J Grid Computing (2022) 20: 25
https://doi.org/10.1007/s10723-022-09613-2

Conformance-oriented Predictive Process Monitoring in BPaaS Based on Combination of Neural Networks

Jiaojiao Wang · Victor Chang · Dongjin Yu · Chang Liu · Xiaoyu Ma · Dingguo Yu

Received: 31 August 2021 / Accepted: 3 June 2022 / Published online: 19 July 2022

Abstract As a new cloud service for delivering complex business applications, Business Process as a Service (BPaaS) is another challenge faced by cloud service platforms recently. To effectively reduce the security risk caused by business process execution load in BPaaS, it is necessary to detect the non-compliant process executions (instances) from tenants in advance by checking and monitoring the conformance of the executing process instances in real-time. However, the vast majority of existing conformance checking techniques can only be applied to the process instances that have been executed completely offline and only focus on the conformance from the single control-flow perspective. We develop an extensible multi-perspective conformance measurement method to address these issues first and then investigate the predictive conformance monitoring approach by automatically constructing an online multi-perspective conformance prediction model based on deep learning techniques. In addition, to capture more decisive features in the model from both local information and long-distance dependency within an executed process instance, we propose an approach, called CNN-BiGRU, by combining Convolutional Neural Network (CNN) with a variant and enhancement of Recurrent Neural Network (RNN). Extensive experiments on two data sets demonstrate the effectiveness and efficiency of the proposed CNN-BiGRU.

Keywords Cloud security · Multi-perspective conformance · Convolution neural networks · Gated recurrent unit · Conformance-oriented predictive process monitoring

J. Wang · C. Liu · X. Ma · D. Yu
Institute of Intelligent Media Technology, Communication University of Zhejiang, Hangzhou, China

J. Wang · C. Liu · X. Ma · D. Yu (✉)
Key Lab of Film and TV Media Technology of Zhejiang Province, Hangzhou, China
e-mail: yudg@cuz.edu.cn

V. Chang (✉)
Department of Operations and Information Management, Aston Business School, Aston University, Birmingham, UK
e-mail: victorchang.research@gmail.com

D. Yu
School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China

1 Introduction

Cloud computing proved to change the supply of computing, storage, and software services. It provides users with computing resources on demand through a pay-per-use approach to offer flexible IT solutions [1]. Since the formulation of cloud security policies always lags behind the use of cloud services, cloud services have many security risks. Generally, cloud service vendors provide three main types of services, software as a service (SaaS), platform as a

service (PaaS), and infrastructure as a service (IaaS). In terms of SaaS, the service provider deploys application software on remote servers and users use these application services via the Internet by paying for them [2]. As we know, the daily operations of organizations and enterprises mainly rely on automated business processes executing on IT infrastructure. On the one hand, the introduction of business process management (BPM) into an enterprise can cause integration problems because not all software involved in a business process (BP) has clear interfaces. On the other hand, enterprises do not need to buy and maintain expensive servers and Process-aware Information Systems (PAIS) to manage and perform their business processes. Accordingly, a new type of SaaS paradigm, business process as a service (BPaaS), has emerged [3, 4]. It is delivered via the Internet through a cloud-based business process (model) execution [5, 6]. In BPaaS, the business process models are usually developed by service providers for tenants to pay for. Since the business process execution is susceptible to resource availability and the configuration of information systems, the process instance executed in the real world may deviate from the defined business process. These non-compliant process executions can bring serious consequences, and most of them prove to be invalid and meaningless at the end of them. The malicious attack posing as a legitimate tenant can especially initiate a large number of non-compliant process executions to attack BPaaS by increasing the process execution load. Accordingly, these invalid and non-compliant process executions waste the resources of cloud services and may result in the security risk of business process execution load in BPaaS. Therefore, it is particularly necessary to predictively monitor the conformance of process executions when they are ongoing [7] to enhance the security of process execution load in BPaaS. Inspired by the definition of *Predictive (business) Process Monitoring* (PPM), which purpose is to predict the future state of an executing process instance [8], we propose the concept of predictive conformance monitoring for business process executions (i.e. conformance-oriented predictive process monitoring, conformance-oriented PPM) in this paper.

Until now, the existing conformance-oriented monitoring technologies mainly focus on the measurement value of conformance by *Conformance Checking*. Conformance checking technologies are designed to find and measure behavioral deviations from actual process executions (that denotes the real behavior) against a predefined process model (that denotes the expected behavior). So the conformance of a process instance only can be determined when it is completed. Accordingly, traditional conformance checking techniques are mostly offline, and they cannot determine in real-time whether the process instance is consistent with the predefined process model. Driven by intelligent business processes management and deep learning techniques, people would like to know whether the process deviates at runtime, rather than a few days later or even longer [9]. Subsequently, some online conformance checking techniques are proposed based on the executed business activities and some behavioral patterns [10, 11]. However, they can only perceive the current conformance of an executing process instance. Different from them, our proposed predictive conformance monitoring for an ongoing process execution (instance) can perceive its future (final) conformance-oriented situation in real-time and take actions to terminate the process instances that are unlikely to be compliant in advance for enhancing the security of the process execution load in BPaaS. In addition, when monitoring the compliance of process execution in BPaaS, the conformance of a process instance should be considered from multiple perspectives rather than a control-flow perspective (the order in which certain activities are performed) because the deviations can not only occur in the control-flow. The deviations from other perspectives such as data, resources, and time can also result in invalid and meaningless process execution. Nevertheless, most existing techniques only study the conformance of control-flow, such as [12–15]. As indicated in [16], it is necessary to propose additional conformance checking techniques from multiple perspectives. Thus, we innovatively propose the *Multi-perspective* (i.e., control-flow, data, resource, and time) *Conformance-oriented Predictive (Business) Process Monitoring* (MCPPM) for tenant-oriented business process executions to reduce the security risk of business process execution load and the tenants' cost in BPaaS.

To achieve the MCPPM task in terms of enhancing the security of process execution load in BPaaS, inspired by the research of PPM on the other tasks

[17–20], we investigate how to predict the multi-perspective conformance of an executing case on the basis of the historical process executions and a predefined process model in BPaaS. Meanwhile, we view this MCPPM problem as a binary classification task ground on a certain multi-perspective conformance threshold. In order to solve this problem, the general approach usually includes two parts by referring to the solution of the outcome prediction for a process instance in [18]. The first one is *offline* part, during which we study the relationship between the historical executed process instances and their multi-perspective (control-flow, data, and resource, etc.) conformance and then build a predictive classification model. The second one is *online* part, during which we forecast the future multi-perspective conformance for an executing case based on this model. Consequently, the effectiveness and efficiency of online real-time predictions are crucial for this task. Driven by the relevant work [17, 20–22], we develop an approach based on deep learning techniques. One reason is that the prediction approaches based on conventional machine learning are less intelligent and have deficiencies in terms of prediction performance, especially the efficiency of online prediction, as demonstrated in [17, 22]. Moreover, considering that an executed process instance consists of a series of events, the MCPPM can be viewed as a sequential data prediction problem. As for this similar problem, some approaches based on Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) prove to be more efficient [23–25] because RNN-based approaches pay more attention to the features between the context of elements in sequential data, while CNN can extract local features [21]. However, the RNNs have obvious disadvantages in terms of gradient vanishing and long-distance dependency feature extraction. To address this issue, two variants of Gated RNN, RNN with a Long Short-term Memory unit (LSTM) [26] and RNN with a Gated Recurrent Unit (GRU) [27] are proposed subsequently while the GRU has some advantages in computational complexity [28]. Besides, some other enhancements of neural networks prove to be more efficient regarding the prediction performance, such as the bidirectional neural networks. Meanwhile, the MCPPM task is relatively complex because the multi-perspective conformance of a case is related to some of the events and the contextual dependencies between these events and the attributes of these events the local dependencies between these attributes. Accordingly, in this paper, we integrate the bidirectional enhancement of GRU with CNN to automatically build a prediction model for MCPPM to enhance the security of process execution load in BPaaS. Here, the purpose of using CNN is first to extract representative attribute features by aggregating attributes of events, while the purpose of BiGRU is to extract more temporal relation features such as the long context dependencies from these events.

In summary, this article makes the following contributions:

- We innovatively propose a conformance-oriented predictive process monitoring solution based on deep learning techniques, which aims to enhance the cloud security of the process execution load in BPaaS by detecting the future non-compliant process execution in advance and taking action.
- We take into consideration the conformance of an executed process instance from multiple perspectives (such as the control-flow, data, resource, and time) rather than just considering the control-flow, and then propose an extensible multi-perspective conformance measurement method.
- We focus on the relationship between historical process executions and their multi-perspective conformances and then put forward the CNN-BiGRU approach that can aggregate attribute features of events and progressively extract temporal features among events to forecast the future multi-perspective conformance of an executing process instance.

The remainder of this article is organized as follows. We first introduce the related work and make a brief discussion in Section 2. Then, in Section 3, we outlined some of the basic concepts and the problem we are trying to solve, as well as the architecture of our proposed approach. Afterward, Section 4 introduces our proposed CNN-BiGRU approach and Section 5 discusses the experiments and their results. Finally, we make a conclusion and outlook on some of the research questions in Section 6.

2 Related Work

2.1 Conformance Checking

As an important part of *Process Mining*, the purpose of *Conformance Checking* is to detect and measure the difference between the business process executions in the real world and the corresponding process models that set expected behaviors. Generally speaking, conformance checking for a certain process requires the input of the corresponding process execution log and the predefined process description. Most current research concentrates on the process models represented in graphical language and views them as technical descriptions of a business process [16]. Accordingly, given a predefined process model of a certain process and the corresponding process executions, the conformance of this process can be calculated by adopting or designing an algorithm to compare both of them. The related research mainly focuses on the two algorithm types, one is *log replay* and the other is *trace alignment*.

The algorithm based on *log replay* aims to replay the trace of each process instance against a predefined process model and then utilize different measurement methods to define and calculate a conformance metric. One of the most commonly used is the log replay based on tokens [29]. Unlike it, Leemans et al. [15] designed another log replay algorithm. The input process model can be transformed into deterministic finite automata (DFAs) firstly and then the event log is replayed on the allowed execution traces of the automata. In addition, Adriansyah et al. [30] presented a cost-based replay approach, which computes the conformance according to the total costs of the arc in Petri net and the insertion or skipping of activities. Similarly, Munoz-Gama et al. [31] investigated another log replay approach for large event logs by dividing a process model into single-entry single-exit sub-models firstly and then replaying each part of the log on its corresponding sub-model. These approaches are only employed for conformance checking from the control-flow (i.e., the order in which activities are performed) perspective because they are all developed according to the structure of the process model. Besides, as for the multi-perspective conformance checking, Burattin et al. [32] designed another log replay method for conformance checking in terms of declarative process models, in which an interpreter is designed to extract Linear Temporal Logic (LTL) constraints from these declarative process models firstly, and then these constraints are employed to check the conformance of executed cases.

The algorithm based on *trace alignment* aims to convert the input used for conformance checking into event sequences and then align them as much as possible. Up to now, a lot of conformance checking techniques based on trace alignment have been proposed. The original trace alignment was developed by a series of heuristic steps such as calculating the score matrix, constructing the guide tree, evaluating and pruning the alignment [33]. Based on this, Adriansyah et al. [30] firstly developed an approach based on the A* algorithm. Meanwhile, they defined a cost function to evaluate each alignment for retrieving the optimal one. In addition, some other measurements of the cost function are available such as distance and legal moves. Besides, Song et al. [12] presented an alignment method based on a heuristic algorithm and divide-and-conquer strategy. As for the multi-perspective conformance checking, Mannhardt et al. [34] put forward a balanced multiple-perspective alignment approach to align contextual data and resources, Alizadeh et al. [35] developed an approach on the basis of a data CRUD matrix that is constructed from a process execution log and the corresponding process model. Similarly, De Leoni et al. [36] employed causal nets with data from BPMN process models to create multi-perspective alignments. Besides, the multi-perspective conformance checking can be taken as an Integer Linear Programming problem and a cost function of alignment can be defined from the perspectives of resource, data, and time [37].

In comparison, the "trace alignment"-based approaches are more extensible in terms of the other perspectives of resource, data, and time rather than the single control flow. After comparing the process execution log with the corresponding process model, a metric is required to define and evaluate their conformance. Usually, four quality metrics are available, including *fitness*, *precision*, *simplicity*, and *generalization* [38]. Among them, the closest concept to conformance is the *fitness*, which denotes the proportion of executed process instances that can successfully replay on the corresponding process model. Accordingly, some different definitions of *fitness* including

the token-based and the cost-based are proposed in [29, 30].

Of all related approaches, the closest to our work is online conformance checking. As for online conformance checking, Burattin [39] developed an approach ground on the behavioral difference between a business process model and a process execution log. Moreover, they proposed an online conformance checking framework based on the Transition Systems (TS) that are extracted from the process models modeled by Petri nets [9]. After that, they employed the behavior patterns to denote the business process and then implemented a framework to detect the compliance between these patterns and the process executions [10]. Additionally, Zelst et al. proposed an online conformance checking approach based on the event stream of an executing process instance and the prefix-alignment [11]. Unlike the online conformance checking, we propose a more forward-looking and meaningful framework to online forecast the future conformance for an executing process instance. This framework can provide a predictive measurement of conformance before a process execution completes. In this way, the process executions that will not conform to the predefined process model in BPaaS can be terminated in advance.

2.2 Predictive (Business) Process Monitoring

Process executions always change due to the dynamic execution settings and external conditions (i.e., law, regulation, and policies), especially in the BPaaS application. As described above, these traditional monitoring methods are mostly passive because deviations are only identified after they have occurred rather than prevent these deviations in the first place. In order to address this issue, PPM techniques are proposed to predict the future status of an executing process instance (case). Generally, the PPM techniques consist of two parts, the first one is *offline training* where one or more predictive models can be constructed ground on a completed process execution log, and the second one is *online prediction*, where the established prediction model(s) is(are) employed to make relevant predictions about the process instances being executed.

The existing vast majority of PPM techniques can be grouped into three categories according to prediction content, including time prediction [17, 40, 41], outcome prediction [18, 42–45], and the next activity (sequences) prediction [19, 21, 41, 46–48]. Moreover, all approaches mentioned above can be grouped into three categories depending on the used techniques, including those based on an extended process model, traditional machine learning, and deep learning. For example, Rogge-Solti et al. [40] employed a specific Petri-net to capture any duration distributions and then utilized it to forecast the remaining execution time for an ongoing process instance. Furthermore, Lakshmanan et al. [44] provided an algorithm to relate the process instances to an expanded space-based Markov chain and then leveraged existing techniques to predict the possibility of performing an activity in the future. Ferilli et al. [46] extended the process model by using the WoMan framework for activity prediction of process. Whereas Appice et al. [41] investigated a data-centric process execution prediction approach based on the shallow machine learning technique. Leontjeva et al. [45] focused on the trace encoding techniques and then proposed an approach ground on Random Forest classification to predict the outcome of cases. As for deep learning-based approaches, Tax et al. [17] investigated the performance of the LSTM network on the predictive tasks such as the next event to be executed and the complete continuation time of a running case. Pasquadibisceglie et al. [21] transformed the event log into 2D image-like data structures and then employed a CNN network to train a prediction model for the next activity prediction. In addition, Park et al. [49] applied deep neural networks to predict the future performance of a business process based on an event log.

2.3 Discussion

According to the above analysis, traditional conformance checking techniques are offline and delayed so that they can not support the real-time checking of conformance. Although online conformance checking techniques are real-time, they cannot forecast the future conformance for an executing case at the present moment because they only compare the currently executed part trace of an ongoing case with a fixed pattern to obtain the conformance result currently instead of predicting the future conformance by learning the certain feature representative. Additionally, the existing majority of studies only focus on

the conformance as for the single control-flow rather than the comprehensive multi-perspective such as the resource, data, and time.

To achieve the business process conformance monitoring in terms of reducing the security risk of process execution load, inspired by the forward-looking nature of PPM, we first focus on and clarify the problem of MCPPM in this paper. After the investigation of existing PPM studies, we find that some of them are based on deep learning and prove to be more automatic and efficient than the most advanced approaches [17, 20, 21]. These deep learning-based approaches treat the completed case as a series of events (i.e., traces), where each event is recorded as multiple attributes, then adopt encoding methods to transform traces into numerical vectors, and finally build a prediction model for capturing the decisive features by utilizing some neural networks such as the RNN and CNN. Likewise, the future conformance of an executing case can be predicted by a prediction model constructed based on these neural networks because its input is sequential data (i.e., trace), which is just suitable for RNN and LSTM. Thus, we explore the efficient deep learning application in the MCPPM task and then propose an approach by combining CNN with a variant and enhancement of RNN to construct an effective and efficient prediction model.

3 Problem Statement and Overall Architecture

3.1 Measurement of Multi-perspective Conformance

As for a defined business process model in the BPaaS application, the future multi-perspective conformance of an executing process instance can be forecasted based on the historical executed cases for a tenant to enhance process execution load security. Generally, these executed cases for each tenant are recorded in an event log, and each case consists of some event records with several attributes. These attributes indicate the detailed execution (behavior) information of a real-world case, such as the resource, data, and time perspective. By comparing them with the predefined process models with multi-perspective constraints, the consistency between executed cases and a specific process model can be measured from multiple perspectives.

Here, we use a specific *Petri net with constraints* (C_PN), in which the transitions of this Petri net should satisfy certain constraints, to model a specific process from a multi-perspective. In reality, such a process model is determined by the original process model with a control-flow perspective (modeled by Petri net) and the constraints with some other perspectives (e.g., resource, data, and time).

Definition 3.1 (C_PN Process Model) A predefined *process model* that sets the desired behavior can be expressed as a *Petri net with constraints* (C_PN) $M = (P,T,F,V,G_d,G_r,G_t)$, which includes:

- a list of places P;
- a list of transitions T that is labeled with activity name;
- a list of directed arcs $F \subseteq (P \times T) \cup (T \times P)$ that are recorded as flow relation;
- a list of variables V;
- a constraint function $G_d : T \to \mathfrak{D}_V$ that relates a set of logical expressions (i.e., guard) in terms of data perspective to each transition.
- a constraint function $G_r : T \to \Upsilon_V$ that relates a set of logical expressions (i.e. guard) in terms of resource perspective to each transition.
- a constraint function $G_t : T \to \Gamma_V$ that relates a set of logical expressions (i.e., guard) in terms of time perspective to each transition.

As shown in Fig. 1, the process modeled by C_PN can be represented as $T = \{a_1,a_2,a_3,a_4,a_5,a_7,a_8,a_6\}$,$P = \{start,c_1,c_2,c_3,c_4,c_5,end\}$,$F = \{(start,a_1),(a_1,c_1),(a_1,c_2),(c_1,a_3),(c_1,a_2),(a_2,c_3),(c_2,a_4),(a_4,c_4),(a_3,c_3),(c_3,a_5),(c_4,a_5),(a_5,c_5),(c_5,a_6),(a_6,c_1),(a_6,c_2),(c_5,a_7),(a_7,end),(c_5,a_8),(a_8,end)\}$. The control-flow (i.e. structure) of a process defines the order where business activities are performed. Table 1 gives the corresponding guards for each transition of this process from multiple perspectives (i.e., data, resource and time), in which the constraint function is composed of multiple logical expressions with "$\vee$", "$\wedge$", and "$\neg$". If a transition t has no constraints in terms of a certain dimension, we set $G_i(t) = true$ ($t \in T,i \in \{d,r,t\}$). For example, the *guard of data* for transition *examine thoroughly* is $G_d(a_1) = (amount > 2000)$ means that the data constraint of this transition is $amount > 2000$. In other words, the transition *examine thoroughly* can be enabled only if the control structure is satisfied first and

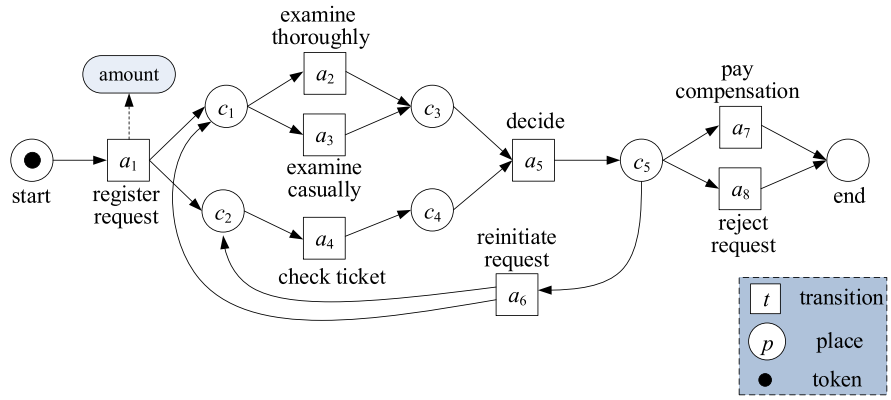

Fig. 1 A flight claim application process modeled by C_PN

Table 1 The guards of multi-perspective, i.e. data, resource, and time

transition	guard of data $G_d(t)$	guard of resource $G_r(t)$	guard of time $G_t(t)$
register request/a_1	*true*	$R_{a_1} \in \{\text{“}Pete\text{”}, \text{“}Mike\text{”}, \text{“}Ellen\text{”}\}$	*true*
examine thoroughly/a_2	$amount > 2000$	$R_{a_2} \in \{\text{“}Sue\text{”}, \text{“}Mike\text{”}, \text{“}Sean\text{”}\} \wedge R_{a_2}^{(i-1)} \neq R_{a_2}^{(i)}$	*true*
examine casually/a_3	$amount \leq 2000$	$R_{a_3} \in \{\text{“}Pete\text{”}, \text{“}Mike\text{”}, \text{“}Sue\text{”}, \text{“}Ellen\text{”}\} \wedge R_{a_3}^{(i-1)} \neq R_{a_3}^{(i)}$	*true*
check ticket/a_4	*true*	$R_{a_4} \in \{\text{“}Pete\text{”}, \text{“}Mike\text{”}, \text{“}Ellen\text{”}\}$	*true*
decide/a_5	*true*	$R_{a_5} \in \{\text{“}Sara\text{”}\}$	*true*
reinitiate request/a_6	*true*	$R_{a_6} \in \{\text{“}Sara\text{”}\}$	$T_{a_6} \leq T_{a_5} + 5days$
pay compensation/a_7	*true*	$R_{a_7} \in \{\text{“}Pete\text{”}, \text{“}Mike\text{”}, \text{“}Ellen\text{”}\}$	*true*
reject request/a_8	*true*	$R_{a_8} \in \{\text{“}Pete\text{”}, \text{“}Mike\text{”}, \text{“}Ellen\text{”}\}$	*true*

then the *amount* of request for flight claim is bigger than 2,000. Likewise, the *guard of resource* $G_r(t)$ specifies the related resource of each transition. In particular, as for transitions of a_2 and a_3, there is another constraint is that the resource for executing each of them in the last time is different from that in the next time when re-initiating a request within an execution of this process. Besides, the *guard of time* $G_t(t)$ defines the time constraints for each transition in addition to the order of transitions described in process model. For transition a_6, the time constraint $T_{a_6} \leq T_{a_5} + 5days$ requires that it must happen within 5 days after the transition a_5 occurs.

Definition 3.2 (Reachable Trace) The *reachable traces* expressed by process model M in terms of control-flow can be represented as $\mathcal{T}(M) \subseteq A_M^*(A_M \subseteq T)$ based on the firing rules of a C_PN. In such a process model, a transition can only be enabled when there are specific tokens at all input places, which is denoted as $\bullet\, t = \{p \in P | (p,t) \in F\}$. Meanwhile, an enabled transition fires (i.e, the activity labeled with a transition can be executed) by consuming tokens at each input place and creating tokens for each output place, which is denoted as $t\bullet = \{p \in P | (t,p) \in F\}$.

Taking the process model described in Fig. 1 for example, a sequence of transitions can be formed based on the above firing rules. At first, the *start* place has a token and it can be fired first and then the transition a_1 can be enabled and it has a token. Afterwards, once a_1 is enabled, each of the two output places c_1 and c_2 has a token. Whereas the place c_1 has only one token so that only one transition a_2 or a_3 can be enabled. In this case, a transition sequence can be generated until the *end* place is fired. As for the process modeled in Fig. 1, we can obtain reachable traces $\mathcal{T}(M) = \{\langle a_1, a_4, a_2, a_5, a_7\rangle, \langle a_1, a_4, a_2, a_5, a_8\rangle, \langle a_1, a_4, a_3, a_5, a_6, a_3, a_4, a_5, a_8\rangle, \dots\}$.

Definition 3.3 (Event, Event Log) An instantiation of a transition is an *event*, which can be defined as a tuple $e = (a,c,t_{start},t_{end},d_1,\dots,d_m)$ where c is the case id representing the specific process execution,

Table 2 Event log of a flight claim application process

caseID	startTimestamp	completeTimestamp	activity	resource	amount	…
1	2010/08/02 10:26:00	2010/08/02 11:06:00	register request/a_1	Pete/r_1	2,000	…
1	2010/08/02 12:12:00	2010/08/03 14:38:00	examine thoroughly/a_2	Sue/r_2	2,000	…
1	2010/08/03 09:24:00	2010/08/09 15:21:00	check ticket/a_4	Mike/r_3	2,000	…
1	2010/08/04 10:49:00	2010/08/04 11:21:00	decide/a_5	Sara/r_4	2,000	…
1	2010/08/08 12:27:00	2010/08/08 12:33:00	reject request/a_8	Pete/r_1	2,000	…
2	2010/08/14 11:11:00	2010/08/14 11:39:00	register request/a_1	Mike/r_3	500	…
2	2010/08/14 12:20:00	2010/08/14 12:31:00	check ticket/a_4	Mike/r_3	500	…
2	2010/08/14 14:20:00	2010/08/14 17:30:00	examine casually/a_3	Pete/r_1	500	…
2	2010/08/16 15:25:00	2010/08/16 16:10:00	decide/a_5	Sara/r_4	500	…
2	2010/08/17 11:25:00	2010/08/17 12:10:00	pay compensation/a_7	Ellen/r_5	500	…
… …	… …	… …	… …	… …	… …	

t_{start} and t_{end} denote the start and complete timestamp respectively, and $d_1, \ldots, d_m (\forall i \epsilon [1, m], d_i \epsilon \mathcal{D}_i)$ denotes a series of additional attributes. All the executed events that are recorded by a process-centered PAIS in BPaaS make up an *event log*. As for the event log L, all occurred events can be represented as a collection of A_L.

Table 2 gives the event log associated with the process model described by Fig. 1. As shown in this table, some events with the same *caseID* indicate that they occur in the same process execution. Meanwhile, the activity attribute of each event associates with a transition in a process model. Generally, the attributes of each event can be divided into two types, *case attribute* and *event attribute*, according to whether their values is distinguished by cases or events. For example, the attributes of *caseID* and *amount* belong to *case attribute* while the attributes of *startTimestamp*, *completeTimestamp*, *activity*, and *resource* belong to *event attribute*. Generally, the value of each *event attribute* and *case attribute* may be numeric data, categorical data, as well as text data. For instance, the value of *activity* and *resource* in Table 2 is categorical data while the value of *amount* is numeric data.

Definition 3.4 (Trace, Prefix Trace) Each process execution (i.e., process instance or case) can generate a non-empty finite time ordered sequence of events, which can be defined as a *trace* $\sigma = \langle e_1, e_2, ..., e_{|\sigma|} \rangle$ with satisfying $\forall i,j \in [1,|\sigma|], e_i \in A, e_j \in A, e_i.c = e_j.c$. For a given trace σ, the preceding part of its event sequence from the beginning represents the executed events at different moments, which can be defined as *prefix trace* $\sigma^l = \langle e_1, e_2, ..., e_l \rangle$ with certain length $l (l \leq |\sigma|)$.

Definition 3.5 (Control-flow Alignment, Optimal Alignment). A *control-flow alignment* between process model M and its executed trace σ can be defined as a sequence of pairs $\{(x, y) | x \in A_M^{\perp}, y \in A_L^{\perp}\}$ where $A_M^{\perp} = A_M \cup \{\text{“}\perp\text{”}\}$ and $A_L^{\perp} = A_L \cup \{\text{“}\perp\text{”}\}$. Here, $A_M^{\perp}$ denotes all the activities (i.e., transition collection T) in this process model and the placeholder “$\perp$” while $A_L^{\perp}$ represents all the activities occurred in the event log and the placeholder “$\perp$”. The possible shifts when they align can be formalized as follows.

- (x,y) denotes a legal *shift* on process model M if $x \in A_M$ & $y =$ “$\perp$”;
- (x,y) denotes a legal *shift* on a trace of event log L if $x =$ “$\perp$” & $y \in A_L$;
- (x,y) denotes a legal *shift* on both process model M and a trace of event log L if $x \in A_M$ & $y \in A_L$;
- (x,y) denotes an *illegal shift* if $x =$ “$\perp$” & $y =$ “$\perp$”.

Given a reachable trace $\sigma_M (\sigma_M \in \mathcal{T}(M))$ from M and a trace σ_L of L, the alignment between them can be defined as a series of *shifts*, $\kappa \in \{\langle (x, y), (x, y), \ldots, (x, y) \rangle | x \in A_M^{\perp}, y \in A_L^{\perp}\}$. In terms of each pair of σ_M and σ_L, many different alignments can be obtained, such as two of them shown below.

$$\kappa_1 = \begin{matrix} a_3 & \perp & a_5 & a_7 & a_9 & a_{11} \\ a_3 & a_4 & \perp & a_7 & a_9 & a_{11} \end{matrix} \quad \kappa_2 = \begin{matrix} a_3 & a_5 & a_7 & a_9 & a_{11} \\ a_3 & a_4 & a_7 & a_9 & a_{11} \end{matrix} \tag{1}$$

To measure these alignments, a cost function about the legal shifts can be expressed as $\delta(\kappa) = \sum_{(x,y)\in\kappa}\delta(x,y)$, in which

$$\delta(x,y) = \begin{cases} 0, & if\ x = y \\ 1, & if\ x = \perp\ or\ y = \perp \\ \infty, & if\ x \neq y \end{cases} \tag{2}$$

To obtain the best complete alignment between process model M and trace σ_L, we define an *optimal alignment* κ as $\forall\kappa' \in K_{\mathcal{T}(M),\sigma_L}, \delta(\kappa') \geq \delta(\kappa)$, in which $K_{\mathcal{T}(M),\sigma_L} = \{\kappa | \exists\sigma_M \in \mathcal{T}(M), \kappa\ is\ an\ alignment\ between\ \sigma_M\ and \sigma_L\}$ because a process model typically has multiple reachable traces. Accordingly, an alignment between a trace σ_L and the optimal aligned reachable trace σ_M can be defined as $\Psi_M(\sigma_L) = \{\kappa \in K_{\mathcal{T}(M),\sigma_L} | \forall\kappa' \in K_{\mathcal{T}(M),\sigma_L}\}$. Based on (2), the cost of this alignment can be represented as $cost(\sigma_L,M) = \delta(\Psi_M(\sigma_L))$.

Definition 3.6 (Control-flow Fitness) As for process model M, the *control-flow fitness* of trace σ_L is introduced to measure its conformance from the control-flow (i.e. structure) perspective. Here, we normalize the *control-flow fitness* in a range of [0,1] by:

$$fitness_{ctrl}(\sigma_L, M) = 1 - \frac{cost(\sigma_L,M)}{|\sigma_L| + min_{\sigma_M\in\mathcal{T}(M)}\sum_{x\in\sigma_M}\delta(x,\perp)} \tag{3}$$

where $|\sigma_L|$ denotes the length of σ_L, $min_{\sigma_M\in\mathcal{T}(M)}\sum_{x\in\sigma_M}\delta(x,\perp)$ denotes the minimum total cost of shifts in process model M only, and $cost(\sigma_L,M)$ is divided by the maximum of the possible cost.

Definition 3.7 (Data Fitness) As for process model M, the *data fitness* of trace σ_L is introduced to measure its conformance from the data perspective in terms of the data constraints described in M. To normalize its value in a range of [0,1], we define it as:

$$fitness_{data}(\sigma_L, M) = \frac{|\{t\in\sigma_L | G_d(t) = true\}|}{|\sigma_L|} \tag{4}$$

where $\{t \in \sigma_L | G_d(t) = true\}$ is the collection of transitions that are also belong to trace σ_L and satisfied the guard function of data $G_d(t)$, $|\{t \in \sigma_L | G_d(t) = true\}|$ indicates the number of transitions while $|\sigma_L|$ indicates the length of σ_L, that is, the number of transitions involved in trace σ_L.

Definition 3.8 (Resource Fitness) As for process model M, the *resource fitness* of trace σ_L is introduced to measure its conformance from the resource perspective in terms of the resource constraints described in M. To normalize its value in a range of [0,1], we define it as:

$$fitness_{res}(\sigma_L, M) = \frac{|\{t\in\sigma_L | G_r(t) = true\}|}{|\sigma_L|} \tag{5}$$

where $\{t \in \sigma_L | G_r(t) = true\}$ is the collection of transitions that are also belong to trace σ_L and satisfied the guard function of resource $G_r(t)$, $|\{t \in \sigma_L | G_r(t) = true\}|$ indicates the number of transitions while $|\sigma_L|$ indicates the length of σ_L.

Definition 3.9 (Time Fitness) As for process model M, the *time fitness* of trace σ_L is introduced to measure its conformance from the time perspective in terms of the time constraints described in M. To normalize its value in a range of [0,1], we define it as:

$$fitness_{time}(\sigma_L, M) = \frac{|\{t\in\sigma_L | G_t(t) = true\}|}{|\sigma_L|} \tag{6}$$

where $\{t \in \sigma_L | G_t(t) = true\}$ is the collection of transitions that are also belong to trace σ_L and satisfied the guard function of time $G_t(t)$, $|\{t \in \sigma_L | G_t(t) = true\}|$ indicates the number of transitions while $|\sigma_L|$ indicates the length of σ_L.

Take the process instance with caseID of 1 in Table 2, its data fitness can be calculated as 4/5(0.8) because the length of this trace is 5 and the $G_d(a_2)$ is false (the amount of a_2 is 2000 which doesn't satisfy $amount > 2000$). Similarly, the resource fitness and the time fitness can be calculated like this.

Definition 3.10 (Multi-perspective Conformance) As for process model M, the *multi-perspective conformance* of trace σ_L can be defined as:

$$Fitness(\sigma_L, M) = \omega_1 * fitness_{ctrl} + \omega_2 * fitness_{data} + \omega_3 * fitness_{res} + \omega_4 * fitness_{time} \tag{7}$$

Here, the weights of ω_1, ω_2, ω_3, and ω_4 satisfy the constraint of $\omega_1 + \omega_2 + \omega_3 + \omega_4 = 1$. To balance the multiple perspectives, each weight can be set equally to 0.25. Besides, (7) can be extended for much more

perspectives rather than the above-mentioned ones, such as the role perspective.

3.2 The Prediction of Multi-perspective Conformance

Definition 3.11 (Multi-perspective Conformance Labeling) As for trace σ_L, its multi-perspective conformance class $y(\sigma_L)$ can be defined as a mapping function $y : (M, \sigma_L, \theta) \rightarrow \{True, False\}$ based on the predefined fitness threshold θ. Here, *True* indicates the process execution is consistent with the process model from multi-perspective while *False* indicates the process execution is completely inconsistent with the process model from multi-perspective. The detailed description is as follows.

$$y(\sigma_L) = \begin{cases} True, & if\ Fitness(\sigma_L, M) \geq \theta \\ False, & otherwise \end{cases} \tag{8}$$

For instance, as shown in Table 2, we can obtain the multi-perspective conformance class $y(\sigma_1)$ for trace σ_1 based on the measurement of multi-perspective conformance and (8).

Definition 3.12 (Attribute Encoding, Event Encoding, Trace Encoding) An *attribute encoding* is defined as a mapping $f_{attr} : attr_i \rightarrow \mathcal{R}^{p_i}(i \in [1, |e|])$ ($|e|$ denotes the number of attributes within event e $(e = \{(attr_1, val_1), (attr_2, val_2), \ldots, (attr_{|e|}, val_{|e|})\})$ that encodes the value of each attribute $attr_i$ as a vector with specific dimensions p_i. In this case, for each event e, its *event encoding* can be also defined as $f_{event} : e \rightarrow \mathcal{R}^p(\mathcal{R}^p = [\mathcal{R}^{p_1} \oplus \mathcal{R}^{p_2} \oplus \cdots \oplus \mathcal{R}^{p_{|e|}}], p = p_1 + p_2 + \cdots + p_{|e|}$, $[\oplus]$ denotes the concatenation of vectors) based on the encoding of each attribute within event e. Similarly, we can define *trace encoding* as a mapping $f_{trace} : \sigma_L \rightarrow \mathcal{R}^{p \cdot |\sigma_L|}$ where $\sigma_L = \langle e_1, e_2, \ldots, e_{|\sigma_L|} \rangle$ and $|\sigma_L|$ is the number of events in trace σ_L.

Definition 3.13 (Prediction Model) A *prediction model* (i.e., a classification model or a classifier) maps a (prefix) trace and its multi-perspective conformance prediction (class) based on trace encoding. It can be defined as $\mathcal{R}^{p \cdot |\sigma_L|} \rightarrow y(\sigma_L)$ where p is the dimension of encoded vector of each event $e(e \in \sigma_L)$.

Problem Statement For a certain process in BPaaS, given its process model M described in C_PN for a tenant, its event log $L = \{\sigma_1, \sigma_2, \ldots, \sigma_s\}$ with s historical executed cases, and an ongoing case $\sigma' = \langle e_1, e_2, \ldots, e_{|\sigma'|} \rangle$ to be predicted, the problem considered in this paper is to forecast the future conformance $Fitness(\sigma', M)$ of each running trace σ' from multiple perspectives, that is, the control-flow, data, resource, and time. This is the so-called *Multi-perspective Conformance-oriented Predictive (Business) Process Monitoring* (MCPPM).

3.3 Overall Architecture

In this paper, we concentrate on the security of process execution load in the BPaaS application and propose a predictive monitoring solution of the multi-perspective conformance. In order to effectively forecast the future final multi-perspective conformance of an executing process instance, we put forward an approach of CNN-BiGRU to establish a prediction model according to a certain process model (combined with a general process structure and some other constraints) and the corresponding historical process executions. The CNN-BiGRU approach, combined with CNN and the variant and enhancement of RNN, aims to determine a specific neural network with the CNN-BiGRU framework and the optimal parameters. Based on it, a prediction model can reveal the impact of the behavior of process executions on its multi-perspective conformance, in which these behaviors are involved in the order where the activities are performed, the resource and time for performing these activities and the process executions are recorded as attribute-value pairs of events. Our proposed solution for the conformance prediction problem mainly includes two parts, *offline* and *online*. The former part aims to construct a prediction model (i.e., classification model or classifier) that indicates the relationship between process executions and their multi-perspective conformance. In contrast, the latter part aims to predict the future real-time conformance of an executing process instance. Figure 2 gives the overview of our proposed solution that includes the following two parts.

Offline Training In this stage, the required inputs are the process model described by C_PN and its historical event log (for a tenant), which contains a series of executed process instances. On the basis of them, we can calculate the fitness of each executed case in

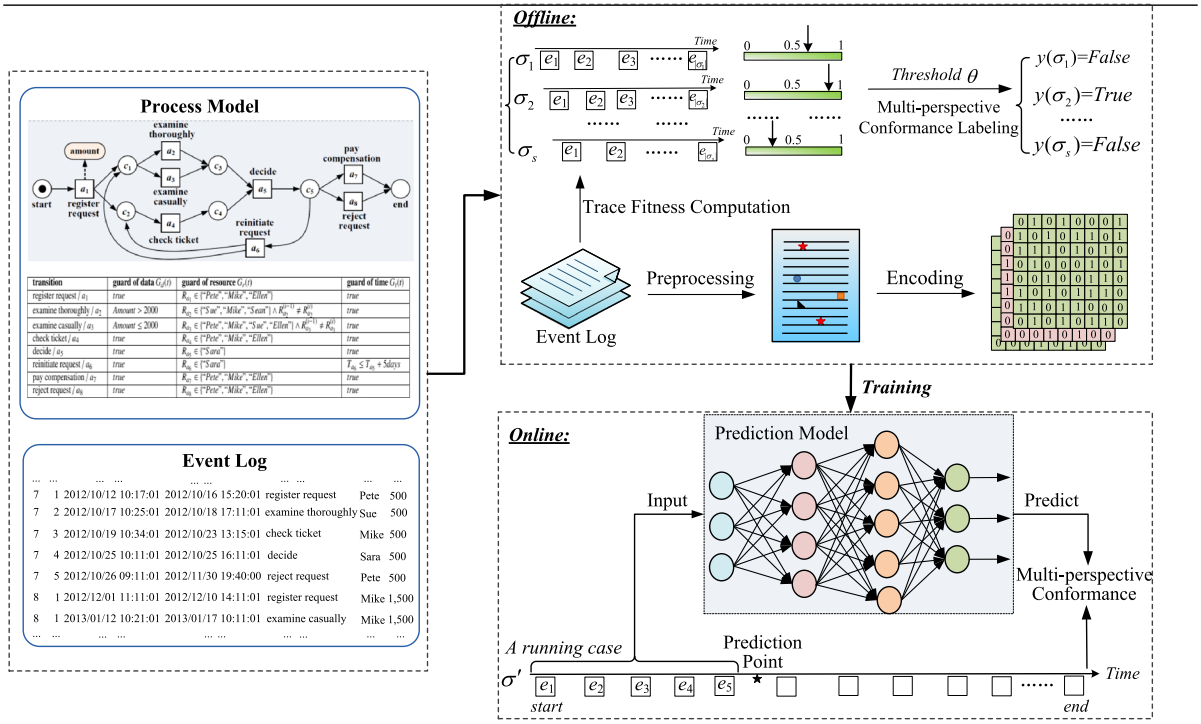

Fig. 2 The overall architecture of our proposal

terms of multiple perspectives. Then, the multi-perspective conformance for each case can be measured based on these fitness values, and each case can also be labeled conformance class based on the predefined threshold. Meanwhile, we preprocess the event log by expanding many more additional attributes for events and then encode each trace according to some coding strategies. Thus, we can obtain the pair of encoded traces and the corresponding multi-perspective conformance class for each case of this event log. A prediction model can be finally obtained through sample training by taking them as the input of a specific neural network.

Online Predicting Based on the built prediction model, the final multi-perspective of an ongoing case can be forecasted. The running case (i.e., a prefix trace) includes some executed events and each of them has a series of pairs of *attribute-value*. By taking this prefix trace as the input of the prediction model, it needs to be encoded and padded with zero according to the length of the encoded vector for historically completed traces. Afterward, the encoded vector of the prefix trace can be input into the prediction model to determine the multi-perspective conformance class of this case.

4 Multi-perspective Conformance-oriented Predictive (Business) Process Monitoring (MCPPM) Approach

The purpose of MCPPM is to forecast the final conformance class of an executing process instance in real-time effectively and efficiently. To solve the multi-perspective prediction problem, we need to first measure the multi-perspective conformance of historical executed cases by comparing it with a certain process model, and then determine their conformance classes (i.e., *True vs. False*). We can then construct a predictive classification model with a specific neural network to extract the relationship between the executed cases and their determined multi-perspective conformance classes. Thus, according to this prediction model, the future conformance class of an executing process instance can be forecasted. To better

introduce our proposed approach, we first describe its whole framework and then describe how to construct a specific neural network based on CNN-BiGRU in detail.

4.1 Measuring the Multi-perspective Conformance of Process Instances

To compute the multi-perspective conformance of an executed case, we propose an algorithm grounded on the process model described by C_PN to measure the conformance from the perspective of the control structure, data, resource, and time. As introduced in Definition 3.1, the process modeled by C_PN can set the expected behavior with some constraints from multiple perspectives, in which the desired behavior of a process in the control-flow aspect is expressed by the transitions T, places P, and flow relations F of C_PN. Similarly, the desired behavior in data, resource, and time perspectives are demonstrated by the guard functions G_d, G_r, and G_t, respectively. Thus, we compute its fitness from these multiple perspectives for each executed process case according to the C_PN process model, respectively. Subsequently, the multi-perspective conformance of an executed process instance can be measured according to these fitness values.

Algorithm 1 Multi-perspective conformance measurement based on C_PN process model.

Input: a C_PN process model $M\langle P, T, F, V, G_d, G_r, G_t\rangle$
an event log $L = \{\sigma_1^L, \sigma_2^L, \cdots, \sigma_s^L\}$
the weight parameter of multiple perspectives $\mathcal{W} = [\omega_1, \omega_2, \omega_3, \omega_4]$
Output: the event log after multi-perspective conformance measurement $L'\langle\sigma^L, Fitness(\sigma^L, M)\rangle$

1: $L'\langle\sigma^L, Fitness(\sigma^L, M)\rangle \leftarrow (\varnothing, 0)$; //initialize the event log L'
2: $\mathcal{T}(M) \leftarrow getReachableTrace(M.P, M.T, M.F)$; //$\mathcal{T}(M) = \{\sigma_1^M, \sigma_2^M, \ldots, \sigma_{|\mathcal{T}(M)|}^M\}$($|\cdot|$ denotes the size)
3: **for** $i = 1, 2, \cdots, s$ **do**
4: $PA\langle(\sigma^M, \sigma^L), cost(\sigma^M, \sigma^L)\rangle \leftarrow (\varnothing, 0)$; //initialize the collection of key-value pair for each trace of L
5: **for** $j = 1, 2, \cdots, |\mathcal{T}(M)|$ **do**
6: $PA \leftarrow getPossibleAlignment(\sigma_i^L, \sigma_j^M)$; //store the possible alignments between σ_i^L and σ_j^M as well as their cost computed by (2)
7: **end for**
8: $((\sigma_i^L, \sigma_{(1)}^M), cost(\sigma_i^L, \sigma_{(1)}^M)) \leftarrow getOptAlignedTrace(PA, \sigma_i^L)$;
9: $cost(\sigma_i^L, M) \leftarrow cost(\sigma_i^L, \sigma_{(1)}^M)$;
10: $fit_{ctrl}(\sigma_i^L, M) \leftarrow getCtrlFit(\sigma_i^L, cost(\sigma_i^L, M), M.P, M.T, M.F)$;
11: $fit_{data}(\sigma_i^L, M) \leftarrow getDataFit(\sigma_i^L, M.G_d)$;
12: $fit_{res}(\sigma_i^L, M) \leftarrow getResourceFit(\sigma_i^L, M.G_r)$;
13: $fit_{time}(\sigma_i^L, M) \leftarrow getTimeFit(\sigma_i^L, M.G_t)$;
14: $Fitness(\sigma_i^L, M) \leftarrow getFit(\mathcal{W}, fit_{ctrl}, fit_{data}, fit_{res}, fit_{time})$;
15: $L'.add(\sigma_i^L, Fitness(\sigma_i^L, M))$;
16: **end for**
17: **return** L';

As analyzed above, we design a multi-perspective conformance measurement algorithm in Algorithm 1. Firstly, we initialize the event log L' that adds the measurement of multi-perspective conformance (line 1). Then, based on the structure of process model M (i.e., $M.T$, $M.p$, and $M.F$) described in C_PN, we get all the reachable traces $\mathcal{T}(M)$ that are allowed to perform in a process model (line 2). After that, for each trace of process case in L, we compute its fitness in terms of multiple perspectives, respectively and then obtain its multi-perspective conformance (lines 3-16). As for the control-flow perspective, we find all of the possible alignments between a trace σ_i^L and each trace σ_j^M of $\mathcal{T}(M)$ and then compute the cost of these alignments based on (2) (lines 4-7). Based on them, we can find an optimal reachable trace $\sigma_{(1)}^M$ that can align

to the trace σ_i^L with the minimum cost from $\mathcal{T}(M)$ (line 8). Meanwhile, we can obtain the cost of trace σ_i^L that is aligned with the process model M based on the minimum cost (line 9). According to (3), in the control-flow aspect, the fitness of trace σ_i^L relative to the process model M can be calculated (line 10). In addition, based on the constraint functions G_d, G_r, and G_t described in the process model M, the fitness of each trace σ_i^L from the data, resource, and time perspectives can be calculated respectively according to (4)-(6) (lines 11-13). After that, the multi-perspective conformance $Fitness(\sigma_i^L, M)$ of trace σ_i^L can be computed on the basis of these fitness and the given weight parameters (line 14). Finally, we add the conformance value for each trace in L' (line 15) and then return the final result L' (line 17).

4.2 Predicting the Multi-perspective Conformance

After the multi-perspective conformance measurement, the key to forecasting the future conformance class for an executing process instance is to establish a prediction model that reveals the relationship between a case and its conformance. The reason is that the multi-perspective conformance of a case is always determined by its execution information. Generally, the actual executed cases are recorded in the event log, where each case consists of a series of events that are logged as many *attribute-value* pairs. Considering the relation between them, the multi-perspective conformance of a case is not only related to some of the events as well as the contextual dependencies between these events, but also to the attributes in these events as well as the local dependencies between these attributes. Therefore, the prediction model to be built requires the ability to learn both the long context dependencies of events and the local dependencies of attributes. However, using traditional machine learning techniques to construct the prediction model is less intelligent. Here, we propose the CNN-BiGRU approach based on neural networks to build a prediction model that performs well.

4.2.1 The Framework of Multi-perspective Conformance Prediction Based on Neural Networks

To forecast the multi-perspective conformance class of an executing process instance in terms of the security of process execution load in BPaaS, here, we describe a framework about how to preprocess the event log for training, how to obtain a prediction model, and how to make predictions about the multi-perspective conformance for a running case. The executed cases recorded as a series of events with attribute-value pairs in the event log can be employed to train a prediction model after their conformance class is determined. In order to obtain a more effective prediction model, the original event log can extend some additional attributes computed from these basic attributes. However, the extended event log cannot be used directly to train a prediction model because each attribute in the event log has different value types, such as numerical data and categorical data. Thus, we adopt some coding strategies to encode them as vectors with a certain dimension before training a prediction model. Taking them as inputs, a prediction model based on neural networks can be trained, and then the future conformance class of an executing process instance can be forecasted through this model.

As shown in Algorithm 2, we design an algorithm to describe a framework that contains the preprocess of the event log, the determination of conformance class, the construction of a prediction model, and the prediction of a running case. Algorithm 2 can be divided into two parts, one is the *offline* part when the historical cases are preprocessed for training, and a prediction model is trained (lines 1-29), and the other is *online* part when the multi-perspective conformance of an executing case is forecasted through the built model (lines 30-31). At the *offline* part, for the executed cases of event log L', we first compute the values of new added *case attributes* (*newAttr*,*val*) according to the given additional case attribute collection *CA* (lines 2-6). Then, for each event of these cases, we compute the values of new added *event attributes* (*newAttr*,*val*) based on the given additional event attribute collection *EA* (lines 8-12). Meanwhile, the new additional attributes are extended in each event $L'.e_{ij}$ of event log L' (line 13). Afterwards, we encode each case for numerical vector $\overrightarrow{v_{\sigma_i^L}}$ with specific dimensions l according to the characteristic of attributes (lines 17-20). For instance, if the value of the attribute is numerical, we standardize it according to the value range of this attribute. For the categorical attribute, we utilize the *one-hot* coding method to transform its value into a vector that consists of 0, 1.

Algorithm 2 Multi-perspective conformance prediction based on neural networks.

Input: the event log $L' = \{(\sigma_1^L, Fitness(\sigma_1^L, M)), (\sigma_2^L, Fitness(\sigma_2^L, M)), \cdots, (\sigma_s^L, Fitness(\sigma_s^L, M))\}$
the conformance threshold θ, the additional case attribute CA and event attribute EA
the neural network with all weight parameters $NN(W, b)$ and its hyper-parameter $HP\langle para, val\rangle$
an ongoing case σ'

Output: the predicted multi-perspective conformance class $\hat{y'}$ of case σ'

1: **for** $i = 1, 2, \cdots, s$ **do**
2: $CaseAttr\langle newAttr, val\rangle \leftarrow (\varnothing, 0)$
3: **for** each $newAttr \in CA$ **do**
4: $(newAttr, val) \leftarrow getCaseAttrVal(newAttr, L'.\sigma_i^L)$;
5: $CaseAttr.add((newAttr, val))$;
6: **end for**
7: **for** $j = 1, 2, \cdots, |L'.\sigma_i^L|$ **do**
8: $EventAttr\langle newAttr, val\rangle \leftarrow (\varnothing, 0)$;
9: **for** each $newAttr \in EA$ **do**
10: $(newAttr, val) \leftarrow getEventAttrVal(newAttr, (L'.e_{ij}))$;
11: $EventAttr.add((newAttr, val))$;
12: **end for**
13: $(L'.e_{ij}).extend(CaseAttr, EventAttr)$;
14: **end for**
15: **end for**
16: $l \leftarrow 0$; $\mathcal{L}\langle idx, \overrightarrow{v}, y(\sigma_{idx}^L)\rangle \leftarrow (0, \varnothing, False)$;
17: **for** $i = 1, 2, \cdots, s$ **do**
18: $\overrightarrow{v_{\sigma_i^L}} \leftarrow getNumCoding(L'.\sigma_i^L)$; $l \leftarrow getAttrDimension(\overrightarrow{v_{\sigma_i^L}})$;
19: $y(\sigma_i^L) \leftarrow isConform(L'.Fitness(\sigma_i^L), \theta)$; $\mathcal{L}.add((i, \overrightarrow{v_{\sigma_i^L}}, y(\sigma_i^L)))$;
20: **end for**
21: $W \leftarrow W_0$; $b \leftarrow b_0$;
22: **for** $i = 1, 2, \cdots, s$ **do**
23: $y(\hat{\sigma_i^L}) \leftarrow getResult(W_0, b_0, NN, \overrightarrow{v_{\sigma_i^L}})$;
24: $Loss(\hat{y_i}, y_i) \leftarrow getLoss(y(\hat{\sigma_i^L}), \mathcal{L}.y(\sigma_i^L))$;
25: $W.update(HP, Loss(\hat{y_i}, y_i))$; $b.update(HP, Loss(\hat{y_i}, y_i))$;
26: **if** $Loss(\hat{y_i}, y_i)$ is convergence **then**
27: break;
28: **end if**
29: **end for**
30: $\overrightarrow{v_{\sigma'}} \leftarrow getFullNumCoding(\sigma', l)$;
31: $\hat{y'} \leftarrow getResult(NN, W, b)$;
32: **return** $\hat{y'}$;

Meanwhile, each case is labeled with a conformance class based on its multi-perspective conformance and the given conformance threshold θ. And the encoded trace and its conformance class can be stored in $\mathcal{L}$ (line 19). By taking these encoded cases as well as their conformance classes as input of a designed neural network, a predictive classification model can be trained. Accordingly, the architecture of neural networks should be determined first, and then the related weight parameters need to be trained by the encoded and labeled event log. Here, we suppose a neural network has been determined. Firstly, we initialize all weight parameters W and b of this neural network randomly (line 21). Next, each pair of the encoded case and conformance class is fed to this neural network for training (lines 22-29). As for each encoded case $\overrightarrow{v_{\sigma_i^L}}$ we compute its result $y(\hat{\sigma_i^L})$ based on the neural network NN with the initialized W_0 and b_0, and then employ a loss function to denote the deviation between the computed result $y(\hat{\sigma_i^L})$ and its real result $y(\sigma_i^L)$ (lines 23-24). Based on the loss, the W and b can be iteratively updated through a loss-based Back-Propagation (BP) algorithm as well as some related parameters *HP*, such as the *batch size*, *learning rate*

and *dropout* (line 25). The above operations are repeated until the loss function converges (lines 26-27). Finally, a prediction model can be obtained according to the final value of weights W and biases b. At *online* part, for an ongoing case σ', we use the obtained prediction model to predict its conformance class according to the executed events with data-payload (lines 30-31). In particular, we encode this case σ' and then fill out its rest part with zero according to the determined dimension l.

4.2.2 The Construction of a Specific Neural Network Based on CNN-BiGRU

To construct a prediction model based on neural networks, we need to determine a specific neural network that can first reveal the relationship between a case and its multi-perspective conformance class. Based on the above analysis, such a neural network requires the ability to learn long context dependencies of events and the local dependencies of attributes. Therefore, the multi-perspective conformance of a case not only relates to some of the events and the contextual dependencies between these events, but also relates to the attributes of these events and the local dependencies between these attributes. Therefore, this paper presents the CNN-BiGRU approach to construct a specific neural network by combining CNN with a variant and improvement of RNN. Figure 3 gives the architecture of this specific neural network. In this figure, we combine a 3-layer CNN (i.e., three convolutional-pooling layers) with a multi-layer BiGRU (i.e., multiple bidirectional GRU layers). The features extracted with CNN and BiGRU can be concatenated to obtain a hybrid feature. Finally, we compute the probabilities of the conformance class for a case. Here, we will give an example to show how the proposed CNN-BiGRU neural network can perform effective feature extraction. Considering an event log $L = \{\sigma_1,\sigma_2,...,\sigma_s\}$ with s cases, one of them can be denoted as $\sigma_t = \langle e_{t1}, e_{t2}, \dots, e_{tn}\rangle (t \in [1,s], n = |\sigma_t|)$, in which $e_{ti}(i \in [1,n])$ denotes the i-th occurred event in this case.

Input Layer Taking the above case σ_t as a input of the CNN-BiGRU, the i-th event is represented as a vector of $\vec{x}_{ti} = [x_{ti,1}, x_{ti,2}, \dots, x_{ti,l}]$ with l-dimension by adopting the event encoding strategy as described in Definition 3.12. Here, l denotes the total length of the event encoded vector based on the attribute encoding strategy. Accordingly, this case can be represented as $\vec{x}_{t1}, \vec{x}_{t2}, \cdots, \vec{x}_{tn}$, and then feature extraction can be carried out by a 3-layer CNN and multi-layer BiGRU, respectively. In particular, as the input of the 3-layer CNN, a padding operation is required to extend the dimension of the input matrix so that the convolution output has the same size as the input. In the subsequent experiments of this paper, we pad with zeros for each trace based on the length of the longest trace when encoding them. In the following description, we assume that n is the length of the longest trace in L.

Attribute Feature Aggregation based on CNN A Convolutional Neural Network extends three additional operations of local filters (i.e., convolution), pooling, and weight sharing based on a simple and fully-connected feed-forward neural network. The CNN shown in Fig. 3 has three pairs of convolution-pooling layers: the *Conv1-Pool1*, *Conv2-Pool2*, and *Conv3-Pool3*, in which each convolution layer utilizes a series of filters to compute the small local context information for each part of the input. Each pooling layer utilizes an optional pooling function, such as the max-pooling, to get the refined local information from the convolutional layer output. In particular, the pooling function can keep translation invariance for some minor differences of the position where some features occur, making sense when we focus on whether a feature appears rather than where it appears. Thus, we use CNN to aggregate the important representative features among the encoded items of attributes in the event log. As for a CNN with multiple layers, after the computations of convolution and pooling many times, a fully connected layer (i.e., the FC Layer is shown in Fig. 3) is finally employed to integrate the feature extracted from all positions. To further demonstrate the application of CNN, Fig. 4 gives a detailed example of the first convolution-pooling Layer in Fig. 3. In Fig. 4, an input case with nine events with 6-dimension item of encoded attributes is considered.

Convolution Layer In a convolutional layer, some filters, i.e. convolution kernels, with different size are applied to extract features. Here, we use K filters with $h \times 1$ size (i.e. a window of h-gram encoded attribute items) to extract the local h-gram features. For this

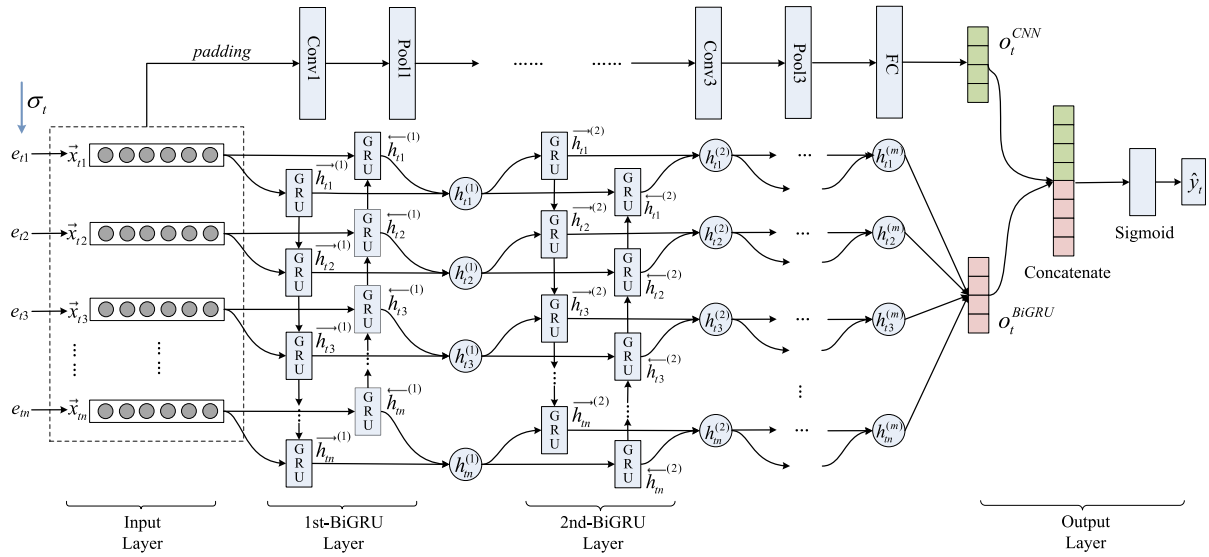

Fig. 3 The architecture of a specific neural network based on CNN-BiGRU

h-gram encoded attribute items, the k-th filter $f^{(k)}(1 \leq k \leq K)$ can generate j-th feature map $c_j^{(k)}$ by:

$$c_j^{(k)} = F_3(W^{(k)} \cdot x_{ti,j:j+h-1} + b^{(k)}) \ (1 \leq j \leq l-h+1) \tag{9}$$

where $c_j^{(k)}$ is the j-th feature map extracted from input matrix X by filter $f^{(k)}$, F_3 denotes *ReLU* activation function, $W^{(k)}$ and $b^{(k)}$ respectively denote the corresponding trainable weight matrix and bias. Accordingly, we obtain all feature maps $c^{(k)} = (c_1^{(k)}, c_2^{(k)}, \ldots, c_{l-h+1}^{(k)})$ after feature extraction through k-th filters.

Pooling Layer Here, we utilize a max-pooling function for each filter output (i.e. feature map) to choose the most important feature by:

$$c_{max}^{(k)} = max\{c_1^{(k)}, c_2^{(k)}, \ldots, c_{l-h+1}^{(k)}\} \tag{10}$$

Accordingly, after all K filters (i.e. convolution kernels) with size h conducted, the extracted features are represented as $c = (c_{max}^{(1)}, c_{max}^{(2)}, \ldots, c_{max}^{(K)})$.

Full-collected Layer After max-pooling operation, all features outputted can be concatenated to represent

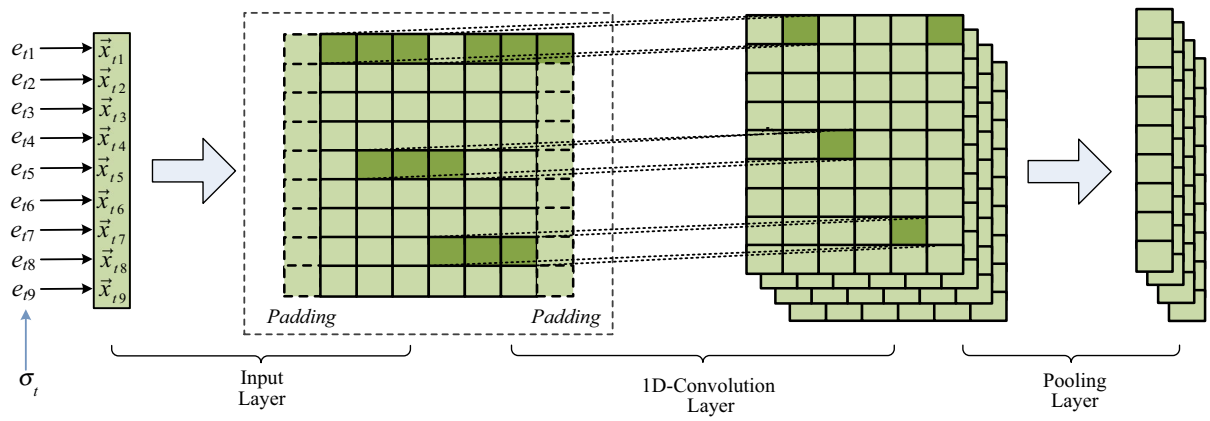

Fig. 4 The overall architecture of the first CNN Layer

the local context h-gram attribute items, which is represented as $o_t^{CNN} = [c_{max}^{(1)} \oplus c_{max}^{(2)} \oplus \cdots \oplus c_{max}^{(K)}]$ ([$\oplus$] is the concatenate operation).

Temporal Relation Extraction based on BiGRU A RNN network [50] originates from a basic feed-forward neural network, which has recurrent hidden states. The hidden state can be activated at each moment, depending on the hidden state of the previous moment. Accordingly, it can handle the variable-length sequential data. However, RNN is difficult to extract long-distance dependencies due to the possible gradient diminishing or explosion. To address this issue, two variants of Gated Recurrent Neural Network, RNN with a Long Short-term Memory (LSTM) unit [26] and RNN with a Gated Recurrent Unit (GRU) [27], are proposed. Both of them have been demonstrated to work well in tasks with sequential input, but the GRU performs better than LSTM when considering a controlled model complexity. Compared with the general RNN, GRU-based RNN adds two gate units in the recurrent hidden state (i.e., cell) so as to extract the dependencies of different time periods adaptively. Here, we adopt a bidirectional improvement and then propose a BiGRU with multiple layers as shown in Fig. 3, in which each BiGRU layer, such as the 1st-BiGRU Layer and the 2nd-BiGRU Layer, consists of a forward propagation layer (i.e., the recurrent direction is consistent with that of events in a case) and a backward propagation layer. Such a BiGRU layer can extract the context information between the preceding events and the current event with the forward propagation layer and the context information between the current event and the subsequence events with the backward propagation layer.

BiGRU Layer Each unit of a BiGRU layer is integrated by the basic GRU unit bidirectionally, i.e. forward and backward, as shown in Fig. 3. Each BiGRU layer extracts the hidden context features from the previous BiGRU layer. For instance, the output of the 1st-BiGRU Layer can be viewed as the input of the 2nd-BiGRU Layer. Taking the first BiGRU layer for example, there are two propagation layers integrated forward and backward. As for the forward propagation layer, a set of hidden states $\overrightarrow{h_{t1}}, \overrightarrow{h_{t2}}, \ldots, \overrightarrow{h_{tn}}$ of GRU units are obtained by updating $\overrightarrow{h_{ti}}$ ground on the input vector of the current event e_{ti} and the hidden state $\overrightarrow{h_{t,i-1}}$ of the previous event $e_{t,i-1}$. Simultaneously, as for the backward propagation layer, a set of hidden states $\overleftarrow{h_{t1}}, \overleftarrow{h_{t2}}, \ldots, \overleftarrow{h_{tn}}$ of GRU units are obtained by updating $\overleftarrow{h_{ti}}$ based on the input vector of the current event e_{ti} and the hidden state $\overleftarrow{h_{t,i+1}}$ of the next event $e_{t,i+1}$. Here, $\overrightarrow{h_{ti}}$ and $\overleftarrow{h_{ti}}$ denote the value of h_{ti} in two directions of respectively. This can be formalized by:

$$\overrightarrow{h_{ti}} = GRU_{fwd}(\vec{x}_{ti}, \overrightarrow{h_{t,i-1}}), i \in [1, n] \tag{11}$$

$$\overleftarrow{h_{ti}} = GRU_{bwd}(\vec{x}_{ti}, \overleftarrow{h_{t,i+1}}), i \in [1, n] \tag{12}$$

$$h_{ti} = [\overrightarrow{h_{ti}} \oplus \overleftarrow{h_{ti}}] \tag{13}$$

where GRU_{fwd} and GRU_{bwd} denote the computations with different inputs of two directions. Similarly, the m-layers BiGRU can be represented as:

$$\begin{aligned}
&\overrightarrow{h_{ti}}^{(1)} = GRU_{fwd}(\vec{x}_{ti}, \overrightarrow{h_{t,i-1}}^{(1)}), \overleftarrow{h_{ti}}^{(1)} \\
&= GRU_{bwd}(\vec{x}_{ti}, \overleftarrow{h_{t,i+1}}^{(1)}), i \in [1, n] \\
&\overrightarrow{h_{ti}}^{(2)} = GRU_{fwd}(\overrightarrow{h_{ti}}^{(1)}, \overrightarrow{h_{t,i-1}}^{(2)}), \overleftarrow{h_{ti}}^{(2)} \\
&= GRU_{bwd}(\overleftarrow{h_{ti}}^{(1)}, \overleftarrow{h_{t,i+1}}^{(2)}), i \in [1, n] \\
&\cdots\cdots \qquad \cdots\cdots \\
&\overrightarrow{h_{ti}}^{(m)} = GRU_{fwd}(\overrightarrow{h_{ti}}^{(m-1)}, \overrightarrow{h_{t,i-1}}^{(m)}), \overleftarrow{h_{ti}}^{(m)} \\
&= GRU_{bwd}(\overleftarrow{h_{ti}}^{(m-1)}, \overleftarrow{h_{t,i+1}}^{(m)}), i \in [1, n]
\end{aligned} \tag{14}$$

Afterwards, the hidden state $h_{ti}^{(m)} = [\overrightarrow{h_{ti}}^{(m)} \oplus \overleftarrow{h_{ti}}^{(m)}]$ in the m-th BiGRU Layer can be obtained, and the extracted features from a m-layer BiGRU can be denoted as $o_t^{BiGRU} = [h_{t1}^{(m)}, h_{t2}^{(m)}, \cdots, h_{tn}^{(m)}]$.

Output Layer In this layer, as for a case σ_t, the extracted features o_t^{CNN} from the 3-layer CNN and o_t^{BiGRU} from the multi-layer BiGRU can be concatenated to obtain a vector $o_t = [o_t^{CNN}, o_t^{BiGRU}]$ firstly. Then, the probability that the multi-perspective conformance class of case σ_t is positive can be calculated by using o_t as the input of the *sigmoid* activation function:

$$\hat{y}_t = sigmoid(W_c \cdot o_t + b_c) \tag{15}$$

where W_c, b_c are the corresponding weight parameters, and the value of $\hat{y}_t$ is in the range of $0 - 1$.

To train the neural network, finally, we determine a binary cross-entropy function that measures the error between the computation result in $\hat{y}_t$ from the

CNN-BiGRU neural network and the actual result y_t for the case σ_t. The detailed loss function is as follows.

$$Loss(\hat{y}_t, y_t) = -((1 - y_t)log(1 - \hat{y}_t) + y_t log\hat{y}_t) \quad (16)$$

Based on the determined CNN-BiGRU neural network and loss function, we can use Algorithm 2 to train a prediction model.

5 Experimental Evaluation

5.1 Experiment Settings

Here, we make a comprehensive comparison with some other deep learning approaches and two typical traditional machine learning approaches to demonstrate the performance of the hybrid CNN-BiGRU approach because no one has proposed other solutions to this problem. The comparative deep learning approaches include the basic RNN, LSTM, and GRU approaches, the corresponding bi-directional improvements Bi-RNN, Bi-LSTM, and Bi-GRU approaches, and the hybrid CNN-RNN, CNN-LSTM, CNN-GRU, CNN-BiRNN, CNN-BiLSTM, and CNN-BiGRU approaches. Moreover, two other traditional machine learning-based approaches, the Gradient Boosted Trees (XGBoost) and the Random Forest (RF), are chosen for comparison because a recent empirical study on 165 datasets show that they are usually better than other traditional machine learning algorithms for classification tasks [51].

NN (Neural Network)-based approaches For further comparison, we develop the RNN approach, LSTM approach and GRU approach by utilizing the original RNN neural network (i.e., RNN approach) and its variants LSTM and GRU neural networks to construct prediction models, respectively. Similarly, we also develop the Bi-RNN approach, Bi-LSTM approach, and Bi-GRU approach by bidirectionally integrating the RNN network, the LSTM network, and the GRU network, respectively. Based on these six approaches, we also develop the other six hybrid approaches by combining with the CNN network, i.e., CNN-RNN, CNN-LSTM, CNN-GRU, CNN-BiRNN, CNN-BiLSTM, and CNN-BiGRU approaches.

RF-based approaches To compare with the traditional machine learning approaches, we choose the RF algorithm to construct a prediction model. As shown in [52], there are some optional operations needed to be determined first, e.g., the bucketing (clustering) and coding for these (prefix) traces extracted from the executed cases. Here, we choose a single bucket method for clustering and two typical methods *laststate* and *aggregation* for encoding. The *laststate* method encodes a (prefix) trace according to the last state (event) information of this trace. In contrast, the *aggregation* method encodes a (prefix) trace by employing an optional aggregation function on this trace. The former only considers the last event of a (prefix) trace, while the latter considers all events of a (prefix) trace but ignores the order of events in this (prefix) trace. Accordingly, two approaches of *RF_single_laststate* and *RF_single_agg* are developed for comparison based on the RF classification algorithm.

XGBoost-based approaches To compare with the traditional machine learning approaches, we also choose the XGBoost algorithm to construct a prediction model inspired by [53]. Similar to RF-based approaches, we develop two approaches *XGBoost_single_laststate* and *XGBoost_single_agg* for comparison.

These 16 approaches mentioned above are employed on two publicly available data sets and then compared from the accuracy and time performance of predictions. For these approaches, we develop them in Python and conduct comparative experiments on a server with three NVIDIA Tesla V100 GPUs and 2 x 12 Intel Xeon 5118 CPU @2.30GHz 256GB memory. Here, we employ these two public event logs to denote the real process execution from tenants in terms of two different processes in BPaaS. To obtain the process modeled by C_PN in BPaaS, in this paper, we use the plug-in from public ProM[1] to mine a process model described by general Petri net and some constraints for each transition from some other perspectives, inspired by [14].

Datasets Two event logs *Traffic Fines* and *BPIC2012* from the public 4TU Centre for Research Data[2] are employed in this experiment. The detailed statistic about these two processed logs is shown in

[1] http://www.promtools.org
[2] https://researchdata.4tu.nl/home/

Table 3 The statistic for the processed event logs *Traffic Fines* and *BPIC2012*

Datasets	#Traces	Positive class(%)	Length range	Truncated length	#Event classes	#Events	#Event attr(s)	#Case attr(s)
Traffic Fines	129,615	97	[2,20]	10	11	460,554	14	4
BPIC2012	11,303	32	[15,159]	40	36	229,868	10	1

Table 3. The *BPIC2012* log records the historical executions for a loan application process and the *Traffic Fines* log mainly includes a set of activities about paying traffic fines and some information related to individual cases, such as the reason and the total amount paid for each traffic fine. At first, we preprocess these logs by removing some noise records. We use ProM to obtain their process model expressed by Petri net with some constraints for each transition from the perspective of data, resource, and time for these preprocessed logs. Based on them, we calculate the multi-perspective conformance of each case through Algorithm 1 and then decide their conformance classes by a defined threshold of $\theta = 0.8$. After that, we extend some additional attributes for each event of these two logs. In addition, we truncate the long cases where the length is greater than a certain one because the long traces can decrease the performance of the prediction model during training. To determine the *truncated length*, we first select the certain conformance class with fewer cases, then group them in ascending order according to the length of cases and find the length at the point of 90%. From Table 3, it is not difficult to discover that the proportion of *positive* and *negative* samples in these two event logs is unbalanced. Especially, the proportion of positive samples of the *Traffic Fines* log is up to 97%. In addition, we also give the corresponding number of *case attribute* and the *event attribute*, respectively. Finally, we encode the executed cases of these event logs according to the value types of attributes.

Evaluation Metrics Generally, the predictive process monitoring techniques expect to obtain an accurate prediction result efficiently in process execution because real-time process monitoring makes sense in terms of the security of process execution load in BPaaS. Therefore, we compare these approaches from the perspectives of *accuracy* and *execution time* when making predictions about the multi-perspective conformance classes. At first, the AUC (the area under the ROC curve), which expresses the probability that a given classifier will rank a positive case higher than a negative one, is determined to measure the prediction *accuracy* [54] because other metrics need to predefine a threshold and the value of threshold has a great influence on the *accuracy*. Furthermore, the ROC curve in AUC can keep unchanged even if the sample ratio is imbalanced. As for *execution time*, two metrics of *offline time* and *online time* are determined for comparison, in which the *offline time* denotes the time it takes to obtain a prediction model while the *online time* denotes the meantime it takes to forecast the multi-perspective conformance (class) for an executing process instance each time.

Implementation Details To simulate the real scenario of multi-perspective conformance-oriented PPM (i.e., the multi-perspective conformance class prediction of an executing case will be made after each event is performed), the processed event logs, including some pairs of an encoded case and its conformance class, are divided into the first 80% *training set* and the last 20% *test set* depending on the time the cases occurred. Furthermore, the *training set* is classified into 80% *training data* and 20% *validation data* randomly to compare the advantages and disadvantages of these approaches after optimization. In other words, we divide these samples of *training set* into two parts, *training data* and *validation data*. The samples of *training data* are employed to train a prediction model, while the samples of *validation data* are viewed as test data to find a set of optimal hyper-parameter combinations for the above-constructed models with the best performance. Since there are many parameters in the NN-based approaches, we choose the *random search* method [55] to make the hyper-parameters optimization. For the above 16 approaches, we set a distribution and value domain for each involved parameter respectively and then initialize these parameters to get a combination of them for optimization in this experiment. In addition,

Table 4 Comparison of prediction accuracy in terms of overall AUC

Approaches	Traffic Fines		BPIC2012		Mean
	original	class weighted	original	class weighted	
RF_single_agg	0.848	0.841	0.767	0.767	0.806
XGBoost_single_agg	0.842	0.851	0.785	0.785	0.816
RF_single_laststate	0.846	0.846	0.695	0.698	0.771
XGBoost_single_laststate	0.835	0.842	0.712	0.698	0.772
RNN	0.854	0.856	0.783	0.789	0.820
LSTM	0.853	0.856	0.793	0.793	0.824
GRU	0.854	0.856	0.803	0.803	0.829
BiRNN	0.854	0.854	0.793	0.793	0.823
BiLSTM	0.842	0.857	0.799	0.799	0.824
BiGRU	0.855	0.857	0.804	0.804	0.830
CNN-RNN	0.856	0.856	0.801	0.800	0.828
CNN-LSTM	**0.858**	0.857	0.797	0.797	0.827
CNN-GRU	0.856	0.858	0.805	0.807	0.831
CNN-BiRNN	0.854	0.855	0.787	0.799	0.824
CNN-BiLSTM	**0.858**	0.858	0.803	0.800	0.830
CNN-BiGRU	**0.858**	**0.860**	**0.808**	**0.808**	**0.833**

for the NN-based approaches, the number of epochs is fixed at 50. Besides, for these approaches CNN-BiRNN, CNN-BiLSTM, and CNN-BiGRU on *Traffic Fines* log, the parameters involved in the CNN part are determined empirically, such as the number of filters in 3 convolution layers is 16, 32, and 64 respectively, the size of the kernel is 3, the stride is 2, and the activation function is *ReLU*.

5.2 Experimental Results

To make online conformance predictions for an executing case in terms of the security in BPaaS, we extract all prefix traces with different lengths for the tenant-oriented historical executed cases from *test set* firstly. Then, we utilize the constructed prediction model based on each approach to forecast the multi-perspective conformance class of each prefix trace. Based on these predictive results, we calculate the AUC values and online prediction time (i.e., *online time*) for each prefix length and each approach on different datasets. Meanwhile, we also calculate the time of training a prediction model (i.e., *offline time*) for each approach on different datasets.

Accuracy Comparison Table 4 gives the overall AUC for each approach on two datasets and the mean overall AUC on two datasets, respectively. In particular, this table also shows the overall AUC for each approach that utilizes a class weight to address the issue of sample imbalance. The overall AUC for an approach on a dataset refers to the weighted average of the AUC values calculated based on the prediction results of all predicted prefix traces with different lengths. Here, the weights are obtained depending on the number of prefix traces with a certain length. As shown in Table 4, it's easy to find our proposed CNN-BiGRU approach outperforms other compared approaches on *Traffic Fines* and *BPIC2012* logs, respectively. Among these 16 approaches, the traditional machine learning-based approaches, i.e., RF-based approaches and XGBoost-based approaches, have the worst performance according to the mean of overall AUC values. Furthermore, *RF_single_laststate* and *XGBoost_single_laststate* have the lower AUC values than other approaches. Thus, we can infer that the reason for this phenomenon is the used encoding method of *last state*. Among these NN-based approaches, in terms of the mean of overall AUC values, the GRU performs best among these basic approaches, followed by LSTM and then RNN. Likewise, among the three bidirectional improvements of them (i.e., BiRNN, BiLSTM, and BiGRU), the BiGRU is also better than the BiLSTM and

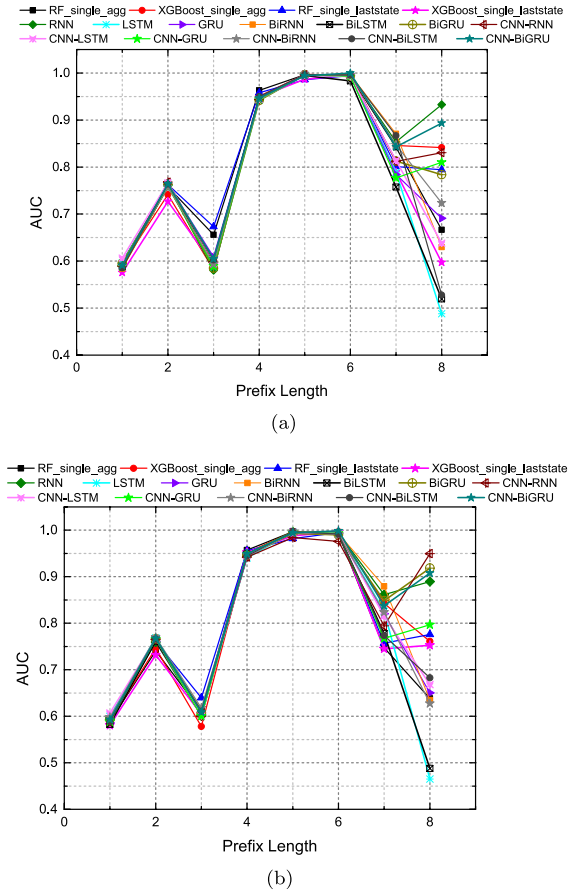

Fig. 5 Comparison of prediction accuracy in terms of AUC on *Traffic Fines* dataset for (**a**) the 16 approaches mentioned above and (**b**) the 16 approaches mentioned above with class weighted

BiRNN approaches. Comparing the BiRNN, BiLSTM and BiGRU with the RNN, LSTM and GRU, respectively, we can find the improved bidirectional approaches perform better indeed. However, among the hybrid approaches, the CNN-RNN approach outperforms the CNN-LSTM, which may be due to the interference of CNN. Similarly, we also find that the CNN-BiRNN is worse than the CNN-RNN. However, the CNN-BiGRU and CNN-BiLSTM still perform better than the CNN-GRU and the CNN-LSTM, respectively. In addition, from the perspective of sample imbalance, these approaches with class weighted have a better performance than the original ones, especially for *Traffic Fines* dataset, which may be because the sample of this dataset is more unbalanced.

For further comparison, Figs. 5 and 6 show the AUC value of making predictions for the prefix traces with different (prefix) lengths. In these subgraphs, the prefix length shown on the x-axis denotes a set of prefix traces that are with a certain length and waiting for prediction. And the corresponding AUC value shown on the y-axis denotes the mean AUC for predicting these prefix traces by using an approach. As for *Traffic Fines* dataset, Fig. 5a and b give the trend of AUC changes with the prefix length increases for the above 16 original approaches, as well as these approaches with class weighted, which is similar to Fig. 6a and b. In Fig. 5, it is easy to find that the changing trend of AUC in subfigures (a) and (b) is similar, but both of them have obvious fluctuations with the prefix length increases, especially at the beginning and end of a case. From a normal point of view, the AUC value should gradually increase as the prefix length increases by considering the larger the prefix length, the more reference information available for prediction. The reason for this phenomenon may be related to the sample imbalance of the dataset. As shown in Figs. 6a and b, at the beginning of cases, the AUC values for different approaches fluctuate significantly in the short term, and they soon tend to increase steadily with the increasing prefix length. Subsequently, it is obvious to identify that the AUC values of these NN-based approaches always keep increasing gradually until most cases are complete. However, the AUC values of *RF_single_laststate* and *XGBoost_single_laststate* begin to fluctuate and decrease once the length of prefix trace is greater than 20. On the one hand, the phenomenon may be caused by the used encoding method in these two approaches. On the other hand, these two approaches may be susceptible to activities that may have a decisive impact on the multi-perspective conformance class when making predictions about an executing case.

In addition, a hypothesis test was further used to evaluate these approaches to demonstrate that the experimental results in this paper are not accidental. Since the predictive performance of these different approaches (classifiers) is for each an ongoing process instance to be predicted, a hypothesis test is required for the predicted results of the test samples

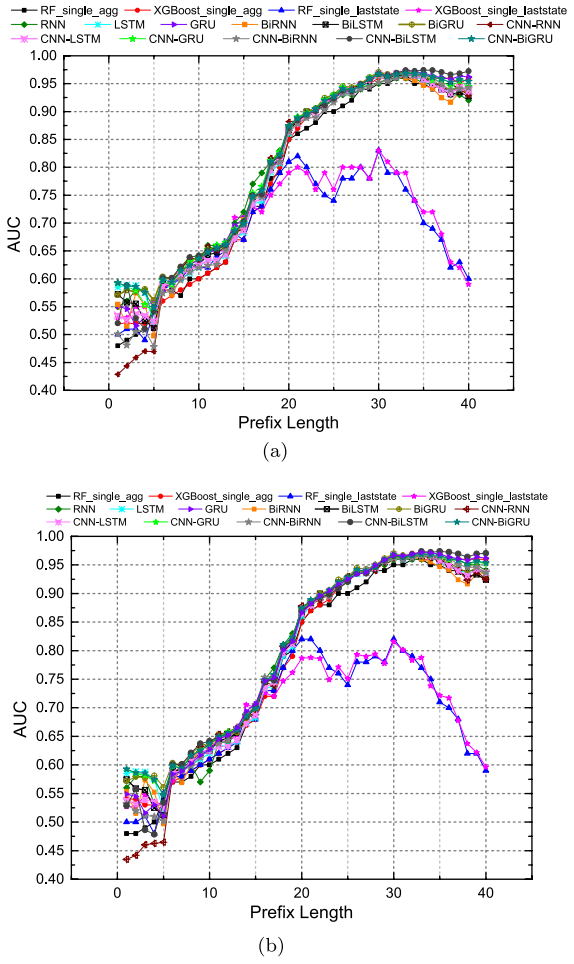

Fig. 6 Comparison of prediction accuracy in terms of AUC on *BPIC2012* dataset for: (**a**) the above-mentioned 16 approaches and (**b**) the above-mentioned 16 approaches with class weighted

of each dataset rather than the dataset. In other words, we can only use a hypothesis test method based on algorithm rank because we compare the performance of multiple approaches on different test samples. Meanwhile, we find that the prediction results with AUC for all test samples under all different approaches (classifiers) do not meet the normal distribution through analysis. Therefore, we choose a non-parametric multivariate hypothesis test, the Friedman test, to refine our evaluation. The *p*-value after the Friedman test was less than 0.05, indicating that there are significant differences between these approaches. However, we could not know which two approaches have the performance differences, and a post-hoc test was further required [56]. The Nemenyi post-hoc test, always used in conjunction with the Friedman test, can demonstrate whether there are significant differences between every two approaches. Therefore, we apply the combined Friedman-Nemenyi test for all test samples and their predicted results of AUC under different approaches (classifiers) in each event log. Then we can calculate the *p*-value (between 0 and 1) between every two approaches among the above 16 approaches. If the value is less than the significance level of 0.05, we can conclude that there is a significant difference between them. Afterward, we find that the *p*-value between most approach pairs is 0.001 (less than 0.05). For example, as for these two event logs, there are significant differences between CNN-BiGRU and BiGRU (*p*-value=0.001), CNN-BiGRU and CNN-GRU (*p*-value=0.001) respectively, which is similar to CNN-BiRNN and BiRNN (*p*-value=0.001), CNN-BiRNN and CNN-RNN (*p*-value=0.001), CNN-BiLSTM and BiLSTM *p*-value=0.001), and CNN-BiLSTM and CNN-LSTM (*p*-value=0.001).

Execution Time Comparison Table 5 gives the *offline* time (in seconds) that is required to train a classification model by using different approaches and the *online* time (in milliseconds) that is required to make predictions about the conformance of a running case (i.e., a prefix trace). First of all, compared with the traditional approaches, we find that the RF-based and the XGBoost-based approaches require less time to build a prediction model (almost within 120 seconds), while the neural network-based approaches require a longer time. In particular, the LSTM approach requires nearly 5,000 seconds on *Traffic Fines* dataset. From the perspective of the datasets, we find that these approaches on *BPIC2012* dataset need less time to construct prediction models. However, in terms of online prediction, the traditional RF-based and XGBoost-based approaches require more time to make predictions based on the built prediction model. As shown in this table, the *online time* required by the neural network-based approaches is less than ten milliseconds, most of which are two milliseconds, which also reflects the lack of efficiency and intelligence in the process prediction and monitoring applications based on the traditional machine learning techniques. Generally, the online prediction time is considered

Table 5 Comparison of prediction efficiency in terms of execution time

Approaches	Traffic Fines		BPIC2012	
	offline time (*s*)	online time (*ms*)	offline time (*s*)	online time (*ms*)
RF_single_agg	117	21	24	9
XGBoost_single_agg	116	16	57	5
RF_single_laststate	117	17	27	3
XGBoost_single_laststate	116	20	27	4
RNN	1,685	5	1,073	2
LSTM	4,937	2	347	2
GRU	1,537	2	789	2
BiRNN	1,553	2	249	11
BiLSTM	1,558	2	948	4
BiGRU	2,417	2	222	2
CNN-RNN	2,687	5	1,077	2
CNN-LSTM	3,662	2	795	2
CNN-GRU	1,634	2	256	2
CNN-BiRNN	2,711	2	982	2
CNN-BiLSTM	2,851	9	665	2
CNN-BiGRU	3,387	2	555	2

to be more crucial than offline training time in real-time prediction or process execution monitoring scenarios. Accordingly, the NN-based approaches have advantages in online prediction tasks compared with these traditional machine learning approaches. In particular, as shown in Table 5, the *online time* of these GRU-based approaches, i.e., the GRU, BiGRU, CNN-GRU, and CNN-BiGRU, can keep the steady real-time prediction.

5.3 Interpretability of Prediction Results

In practical application, we can use our proposed approach to make predictions about the final multi-perspective conformance checking for an ongoing process instance. As we know, the ongoing case indicates a prefix trace that consists of a series of performed activities (i.e., events). These performed activities have many attribute values, such as the *activity* name, *resource*, *timestamp*, and *amount*. In addition, there are some attributes can generated from these original attributes, such as the *open_case*, *event_nr*, and *time_since_midnight*. As shown in Figs. 7 and 8, we take two decision points of an ongoing case in *BPIC2012* for example. Here, 'A_SUBMITTED-COMPLETE' is the loan application submission stage and the first decision point 'A_PARTLYSUBMITTED-COMPLETE' is the supplementary submission stage. The current state of this process case is called prefix trace. Before prediction, the above-mentioned approaches (classifiers) have the ability to learn the relationship between the encoded features of a case and the class of its multi-perspective conformance based on the similar prefix traces generated from the training set. Take a prediction for a case in this decision point, the input of this ongoing case for each approach (classifier) includes all attributes of the completed activities and these attributes are encoded as a numerical value according to the previous description. As shown in Figs. 7 and 8, based on our approach CNN-BiGRU, the prediction probability of prediction target *True* (conformance) is 0.38 at the first decision point and then the prediction probability rises to 0.39 when this case further completes activity 'W_Afhandelen leads-SCHEDULE'. As the process instance moves forward, the prediction probability of the prediction target *True* (conformance) doesn't always increase. Take the case, and for example, the prediction probability will drop to 0.33 after completing 5 activities and then increase to 0.36 after completing 8 activities. Moreover, this change will happen again. As for this case, the prediction probability gradually increases from 0.54 to 0.89

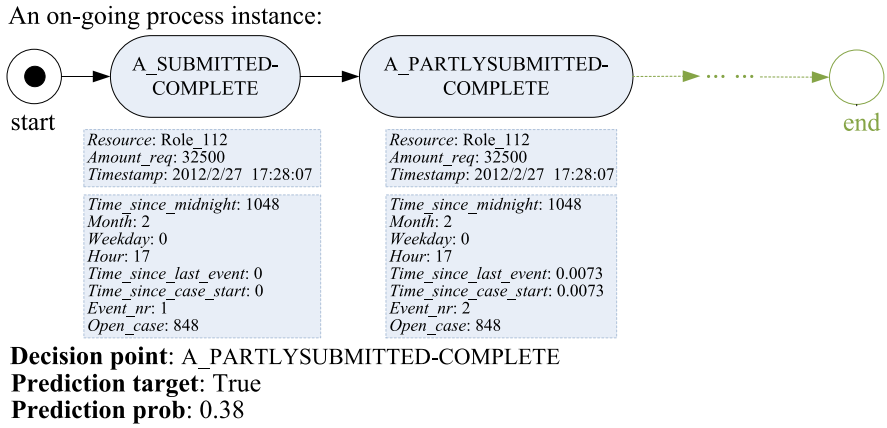

Fig. 7 The explanation for *True* (conformance) prediction target at 'A_PARTLYSUBMITTED-COMPLETE' decision point

(after completing 39 activities) when 25 activities are completed.

6 Conclusion and Future Work

It is very important to predictively monitor the final conformance for an executing process instance regarding the security of process execution load in the cloud-based BPaaS application. In this article, we concentrated on the multi-perspective conformance-oriented predictive process monitoring task for enhancing the security of BPaaS. We then proposed an extensible multi-perspective (i.e., the structure, data, resource, and time) conformance measurement first. Based on it, given a predefined process model in BPaaS with some multi-perspective constraints (determined by tenants), the multi-perspective conformance of an executed case in historical event log can be determined and viewed as supervised knowledge. To predict the multi-perspective conformance of an executing process instance for a tenant, we proposed the CNN-BiGRU approach to build a prediction model from the historical executed cases that correspond to this tenant by combining the CNN neural network with the variant and enhancement of the RNN neural network. The proposed CNN-BiGRU simultaneously uses a 3-layer CNN network to aggregate the features of attributes and a multi-layer bidirectional GRU network to extract the temporal relation of events. In addition, we developed a framework in which we can make a multi-perspective conformance prediction for an executing process instance based on neural networks to enhance the BPaaS security. Extensive experimental results on two event logs demonstrated the superiority of our CNN-BiGRU

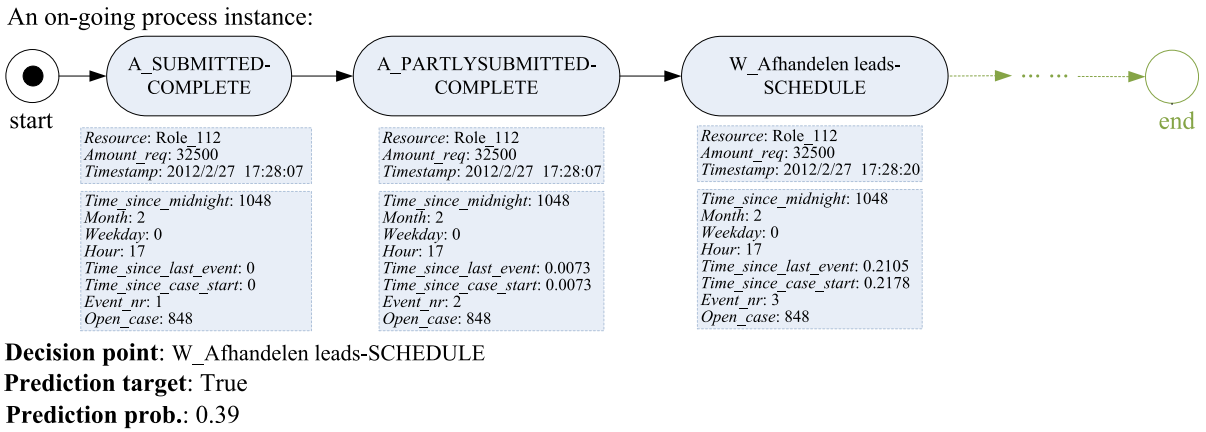

Fig. 8 The explanation for *True* (conformance) prediction target at 'W_Afhandelen leads-SCHEDULE' decision point

approach by comparing it with a bundle of state-of-the-art technologies on process prediction tasks.

However, in terms of the applicability, there are some limits of the proposed method in this paper. For example, our proposed solution for conformance-oriented predictive process monitoring needs to have an original regulatory process model as a baseline when measuring the multi-perspective conformance of an executed process instance. Moreover, due to page limitations, this paper does not explore what to do after the compliance prediction for an ongoing case. Therefore, we are going to develop a strategy of how to take action on an ongoing case initiated by tenants based on the result of conformance predictions in the future. Besides, as some more efficient feature representation learning techniques are proposed, our future work will consider more contextual information to improve the performance of a multi-perspective conformance prediction model. Last but not least, with the continued execution of a process in BPaaS for a tenant, we also plan to investigate the incremental conformance prediction based on neural networks to avoid duplicate offline training.

Acknowledgments This work is supported by the Natural Science Foundation of China (No. 62002316), the VC Research (VCR 000067) for Prof. Chang, the Key Research and Development Program of Zhejiang Province (No. 2019C03138), the Key Science and Technology Project of Zhejiang Province (No. 2017C01010), and Zhejiang Provincial Natural Science Foundation (No. LQ20F020017).

Data Availability The experiment data supporting this experiment analysis are from the website (https://researchdata.4tu.nl/home/), which has been described in a footnote to the article. The experiment data used to support the findings of this study are included in the article. The experiment data are described in Section 5 in detail. And the source code is available in Github (https://github.com/jiaojiaowang1992/multi-perspective_conformance-oriented_PPM.git).

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Cusumano, M.: Cloud computing and saas as new computing platforms. Communation of the ACM **53**(4), 27–29 (2010). https://doi.org/10.1145/1721654.1721667
2. Tsai, W.T., Bai, X.Y., Huang, Y.: Software-as-a-service (saas): perspectives and challenges. Science China Information Sciences **57**(5), 1–15 (2014). https://doi.org/10.1007/s11432-013-5050-z
3. Sun, Y., Su, J., Yang, J.: Separating execution and data management: A key to business-process-as-a-service (bpaas). In: Sadiq, S., Soffer, P., Völzer, H. (eds.) Business Process Management, pp. 374–382. https://doi.org/10.1007/978-3-319-10172-9_25 (2014)
4. Bentounsi, M., Benbernou, S., Atallah, M.J.: Security-aware business process as a service by hiding provenance. Computer Standards and Interfaces **44**, 220–233 (2016). https://doi.org/10.1016/j.csi.2015.08.011
5. Woitsch, R., Hinkelmann, K., Ferrer, A.M.J., Yuste, J.I.: Business process as a service (bpaas): The smart bpaas design environment CAiSE 2016 Industry Track. https://doi.org/10.26041/fhnw-1020 (2016)
6. Gzik, T.: Business process as a service - a systematic literature review. Towards Industry 4.0—Current Challenges in Information Systems, pp. 163–181. https://doi.org/10.1007/978-3-030-40417-8_10 (2020)
7. Qi, M., Wang, Y., Xiang, J., Li, T.: A correctness checking approach for collaborative business processes in the cloud. Science China Information Sciences, pp. 2020 (2020)
8. Verenich, I.: A general framework for predictive business process monitoring. In: Proceedings of CAiSE 2016 Doctoral Consortium, pp. 1–9. http://ceur-ws.org/Vol-1603/10000053.pdf (2016)
9. Burattin, A., Carmona, J.: A framework for online conformance checking. In: International Conference on Business Process Management, pp. 165–177. Springer. https://doi.org/10.1007/978-3-319-74030-0_12 (2017)
10. Burattin, A., van Zelst, S.J., Armas-Cervantes, A., van Dongen, B.F., Carmona, J.: Online conformance checking using behavioural patterns. In: International Conference on Business Process Management, pp. 250–267. Springer. https://doi.org/10.1007/978-3-319-98648-7_15 (2018)
11. van Zelst, S.J., Bolt, A., Hassani, M., van Dongen, B.F., Van der Aalst, W.M.P.: Online conformance checking: relating event streams to process models using prefix-alignments. International Journal of Data Science and Analytics **8**(3), 269–284 (2019). https://doi.org/10.1007/s41060-017-0078-6
12. Song, W., Xia, X., Jacobsen, H.-A., Zhang, P., Hu, H.: Efficient alignment between event logs and process models. IEEE Trans. Serv. Comput. **10**(1), 136–149 (2016). https://doi.org/10.1109/TSC.2016.2601094

13. de Leoni, M., Marrella, A.: Aligning real process executions and prescriptive process models through automated planning. Expert Syst. Appl. **82**, 162–183 (2017). https://doi.org/10.1016/j.eswa.2017.03.047
14. García-Bañuelos, L., Van Beest, N.R.T.P., Dumas, M., La Rosa, M., Mertens, W.: Complete and interpretable conformance checking of business processes. IEEE Trans. Softw. Eng. **44**(3), 262–290 (2017). https://doi.org/10.1109/TSE.2017.2668418
15. Leemans, S.J.J., Fahland, D., Van der Aalst, W.M.P.: Scalable process discovery and conformance checking. Softw. Syst. Model. **17**(2), 599–631 (2018). https://doi.org/10.1007/s10270-016-0545-x
16. Dunzer, S., Stierle, M., Matzner, M., Baier, S.: Conformance checking: a state-of-the-art literature review. In: Proceedings of the 11th International Conference on Subject-Oriented Business Process Management, pp. 1–10. https://doi.org/10.1145/3329007.3329014 (2019)
17. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with lstm neural networks. In: International Conference on Advanced Information Systems Engineering, pp. 477–492. Springer. https://doi.org/10.1007/978-3-319-59536-8_30 (2017)
18. Teinemaa, I., Dumas, M., La Rosa, M., Maggi, F.M.: Outcome-oriented predictive process monitoring: Review and benchmark. ACM Transactions on Knowledge Discovery from Data (TKDD) **13**(2), 1–57 (2019). https://doi.org/10.1145/3301300
19. Mehdiyev, N., Evermann, J., Fettke, P.: A novel business process prediction model using a deep learning method. Business and Information Systems Engineering **62**(2), 143–157 (2020). https://doi.org/10.1007/s12599-018-0551-3
20. Weytjens, H., De Weerdt, J.: Process outcome prediction: Cnn vs. lstm (with attention). In: International Conference on Business Process Management, pp. 321–333. Springer. https://doi.org/10.1007/978-3-030-66498-5_24 (2020)
21. Pasquadibisceglie, V., Appice, A., Castellano, G., Malerba, D.: Using convolutional neural networks for predictive process analytics. In: 2019 International Conference on Process Mining (ICPM), pp. 129–136. IEEE. https://doi.org/10.1109/ICPM.2019.00028 (2019)
22. Kratsch, W., Manderscheid, J., Röglinger, M., Seyfried, J.: Machine learning in business process monitoring: a comparison of deep learning and classical approaches used for outcome prediction. Business Information System Engineering **63**(3), 261–276 (2021). https://doi.org/10.1007/s12599-020-00645-0
23. Liu, P., Qiu, X., Huang, X.: Recurrent neural network for text classification with multi-task learning. In: Kambhampati, S. (ed.) Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence IJCAI, pp. 2873–2879. IJCAI/AAAI Press. https://www.ijcai.org/Proceedings/16/Papers/408.pdf (2016)
24. Wang, J., Yu, L.-C., Robert Lai, K., Zhang, X.: Dimensional sentiment analysis using a regional cnn-lstm model. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp 225–230. https://www.aclweb.org/anthology/P16-2037.pdf (2016)
25. Wang, S., Huang, M., Deng, Z.: Densely Connected Cnn with Multi-Scale Feature Attention for Text Classification. In: IJCAI, pp. 4468–4474. https://doi.org/10.24963/ijcai.2018/621 (2018)
26. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
27. Cho, K., Van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder-decoder approaches. arXiv:1409.1259. https://www.aclweb.org/anthology/W14-4012.pdf (2014)
28. Shewalkar, A., Nyavanandi, D., Ludwig, S.A.: Performance evaluation of deep neural networks applied to speech recognition: Rnn, lstm and gru. Journal of Artificial Intelligence and Soft Computing Research **9**(4), 235–245 (2019). https://doi.org/10.2478/jaiscr-2019-0006
29. Rozinat, A., Van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. **33**(1), 64–95 (2008). https://doi.org/10.1016/j.is.2007.07.001
30. Adriansyah, A., Sidorova, N., van Dongen, B.F.: Cost-based fitness in conformance checking. In: 2011 15Th International Conference on Application of Concurrency to System Design, pp. 57–66. IEEE. https://doi.org/10.1109/ACSD.2011.19 (2011)
31. Munoz-Gama, J., Carmona, J., Van der Aalst, W.M.P.: Conformance checking in the large: Partitioning and topology. In: Business Process Management, pp. 130–145. Springer. https://doi.org/10.1007/978-3-642-40176-3_11 (2013)
32. Burattin, A., Maggi, F.M., Sperduti, A.: Conformance checking based on multi-perspective declarative process models. Expert Syst. Appl. **65**, 194–211 (2016). https://doi.org/10.1016/j.eswa.2016.08.040
33. Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Process diagnostics using trace alignment: opportunities, issues, and challenges. Inf. Syst. **37**(2), 117–141 (2012). https://doi.org/10.1016/j.is.2011.08.003
34. Mannhardt, F., De Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Balanced multi-perspective checking of process conformance. Computing **98**(4), 407–437 (2016). https://doi.org/10.1007/s00607-015-0441-1
35. Alizadeh, M., Lu, X., Fahland, D., Zannone, N., van der Aalst, W.M.P.: Linking data and process perspectives for conformance analysis. Computers and Security **73**, 172–193 (2018). https://doi.org/10.1016/j.cose.2017.10.010
36. De Leoni, M., Van Der Aalst, W.M.P., Van Dongen, B.F.: Data-and resource-aware conformance checking of business processes. In: International Conference on Business Information Systems, pp. 48–59. Springer. https://doi.org/10.1007/978-3-642-30359-3_5 (2012)
37. De Leoni, M., Van Der Aalst, W.M.P.: Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming. In: Business Process Management, pp. 113–129. Springer. https://doi.org/10.1007/978-3-642-40176-3_10 (2013)
38. Van der Aalst, W., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **2**(2), 182–192 (2012). https://doi.org/10.1002/widm.1045

39. Burattin, A.: Online conformance checking for petri nets and event streams. In: 15Th International Conference on Business Process Management (BPM 2017). https://core.ac.uk/download/pdf/97180593.pdf (2017)
40. Rogge-Solti, Andreas, Weske, Mathias: Prediction of business process durations using non-markovian stochastic petri nets. Inf. Syst. **54**, 1–14 (2015). https://doi.org/10.1016/j.is.2015.04.004
41. Appice, Annalisa, Mauro, Nicola Di, Malerba, Donato: Leveraging shallow machine learning to predict business process behavior. In: 2019 IEEE International Conference on Services Computing (SCC), pp. 184–188. IEEE. https://doi.org/10.1109/SCC.2019.00039 (2019)
42. Harl, Maximilian, Weinzierl, Sven, Stierle, Mathias, Matzner, Martin: Explainable predictive business process monitoring using gated graph neural networks. J. Decis. Syst., pp. 1–16. https://doi.org/10.1080/12460125.2020.1780780 (2020)
43. Maria Maggi, Fabrizio, Di Francescomarino, Chiara, Dumas, Marlon, Ghidini, Chiara: Predictive monitoring of business processes. In: International conference on advanced information systems engineering, pp. 457–472. Springer. https://doi.org/10.1007/978-3-319-07881-6_31 (2014)
44. Lakshmanan, Geetika T, Shamsi, Davood, Doganata, Yurdaer N, Unuvar, Merve, Khalaf, Rania: A markov prediction model for data-driven semi-structured business processes. Knowl. Inf. Syst. **42**(1), 97–126 (2015). https://doi.org/10.1007/s10115-013-0697-8
45. Leontjeva, Anna, Conforti, Raffaele, Di Francescomarino, Chiara, Dumas, Marlon, Maria Maggi, Fabrizio: Complex symbolic sequence encodings for predictive monitoring of business processes. In: International Conference on Business Process Management, pp. 297–313. Springer. https://doi.org/10.1007/978-3-319-23063-4_21 (2016)
46. Ferilli, Stefano, Esposito, Floriana, Redavid, Domenico, Angelastro, Sergio: Extended process models for activity prediction. In: International Symposium on Methodologies for Intelligent Systems, pp. 368–377. Springer. https://doi.org/10.1007/978-3-319-60438-1_36 (2017)
47. Taymouri, F., La Rosa, M., Erfani, S.M., Bozorgi, Z.D., Verenich, I.: Predictive business process monitoring via generative adversarial nets: The case of next event prediction. In: International Conference on Business Process Management, vol. 12168, pp. 237–256. Springer. https://doi.org/10.1007/978-3-030-58666-9_14 (2020)
48. Bukhsh, Z.A., Saeed, A., Dijkman, R.M.: Processtransformer: Predictive business process monitoring with transformer network. arXiv:2104.00721 (2021)
49. Park, G., Song, M.: Predicting performances in business processes using deep neural networks. Decis. Support. Syst. **129**, 113191 (2020). https://doi.org/10.1016/j.dss.2019.113191
50. Elman, J.L.: Finding structure in time. Cognitive Science **14**(2), 179–211 (1990). https://doi.org/10.1207/s15516709cog1402_1
51. Olson, R.S., La Cava, W., Mustahsan, Z., Varik, A., Moore, J.H.: Data-driven advice for applying machine learning to bioinformatics problems. Pac Symp Biocomput, 23. https://doi.org/10.1142/9789813235533_0018 (2018)
52. De Leoni, M., van der Aalst, W.M.P., Dees, M.: A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. Inf. Syst. **56**, 235–257 (2016). https://doi.org/10.1016/j.is.2015.07.003
53. Senderovich, A., Di Francescomarino, C., Ghidini, C., Jorbina, K., Maggi, F.M.: Intra and inter-case features in predictive process monitoring: A tale of two dimensions. In: International Conference on Business Process Management, pp. 306–323. Springer. https://doi.org/10.1007/978-3-319-65000-5_18 (2017)
54. Andrew, P.: Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern Recogn. **30**(7), 1145–1159 (1997). https://doi.org/10.1016/S0031-3203(96)00142-2
55. Bergstra, James, Bengio, Yoshua: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**(1), 281–305 (2012). https://doi.org/10.5555/2503308.2188395
56. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006). https://doi.org/10.5555/1248547.1248548