From (1), the attention operation can be performed with,

$$Z_i^l = \text{softmax}(\frac{Q_i^l {K_i^l}^T}{\sqrt{d}}) V_i^l, \tag{2}$$

where d is a dimension of Q_i and K_i. Eq. (2) weights the matrix value V_i based on all input sequences of E_n according to Q_i and K_i. In our model, we used the multi-head self-attention layer, where several self-attention layers were concatenated h times, such that

$$Z_n^l = \text{concat}(Z_1^l, Z_2^l, \ldots, Z_h^l) W^l, \tag{3}$$

where W^l is the weight to concatenate Z_i^l such that it learns the cross-relation of all individual heads. In practice, a multi-head self-attention layer was used to split the dimensions of Q_i and W_i to $d_{head_i} = d/h$, so the total computational cost is the same, while outputting the same d dimension of Z_n^l in (3). A single head of the self-attention layer learns a single representational space. Thus with the multihead self-attention layer the model can learn from the different information of several representational spaces.

Focusing on (2), the operation inside the softmax function is a full attention operation where the attention of the current input v was evaluated on all other input sequences. In our case, this is a kind of cheating because the prediction of the current state can see all future activities. Thus, the upper triangular matrix masking M was used to prevent the current position from seeing future activities. Accordingly, (2) can be rewritten as

$$Z_i^l = \text{softmax}(\frac{Q_i^l {K_i^l}^T}{\sqrt{d}} + M) V_i^l, \tag{4}$$

where, the value of M is filled with $-\infty$ on the region above the diagonal so the output of this region becomes zero after applying the softmax operation, which prevents attending future activity features.

After passing the self-attention layer, Z_n^l obtained from (4) is forwarded to layer normalization with skip connection and a fully connected layer, such that

$$\begin{aligned} H_n^l &= \text{LayerNorm}(E_n^l + Z_n^l) \\ O_n^l &= \text{LayerNorm}(\text{ReLU}(H_n^l W^n) + H_n^l) \end{aligned} \tag{5}$$

where W_n is the weight of the fully connected layer. Then, O_n^l becomes the input of the next encoder layer. In other words, $E_n^{l+1} = O_n^l$.

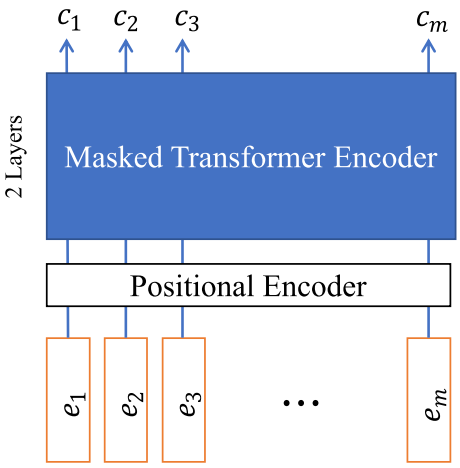

FIGURE 2. Transformer encoder model: e_i is the embedding vector which is 92 features for each day or week and $c_i \in \{0, 1\}$ is the prediction of each day or week.

In this work, two layer transformer encoders with 100 hidden nodes were used. The prediction of the student performance that differentiates between at-risk and non at-risk students is a binary classification task. Therefore, we added the final classification layer using a linear layer, so that it outputs one dimensional output features, such that

$$Y_n = \text{sigmoid}(O_n^L W^y + b^y) \tag{6}$$

where Y_n is a probability sequence at-risk students per each day/week, O_n^L is an output of the last encoder layer, W^y is the corresponding weights, and b^y is the bias. Here, each element of $Y_n = [y_1, y_2, \ldots, y_D]$ can be interpreted as a probability of at-risk students at that day given all their past activities, i.e.,

$$y_k = P(C = 1 | e_1, e_2, \ldots, e_k).$$

The embedding vector e_n^1 was modified using positional encoding using sinusoidal functions similar to the original transformer [15]. Positional encoding helps the model understand the position of the embedding vector because the transformer model is basically a fully connected layer. The transformer encoder model is depicted in Fig. 2

C. EXPERIMENTAL SETTINGS

1) TRAINING DATA

After the preprocessing, the dataset was divided into two sets of data with a ratio of 7:3 using the stratified shuffle method, so we had a proportional sample ratio for each class. The 30% of data were used as testing dataset. The rest 70% of the data were split into training and validation sets with a ratio of 8:2. The training data were used for tuning the parameter of the models on each task. The validation data were used to choose the best model from the training phase based on validation losses. Then, we assessed the performance of the model using the testing data based on the evaluation criteria, i.e., accuracy and F1-score.

2) LOSS FUNCTION

Because the model is used for a binary classification problem, the loss function used in this work is a binary cross entropy loss function described as,

$$\mathcal{L}(T^B, Y^B) = -\frac{1}{BD}\sum_{i=1}^{B}\sum_{j=1}^{D}[p_w t_{ij}\log(y_{ij}) + (1-t_{ij})\log(1-y_{ij})] \quad (7)$$

where, $t_i \in \{0, 1\}$ is the ground truth class, B is the length of a batch, and p_w is the positive class weight for the imbalanced problem.

3) IMBALANCED PROBLEM

The problem with an imbalanced class distribution is that instances of the majority class dominate the influence to the loss function. In this case, we need to balance the influence by tuning the positive class weight p_w in the loss function such that

$$p_w = \frac{\#\text{neg\_class}}{\#\text{pos\_class}} \quad (8)$$

which is the imbalanced ratio (IR) between the negative class and the positive class. Thus, from (8), the loss on the minority class is amplified depending on the p_w as shown in (7).

4) EVALUATION METHODS

The criteria used in our work are accuracy and f1-score. The accuracy can be calculated as,

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \quad (9)$$

where, TP, TN, FP, and FN are the true positive, true negative, false positive, and false negative, respectively. The F1-score which is the harmonic mean of the recall and precision was used to evaluate the performance of a balanced trade-off between the recall and precision values, defined as follows,

$$\mathrm{F}_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \quad (10)$$

where, precision $= TP/(TP + FP)$ and recall $= TP/(TP + FN)$. Semantically, the precision of a particular class is the probability of how the model performs when it votes for a class, whereas recall is the sensitivity of the model for detecting the positive class. If we have a high precision value, then we can highly believe the model prediction of the class. If we have a high recall value, then we can highly ensure that the model can detect that particular class.

Statistical analysis between two models was calculated using a t-test with one sample. The sample distribution is based on the difference of the metric on each time step of the two models. Let us have two models, i.e., a and b, the average difference of these two models is

$$\text{AD}(a, b) = \frac{1}{n}\sum_{i=1}^{n}\text{metric}(a_i) - \text{metric}(b_i), \quad (11)$$

where the number of the time step for daily tasks is 270 and the number of the time step for weekly tasks is 39. In our case, the two-tailed t-test has two hypotheses:

- $\mathcal{H}_0$: The average difference, AD, has zero mean.
- $\mathcal{H}_1$: The average difference, AD, does not have zero mean.

Thus, when the null hypothesis, $\mathcal{H}_0$, is accepted then the performance of the two models is no significantly different. Otherwise, $\mathcal{H}_0$, is rejected, then there is a significant performance difference between the two models. Moreover, If AD has a positive value, the model a performed better than the model b. Otherwise, model b is better than model a, for a negative AD value.

5) TRAINING SCHEME

This work used popular frameworks to train the transformer encoder with two encoder layers and 100 hidden nodes. PyTorch [26] was used as the deep learning framework, and its wrapper, PyTorch Lightning API, was used to organize the code so that it is easy to read and reproduce. The optimization algorithm used in the training phase was ADAM [27] due to its fast convergence performance in our pre-experimental trials. The training phase of each task was divided into two cycles of 20 and 30 epochs, with initial learning rates of 10^{-3} and 10^{-5}, respectively. The learning rate on each cycle was annealed using the one-cycle learning policy [28]. Thus, we trained the model using 50 epochs.

V. RESULTS

Three binary classification tasks are defined based on at-risk label groups, i.e., withdrawn and failed. Passed and withdrawn students are grouped as a single class because these labels describe non at-risk students. The tasks are described as follows:

- Withdrawn-fail (WF) versus pass-distinction (PD), WF-PD task. This task indicates whether the student passed the course (PD) or not (WF). The withdrawn and failed categories are grouped as one category. The at-risk students are defined as those who failed or dropped out.
- Fail (F) versus pass-distinction (PD), F-PD task. This task is the same as WF versus PD but dropped-out students are excluded. Thus, this task focuses on identifying failed students toward students who completed the course.
- Withdrawn (W) versus pass-distinction (PD), W-PD task. This task is the same as WF versus PD, but failed students are excluded. Thus, the focus of this task is the identification of dropped-out students versus the non at-risk student.

In these tasks, we tried to classify failed or withdrawn or both classes to the non at-risk students (PD). A model was created for each task either based on daily or weekly prediction approaches. In the daily model, we can predict a student category daily based on their daily activities. Conversely, in the weekly model, the category of students is predicted

TABLE 3. Performance comparison of different time period between models with a positional encoding module versus without positional encoding module. The overall performance is the average performance of the daily prediction.

Models	Stages and Performance (%)												AD(a, b)	
	20%		40%		60%		80%		100%		Overall			
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
f-pd daily^a	*77.27	*52.25	*82.75	*67.43	*86.87	*75.55	*89.59	*81.53	*91.34	*85.58	*83.58	*67.84	**5.66	**16.52
f-pd daily pe^b	71.09	39.71	75.89	55.98	82.19	66.36	83.55	70.02	83.46	69.79	77.92	51.33		
w-pd daily^a	*82.93	*67.80	*90.27	*83.47	*94.52	*90.88	*96.69	*94.67	*98.14	*97.08	*89.59	*80.82	**4.78	**12.92
w-pd daily pe^b	73.61	55.13	84.70	73.22	90.98	85.03	92.75	88.30	93.25	89.13	84.81	67.90		
fw-pd daily^a	*75.41	*71.50	*82.83	*80.91	*88.23	*86.77	*90.78	*89.93	*93.42	*93.01	*83.03	*81.02	**5.75	**6.37
fw-pd daily pe^b	67.40	64.82	75.99	71.67	83.28	80.48	86.49	84.60	86.85	84.94	77.28	74.65		

*better than the other model based on numerical value. A positive AD indicates that model a is better, otherwise model b is better for a negative AD.
**significantly different based on statistical testing with p-value < 0.01

weekly based on the weekly accumulated activity records of the students.

For compact comparison, we only show the numerical result performance of the models at specific stages or time steps which increased by 20% according to the course duration, e.g., at i-th day or i-th week stages, where $i \in \{20\%, \ldots, 100\%\}$. However, we also provide the overall average performance of the first days/weeks until the end of the course period. In this paper, models were compared using statistical testing based on their architectural settings. We compared several aspects as follows:

- the use of positional encoding module,
- the effect of aggregation method, i.e., daily versus weekly feature aggregation,
- the effect of the weighted loss function to the imbalanced problem, and
- the comparison with the recurrent model, LSTM.

A. THE USE OF POSITIONAL ENCODING

In this case, the models with positional encoding were compared to the models without positional encoding (PE). The model without positional encoding performed better than the model with positional encoding for the three tasks, as shown in Table 3. For the F-PD task, the model without PE achieved 77.27% accuracy, which is 6% better than that of the model with PE at the stage of 20%. For the F1-score metric, the model without PE also performed better than the model with PE with a 13% of difference. The AD values for both metrics also have a positive value which means model b (without PE) is better than the other. The model without PE is also significantly better based on the statistical test result. For the other tasks, W-PD and WF-PD, the results are consistently the same as the F-PD task. Thus, the models without PE are significantly better than the models with PE on both metrics, as shown in Table 3. With these models, we can predict the at-risk students in the earlier stage 20% for F-PD, W-PD, and WF-PD tasks with 77%, 82%, and 75% of accuracy, respectively. The use of PE in the transformer encoder led to the degradation of the performance on both metrics because the PE could change the important feature values, which led to incorrect information about the features, especially for the assignment score that should not be changed. Therefore, the non-embedding feature vectors, such as OULAD feature sets, should not use positional embedding because it leads to wrong information that degrades the model performance.

B. DAILY VERSUS WEEKLY FEATURE AGGREGATION

The effect of aggregation was investigated for the three tasks and both metrics. Table 4 shows the result of the experiment. For the three tasks, the weekly models performed slightly better than the daily models. The AD values had negative signs, which means that the model b (weekly) performed better. The statistical testing showed all accuracy metrics for the three tasks were significantly different. On the other hand, the F1-score metrics were not significantly different, although the AD values showed the weekly model performed better. Based on the numerical values, weekly models tend to be better than the daily models for all cases, from 0.1% to 0.4% performance boosting. Moreover, the weekly models are much lighter because the prediction can be made in each week instead of daily basis. Thus, it reduces the computational expense either on training or testing.

C. THE USE OF WEIGHTED LOSS FUNCTION

As shown in Table 2, the class distribution becomes one of the problems. F-PD and W-PD task class distribution is not balanced. The F-PD task has an IR of 2.3, and the W-PD task has an IR of 2.1. On the other hand, the WF-PD task has a slightly balanced distribution with an IR of 1.1. Therefore, the loss function was used by tuning the p_w in (7) based on these ratios, and the positive classes were weighted using the IR values. As shown in Table 5, the weighted loss function improved the performance, especially for the F-PD and W-PD tasks with an IR larger than 2. For F-PD daily models, there is no significant difference between the model with and without the weighting factors based on the statistical testing. However, the weighting factor improved the F1-score on the weekly model with 0.5% overall improvement with a p-value less than 0.01. Similarly, the weighted loss function improved the W-PD task for the daily model in terms of accuracy and the weekly model in terms of F1-score, based

TABLE 4. Performance comparison of different time period between the daily model versus the weekly model. The overall performance is the average performance of the daily or weekly prediction.

Models	Stages and Performance (%)												AD(a, b)	
	20%		40%		60%		80%		100%		Overall			
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
f-pd daily^a	77.27	52.25	82.75	67.43	86.87	75.55	89.59	81.53	91.34	85.58	83.58	67.84	**-0.19	-0.11
f-pd weekly^b	*78.44	*54.96	*83.69	*68.60	*87.58	*76.48	*90.01	*82.07	*91.70	*86.13	*84.18	*69.00		
w-pd daily^a	82.93	67.80	90.27	83.47	94.52	90.88	96.69	94.67	*98.14	*97.08	89.59	80.82	**-0.27	-0.26
w-pd weekly^b	*83.67	*69.90	*91.24	*85.10	*95.30	*92.25	*97.30	*95.68	98.11	97.05	*90.33	*82.49		
fw-pd daily^a	75.41	71.50	82.83	80.91	88.23	86.77	90.78	89.93	93.42	93.01	83.03	81.02	**-0.34	-0.38
fw-pd weekly^b	*76.76	*73.37	*83.93	*81.87	*88.96	*87.62	*91.81	*91.10	*93.57	*93.21	*83.91	*82.17		

*better than the other model based on numerical value. A positive AD indicates that model a is better, otherwise model b is better for a negative AD.
**significantly different based on statistical testing p-value < 0.01

TABLE 5. Performance comparison of different time period of the models using weighted loss function versus ordinary cross-entropy loss function. The overall performance is the average performance of the daily or weekly prediction.

Models	Stages and Performance (%)												AD(a, b)	
	20%		40%		60%		80%		100%		Overall			
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
f-pd daily^a	77.27	52.25	82.75	67.43	86.87	75.55	*89.59	*81.53	91.34	85.58	*83.58	67.84	0.05	0.00
f-pd daily bl^b	*77.80	*52.50	*82.95	*67.70	*87.04	*75.92	89.29	81.04	*91.85	*86.22	83.53	*67.85		
f-pd weekly^a	78.44	54.96	83.69	68.60	87.58	76.48	90.01	82.07	91.70	86.13	84.18	69.00	-0.09	**-0.50
f-pd weekly bl^b	*78.53	*55.68	*83.73	*69.20	*87.85	*77.41	*90.43	*83.09	*91.75	*86.27	*84.26	*69.49		
w-pd daily^a	82.93	67.80	90.27	83.47	*94.52	*90.88	96.69	94.67	*98.14	*97.08	89.59	*80.82	**-0.08	0.07
w-pd daily bl^b	*83.17	*68.03	*90.49	*83.84	94.46	90.77	*96.78	*94.80	98.02	96.88	*89.67	80.75		
w-pd weekly^a	83.67	69.90	*91.24	85.10	95.30	92.25	*97.30	*95.68	98.11	97.05	90.33	82.49	0.00	**-0.30
w-pd weekly bl^b	*83.79	*70.63	91.01	*84.79	*95.51	*92.61	97.22	95.58	*98.14	*97.10	90.33	*82.79		
fw-pd daily^a	75.41	*71.50	82.83	*80.91	88.23	86.77	*90.78	*89.93	93.42	93.01	83.03	*81.02	-0.02	**0.12
fw-pd daily bl^b	*75.43	71.30	*83.00	80.89	*88.42	*86.97	90.76	89.88	*93.60	*93.21	*83.05	80.90		
fw-pd weekly^a	*76.76	*73.37	*83.93	*81.87	*88.96	*87.62	*91.81	91.10	*93.57	*93.21	*83.91	*82.17	**0.12	**0.13
fw-pd weekly bl^b	76.45	72.89	83.71	81.58	89.09	87.77	91.84	*91.15	93.42	93.08	83.79	82.04		

*better than the other model based on numerical value. A positive AD indicates that model a is better, otherwise model b is better for a negative AD.
**significantly different based on statistical testing with p-value < 0.01

TABLE 6. Performance comparison of different time period between the LSTM model versus the transformer encoder model. The overall performance is the average performance of the daily or weekly prediction.

Models	Stages and Performance (%)												AD(a, b)	
	20%		40%		60%		80%		100%		Overall			
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
LSTM f-pd daily bl^a	72.97	37.53	79.75	59.68	85.44	70.84	89.06	79.78	88.03	80.21	81.81	60.41	**-1.72	**-7.44
f-pd daily bl^b	*77.80	*52.50	*82.95	*67.70	*87.04	*75.92	*89.29	*81.04	*91.85	*86.22	*83.53	*67.85		
LSTM f-pd weekly bl^a	73.41	38.26	81.02	61.86	86.22	72.87	89.87	81.60	88.70	81.40	82.51	62.77	**-1.75	**-6.72
f-pd weekly bl^b	*78.53	*55.68	*83.73	*69.20	*87.85	*77.41	*90.43	*83.09	*91.75	*86.27	*84.26	*69.49		
LSTM w-pd daily bl^a	78.29	57.02	86.46	76.64	93.30	88.77	96.29	94.12	95.97	93.94	88.09	77.55	**-1.58	**-3.20
w-pd daily bl^b	*83.17	*68.03	*90.49	*83.84	*94.46	*90.77	*96.78	*94.80	*98.02	*96.88	*89.67	*80.75		
LSTM w-pd weekly bl^a	79.59	59.18	89.24	81.33	94.68	91.09	97.37	95.79	98.46	97.61	89.43	79.48	**-0.90	**-3.31
w-pd weekly bl^b	*83.79	*70.63	*91.01	*84.79	*95.51	*92.61	*97.22	*95.58	*98.14	*97.10	*90.33	*82.79		
LSTM fw-pd daily^a	68.75	58.96	77.82	74.11	86.03	83.85	90.19	89.23	90.56	90.10	80.02	76.71	**-3.01	**-4.31
fw-pd daily^b	*75.41	*71.50	*82.83	*80.91	*88.23	*86.77	*90.78	*89.93	*93.42	*93.01	*83.03	*81.02		
LSTM fw-pd weekly^a	71.33	63.53	80.69	77.77	87.48	85.73	91.25	90.49	90.93	90.61	81.64	78.75	**-2.27	**-3.43
fw-pd weekly^b	*76.76	*73.37	*83.93	*81.87	*88.96	*87.62	*91.81	*91.10	*93.57	*93.21	*83.91	*82.17		

*better than the other model based on numerical value. A positive AD indicates that model a is better, otherwise model b is better for a negative AD.
**significantly different based on statistical testing with p-value < 0.01

on the statistical testing. On the other hand, for the WF-PD task, the use of the weighted loss function in the training phase led to a degradation in performance. The F1-score on the daily model and both metrics of the weekly model show that models trained without weighted loss function performed better based on the statistical testing. It can be concluded that the weighting factor was not required because of the slightly balanced distribution of the class.

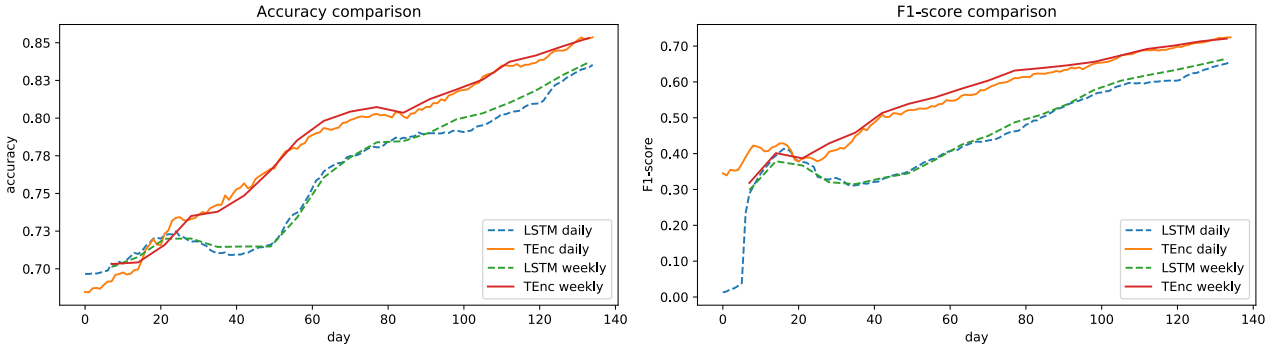

FIGURE 3. Model performance comparison based on accuracy and F1-score metrics on each time step of F-PD tasks until the 50% length of courses.

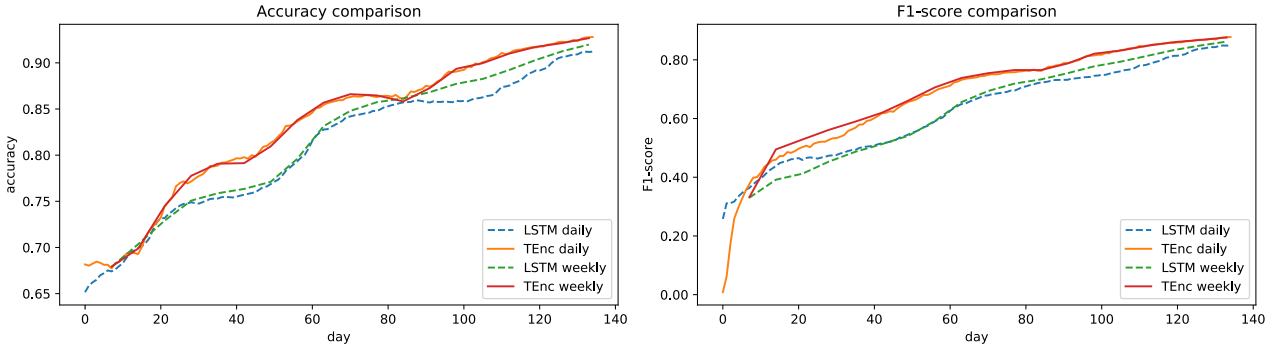

FIGURE 4. Model performance comparison based on accuracy and F1-score metrics on each time step of W-PD tasks until 50% length of courses.

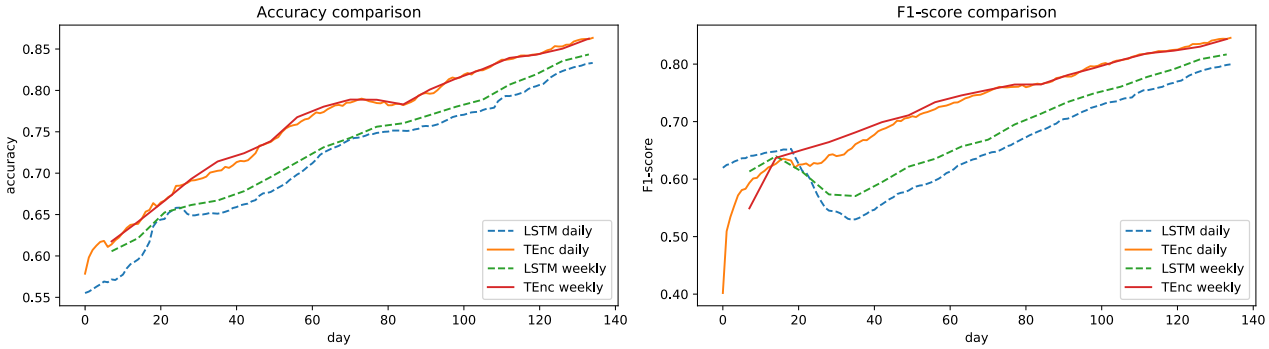

FIGURE 5. Model performance comparison based on accuracy and F1-score metrics on each time step of WF-PD tasks until 50% length of courses.

D. THE COMPARISON WITH RECURRENT MODELS

In this section, the transformer models were compared to the recurrent models, i.e., long short-term memory (LSTM), in terms of accuracy and F1-score metrics. LSTM is commonly used for processing sequential data such as OULAD [14]. As the results of the previous experiments, all models with F-PD and W-PD tasks were trained using a weighted loss function. On the other hand, all models with WF-PD tasks were trained without a weighting factor. Table 6 shows the comparison results. The performance of the transformer encoder model is better than the recurrent models for all cases based on the statistical results. In the case of the F-PD task, the AD values are 1.7% in terms of accuracy and 7% in terms of F1-score for both daily and weekly cases. Similarly to the other tasks, the AD of the F1-score is relatively significant, at least 3%, which means that the transformer encoder models are robust to the imbalanced data compared to the LSTM model.

1) EARLY PREDICTION COMPARISON

Here, the transformer encoder models were compared to the LSTM models in the case of early prediction cases. As previously mentioned, the transformer encoder models performed better in all tasks. Fig. 3 shows the comparison of transformer encoder and LSTM models based on daily or weekly feature aggregation until the 50% stage (half of

the course period) for the F-PD task. Transformer encoder models outperformed the LSTM models almost in each time step in terms of accuracy and F1-score. Similarly, for W-PD and WF-PD tasks, all transformer encoder models performed better than the LSTM models, as shown in Fig. 4 and Fig. 5. These results confirmed the statistical testing in the previous section. Note that the longer the time step is, the better the model performance is. This is because the longer time step led to much information about the student activities that can be remembered by the models through the hidden state and the gate channel for the LSTM, and the attention module for the transformer encoder.

VI. CONCLUSION

In this paper, we propose a transformer encoder model for predicting at-risk students based on their interactions in an LMS. From the experiments on the OULAD, predicting the at-risk student at 20% of the course period using the proposed model gave an accuracy of at least 76%. For the W-PD task, the model achieved 83% of accuracy and 70% of the F1-score at the 20% stage, whereas the overall accuracy until the end of courses was 90% and the overall F1-score was 83%. From the experimental results, the use of positional encoding (PE) in the transformer encoder led to the degradation of performance on both metrics because the PE could change the feature values, which led to incorrect information, especially for the assignment score that should not be changed. Moreover, the feature aggregation significantly affected the accuracy, although the average difference is around 0.1% to 0.4%. On the other hand, the feature aggregation did not affect the F1 score. The F1-score metric was used to fairly measure the performance of the model on imbalanced data, such as F-PD and W-PD tasks. The weighted loss function was used to overcome the problem. The results show that the model gains a performance improvement in terms of the F1-score, especially for weekly feature aggregation. Moreover, a transformer encoder performed better than an LSTM on all tasks on both metrics based on the statistical testing with average difference (AD) values between 1% to 3% for the accuracy metric and 3% to 7% for the F1-score metric. LA using a deep learning model such as a transformer was rarely used due to its sophisticated architecture. However, the model has great potential to be used in LA. Another important issue is the compatibility of the dataset with the model. Thus, the data collection mechanism should be integrated with the LMS, such as in OULAD. In the future, this method can be applied to any LMS that provides a data architecture similar to the OULAD.

ACKNOWLEDGMENT

The authors would like to extend their heartfelt gratitude to the proofreading support service from Universitas Gadjah Mada for their invaluable assistance in improving the quality of this manuscript.

REFERENCES

[1] O. Pilli and W. Admiraal, "Students learning outcomes in massive open online courses (MOOCs): Some suggestions for course design," *Yuksekogretim Dergisi*, vol. 7, no. 1, pp. 46–71, Apr. 2017.
[2] K. T. Chui, R. W. Liu, M. Zhao, and P. O. D. Pablos, "Predicting students performance with school and family tutoring using generative adversarial network-based deep support vector machine," *IEEE Access*, vol. 8, pp. 86745–86752, 2020.
[3] J. Kasurinen and U. Nikula, "Estimating programming knowledge with Bayesian knowledge tracing," *ACM SIGCSE Bull.*, vol. 41, no. 3, pp. 313–317, Aug. 2009.
[4] Y. Qiu, Y. Qi, H. Lu, Z. A. Pardos, and N. T. Heffernan, "Does time matter? Modeling the effect of time with Bayesian knowledge tracing," in *IEDM Tech. Dig.*, 2011, pp. 139–148.
[5] F. Hlioui, N. Aloui, and F. Gargouri, "Withdrawal prediction framework in virtual learning environment," *Int. J. Service Sci., Manag., Eng., Technol.*, vol. 11, no. 3, pp. 47–64, Jul. 2020.
[6] H. Waheed, S.-U. Hassan, N. R. Aljohani, J. Hardman, S. Alelyani, and R. Nawaz, "Predicting academic performance of students from VLE big data using deep learning models," *Comput. Hum. Behav.*, vol. 104, Mar. 2020, Art. no. 106189.
[7] K. Lee, J. Chung, Y. Cha, and C. Suh, "Machine learning approaches for learning analytics: Collaborative filtering or regression with experts," in *Proc. NIPS*, Dec. 2016, pp. 1–11.
[8] Y. Bengio, I. Goodfellow, and A. Courville, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2016.
[9] D. Shen, G. Wu, and H. Suk, "Deep learning in medical image analysis," *Annu. Rev. Biomed. Eng.*, vol. 19, pp. 221–248, Jun. 2017.
[10] S. A. I. Alfarozi, K. Pasupa, H. Hashizume, K. Woraratpanya, and M. Sugimoto, "Robust and unified VLC decoding system for square wave quadrature amplitude modulation using deep learning approach," *IEEE Access*, vol. 7, pp. 163262–163276, 2019.
[11] S. Nosratabadi, A. Mosavi, P. Duan, P. Ghamisi, F. Filip, S. Band, U. Reuter, J. Gama, and A. Gandomi, "Data science in economics: Comprehensive review of advanced machine learning and deep learning methods," *Mathematics*, vol. 8, no. 10, p. 1799, Oct. 2020.
[12] Y. Choi, Y. Lee, J. Cho, J. Baek, B. Kim, Y. Cha, D. Shin, C. Bae, and J. Heo, "Towards an appropriate query, key, and value computation for knowledge tracing," in *Proc. 7th ACM Conf. Learn. Scale*, Aug. 2020, pp. 341–344.
[13] X. Song, J. Li, S. Sun, H. Yin, P. Dawson, and R. R. M. Doss, "SEPN: A sequential engagement based academic performance prediction model," *IEEE Intell. Syst.*, vol. 36, no. 1, pp. 46–53, Jan. 2021.
[14] S. Hassan, H. Waheed, N. R. Aljohani, M. Ali, S. Ventura, and F. Herrera, "Virtual learning environment to predict withdrawal by leveraging deep learning," *Int. J. Intell. Syst.*, vol. 34, no. 8, pp. 1935–1952, Aug. 2019.
[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017, *arXiv:1706.03762*.
[16] J. Kuzilek, M. Hlosta, and Z. Zdrahal, "Open university learning analytics dataset," *Sci. Data*, vol. 4, no. 1, pp. 1–8, Nov. 2017.
[17] Y. He, R. Chen, X. Li, C. Hao, S. Liu, G. Zhang, and B. Jiang, "Online at-risk student identification using RNN-GRU joint neural networks," *Information*, vol. 11, no. 10, p. 474, Oct. 2020.
[18] E. G. Poitras, R. F. Behnagh, and F. Bouchet, "A dimensionality reduction method for time series analysis of student behavior to predict dropout in massive open online courses," in *Adoption of Data Analytics in Higher Education Learning and Teaching*. Berlin, Germany: Springer, 2020, pp. 391–406.
[19] R. Alshabandar, A. Hussain, R. Keight, A. Laws, and T. Baker, "The application of Gaussian mixture models for the identification of at-risk learners in massive open online courses," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2018, pp. 1–8.
[20] L. Haiyang, Z. Wang, P. Benachour, and P. Tubman, "A time series classification method for behaviour-based dropout prediction," in *Proc. IEEE 18th Int. Conf. Adv. Learn. Technol. (ICALT)*, Jul. 2018, pp. 191–195.
[21] H. Heuer and A. Breiter, "Student success prediction and the trade-off between big data and data minimization," *DeLFI 2018: Die 16. E-Learning Fachtagung Informatik*, D. Krömker and U. Schroeder, Eds. Bonn, Germany: Gesellschaft für Informatik e.V., 2018, pp. 219–230.
[22] S. Rizvi, B. Rienties, and S. A. Khoja, "The role of demographics in online learning; A decision tree based approach," *Comput. Educ.*, vol. 137, pp. 32–47, Aug. 2019.

[23] M. Hussain, W. Zhu, W. Zhang, and S. M. R. Abidi, "Student engagement predictions in an E-learning system and their impact on student course assessment scores," *Comput. Intell. Neurosci.*, vol. 2018, pp. 1–21, Oct. 2018.
[24] D. Shin, Y. Shim, H. Yu, S. Lee, B. Kim, and Y. Choi, "SAINT+: Integrating temporal features for EdNet correctness prediction," in *Proc. 11th Int. Learn. Anal. Knowl. Conf.*, Apr. 2021, pp. 490–496.
[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
[26] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2019, pp. 8024–8035. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf
[27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[28] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Proc. SPIE*, vol. 11006, May 2019, Art. no. 1100612.

SRI SUNING KUSUMAWARDANI (Member, IEEE) received the bachelor's (S.T.), master's (M.T.), and Ph.D. (Dr.) degrees in electrical and information engineering from Universitas Gadjah Mada (UGM), Yogyakarta, Indonesia. In 2009, she worked as a Postdoctoral Research Fellow with the Faculty of Engineering Education, Utah State University, Logan, UT, USA. From 2012 to 2013, she also worked as a Postdoctoral Research Fellow (Erasmus Mundus Research Program) with the Universitat Politècnica de Catalunya, Barcelona, Catalonia, Spain. She is currently an Associate Professor with the Department of Electrical and Information Engineering, Faculty of Engineering, UGM. Her research interests include MOOCs and micro credentials, where artificial intelligence and machine learning can be applied to support students/learners to increase the motivation and learning performances.

SYUKRON ABU ISHAQ ALFAROZI (Member, IEEE) received the bachelor's (S.T.) degree in electrical and information engineering from Universitas Gadjah Mada (UGM), Yogyakarta, Indonesia, in 2014, and the Ph.D. degree in information technology from the King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand, under the sandwich program with Hokkaido University, Sapporo, Japan, in 2020. He is currently with the Department of Electrical and Information Engineering, Faculty of Engineering, UGM. His research interests include machine learning and its application, computer vision, and signal processing.