Log-Driven Conformance Checking Approximation Method Based on Machine Learning Model

Huan Fang, Sichen Zhang, Zhenhui Mei
School of Mathematics and Big Data, Anhui University of Science and Technology, Huainan, China, 232001

***Abstract*—Conformance checking techniques are usually used to determine to what degree a process model and real execution trace correspond to each other. Most of the state-of-the-art techniques to calculate conformance value provide an exact value under the circumstance that the reference model of a business system is known. However, in many real applications, the reference model is unknown or changed for various reasons, so the initial known reference model is no longer feasible, and only some historical event execution traces with its corresponding conformance value are retained. This paper proposes a log drivened conformance checking method, which tackles two perspective issues, the first is presenting an approach to calculate the approximate conformance checking value much faster than the existing methods using machine learning method. The second is presenting an approach to conduct conformance checking in probabilistic circumstances. Both kinds of approaches are from the perspective of no reference model is known and only historical event traces and their corresponding fitness can be used as train data. Specifically, for large event data, the computing time of the proposed methods is shorter than those align-based methods, and the baseling methods includes k-nearest neighboring, random forest, quadratic discriminant analysis, linear discriminant analysis, gated recurrent unit and long short-term memory. Experimental results show that adding a machine learning classification vector in the training set as preprocessing for train data can obtain a higher conformance checking value compared with the training sample without increasing the classification vector. Simultaneously, when conducted in processes with probabilities, the proposed log-log conformance checking approach can detect more inconsistent behaviors. The proposed method provides a new approach to improve the efficiency and accuracy of conformance checking. It enhances the management efficiency of business processes, potentially reducing costs and risks, and can be applied to conformance checking of complex processes in the future.**

Keywords*—Conformance checking; fitness; log driven; machine learning; deep learning; probabilities

I. Introduction

Process mining mainly extracts valuable process information from events. It is a supplement and innovation to business process management methods. Process mining is mainly composed of three parts, namely, discovery, conformance checking, and enhancement [1]. Conformance checking is designed to check the conformity of discovered process model with the event executions, so conformance checking value or fitness is used to describe the degree that the event execution conforms to the process model.

Two problems and two major challenges are encountered in conformance checking studies [2]. The two problems are described as follows: (1) Does the process execute the process model in the manner recorded in the model? (2) How much flexibility does the log trace allow in the execution of the process in the case of violation of the rules? The two challenges are described as follows: (1) How to improve the performance of conformance checking when the models and logs become larger? (2) How to balance between precision and deliberate vagueness?

For the two challenges, we do not need to obtain a specific value for conformance checking in several cases as long as we can acquire an approximate value to meet our needs. Therefore, studying efficient approximate consistency methods is important for large-scale log situations or situations where the reference model is unknown.

The state-of-the-art studies on conformance checking methods are mostly based on rule checking [20], token-based replay [7], and alignment [9]. The main starting points of these existing methods are based on the assumption that the process reference model is known. However, in some real cases, the process reference model is unknown for some reasons, such as some changing operations are introduced as software maintenance and business integrations. In such cases, the initial reference model is no longer suitable for current use, and the process reference model is not saved for further use in some other situations. Thus, considering how to efficiently measure conformance checking value only on the basis of historical event execution traces with the absence of the process reference model is crucial.

This paper proposes a new approach to calculate the conformance checking value through machine learning method from event logs. The designed method uses some machine learning and deep learning algorithms to calculate the approximate conformance checking value, and a collection of experiments is implemented. The results show two fold conclusions. On the one hand, our method provides an approximate one but quicker, specifically in large event data for the reason of introducing machine learning algorithm compared with the alignment method that usually provides a precise conformance value and takes long computation time. On the other hand, adding a machine learning classification vector in the training set can obtain an approximate conformance checking value with higher precision compared with the training sample without introducing the classification vector obtained by machine learning. Furthermore, a series of experiments were also conducted in probabilistic processes. A stochastic process miner was used to mine models from real event logs, and the generated stochastic process model was used to simulate event logs, which were then subjected to a series of variations. The real logs were compared with the simulated logs through a conformance check to obtain a conformance score, which was then compared with alignment methods. This approach

does not require maintaining organizational process models but instead detects noncompliant process traces based on historical data. The results indicate that when considering probabilities, the conformance checking technique detects more inconsistent behaviors.

The remainder of this paper is structured as follows. Section II discusses related work. Section III reviews some basic concepts and notations. Section IV introduces the proposed method. Section V conducts experiments and analyzes the results. Section VI provides the conclusions and presents some future work.

II. Related Work

For the first question posed in the previous section of the conformance checking study is as follows: are the logs executed in the manner the model records it? The state-of-the-art research has provided relatively complete methods and conclusions. The early research of conformance checking is dedicated to determining whether a given process instance conforms to a given process model [3], [4], [5], that is, whether the process log conforms to the model is quantified through discrete values 0 and 1, where 0 indicates that the log does not conform to the process model, and 1 indicates that the log fully conforms to the process model. In [6], a recurrent neural network (RNN) is used to classify the traces in the log, where the discrete values 0 and 1 are used for classification. Some scholars aimed to provide diagnosis at the event log level, that is, to observe the extent to which the log instance violates the process model rather than simply providing a simple yes/no answer. This type of method usually assigns a value between 0 and 1 to quantify the degree to which the process log conforms to the process model. The larger the value, the higher the degree to which the process log conforms to the model, thereby solving the second problem of the conformance checking research. The method proposed in [7] can accurately point out where the deviations occur more frequently and the severity of process instances that do not conform to the process model. The early work of conformance checking is mostly based on token-based replay [8]. This technology replays each trace of the event log in the process model by executing tasks in accordance with the sequence of each event and by observing the process during the replay. The final state of the model can determine whether and to what extent the tracking actually corresponds to the effective execution sequence of the model. However, this method may be constrained by its dependence on the final model state, which might not capture all the subtle nuances of process deviations. Alignment was introduced in [9] and quickly developed into the mainstream of conformance checking technology, and many alignment-based extension methods were developed. The work in [10] proposed an incremental method to check the consistency of the process model and the event log. It may still face challenges when dealing with extremely large datasets. The work in [11] presented a conformance checking method based on multiperspective declarative, adding other perspectives, such as data or time for consistency testing, such as describing process behavior. Most conformance checking techniques using alignments provide an exact solution for fitness values. However, in many applications, having an approximation of the conformance value is sufficient. Specifically, for large event data, the computing time for alignments is considerably long by using current techniques, making them inapplicable in reality [12].

Some studies have investigated approximate consistency calculation methods. The work in [12] used subset selection and edit distance for conformance checking, thereby improving the performance of the conformance checking method compared with alignment. But the selection of subsets may ignore some key process behaviors. The work in [13] applied bound approximation guides for the selection of the relevant subsets of the process model behavior, further improving the approximate accuracy of the consistency calculation value. The work in [14] presented a statistical approach to ground conformance checking in trace sampling and conformance approximation. This type of method significantly reduces the running time while still ensuring the accuracy of the estimated conformance checking results, And the author has improved it in the latest work [15]. The work in [16] used the simulation behavior of the process model to approximate the conformance checking value. The simulation method generates a trace that is more similar to the behavior recorded in the event log and uses these simulated traces and edit distance functions to approximate the conformance checking value. The work in [17] developed an approximation method for calculating the fitness value by applying the relaxation labeling to the process partial order representation of the model. The work in [18] proposes a method to compute the alignment of logs to a reference process using trie data structures to improve efficiency through compact representation of process proxy behavior and attempts to reduce the search space. The work in [19] proposes an online approximate consistency detection method that clusters event logs and selects representative traces to construct support sets for consistency detection. However, the clustering quality directly affects the detection accuracy.

III. Preliminaries

In this section, we give a brief introduction to basic process mining, especially the conformance checking terminology and notation that can improve the readability of this paper.

A. Log Trace and Log

An event trace (or event executions) over an alphabet of activity names Σ is a finite word $\sigma \in \Sigma^*$ that corresponds to an event sequence. A log is a collection of log traces.

Denoting $L = \{\tau_0, \tau_1, \tau_2, \cdots, \tau_n, \}$ as an event log, and τ_i as a log trace. Each event in the process is recorded in a trace, that is, $\tau = \{e_1 e_2 e_3 \cdots e_n\}$. $len(\tau)$ indicates the number of cases recorded in the trace, and $\tau(j)$ indicates the $j-th$ event in τ.

As shown in Table I, 6676 event traces constitute log L, $\langle A, C, G^7, H, D, F, I\rangle$is an event trace in L, and $\Sigma = \{A, B, C, D, E, F, G, H, I\}$ is the set of activities corresponding to events. The labels 0 and 1 denote the a binary output.

B. Classification Learning Method

1) Quadratic Discriminant Analysis (QDA): The idea of QDA classification is to first construct a discriminant function F and use it to determine the decision boundary between classes. The discriminant function F is used to establish the

TABLE I. AN EVENT LOG EXAMPLE

ID	Event sequence	label
1	$\langle D, B, D, E, I\rangle$	0
2	$\langle A, C, D, A, C, F, I\rangle$	0
3	$\langle A, C, B, H, F, I\rangle$	0
4	$\langle A, C^4, D, G, F, I\rangle$	0
.....	.....	.....
6673	$\langle A, C, G^7, H, D, F, I\rangle$	1
6674	$\langle A, C, G^7, D, H, F, I\rangle$	1
6675	$\langle A, C, G^7, D, G, F, I\rangle$	1
6676	$\langle A, C, G^8, D, F, I\rangle$	1

decision boundary for distinguishing different categories into different regions [21].

Assuming that the data follow the Gaussian mixture model, the observations in the category conform to the multivariate Gaussian distribution of mean and covariance, that is,

$$x \in C_i \Leftrightarrow x = \mu_i + \sum_i^{1/2} z,\ with\ z \sim N(0, I_p) \quad (1)$$

where I_p represents the size of the $p \times p$ unit matrix.

Let $\pi_i,\ i \in \{0, 1\}$ denote the prior probability that x belongs to class C_i. The classification rules related to QDA classifier are given as (Eq. 2).

$$W^{QDA}(x) = -\frac{1}{2} log \frac{|\Sigma_0|}{|\Sigma_1|} - \frac{1}{2} x^T (\sum_0^{-1} - \sum_1^{-1}) x + x^T \sum_0^{-1} \mu_0 - x^T \sum_1^{-1} \mu_1 - \frac{1}{2} \mu_0^T \sum_0^{-1} \mu_0 + \frac{1}{2} \mu_1^T \sum_1^{-1} \mu_1 - log \frac{\pi_1}{\pi_0} \quad (2)$$

The number of training observations for each class $C_i,\ i \in \{0, 1\}$ is denoted as $n_i,\ i \in \{0, 1\}$, and $T_0 = \{x_l \in C_0\}_{l=1}^{n_0}$ and $T_1 = \{x_l \in C_1\}_{l=n_0+1}^{n_0+n_1}$ are used to represent their respective samples.

$$\widehat{\mu_i} = \frac{1}{n_i} \sum_{l \in T_i} x_l,\ i \in \{0, 1\} \quad (3)$$

$$\widehat{\Sigma_i} = \frac{1}{n_i - 1} (x_l - \widehat{\mu_i})(x_l - \widehat{\mu_i})^T,\ i \in \{0, 1\} \quad (4)$$

$$\begin{cases} x \in C_0, if\ W^{QDA} > 0 \\ x \in C_1,\ otherwise \end{cases} \quad (5)$$

2) AdaBoost: AdaBoost [22] is a popular integrated learning technology due to its adaptability and simplicity. AdaBoost has been successfully extended to the field of pattern recognition, computer vision, and has been used in many fields, such as two class and multiclass scenes. The main idea of AdaBoost is to build a series of weak learners by using different training sets, which are obtained by resampling the original data. These learners are combined through weighted voting to predict the class label of the new test instance.

3) Long Short-term Network (LSTM) network: LSTM network is an improvement of RNN [23], which effectively solves the gradient disappearance and gradient explosion of RNN by adding a gate structure. The LSTM network is widely used in time series forecasting and has been utilized in process monitoring and forecasting in recent years.

4) Gated Recurrent Unit (GRU): The GRU network is a variant of the LSTM network [24], which combines the forget gate and the input gate in the LSTM network into one gate, which is called the update gate. It has two door structures, the update door and the reset door. The update gate is used to determine the degree of retention of the state information at the previous moment in the current moment of learning. The larger the update gate value, the greater the degree of retention. The reset gate is used to control the degree of combination between the state information at the previous moment and the state information at the current moment. The larger the reset gate value, the greater the degree of combination. A simplified GRU network maintains the LSTM effect, has a simpler structure, fewer parameters, and a better convergence model.

C. Regression Learning Method

1) Light Gradient Boosting Machine (LGBM): Light GBM is a gradient boosting framework originally developed by Microsoft and uses a tree-based learning algorithm. Its main idea is to use weak classifiers (decision trees) for obtaining the optimal model through iterative training. This framework has good training effect, difficult overfitting, and is widely used to solve regression problems.

2) Random forest: Random forest is an ensemble learning method for classification and regression [25]. It runs by constructing a large number of decision trees during training. The random forest regression model is a model obtained by synthesizing the results obtained from several established decision tree models, and the final prediction result is obtained by averaging the prediction results of all decision tree models.

D. One-hot Encoding Method

One-Hot encoding, also known as one-bit effective encoding, mainly uses N-bit status registers to encode N states. Each state has its own independent register bit, and only one bit is valid at any time. One-hot coding represents the categorical variables as binary vectors.

IV. LOG-DRIVEN APPROXIMATE CONFORMANCE CHECKING VALUE CALCULATION METHOD

A. Method Framework

The method proposed in literature [6] has some commonalities with this paper in that they are based on classifying logs and error logs to obtain the conformance checking values. The difference is that the values in literature [6] use RNN methods for classification to obtain global accuracy and recall between logs and models, whereas our proposed method obtains the approximate conformance checking values for each trace in the logs. In this paper, we propose a method to approximate the conformance checking values by using machine learning. The results are improved by adding an intermediate vector for the classification to the original data's method for fitting.

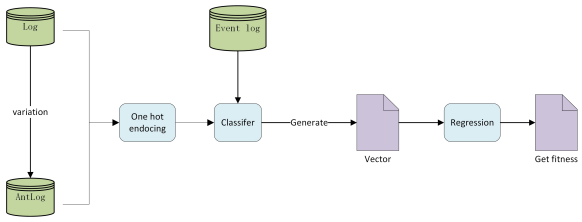

Fig. 1. Overall implementation framework of the proposed method.

Fig. 1 shows the overall framework of the proposed method proposed. In accordance with the existing event log and its consistency training samples, the traces in the log are divided into "correct traces" and "error traces," and label values of 1 and 0 are assigned, respectively. The two types of trace data are preprocessed. For simplicity of expression, the set of "correct traces" is recorded as Log, and the set of "error traces" is recorded as Antilog. To better maintain the order relationship of the traces in the log, this article uses one-hot encoding to process all the traces. The use of GRU, LSTM, k nearest neighbor, Gaussian process, decision tree, random forest, AdaBoost, and QDA learning with algorithms, such as LDA, can obtain the classification accuracy of the "correct trace" and "wrong trace" in the log. On the basis of the classifier, the approximate value of the consistency is further obtained.

Let all gather event executions as set $L = \{L_T, L_X\}$, where L_T denotes the trace set that all event execution trace in it have conformance checking value between 0 and 1, and L_X denotes the trace set that all event trace in it have no conformance checking value.

Algorithm 1: Conformance checking value approximate calculation method

Input: log $L = \{L_T, L_X\}$, $Fit = \{c_i \mid c_i \in [0,1] \wedge c_i = fitness(\pi_i) \wedge \pi_i \in L_T\}$

Output: the fitness value of each trace in the trace set L_x.

Procedures:

Step 1: In accordance with the fitness value in Fit, if the fitness value of a trace is less than 1, then mark its classification label as 0, else if the fitness value equals 1, then set its classification label to 1.

Step 2: The traces in L are processed by using one-hot encoding and machine learning algorithms. k nearest neighbor, decision tree, random forest, and QDA are used to classify the traces in accordance with the labels in step 1.

Step 3: Perform one-hot encoding on the trajectory in L, and use the LSTM and GRU deep learning algorithms to classify in accordance with the label in step 1, where the coding of each activity is inputted into each time step.

Step 4: Use the classification model with the highest score in Steps 2 and 3 (known as the QDA algorithm), and apply the classification model to all the trajectories in L to obtain the classification vector.

Step 5: Add the classification vector to the original data.

Step 6: Use regression algorithm to fit and obtain the fitness value of each trace.

B. Metric Method

In this paper, the accuracy [26] is used to evaluate the classification algorithm, and the conformance checking value of fitting is measured in terms of mean absolute error (MAE), mean square error (MSE), and R-squared [27]. The related evaluation index calculation methods are expressed as Eq. (6), (7), (8) and (9).

For binary classification problems, the samples can be divided into true positive (TP) in accordance with their true categories and the predicted categories of the classifier. The true category and the predicted category are positive examples. False positive (FP): The true category is negative, and the predicted category is positive. False negative (FN): The true category is positive, and the predicted category is negative. True negative (TN): The true category is a negative case, and the predicted category is a negative case. The accuracy rate is calculated, as shown in Eq. (6).

$$accuracy = \frac{TP + NP}{TP + TN + FP + FN} \tag{6}$$

y_i is the true value of the $i-th$ sample, and $\widehat{y_i}$ is the observed value of the $i-th$ sample.

The MAE is used to measure the average value of the absolute difference between the predicted value and the true value. The smaller the MAE, the better the model. The calculation method is shown in Eq. (7).

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |(y_i - \widehat{y_i})| \tag{7}$$

The MSE represents the average of the squared difference between the original value and the predicted value in the data set. It measures the variance of the residuals. The smaller the value, the better. The calculation method is shown in Eq. (8).

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \widehat{y_i})^2 \tag{8}$$

R-squared represents the coefficient of the degree of fit between the predicted value and the original value. The larger the value, the better the model. The calculation method is shown in Eq. (9).

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (\widehat{y_i} - y_i)^2}{\sum_{i=1}^{n} (\bar{y} - y_i)^2} \tag{9}$$

C. Probabilistic Log-Driven Stochastic Process Conformance Checking

First, using the stochastic process miner from the [34] and the Prom tool, a probabilistic stochastic process model was mined from the real event log L. Then, the generated model was simulated using Pm4py to produce event logs, which underwent a series of variations to obtain the simulated event log $L^{'}$. The real event log L and the simulated event log $L^{'}$

were then subjected to a Log-Log conformance check using the method from the [35] to obtain the conformance value. The specific steps are as follows:

Algorithm 2: Probabilistic log-driven stochastic process conformance checking

Input: Event log L, $L^{'}$
Output: $L - L^{'}$ conformance score
Procedures:
Step 1: Mine a probabilistic stochastic process model M from the real event log L.
Step 2: Use the Pm4py tool to simulate event logs using M, and apply a series of variations (including some traces that do not conform to the model) to obtain the log $L^{'}$.
Step 3: Compute the reallocation matrix for L and $L^{'}$.
Step 4: Compute the distance matrix for L and $L^{'}$.
Step 5: Calculate the $L - L^{'}$ conformance score based on the reallocation matrix and distance matrix of L and $L^{'}$.

V. Evaluation

A. Artificial Log Acquisition and Preprocessing

No benchmark case library can be used for the calculation and evaluation method of the approximate conformance checking value based only on logs. Therefore, this paper adopts the following methods to generate the associated manual integration log. A process is customized by using the Petri net model SN, and then the token replay technology in pm4py library [28] is used to generate the "correct traces" set (Log) that conforms to the process model for the process model SN. We then mutate these traces to make them noncompliant with the process model "error traces" set ($Antilog$) and use the alignment technology in pm4py to calculate the consistency value of each trace with respect to the model. These traces and their fitness are used as the data set of the experimental work in this paper.

This paper obtains two real event case datasets from the public datasets, a real event log of a sepsis case [29] and a real event log of the information system for managing road traffic fines [30]. No traces in these real logs that do not conform to the real process model. Therefore, we first use the mining algorithm to obtain a model of real logs and then use the real model to simulate real logs as the dataset for the experimental work.

Fig. 2, 3 and 4 show the three Petri net models used in this paper, respectively. This paper uses the models in Fig. 2, 3 and 4 to generate $Log1$, $Log2$, and $Log3$, respectively. The three models contain different loops and concurrent behavior that can be used to test the applicability of the proposed method to various behavior. As shown in Table I, the set of "correct trace" and "error trace" is generated by model 1. Given that cycles are found in the model and generating all traces is impossible, we categorize the log into two disjoint parts, which are $L_{=K}$ and $L_{<K}$, where $L_{<K}$ denotes the completeness log set under the k constraint, that is, the trace length is k, and $L_{<K}$ contains all traces of the model that have length less than k.

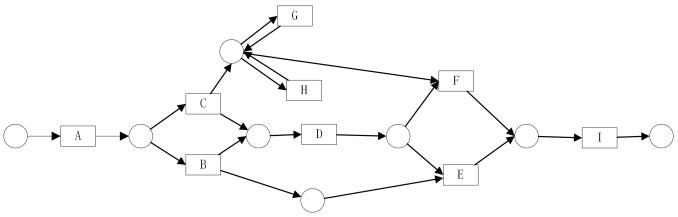

Fig. 2. Petrinet model SN1.

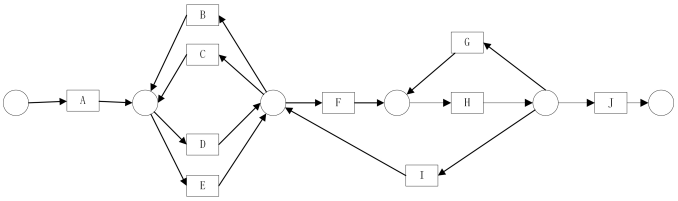

Fig. 3. Petrinet model SN2.

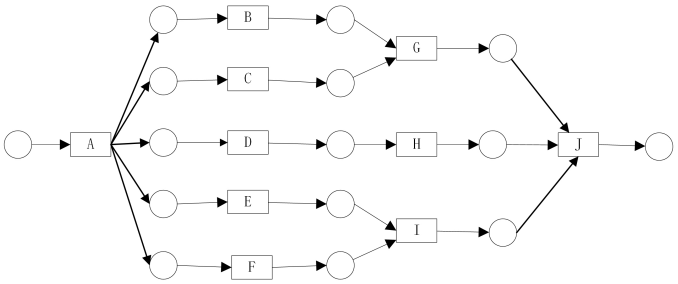

Fig. 4. Petrinet model SN3.

B. Practical Logs

The experiments implemented here uses five different types of event logs, three of which are manually generated logs (named Log1, Log2 ,and Log3 produced in last section), and the other two are real event logs (named sepsis and road fine). The data presented in this study are openly available in at https://pan.baidu.com/s/1TFEobiqrXTWjQF4R-cLHwQ (extract code: aidw).

Taking $Log1$ for an example. As shown in Table I, the first log contains 6667 different traces. Its consistency is marked as 1 and 0 in accordance with whether the trace conforms to the process in the data. The detailed information of the five logs is shown in Table II.

C. Classification Result Analysis

We selected two types of learning algorithms for classification experiments, which are deep learning algorithm and machine learning algorithm. The use of machine learning and deep learning methods require different processing of the generated logs. To better preserve the sequence relationship in the logs, we perform one-hot encoding on the log.

The traces in $Log1$ (Table I) are taken as an example. $Log1$ has a total of nine activities, and the longest trace length is 18. The length of trace $\langle D, B, D, E, I\rangle$ is 5. In the procedure of one-hot encoding, the character "0" is filled to make the trace with a length of

TABLE II. INFORMATION ABOUT THE FIVE LOGS

Log	Complete-ness	Num. activities	Max. case length	Min. case length	cases Normal	cases Deviant
Log1	true	9	18	1	4098	2578
Log2	true	10	20	1	4862	4005
Log3	true	10	15	1	2240	2025
Sepsis	false	16	14	3	41143	17397
Road fine	false	11	14	3	21868	11378

18, that is, $\langle D, B, D, E, I, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\rangle$ after filling. One-hot encoding is performed for each activity of the trace. Given that 10 elements are found in the activity table $[A, B, C, D, E, F, G, H, I, 0]$, each element of the activity table can be represented by 10 bits. For example, A is represented by $[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]$, and B is represented by $[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]$, and similar means for other elements. Therefore, each trace is filled as a vector of 180 dimensions. The trace $\langle D, B, D, E, I, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\rangle$ is represented by a 180 bit as $[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0. \ldots\ldots\ldots\ldots 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]$.

For deep learning, we directly use the encoded vector, each trace has a label, and each activity is inputted into the LSTM and GRU as an event step. We use LSTM and GRU for classification experiments, which are different from [6]. Our training data have more variables than [6], and the length of the trace is longer than that used in [6]. RNN specializes in processing time series data. The disadvantage of RNN is that it cannot handle long-term dependencies, whereas LSTM and GRU can handle long-term dependencies well. We use the LSTM and GRU network in Pytorch library [31] for training. For machine learning, we use the PCA dimensionality reduction algorithm to reduce the dimensionality of the data after one-hot encoding and then we use several typical methods, such as random forest, secondary discrimination, decision tree, and k nearest neighbors. These algorithms are integrated into the sklearn library [32] in Python and can be easily used. The supposed framework is shown in Fig. 5.

In this paper, we conduct two type of experiments. In the first experiment, we perform a classification experiment on three different logs. These logs are divided into two categories. The first category is in line with the process model and is marked as 1. Another type of log that does not conform to the process model is marked as 0. We use machine learning and deep learning methods to classify these data, so we can obtain a classification method with a higher accuracy score.

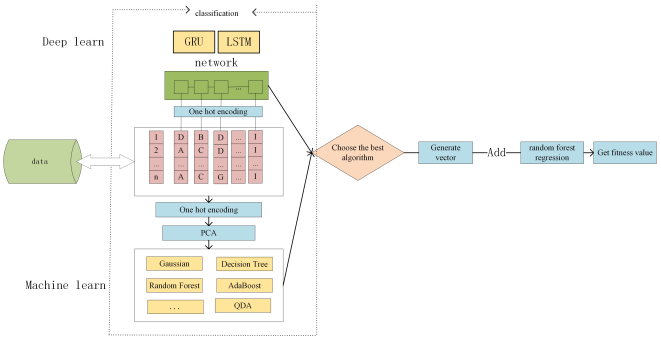

Fig. 5. Supposed framework.

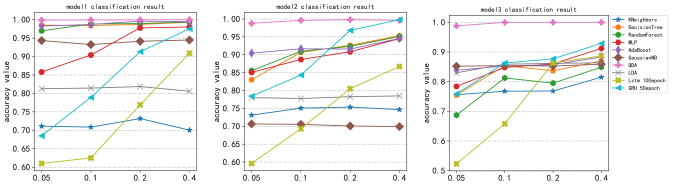

Fig. 6. Comparison of the classification accuracy of various methods.

In the second experiment, we use the method with the best classification effect in experiment 1 to process the log and obtain the classification vector of each trace in the log. The classification vector and the original trace vector are used to fit the fitness, so that we can obtain a good fitting effect.

In the classification experiment of different labels, for three different logs, we divided the training set of 5%, 10%, 20%, and 40%, and the rest is the test set. Fig. 6 shows the classification results under $5\%$of the training set. We use a line chart to easily observe the effect of the classification model. The QDA classification method has the highest accuracy by observing the different classification methods in Fig. 6 for the three different log classification results. It can obtain 99% accuracy under $5\%$of the training set, and the QDA method is better than the method ranked as second. The classification accuracy is higher, indicating that the QDA method can better classify the logs generated by the Petri net structure. Therefore, we use the QDA classifier to generate a classification vector for the classification to fit the conformance checking value. The vector is added to the PCA dimensionality reduction vector, and then the fit method is used to fit the fitness value, as shown in the next section.

The experimental results in Fig. 6 show that using QDA to classify data can obtain higher classification accuracy. Therefore, the probability generated by the decision function of QDA classification is added to the data as a classification vector for fitting.

D. Analysis of Fitting Experiment Results

This experiment is divided into two groups. The experimental results of fitting the data without the classification vector are used as a control, and the consistency calculation of data after adding the classification vector is analyzed specifically. In the experiment, 60% of the three artificial log samples are used as the training set, and the remaining 40% are used as the test set. To evaluate the fitting effect, we use three fitting metrics: MSE, MAE, and coefficient of determination (R^2).

The experimental results of the three different logs are shown in Fig. 7, 8 and 9. In the three different logs, the fitting effect of the various measurement methods after adding the classification vector is significantly improved than the one without adding the classification vector. The R^2 score of log1 added to the classification vector is 2 percentage points higher than that of the unadded vector, and log2 is 4 percentage points higher. The other two measurement methods

have different degrees of improvement, showing the superiority of the proposed methods.

This work compares the conformance checking technology based on token replay in [33] and the proposed method to evaluate their differences. We still use MAE, MSE, and R^2 metrics to evaluate and calculate the difference between the proposed method and the alignment, and the difference

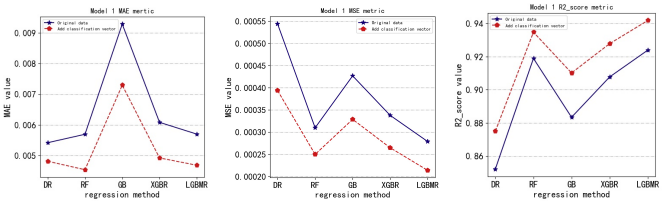

Fig. 7. Fitness for Model 1, PCA=40.

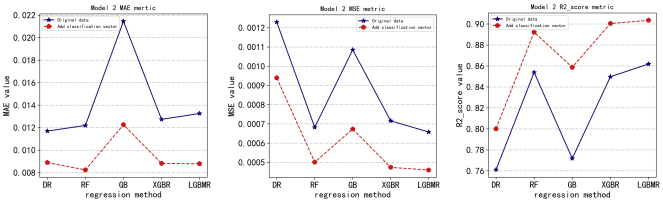

Fig. 8. Fitness for Model 2, PCA=40.

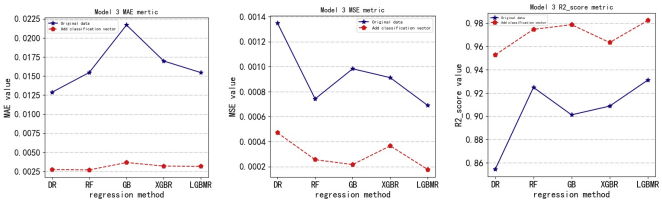

Fig. 9. Fitness for Model 3, PCA=40.

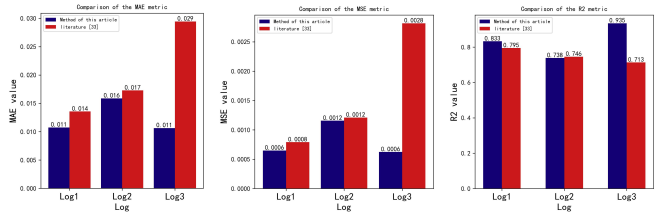

Fig. 10. Comparison of manual event log results.

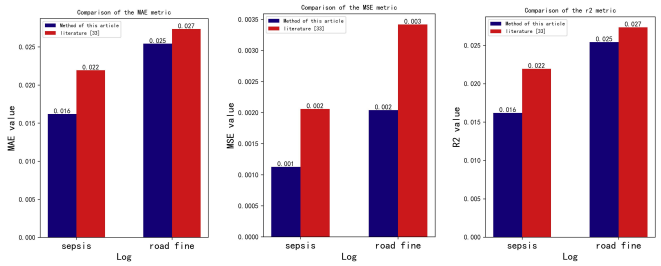

Fig. 11. Comparison of real event log results.

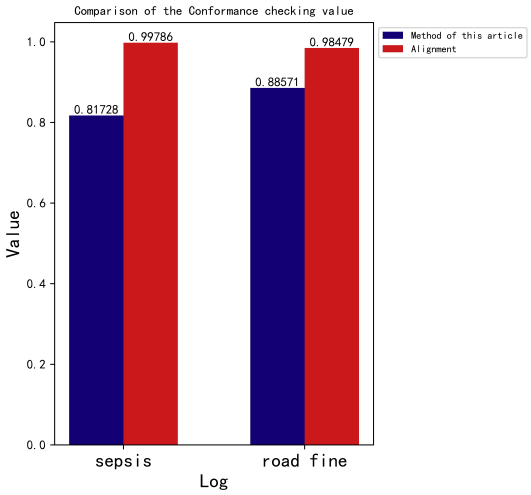

Fig. 12. Comparison of real event log Conformance checking results.

between [33] and the alignment separately. We can obtain a better approximation effect than the consistency of [33] when we use the proposed method to train the training set of 5% manual logs. The comparison between the proposed method and the method in [33] on the synthetic log results is shown in Fig. 10. The comparison between the proposed method and the method in [33] on the real log is shown in Fig. 11. The proposed method is found to obtain lower MAE and MSE than the method in [33] on real and synthetic logs for 5% of the training set and obtain higher R^2 scores. Our method is slightly less effective than the artificial logs for real logs because the artificial logs are complete logs and the real logs are noncomplete logs. Our method cannot learn more behavior from fewer training logs. After evaluating different artificially generated and real event logs, we verify that the proposed method can be used to evaluate the consistency of other traces in the logs when the consistency of some traces in the logs is known and can be closer to the consistency with the real ones than other methods.

Probabilistic log-driven stochastic process conformance checking results are shown in Fig. 12. As seen in the figure, the results calculated using the probabilistic conformance checking technique may detect more inconsistent behaviors, resulting in a lower conformance score.

VI. Conclusion and Future Work

In this paper, we propose a novel method to calculate the trace approximate conformance checking value based on logs The proposed method integrates traditional machine learning in obtaining conformance checking, thereby enabling the calculation of the trace conformance checking value without a systematic reference model and extending the breadth of existing studies. It provides a new approach to improve the efficiency and accuracy of conformance checking, reducing the difficulty of conformance verification in complex processes, enhancing the management efficiency of business processes, and potentially lowering costs and risks. However, this method has some limitations. The proposed method belongs to supervised learning. The fitness of the training samples must be

determined in advance to perform related machine learning and fitting operations. Therefore, the format of the data set has certain requirements. No enterprise-level, large-scale benchmark case library is used at present.

The proposed machine learning calculation method for the approximate conformance checking value can be further applied to change mining and business process prediction and monitoring. As the business system progresses over time, many changes, such as software maintenance, business fusion, and other factors, are inevitably introduced. Detection of log behavior deviation and verification through machine learning and deep learning without a reference model are necessary. Therefore, machine learning and deep learning will be used to detect the behavior changes of logs in future studies. Related analysis and discussion are important branches in the study of process mining, and they are future extension of the work in this paper.

ACKNOWLEDGMENT

This work was supported by the National Nature Science Foundation of China (No.61902002), and the National Key R&D Program of China (Nos. 2023YFC3807500 and 2023YFC3807501).

We also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] W. M. van der Aalst, *Process mining*, Communications of the ACM, vol. 55, no. 8, pp. 76-83, 2012.

[2] J. Carmona, B. van Dongen, A. Solti, and M. Weidlich, *Conformance checking*, Springer, 2018.

[3] G. Governatori, Z. Milosevic, and S. Sadiq, *Compliance checking between business processes and business contracts*,in 2006 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06), 2006, pp. 221-232: IEEE.

[4] A. Awad, G. Decker, and M. Weske, *Efficient compliance checking using BPMN-Q and temporal logic*,in International Conference on Business Process Management, 2008, pp. 326-341: Springer.

[5] W. M. van der Aalst, H. De Beer, and B. van Dongen, *Process mining and verification of properties: An approach based on temporal logic*,in OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", 2005, pp. 130-147: Springer.

[6] J. Peeperkorn and J. Weerdt, *Supervised Conformance Checking Using Recurrent Neural Network Classifiers*, in International Conference on Process Mining, 2020, pp. 175-187: Springer.

[7] M. De Leoni, F. M. Maggi, and W. M. van der Aalst, *Aligning event logs and declarative process models for conformance checking*,in International Conference on Business Process Management, 2012, pp. 82-97: Springer.

[8] A. Rozinat and W. M. Van der Aalst, *Conformance checking of processes based on monitoring real behavior*,Information Systems, vol. 33, no. 1, pp. 64-95, 2008.

[9] W. Van der Aalst, A. Adriansyah, and B. F. van Dongen, *Replaying history on process models for conformance checking and performance analysis*,Wiley Interdisciplinary Reviews: Data Mining Knowledge Discovery, vol. 2, no. 2, pp. 182-192, 2012.

[10] A. Rozinat and W. M. van der Aalst, *Conformance testing: measuring the alignment between event logs and process models*. Citeseer, 2005.

[11] A. Burattin, F. M. Maggi, and A. J. E. s. w. a. Sperduti,*Conformance checking based on multi-perspective declarative process models*,vol. 65, pp. 194-211, 2016.

[12] M. F. Sani, S. J. van Zelst, and W. M. van der Aalst, *Conformance checking approximation using subset selection and edit distance*,in International Conference on Advanced Information Systems Engineering, 2020, pp. 234-251: Springer.

[13] M. F. Sani, M. Kabierski, S. J. van Zelst, and W. M. van der Aalst, *Model Independent Error Bound Estimation for Conformance Checking Approximation*,arXiv preprint arXiv:.13315, 2021.

[14] M. Bauer, H. Van der Aa, and M. Weidlich, *Estimating process conformance by trace sampling and result approximation*,in International Conference on Business Process Management, 2019, pp. 179-197: Springer.

[15] M. Bauer, H. van der Aa, and M. Weidlich, *Sampling and approximation techniques for efficient process conformance checking*,Information Systems, vol. 104, p. 101666, 2022.

[16] M. F. Sani, J. J. G. Gonzalez, S. J. van Zelst, and W. M. van der Aalst, *Conformance checking approximation using simulation*,in 2020 2nd International Conference on Process Mining (ICPM), 2020, pp. 105-112: IEEE.

[17] L. Padró and J. Carmona, *Approximate computation of alignments of business processes through relaxation labelling*,in International Conference on Business Process Management, 2019, pp. 250-267: Springer.

[18] A. Awad, K. Raun, and M. Weidlich, *Efficient Approximate Conformance Checking Using Trie Data Structures*,in 2021 3rd International Conference on Process Mining (ICPM), 2021, pp. 1-8: IEEE.

[19] X. Guo, X. Fang, and G. Mao, *Online Approximate Conformance Checking*,in 2021 International Conference on Cyber-Physical Social Intelligence (ICCSI), 2021, pp. 1-6: IEEE.

[20] A. Baumgrass, T. Baier, J. Mendling, and M. Strembeck, *Conformance checking of RBAC policies in process-aware information systems*,in International Conference on Business Process Management, 2011, pp. 435-446: Springer.

[21] A. Bejaoui, K. Elkhalil, A. Kammoun, M. S. Alouni, and T. Al-Naffouri, *Improved design of quadratic discriminant analysis classifier in unbalanced settings*, arXiv preprint arXiv:.06355, 2020.

[22] Y. Freund and R. E. Schapire, *Schapire R: Experiments with a new boosting algorithm*,in in: Thirteenth International Conference on ML, 1996: Citeseer.

[23] A. Sherstinsky, *Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network*,Physica D: Nonlinear Phenomena, vol. 404, pp. 132306, 2020.

[24] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, *Learning phrase representations using RNN encoder-decoder for statistical machine translation*,arXiv preprint arXiv:. 2014.

[25] A. Liaw and M. J. R. n. Wiener, *Classification and regression by randomForest*, vol. 2, no. 3, pp. 18-22, 2002.

[26] D. M. Powers, *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*,arXiv preprint arXiv:.16061, 2020.

[27] D. Chicco, M. J. Warrens, and G. Jurman, *he coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation*,PeerJ Computer Science, vol. 7, pp. e623, 2021.

[28] A. Berti, S. J. van Zelst, and W. van der Aalst, *Process mining for python (PM4Py): bridging the gap between process-and data science*,arXiv preprint arXiv:.06169, 2019.

[29] F. Mannhardt, *Sepsis cases - event log*,2016. https://data.4tu.nl/repository/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460

[30] M. De Leoni and F. Mannhardt, *Road traffic fine management process*,2015. https://data.4tu.nl/repository/uuid: 270fd440-1057-4fb9-89a9-b699b47990f5

[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, et al. *Pytorch: An imperative style, high-performance deep learning library*,vol. 32, pp. 8026-8037, 2019.

[32] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, et al. *Scikit-learn: Machine learning in Python*,vol. 12, pp. 2825-2830, 2011.

[33] A. Berti and W. M. van der Aalst, *A Novel Token-Based Replay Technique to Speed Up Conformance Checking and Process Enhancement*,Trans. Petri Nets Other Model. Concurr., vol. 15, pp. 1-26, 2021.

[34] A. Rogge-Solti, W. van der Aalst and M. Weske, *Discovering stochastic Petri nets with arbitrary delay distributions from event logs*,in: BPM Workshops, 2013, pp. 15–27.

[35] S. J. Leemans, W. Van der Aalst, T. Brockhoff , A. Polyvyanyy, *Stochastic process mining: Earth movers' stochastic conformance*, Information Systems, 2021, vol. 102, pp. 101724.